

A SMART FUZZY AUTO SUGGESTION SYSTEM FOR A MULTILAYER QR CODE GENERATOR

¹BAKRI BADAWI, ²TEH NORANIS MOHD ARIS, ³NORWATI MUSTAPHA, ⁴NORIDAYU MANSHOR

^{1,2,3,4}Department of Computer Science, Faculty of Computer Science and Information Technology, 43400 UPM Serdang, Selangor, Malaysia

E-mail: ¹bakri.info@gmail.com, ²nuranis@upm.edu.my (corresponding author)
³norwati@upm.edu.my, ⁴ayu@upm.edu.my

ABSTRACT

One of the current major developments in color Quick Response (QR) code requires larger data capacity. Many research works proposed color QR code with a data capacity three times larger compared to black and white QR code. These works emphasize on the algorithm about how to generate color QR code but, it does not provide sufficient information on color QR code generator implementation. In this paper, we will show the implementation for multilayer QR code generator with smart suggestion for the number of colors, based on size available on paper and size of data file. Fuzzy technique is utilized for the smart suggestion. The proposed generator also explains how to add color reference for the generated QR code. This color reference indicates how many colors are used for this QR code. The proposed generator can generate color QR code with data capacity four times larger compared to black and white QR code and a 25% larger color QR code compared to any existing work.

Keywords: *Smart, Fuzzy, Suggestion, Color QR Code, Generator*

1. INTRODUCTION

QR code is a special type of 2D barcodes. The key future for QR code is larger data capacity, and fast decoding rate compared to normal barcode [3,6,7]. Applications that use QR code have increased in recent year [1,4,5,7,8,3,2]. This increase is due to the popularity of QR code that is used in various applications, such as bus ticket, cinema ticket, store front, business card, products in supermarket, new event posters, URL links, street advertisement, etc., [1,4,5,8,9,6]. The main limitation for current QR code is the data size that can be encoded in single QR code. Many researches work on QR code provide custom solution to overcome the size limitation [10,11,12,14,15]. The new generation of QR code is color QR code which provides same key features for QR code like error correction, 360-degree reading, high-speed reading, with higher data capacity up to three times larger than normal black and white QR code. The increase in data size is due to multiple color layer. Current color QR code generators can generate QR code with maximum of 3 layers which is due to using Red Green Blue (RGB) color format. This color format

assigns one main color for each layer [1,2,5,6,7,8,9]. To increase the size of color QR code we need to add more color layers, which will create two major issues: first, is to create the color format and second, how the decoder can identify what colors are used in the encoder. To date, there does not exist any algorithm for color format with four basic colors and decoder to identify the colors used in the encoder. Current research on color QR code generator addressed the generate algorithm but does not address the implementation for the color QR code. In this paper, we will show the implementation of our proposed color QR code generator with a smart color suggestion utilizing fuzzy technique. Our proposed algorithm suggests the number of layers based on available size on paper and data file. Our generator can generate color QR code up to four color layers with color reference that help the decoder to identify the color used in the generator.

2. BLACK AND WHITE QR CODE STRUCTURE

QR code has eight functionality patterns that use to facilitate the decoding process, which are:

2.1 Finder Pattern

The structure and position of the finder pattern allow detection of the QR code with the size of the angle, to enable QR code to be detected in all directions (360°). In Fig. 1, the finder pattern is shown in the three corners of QR code, labeled with number 1.

2.2 Separator Pattern

The internal empty space finder pattern and other QR code pattern, help the decoder to ease the detection for finder pattern. The separator pattern in Fig. 1 is next to finder pattern, labeled with number 2.

2.3 Timing Pattern

Timing pattern are two lines, one horizontal and one vertical, located between the separators. These lines consist of alternating dark and light modules. The horizontal timing pattern is placed on the 6th row of the QR code between the separators. The vertical timing pattern is placed on the 6th column of the QR code between the separators. This pattern helps to detect the position of each cell in the QR code. The timing pattern is labeled with number 3 in Fig. 1, which is between the Separator Pattern.

2.4 Alignment Pattern

The Alignment Pattern can be found in QR code version 2 or higher. This pattern is used for correcting the distortion of QR code. The center of the alignment pattern will be used for correcting the distortion, therefore, a black isolated cell is placed in the alignment pattern to make it easier to detect. From Fig. 1, the Alignment Pattern is in the lower right-hand corner, labeled number 4.

2.5 Format Information

The Format Information in Fig. 1 is next to separator pattern, labeled as number 5. This pattern contains the error correction rate and mask of the QR code. The format information is read first when the code is decoded.

2.6 Data Pattern

The encoded data will be stored in this pattern. The data will be encoded in binary format '1' and '0'. The binary numbers of 0 and 1 will be converted into black and white cell. Error correction codes are inserted into this area as well. Data Pattern is shown in Fig.1, number 6.

2.7 Remainder Pattern

For some QR versions, this pattern consists of empty bits. The data and error correction bits cannot be divided into 8-bit code words. In this case, it is necessary to add a certain number of 0's to the end of the final message to make it have the correct length. These extra 0's are called remainder bits. Remainder pattern is shown in Fig. 1, number 7.

2.8 Quiet Zone

Quiet Zone is a margin space necessary for reading the QR code. This quiet zone makes it easier to detect the QR code from among the image. Quiet Zone is shown in Fig. 1, number 8.

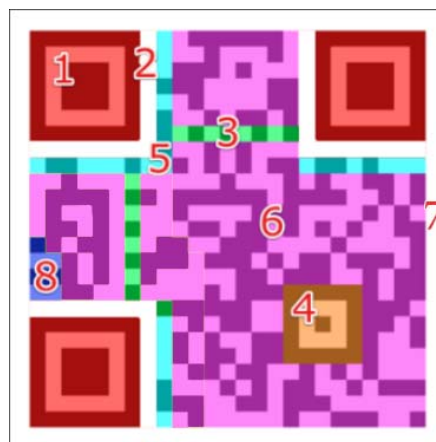


Figure 1: Qr Code Strucher

3. BLACK AND WHITE QR CODE SPECIFICATION

Black and white QR codes have different version that ranges from version 1 to version 40. Each version has different model configuration which refer to the number of cells that contains data. Version 1 has 21 * 21 modules, with increment of four models for each version. There are four errors Correction Level:

- 1- Low (L) up to 7% damage

- 2- Medium (M) up to 15% damage
- 3- Quality (Q) up to 25% damage
- 4- High (H) up to 30 % damage

There are 2 factors that we need to consider to select the data capacity for each model:

- 1- Error Correction Level
- 2- Data Type

The data type may vary: Data bits, Numeric, Alphanumeric, Binary, Kanji. Table 1 summarizes black and white QR code specification.

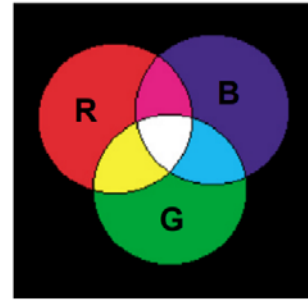


Figure 2: Rgb Color Model

Table 1: 2d-Barcodes

Symbol size	Min. 21x21 cell - Max. 177x177 cell (with 4-cells interval)	
Maximum Data Size (version 40)	Data bits	10,208
	Numeric	3,057
	Alphanumeric	1,852
	Binary	1,273
Error correction	Kanji	784
	Level L	UP to 7%
	Level M	UP to 15%
	Level Q	UP to 25%
	Level H	UP to 30%

4. COLOR MODELS

There are two main color models that are used nowadays in computer systems. We will show a brief for those models, in order to explain the limitation of the number of color layers for color QR code.

4.1 RGB Color Model

This color model is mostly used in computer screens. The name for this model is derived from main primary colors (Red, Green and Blue). The color in this model is created by mixing different light colors. More colored lights mixed will produce more brightness of the light. When mixing all red, green and blue components equally, will result a white color. Therefore we call this model additive color. Model in Fig. 2 show an example of RGB color model [26, 27].

4.2 CMYK Color Model

This color model is mostly used in printers. The name for this model is derived from four inks used in some color printers (Cyan, Magenta, Yellow and Black). The color in this model is created by mixing different inks. More mixed inks will produce more darkness of the light. When mixing all Cyan, Magenta and Yellow components equally, will result in a black color. Therefore we call this model subtractive color model. Black ink is added for this model because by mixing the three-primary color for this model (Cyan, Magenta, Yellow) will not result in a pure black color but will result in a very dark brown color. Fig. 3 show an example of CMYK color model [26, 27].

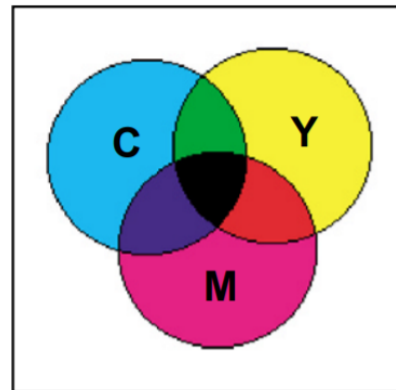


Figure 3: Cmyk Color Model

5. COLOR QR CODE

Color QR code consist of color multiplexing for many layers of monochrome QR code, resulting a colored QR code with data capacity equal to black and white QR code data size multiply by the number of layers [1,2,3]. Since the color QR code is resulting of color multiplexing between the primary color for the color model, the maximum

number of layers for the color QR code is three layers, since we have only three main colors for RGB/CMYK color model. In our proposed color QR code, we will show how we can extend this limitation. To get color QR code for specific data, we need to first split the data into three equal chunks, then we generate three monochrome QR code, the color of those QR code is based on the selected color model either red, green, blue or Cyan, Magenta, Yellow, then we mix those three QR code to get one colored QR code. The number of colors = $2^{\text{number of layers}}$. Which mean, for three layers (the maximum) we have $2^3 = 8$ colors. Fig. 4 shows the steps of how to generate three layers of color QR code.

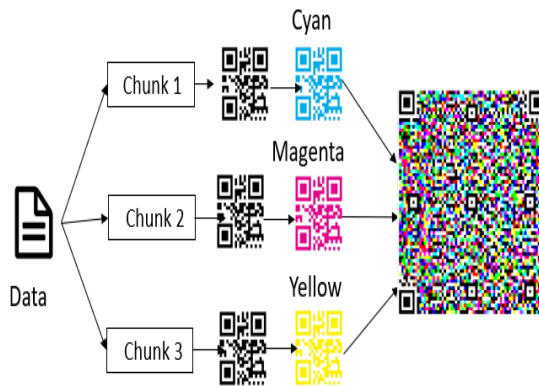


Figure 4: Current Color Qr Code Generator Process

6. RELATED RESEARCH WORKS

This section explains the algorithm for QR code generator implemented in other research work. Then we will make a comparison between existing generators. These generators will be used as benchmarks in our research work.

6.1 Research Work by Zhibo Yang, 2018

The researchers for this work have proposed encoder and decoder for color QR code. The color QR code encoder algorithm is as follows:

- i. Split the file into three equal parts.
- ii. Encode each part into black and white QR code.
- iii. Assign color for each QR code, which will produce red, green and blue QR codes.
- iv. Merge all those QR code into the colored QR code using color multiplexing.

Fig. 5 shows the encoder process.

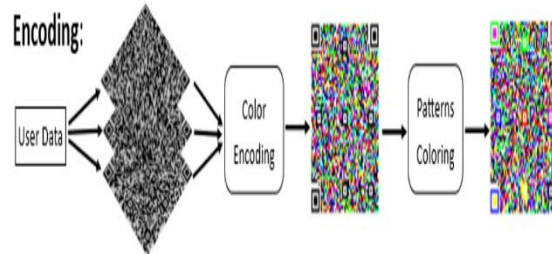


Fig. 5: Encoder By Zhibo Yang, 2016

From our review, this research work provides standard color QR code with data capacity equal to three times larger than black and white QR code. However, the encoder algorithm proposed to add one color reference in the finder pattern which will give incorrect color sampling at the shadow area. This algorithm is using the RGB color format which allow maximum up to 3 color layers only.

6.2 Research Work by Blasinski, Henryk, 2013

The researchers proposed color QR code encoder and decoder based on CMYK color model. The encoder works as follow:

- i. Split the file into three equal parts.
- ii. Encode each part into black and white QR code.
- iii. Assign color for each QR code, which will produce cyan, magenta and yellow QR codes.
- iv. Marge all those QR code into the colored QR code using color multiplexing.

The difference between this research work with research work in section 6.1 is in step iii., where the result is cyan, magenta and yellow QR codes.

From our review, this system provides color QR code based on CMYK with data capacity three times larger of black and white QR code. However, their algorithm is using CMYK color system which will create more color illumination on QR code from computer screens which will make the decoding part much harder than using RGB color format. In their algorithm there is no color reference to guide the decoder in what colors are used in the QR code.

6.3 Research Work by Thilo Fath, Falk Schubert, and Harald Haas, 2014

This research used stream of color QR code which is available in airplane data transmutation. Since there is no wireless or mobile data QR code in

airplane, it provides a good example of offline data transmission. The algorithm work as follows:

- i. The file to be transferred on the airplane, for instance text documents or images, is compressed to reduce the amount of data.
- ii. The compressed data is segmented into several packets.
- iii. Each packet is encoded by a black and white QR code resulting in a sequence of several visual codes.
- iv. Each three black and white QR code is converted to one colored QR code.
- v. Those colored QR code sequences are displayed in a continuous loop on the In-Flight Entertainment (IFE) screen like a common video film.

The encoding process is shown in Fig. 6.

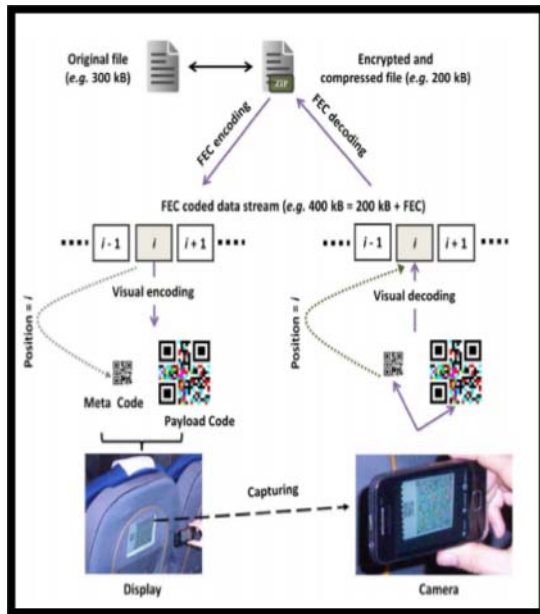


Fig. 6: IFE Screen by Thilo Fath, Falk Schubert and Harald Haas, 2014

From our review, this research work provides a new aircraft data transmission, where the system suggests video streaming of colored QR code for the data that exceeds the maximum of three layers color QR code. Their algorithm is using RGB color format which allow maximum up to 3 color layers only. In addition, the algorithm does not provide color reference to guide the decoder on what color are used in the QR code.

6.4 Research Work by Sin Rong Toh, Weihan Goh and Chai Kiat Yeo, 2016

This research work proposes color QR code encoder and decoder using color multiplexing. The sender algorithm works as follows:

- i. Insert the black and white QR code version.
- ii. Split the data that needs to be encoded in many equal chunks, where each chunk has the size of the selected black and white QR code.
- iii. Encode each three chunks into three monochrome QR code (red, green and blue).
- iv. Using color multiplexing merge each 4 monochrome QR code into 1 colored QR code.

The sending and receiving process is shown in Fig. 7.

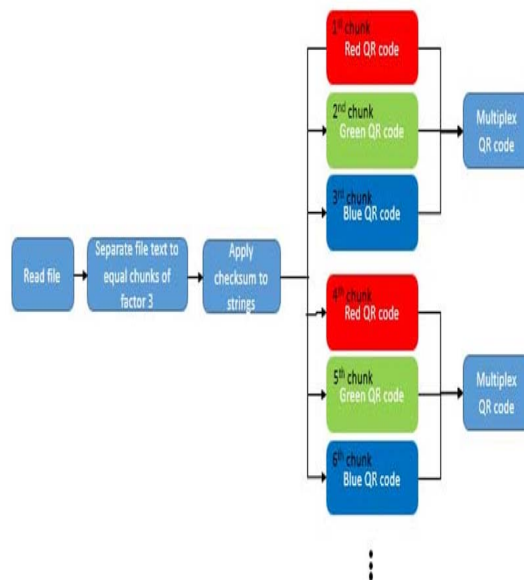


Fig. 7: Thilo Fath, Falk Schubert and Harald Haas, 2014

From our review, this research work provides standard color QR code with data capacity equal to three times larger than black and white QR code. However, the proposed encoder is using three monochrome color layers which will create color QR code with maximum capacity up to 3 times larger compared to black and white QR code. From our findings, the algorithm does not provide color reference to guide the decoder what color are used in the QR code.

6.5 Research Work by Tian Hao, Ruogu Zhou, Guoliang Xing 2012

These researchers proposed colored barcode to increase the capacity of the QR code. They are using color encoding to achieve the color barcode.

The encode algorithm is as follows:

- i. Convert the data that needs to be encoded into bit stream 0's and 1's.
- ii. Convert each two bits into single color.
- iii. Red color represents 00, green color represents 01, blue color represents 10, and white color represent 11.
- iv. Add a frame of black dots, and color reference, to help the decoder to identify the color used and to locate the proposed barcode.

The size for the generated barcode on a 4-inch phone screen with 800×480 resolution, with 6-pixel block size, is 18.8K bits.

The proposed barcode is shown in Fig. 8

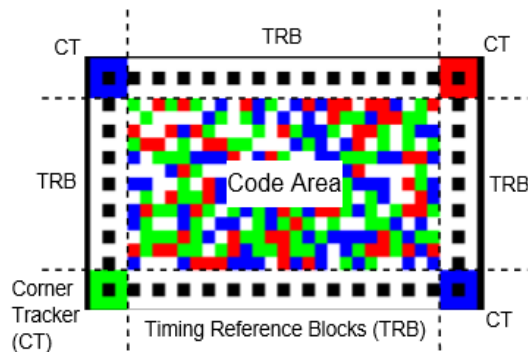


Fig. 8 Tian Hao, Ruogu Zhou and Guoliang Xing 2012

From our review, this research work provides new color barcode with data capacity larger than black and white QR code. In addition, this system has color reference for the used color which makes the decoding easier. However, their encoder algorithm used only 5 colors which will create barcode with only 2 times larger than black and white barcodes.

6.6 Research Work by Rayana Boubezari, Hoa Le Minh, Zabih Ghassemlooy and Ahmed Bouridane, 2016

They propose mobile-to-mobile communications using barcode stream. Their encoder algorithm is as follows:

- i. Convert the data that needs to be encoded into bit stream 0's and 1's.
- ii. Represent the bit with value 0 by black color.

- iii. Represent the bit with value 1 by white color.
- iv. Add black frame for the barcode for easier detection.

The proposed barcode is shown in Fig. 9.



Fig. 9: Rayana Boubezari, Hoa Le Minh, Zabih Ghassemlooy and Ahmed Bouridane, 2016

From our review, this research work provides a new black and white barcode with data capacity larger than black and white QR code. However, this system does not provide important features that most QR code has, like 360 degree and error correction. In addition, the data size encoded within the proposed barcode can be enhanced by adding more colors.

6.4 Comparison with Existing Systems

We will use this comparison as our benchmark. We compare our proposed decoder with six other existing works, in seven aspects consisting of: (a) Generator, (b) Max data size, (c) Number of colors, (d) Algorithm used, (e) Color Selection, (f) Number of layers and (g) Implementation guideline. As shown in Table 2, we can summarize that, current color QR code encoder can encode maximum of three layers QR code with 8 colors. The selection for the color is static and the maximum data size that can be encoded is 8868 bytes. Our proposed system can select the number of colors dynamically based on the data size. In addition, our proposed encoder can encode four layers of QR code producing a color QR code with 16 colors. The maximum data size for our encoder is 11824 bytes. Applying the increase percentage calculation, we can obtain the increment percentage from our proposed encoder. The calculation is as follows:

$$\text{Increase byte} = \frac{\text{Max data size for our generator} - \text{max data size for existing generator}}{\text{max data size for existing generator}} \times 100\%$$

Increase percentage = Increase byte ÷ max data size for our proposed generator $X 100 = ((11824 - 8868) \div 11824) * 100 = 25\%$

7. PROPOSED COLOR QR CODE GENERATOR ALGORITHM

The process for the proposed color QR code is shown in Fig. 10. The user first needs to select the size on paper and the data file that needs to be encoded. Then, the selection of the QR code version and number of layers is done using fuzzy technique. Next, the data is split into chunks, equal to the number of layers, with the selection done using fuzzy technique. After that, each chunk is encoded into black and white QR code, based on the number of layers which is assigned to the color for each layer. This mean, black and white if only one layer; red and green, if two layers; red, green and blue, for three layers and red, green, blue and 100 color value for four layers. 100 color value layers is to over come the limitation of three main colors in the RGB color model. This will be explained in more detail in section 7.1. The next step is using color multiplexing which will produce the color QR code. The final process is to add color reference to the finder pattern.

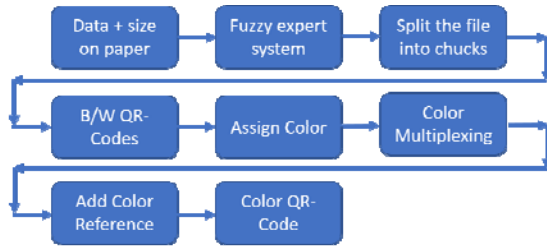


Figure 10: Proposed Color Qr Code Generator Process

The proposed color QR code encoder algorithm work as follows:

- i. First the user inserts the max size in paper and its data to encode.
- ii. Our framework will check the max size on paper and select the best fit for black and white QR code for that space using fuzzy technique.
- iii. Select the number of chunks = data size / max size for the selected QR code.
- iv. Number of colors = 2[^] number of chunks.
- v. Generate black and white QR code for each data chunk.

- vi. Give each chunk a specific color.
- vii. Merge all generated QR code into one color QR code.
- viii. Replace the identification pattern with color reference identification pattern.

Fig. 11 shows our proposed color QR code generator.

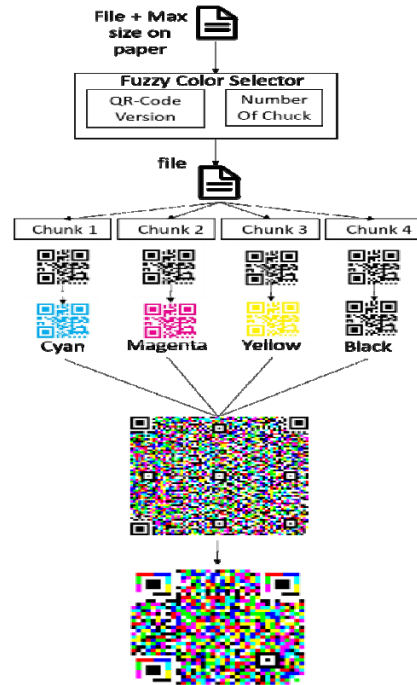


Figure 11: Proposed Color QR Code Generator

7.1 Fuzzy Process

The fuzzy process has two inputs: (a) Data that need to be encoded and (b) the size on paper. After the fuzzy process, the QR code version and number of chunks will be the fuzzy output. Fig. 12 is an overview of the fuzzy process.

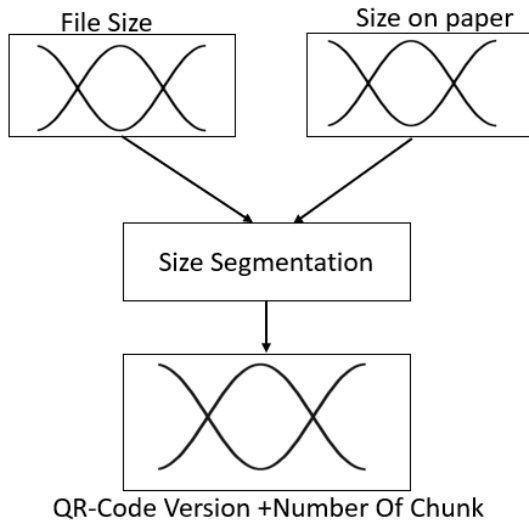


Figure 12: Overview Of The Proposed Fuzzy Process

7.1.1 Fuzzification

This process converts the input to fuzzy input. In our proposed fuzzy algorithm, the size on paper will be converted to QR code version v and QR code version $V+$, and the size of file will be divided to the QR code data size. If the result is more than four, then our algorithm will select the QR code with higher version and repeat the fuzzification process.

7.1.2 Membership Function

The character of the membership function for the minimum number of chunk is 1 and maximum is 4. The output will be the number of chunks with the selected version (1, 2, 3, 4) based on the input or selected bigger version (1+,2+,3+,4+). Fig. 13 shows the membership function.

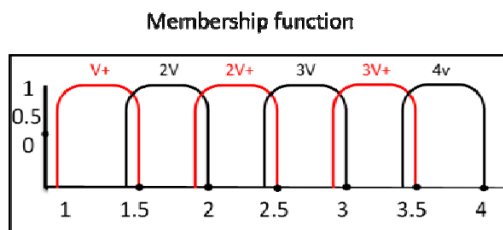


Figure 11: Membership Function

The input membership functions are:

- Minimum: 1.
- Maximum: 4.
- Values ($v+$, $2v$, $2v+$, $3v$, $3v+$, $4v$).

7.1.3 Fuzzy Rules

In our proposed QR code generator, fuzzy rules are used to select the number of chunks plus the QR code version. In the defuzzification process, the selected version and the resulting number are obtained and we consider it as the number of chunks.

The generator rules is as follows:

```

if (selected value in range [1-4]) then
  if (decimal value  $\leq 0.5$ ) then
    version = v
    number of chunk = selected value+1-
    decimal value;
  
```

else

```

    version= $v+$ 
    number of chunk = selected value-decimal
    value;
  
```

else $v = v+$ and $v+ = v++$ (select higher value and go for another) and go for a new round of the fuzzy membership function.

7.1.4 Defuzzification

For this function, we obtained the QR code version and number of chunks from the fuzzy output. Then we convert v to QR code version and we return the number of chunk.

7.1.5 Pseudo Code

The pseudo code for the fuzzy process is as follows:

7.1.5 .1 Fuzzification Pseudo Code

QR Ccde size and version data structure:

```

class QRCodeInfo
{
  var version;
  var QRCodeErrorLevel;
  var Size;
}
  
```

Function FuzzyMain()

```

{
  var v, v+, datasize = Fuzzification()
  var version, numberOfchunk = FuzzyRole(v,v+,
    1,datasize)
  return version, numberOfchunk;
}
  
```



```

Function Fuzzification
{
    var numberOfLayer = 1;
    var data=getfile();
    if(date.length> 23648*4)
        Error:file size to large;
    var sizeOnPaper=getsize();
    var v= QRCodeInfoArray.find(sizeOnPaper);
    var v+= QRCodeInfoArray[v+1];
    return v,v+;
}

```

7.1.5.2 Membership Function Pseudo Code

```

Function Membership(v, v+, datasize)
{
    var numberOfLayers = 1;
    while(v <> null)
    {
        if(v.size >= datasize / numberOfLayers)
            break;
        if(numberOfLayers<4)
            numberOfLayers+=1;
        else{
            numberOfLayers=1;
            v=v+;
            v+= QRCodeInfoArray[(v+)+1];
        }
    }
    return numberOfLayers,v,v+;
}

```

7.1.5.3 Fuzzy Rule Pseudo Code

```

Function FuzzyRule(v,v+, numberoflayer,datasize)
{
    membership(v,v+, datasize);
    var decimal= datasize % numberoflayer;
    if (numberoflayer<=4)
    {
        NumberOfchunk = numberoflayer;
        if(decimal<=0.5)
        {
            version=v;
        }
        else
        {
            version=v+
        }
    }
    else{
        NumberOfchunk=1;
        v=v+;
        v+= QRCodeInfoArray[(v+)+1];
        FuzzyRole(v,v+, numberoflayer,datasize)
    }
}

```

```

}
return version, numberOfchunk;
}

```

7.2 Split the File into Chunks

In this process, the data that needs to be encoded is divided based on the fuzzy result. The number of chunk is checked based on the fuzzy result and check to make sure the file are able to be divided to the number of chunks otherwise add zeros byte to the end of the file to make sure we will get correct equal chunks.

7.2.1 Split the File into Chunks Pseudo Code

```

Function SplitFile(data)
{
    var version, numberOfchunk= FuzzyMain();
    var fullencodesize = version.Size *
        numberOfchunk;
    var extrazeros = fullencodesize - data.size;
    while (extrazeros <> 0)
    {
        data = data + 0;
    }
    Chunks[] = data / numberOfchunk;
    return Chunks;
}

```

7.3 Generate Black and White QR Code

In this process, each chunk is taken and encoded into black and white QR code using any open source QR code generator. In our implementation, we are using Zxing library to generate the black and white QR code.

7.3.1 Generate Black and White QR Code Pseudo Code

```

Function GenerateCodes(data)
{
    var chunks= SplitFile(data)
    var codes[];
    for(i=0;i<chunks.count;i++)
    {
        Codes[i]=Zxing.Encode(chunks[i]);
    }
    return Codes;
}

```

7.4 Assign Color

In our proposed generator, the color selection is dynamic based on the number of layers selected from the fuzzy technique. Therefore, we have four cases (a) one layer, (b) two layers, (c) three layers, and (d) four layers. We will describe the color selection for each case in the next section.

7.4.1 One Layer

In this case, adding color to the QR code is not required, since all the data can be encoded in one QR code. So, the result will be black and white QR code.

7.4.2 Two Layers

This case happens when assigning 1 to the primary color for each QR code. Red color will be assigned to the first QR code and green color to the second QR code. So, the result will be 4 color QR code.

7.4.3 Three Layers

For this case, primary color is assigned to 1 for each QR code. Red color is assigned to the first QR code, green color to the second QR code and blue color to the third QR code. The result will produce 8 color QR code.

7.4.4 Four Layers

The case is also the same as two and three layers which will assign 1 to primary color for each QR code. Red color is assigned to the first QR code, green color to the second QR code, blue color to the third QR code, and the fourth QR code is assigned black color. So, the result will be 16 color QR code.

7.4.5 Assign Color Pseudo Code

```
Function AssignColor(data)
{
    var codes= GenerateCodes(data);
    switch(codes.count)
    {
        Case 2:
            UpdateColor(codes[0], red)
            UpdateColor(codes[1], green)
            Break;
        Case 3:
            UpdateColor(codes[0], red)
```

```
            UpdateColor(codes[1], green)
            UpdateColor(codes[2], Blue)
            Break;
        Case 4:
            UpdateColor(codes[0], red)
            UpdateColor(codes[1], green)
            UpdateColor(codes[2], Blue)
            UpdateColor(codes[2], Black)
            Break;
    }
    return codes;
}
```

7.5 Color Multiplexing

In this stage, the monochrome QR codes are merged into one colored QR code. The number of color is equal to $2^{\text{number of layers}}$. The representation for RGB color format is (red value, green value and blue value), mixing this three color value will produce one color. The value for each color is in the range between 0 and 255. Therefore, in color multiplexing process, 255 will be added when there is color and 0 when there is no color. In the case of 4 layers, the value of the three color element is changed from 255 to 100. For example, we have (255,0,255). 255 for the forth layer will produce (100,0,100). The explanation of the result color for each case is as follows:





7.5.1 One Layer

The number of colors = 2^1 so we will have only 2 colors black and white.

7.5.2 Two Layers

The number of colors = 2^2 which will produce color QR code with 4 colors. Red, green, yellow and black colors are used in this QR code, as shown in Table 3.









Table 3. Color Used For 2 Layers QR Code

Color Representation (Red, Green, Blue)	Color
(255,0,0)	
(255,255,0)	
(0,255,0)	
(0,0,0)	

7.5.3 Three Layers

The number of colors = 2^3 which will produce color QR code with 8 colors. Table 4 shows red, fuchsia, yellow, white, black, blue, green and aqua color used in this QR code.








Table 4. Color Used For 3 Layers QR Code










Color Representation (Red, Green, Blue)	Color
(255,0,0)	
(255,0,255)	
(255,255,0)	
(255,255,255)	
(0,0,0)	
(0,0,255)	
(0,255,0)	
(0,255,255)	

7.5.4 Four Layers

The number of colors = 2^4 , so, we will have color QR code with 16 colors. In Table 5, there are red, fuchsia, yellow, white, black, blue, green, aqua, maroon, purple, olive, gray, brown, navy, dark green and teal colors used in this QR code.

Table 5. Color used for 4 layers QR code

Color Representation (Red, Green, Blue)	Color
(255,0,0)	
(255,0,255)	
(255,255,0)	
(255,255,255)	
(0,0,0)	
(0,0,255)	
(0,255,0)	

(0,255,255)	
(100,0,0)	
(100,0,100)	
(100,100,0)	
(100,100,100)	
(175,50,50)	
(0,0,100)	
(0,100,0)	
(0,100,100)	

7.5.5 Color Multiplexing Pseudo Code

```

Function ColorMultiplexing(data)
{
    var codes= AssugnColor(data);
    var colorCode;
    switch(codes.count)
    {
        case 1:
            return code[0];
            break;
        case 2:
            for (int i = 0; i < codes[0].length; i++)
            {
                for (int j = 0; j < codes[0]; j++)
                {
                    r = codes [0][i,j];
                    g = codes[1][i,j];
                    b = 0;
                    colorCode[i, j] = Color.RGB (r, g, b);
                }
            }
            break;
        case 3:
            for (int i = 0; i < codes[0].length; i++)
            {
                for (int j = 0; j < codes[0]; j++)
                {
                    r = codes [0][i,j];
                    g = codes[1][i,j];
                    b = codes[2][i,j];
                    colorCode[i, j] = Color.RGB (r, g, b);
                }
            }
            break;
        case 4:
            for (int i = 0; i < codes[0].length; i++)
            {
    
```

```

for (int j = 0; j < codes[0]; j++)
{
    r = codes [0][i,j];
    g = codes [1][i,j];
    b = codes [2][i,j];
    if(codes[3][i,j]!=Color.Black)
    {
        if(r==255)
            r=100
        if(g==255)
            g=100
        if(b==255)
            b=100
    }
    colorCode[i, j] = Color.RGB (r, g, b);
}
}
break;
}
return codes, colorCode;
}
    
```

7.6 Add Color Reference

In this stage, color reference is added at finder pattern, to help the decoder to identify the color used in the encoder. To add color reference, the locator pattern is divided into regions equal to maximum number of color used which consists of sixteen regions in our proposed encoder. Fig. 12 shows the regions division in the finder pattern.

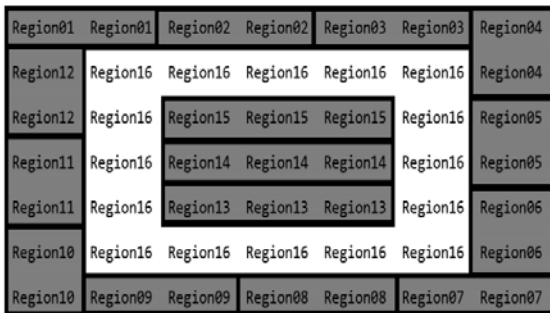


Figure 12: Locator Pattern Regions

We will explain how the color is assigned to each color QR code for different cases as follows:

7.6.1 One Layer

For this case, adding color reference is not required to the finder pattern since only black and white color are used in this QR code.

7.6.2 Two Layers

For this case, red, green, yellow and black is assigned to the finder pattern since we only have four colors. One color is assigned to each four regions. The color assignment for each region is as follows:

- Red color for regions 1, 2, 3 and 4.
- Green color for regions 5, 6, 7 and 8.
- Yellow color for regions 9, 10, 11 and 12.
- Black color for regions 13, 14 and 15.
- White color for region 16.

Fig. 13 shows an example of finder pattern for color QR code with 4 colors.

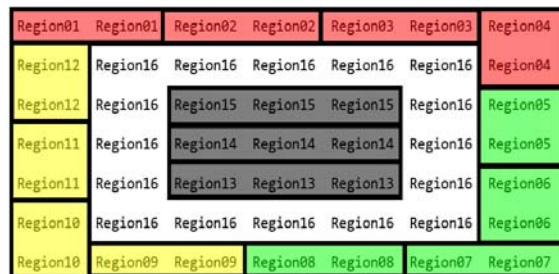


Figure 13: Finder Pattern for Color QR code with 4 Colors

7.6.3 Three layers

For this case, red, fuchsia, yellow, white, black, blue, green and aqua is assigned to the finder pattern since we only have eight colors. One color is assigned to each two regions. The color assignment for each region is as follows:

- Red color for regions 1 and 2.
- Fuchsia color for regions 3 and 4.
- Yellow color for regions 5 and 6.
- Blue color for regions 7 and 8.
- Green color for regions 9 and 10.
- Aqua color for regions 11 and 12.
- Black color for regions 13, 14 and 15.
- White color for region 16.

Fig. 14 shows an example for the finder pattern for color QR code with 8 colors.

Region01	Region01	Region02	Region02	Region03	Region03	Region04
Region12	Region16	Region16	Region16	Region16	Region16	Region04
Region12	Region16	Region15	Region15	Region15	Region16	Region05
Region11	Region16	Region14	Region14	Region14	Region16	Region05
Region11	Region16	Region13	Region13	Region13	Region16	Region06
Region10	Region16	Region16	Region16	Region16	Region16	Region06
Region10	Region09	Region09	Region08	Region08	Region07	Region07

Figure 14: Finder Pattern for color QR Code with 8 Colors

7.6.4 Four Layers

For this case, red, fuchsia, yellow, white, black, blue, green, aqua, maroon, purple, olive, gray, brown, navy, dark green and teal is assigned to the finder pattern since we only have 16 colors. One color is assigned to each region. The color assignment for each region is as follows:

- Red color for regions 1.
- Fuchsia color for regions 2.
- Yellow color for regions 3.
- Blue color for regions 4.
- Green color for regions 5.
- Aqua color for regions 6.
- Maroon color for regions 7.
- Purple color for regions 8.
- Olive color for regions 9.
- Gray color for regions 10.
- Brown color for regions 11.
- Navy color for regions 12.
- Dark Green color for regions 13.
- Teal color for regions 14.
- Black color for regions 15.
- White color for region 16.

Fig. 15 shows an example of finder pattern for color QR code with 16 colors.

Region01	Region01	Region02	Region02	Region03	Region03	Region04
Region12	Region16	Region16	Region16	Region16	Region16	Region04
Region12	Region16	Region15	Region15	Region15	Region16	Region05
Region11	Region16	Region14	Region14	Region14	Region16	Region05
Region11	Region16	Region13	Region13	Region13	Region16	Region06
Region10	Region16	Region16	Region16	Region16	Region16	Region06
Region10	Region09	Region09	Region08	Region08	Region07	Region07

Figure 15: Finder Pattern for Color QR code with 4 Colors

7.6.5 Add Color Reference Pseudo Code

Function ColorRefrence(data)

```

{
    var codes, finalcode= ColorMultiplexing (data);
    switch(codes.count)
    {
        case 1:
            Finalcode.finder.regions[1]= color.black;
            Finalcode.finder.regions[2]= color.black;
            Finalcode.finder.regions[3]= color.black;
            Finalcode.finder.regions[4]= color.black;
            Finalcode.finder.regions[5]= color.black;
            Finalcode.finder.regions[6]= color.black;
            Finalcode.finder.regions[7]= color.black;
            Finalcode.finder.regions[8]= color.black;
            Finalcode.finder.regions[9]= color.black;
            Finalcode.finder.regions[10]= color.black;
            Finalcode.finder.regions[11]= color.black;
            Finalcode.finder.regions[12]= color.black;
            Finalcode.finder.regions[13]= color.black;
            Finalcode.finder.regions[14]= color.black;
            Finalcode.finder.regions[15]= color.black;
            Finalcode.finder.regions[16]= color.white;
        case 2:
            Finalcode.finder.regions[1]= color.red;
            Finalcode.finder.regions[2]= color.red;
            Finalcode.finder.regions[3]= color.red;
            Finalcode.finder.regions[4]= color.red;
            Finalcode.finder.regions[5]= color.green;
            Finalcode.finder.regions[6]= color.green;
            Finalcode.finder.regions[7]= color.green;
            Finalcode.finder.regions[8]= color.green;
            Finalcode.finder.regions[9]= color.yellow;
            Finalcode.finder.regions[10]= color.yellow;
            Finalcode.finder.regions[11]= color.yellow;
            Finalcode.finder.regions[12]= color.yellow;
            Finalcode.finder.regions[13]= color.black;
            Finalcode.finder.regions[14]= color.black;
            Finalcode.finder.regions[15]= color.black;
            Finalcode.finder.regions[16]= color.white;
        break;
        case 3:
            Finalcode.finder.regions[1]= color.red;
            Finalcode.finder.regions[2]= color.red;
            Finalcode.finder.regions[3]= color.fuchsia;
            Finalcode.finder.regions[4]= color.fuchsia;
            Finalcode.finder.regions[5]= color.yellow;
            Finalcode.finder.regions[6]= color.yellow;
            Finalcode.finder.regions[7]= color.blue;
            Finalcode.finder.regions[8]= color.blue;
            Finalcode.finder.regions[9]= color.green;
            Finalcode.finder.regions[10]= color.green;
            Finalcode.finder.regions[11]= color.aqua;
            Finalcode.finder.regions[12]= color.aqua;
    }
}
    
```

```

Finalcode.finder.regions[13]= color.black;
Finalcode.finder.regions[14]= color.black;
Finalcode.finder.regions[15]= color.black;
Finalcode.finder.regions[16]= color.white;
break;
case 4:
Finalcode.finder.regions[1]= color.red;
Finalcode.finder.regions[2]= color.fuchsia;
Finalcode.finder.regions[3]= color.yellow;
Finalcode.finder.regions[4]= color.blue;
Finalcode.finder.regions[5]= color.green;
Finalcode.finder.regions[6]= color.aqua;
Finalcode.finder.regions[7]= color.maroon;
Finalcode.finder.regions[8]= color.purple;
Finalcode.finder.regions[9]= color.olive;
Finalcode.finder.regions[10]= color.gray;
Finalcode.finder.regions[11]= color.brown;
Finalcode.finder.regions[12]= color.navy;
Finalcode.finder.regions[13]=color.darkgreen;
Finalcode.finder.regions[14]= color.teal;
Finalcode.finder.regions[15]= color.black;
Finalcode.finder.regions[16]= color.white;
break;
}
return codes;
}
    
```

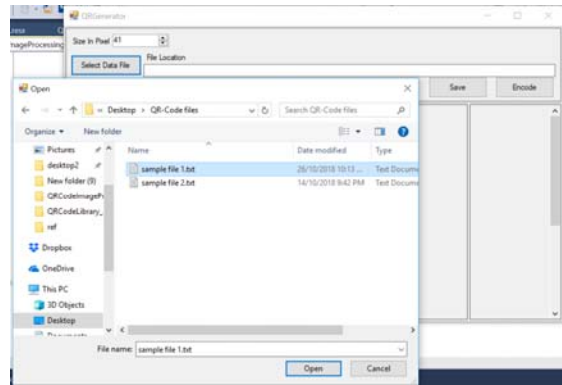


Figure 16: Our System Input





Figure 17: Our System Output

8. SYSTEM OVERVIEW

First, the user needs to insert the available size on paper and select the data file. Then click the encode button. Next, the system will generate the best fit color QR code based on the user inputs. The result will be QR code with 2 colors, 4 colors, 8 colors or 16 colors. The system will show the color QR code and black and white QR code layers, for the user reference. The user can select the color QR code or any layer and click save to save the QR code as an image. Fig. 16 shows the system input, Fig. 17 shows the system output and Table 5 shows an example for all types of color QR code that our generator can generate.

Table 5. System Output

Number of layers	output
1	
2	



($2^4=16$). Previous researches can encode only 3 Black and White QR code in one color QR code that will produce 8 colors ($2^3=8$).

The color selection from previous work is fixed to 8 colors, meaning that it is static and no selection can be done. In our proposed work, it is dynamic because we can choose 2, 4, 8 or 16 colors based on the data size. Therefore, we can choose the number of colors that can fit the data size efficiently and saves spaces.

Our proposed system provides reference color in the finder pattern, which help the decoder to determine the number of colors used in the encoder. This feature also exists in research works by Zhibo Yang [26].

9. RESULTS AND DISCUSSION

We have compared our proposed generator with existing QR code generator. Our proposed system can encode 4 layers of QR code compared to existing system that can encode at most 3 layers [3,6,8,10]. Based on, the data size measurement in Table 6, our system achieved 25% higher capacity than existing color QR code generator. We got this percentage by applying the increase percentage (in bytes) calculation as follows:

$$\begin{aligned} \text{Increase} &= \text{Our Generator} - \text{Existing Generator} \\ &= 11824 - 8868 \\ &= 2956 \end{aligned}$$

$$\begin{aligned} \% \text{ Increase} &= \text{Increase} \div \text{Our Generator} \times 100 \\ &= 2956 \div 11824 * 100 \\ &= 25\% \end{aligned}$$

11824 bytes is obtained from our proposed system and we choose 8868 bytes because it is the highest data size achieved from previous work.

Besides data size, Table 6 illustrate other features from research works which are number of colors, color selection and color reference.

The highest number of colors produced from previous works is 8 colors only compared to our proposed work which is 16 colors. This is because in our proposed work, we encode one more layer, so our encoder encodes 4 Black and White QR code in one color QR code that will produce 16 colors

Table 6. Feature Comparison

Research Works	Max Data Size (bytes)	Number of Colors	Color Selection	Color Reference
(Zhibo Yang , 2018)[26]	8,868	8	Static	Yes
(Blasinski, Henryk,2013)[19]	8,868	8	Static	No
(Thilo Fath, Falk Schubert, and Harald Haas, 2014)[6]	1,158	8	Static	No
(Sin Rong Toh, Weihan Goh, and Chai Kiat Yeo, 2016)[8]	8,868	8	Static	No
(Tian Hao, Ruogu Zhou, Guoliang Xing 2012)[9]	2,350	5	Static	No
(Rayana Boubezari, Hoa Le Minh, Zabih Ghassemlooy,2016)[5]	500	2	Static	No
Proposed system	11,824	16	Dynamic	Yes

Table 7 shows the comparison of data size (in bytes) for QR code version 1, 11, 21, 31, and 40 with low error correction. The second column refers to other research work by (Zhibo Yang, 2018)[26], (Blasinski, Henryk,2013)[19] and (Sin Rong Toh, Weihai Goh, and Chai Kiat Yeo, 2016)[8] where they used the same encoding data size for each of the QR code version. For each version presented in Table 7, there is a 25% increase in our proposed generator compared to existing color QR code generator.

Table 7. Data Size Comparison

QR Code Version	B/W QR Code (bytes)	Other Research Work [26], [19], [8] (Color QR Code in bytes)	Our Proposed Generator (Color QR Code in bytes)
1	19	57	76
11	325	975	1,300
21	932	2,796	3,728
31	1,843	5,529	7,372
40	2,956	8,868	11,824

Fig. 18 shows the data size (in bytes) comparison between our proposed color QR code generator and existing generator which shows our proposed system produced 25% larger data capacity for all QR code version, compared to existing color QR code generator.

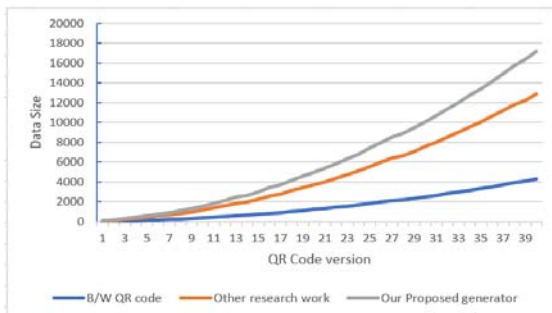


Figure 18: Data Size Comparison

Fig. 18 shows the overview result of our work compared to the maximum data size from existing work.

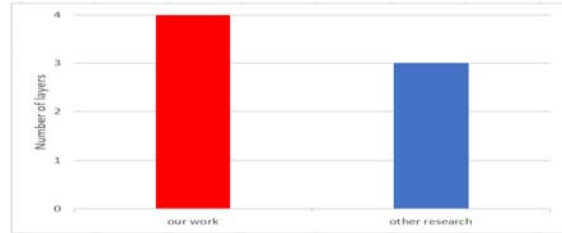


Figure 19: Number of Layers Comparison

10. Assumptions

Several assumptions made for our proposed system is as follows:

- i. The system works under windows operating system.
- ii. The system used RGB as color model.
- iii. The data modeling for each color is 4 pixels in width and 4 pixels in height.

11. CONCLUSION AND FUTURE WORK

Increasing the data size of QR code became a major research topic due to the popularity of QR code. Although the size of color QR code has been increased compared to black and white QR code, but there is still size limitation issue due to RGB and CMYK color format. In this paper, we explained our proposed color QR code generator which can encode 4 layers of QR code with color references. We provide suggestion for color modeling to overcome the limitation of 3 layers QR code. The result shows that there is an increase of 25% in the size of current color QR. The pseudo code for our system implementation is also explained.

For future works, we will propose color QR code decoder that will use the color reference in our proposed QR code generator. We also aim to generate QR code with 32 colors.

ACKNOWLEDGEMENT

We would like to thank Universiti Putra Malaysia for support given under the Putra Grant, code project GP/2018/9621800.

REFERENCES

- [1] Bakri Badawi, TEH NORANIS MOHD ARIS, Norwati Mustapha, and Noridayu Manshor. "EVALUATION OF A FUZZY 3D COLOR QR CODE DECODER." *Journal of Theoretical and Applied Information Technology* 96.19 (2018).
- [2] Bakri Badawi, TEH NORANIS MOHD ARIS, Norwati Mustapha, and Noridayu Manshor. (2017). COLOR QR CODE RECOGNITION UTILIZING NEURAL NETWORK AND FUZZY LOGIC TECHNIQUES. *Journal of Theoretical & Applied Information Technology*, 95(15).
- [3] Yang, Zhibo, et al. "Robust and fast decoding of high-capacity color QR codes for mobile applications." *IEEE Transactions on Image Processing* 27.12 (2018): 6093-6108.
- [4] Nivedan Bhardwaj¹, Ritesh Kumar², Rupali Verma¹, Alka Jindal¹ and Amol P. Bhondekar², "Decoding Algorithm for color QR code: A Mobile Scanner Application", *International Conference on Recent Trends in Information Technology (ICRTIT)* (2016) 1-6.
- [5] Lee, S. J., Tewolde, G., Lim, J., & Kwon, J. (2015, July). QR code based localization for indoor mobile robot with validation using a 3D optical tracking instrument. In *Advanced Intelligent Mechatronics (AIM)*, 2015 IEEE International Conference on (pp. 965-970). IEEE.
- [6] Blasinski, H., Bulan, O., & Sharma, G. (2013). Per-colorant-channel color barcodes for mobile applications: An interference cancellation framework. *IEEE Transactions on Image Processing*, 22(4), 1498-1511.
- [7] Yang, Zhibo, et al. "Towards robust color recovery for high-capacity color QR codes." *Image Processing (ICIP)*, 2016 IEEE International Conference on. IEEE, 2016.
- [8] Thilo Fath, Falk Schubert, and Harald Haas, "Wireless data transmission using visual codes." *Photonics Research Volume 2 Issue 5* Page 150 (2014).
- [9] Tian Hao, Ruogu Zhou, Guoliang Xing "COBRA: Color Barcode Streaming for Smartphone Systems" *MobiSys '12* (2012)
- [10] Sin Rong Toh, Weihang Goh, and Chai Kiat Yeo "Data Exchange via Multiplexed Color QR Codes on Mobile Devices" *MobiCom* Wireless Telecommunications Symposium (WTS), (2016).
- [11] R. Boubezari, H. Le Minh, Z. Ghassemlooy and A. Bouridane "Novel Detection Technique for Smartphone to Smartphone Visible Light Communications." 10th International Symposium on Communication Systems (2016).
- [12] Bingsheng Zhang, Kui Ren, Senior Member, IEEE, Guoliang Xing, Senior Member, IEEE, Xinwen Fu, Senior Member, IEEE, and Cong Wang, Member, IEEE "SBVLC: Secure Barcode-Based Visible Light Communication for Smartphones" *IEEE TRANSACTIONS ON MOBILE COMPUTING*, VOL. 15, NO. 2, FEBRUARY (2016).
- [13] Sa Rayana Boubezari, Hoa Le Minh, Zabih Ghassemlooy, and Ahmed Bouridane, "Smartphone Camera Based Visible Light Communication" *JOURNAL OF LIGHTWAVE TECHNOLOGY*, VOL. 34, NO. 17, SEPTEMBER 1, (2016) 4120-4126.
- [14] Kikuchi, M., Fujiyoshi, M., & Kiya, H. (2013, November). A new color QR code forward compatible with the standard QR code decoder. In *Intelligent Signal Processing and Communications Systems (ISPACS)*, 2013 International Symposium on (pp. 26-31). IEEE.
- [15] Singh, A., Verma, V., & Raj, G. (2017, January). A novel approach for encoding and decoding of high storage capacity color QR code. In *Cloud Computing, Data Science & Engineering-Confluence*, 2017 7th International Conference on (pp. 425-430). IEEE.
- [16] Cui Liu, Lianming Wang, "Fuzzy Color Recognition and Segmentation of Robot Vision Scene", (2015) 8th *International Congress on Image and Signal Processing*.
- [17] Priyanka Gaur¹, Shamik Tiwari² "2D QR Barcode Recognition Using Texture Features and Neural Network" *International Journal of Research in Advent Technology*, Vol.2, No.5, May 2014.
- [18] Samuel David Perli, Nabeel Ahmed, and Dina Katabi "PixNet: Interference-Free Wireless Links Using LCD-Camera pairs" *MobiCom* 10 (2010) 137-148.
- [19] P.S. André and R.A.S. Ferreira "Colour multiplexing of quick-response (QR) codes" *ELECTRONICS LETTERS* 20th November (2014) Vol. 50 No. 24

- [20] Max E.Vizcarra Melgar, and Luz M. Santander "Channel Capacity Analysis of 2D Barcodes: QR Code and CQR Code-5". *IEEE COLCOM (2016)*
- [21] Hanmandlu, Madasu, et al. "A novel optimal fuzzy system for color image enhancement using bacterial foraging." *IEEE Transactions on Instrumentation and Measurement* 58.8 (2009): 2867-2879.
- [22] Keng T. Tan, Douglas Chai, Hiroko Kato, and Siong Khai Ong " Designing a color Barcode for Mobile applications" *ECU PUBLICATIONS (2012)*
- [23] Ashwin Ashok, Shubham Jain, Marco Gruteser, Narayan Mandayam, Wenjia Yuan, Kristin Dana, "Capacity of screen-camera communication sunder perspective distortions" *Pervasiveand Mobile Computing* 16 (2015)
- [24] Meruga, J. M., Fountain, C., Kellar, J., Crawford, G., Baride, A., May, P. S., ... & Hoover, R. (2015). Multi-layered covert QR codes for increased capacity and security. *International Journal of Computers and Applications*, 37(1), 17-27.
- [25] Neshat, Mehdi, et al. "A new skin color detection approach based on fuzzy expert system." *Indian Journal of Science and Technology* 8.21 (2015).
- [26] Yang, Ching-Nung, and Tse-Shih Chen. "Colored visual cryptography scheme based on additive color mixing." *Pattern Recognition* 41.10 (2008): 3114-3129.
- [27] Hou, Young-Chang. "Visual cryptography for color images." *Pattern recognition* 36.7 (2003): 1619-1629.

Table 2. Comparison with Existing Works

Research Works	Generator	Max data size byte	Number of colors	Algorithm used	Color Selection	Number of layers	Implementation guideline
(Zhibo Yang , 2018)[26]	8 Color	8868	8	Color reference	Static	3	No
(Blasinski, Henryk,2013)[19]	8 Color	8868	8	Diffuse reflection	Static	3	No
(Thilo Fath, Falk Schubert, and Harald Haas, 2014)[6]	B/W or 8 Color	1158	8	Color multiplexing	Static	3	NO
(Sin Rong Toh, Weihan Goh, and Chai Kiat Yeo, 2016)[8]	8 Color	8868	8	Color multiplexing	Static	3	NO
(Tian Hao, Ruogu Zhou, Guoliang Xing 2012)[9]	5 Color	2350	5	Bit encoding	Static	2	NO
(Rayana Boubezari, Hoa Le Minh, Zabih Ghassemlooy,2016)[5]	B/W	500	2	Colored Bit	Static	1	NO
Proposed system	B/W, 4 color, 8 color, 16 color	11824	16	Fuzzy	Dynamic	1,2,3,4	Yes