Malaysian Journal of Computer Science, Vol. 9 No. 1, June 1996, pp. 32-36

REPRODUCING COMPUTERIZED PICTURE (RCP) USING ROTATED ORDERED-DITHER MATRIX

Zarinah Mohd Kasirun
Faculty of Computer Science and
Information Technology
University of Malaya
50603 Kuala Lumpur
Malaysia
Tel. No.: 603-7593167
email: zarin@fsktm.um.edu.my

ABSTRACT

Reproducing Computerized Picture (RCP) is a windows-based system that can be used to convert pictures with multi-level intensities into a continuous gray-scale intensity of pictures. RCP has shown that a rotated ordered-dither matrix would produce an output much better than the usual ordered-dither matrix for .BMP files. With the successful development of RCP, it will overcome the main problem of some devices, which can only display two levels of intensity, i.e., black and white.

Keywords: ***Reproducing computerised picture, windows-based system, rotated ordered-dither matrix, .BMP files, bilevel devices****.*

1.0 INTRODUCTION

Many raster-type displays and hardcopy devices can only display two intensities for each pixel. These devices are called bilevel devices. The problems with these devices are that they cannot produce pictures with continuous gray tones or shaded colours. However, it is possible to create the effect of shading with bilevel pixel values by varying the ratio of black to white pixels in each small area of the screen. This is because due to the limited resolving power of our eye, we will see this effect as a continuous intensity at normal viewing.

This effect of shading has been used in the printing industry for centuries. It is called halftoning technique. It uses dots of ink whose sizes or densities vary according to the required local intensity. The dot is placed in the middle of the pixel area and the ratio of the black dot area to the pixel area is inversely proportional to the required intensity. The diameter of the ink dot can vary continuously, thus producing continuous halftones levels.

Unfortunately, a computer display device cannot change the size of its electron beam, nor can a matrix printer change the size of its printed dot. Therefore, continuous halftone cannot be produced on these devices using the same technique. For a bilevel display device, the only possible way of producing shaded pictures is to vary the ratio of the number of discrete intensity levels it has at each pixel. The pixels are usually arranged on a rectangular mesh and the picture data can be expressed as an array of integers ranging from 0 to a maximum intensity value that can be represented by an array called PictureData. The definition of PictureData is Picture-Data [0..Xpicture,0..YPicture] : 0..MaxPictureIntensity.

There are many halftoning techniques [1,2,6,7,8] such as simulated halftoning, Floyd-Steinberg, error-diffusion and some others. Here, we will only make a thorough explanation for ordered-dithering technique and improve it's output quality.

2.0 ORDERED-DITHERING TECHNIQUE

The term dithering [1] refers to the technique for approximating halftones without reducing resolution. This is applied especially to sharp edges when intensities jumped from one discrete value to another. But the term is also applied to halftone approximation methods using pixel grids. Sometimes, however, it is used to refer to color halftone approximation only. This approximation is accomplished by adding small random values or intensities to pixel intensities to break up contours and referred to as dither noise. This will soften the boundary's intensity when it is added over an entire picture.

In ordered-dithering technique, the picture is divided into small areas that have same size compared to the areas of dithering matrix. The matrix is then superimposed with the picture. In the case of using 2x2 intensity matrix (Fig. 1), the superimposing process will generate an output as shown in Fig. 2. In this technique, dithering matrices of 2x2, 3x3, 4x4 and 5x5 has always been used. For any given scaled picture intensity, the pixel is turned ON if the integer number stored in the matrix is smaller than or equal to the scaled picture intensity. Otherwise, the pixel is turned OFF.

	Xpixel	
YPixel	80	130
	180	255

Fig. 1: Matrix 2X2 with intensity

This matrix differs from simulated halftoning [2] in which an average of intensity is used over entire rectangular area. While in this technique, each picture intensity is used at each pixel.

The technique just discussed can be represented by Code Segment 1 below. Function fmod() is used to determine the index numbers in the dithering matrix from the pixel numbers.

```
double X;           //frame buffer row index
double HX, HY;  //picture row and column indexes
TColor intensity;
if (X1 == X2) return;
HX = fmod (X1, Half_X);
HY = fmod (Y, Half_Y);
for (X = X1; X <= X2; ++X) {
       /* get the pixel intensity */
       if (intensity > pH->HalfToneCell[HX][HY])
           FrameBuffer[X][Y] = ON
       else FrameBuffer[X][Y] = OFF;
        if (HX == Half_X)
          HX = 0
        else HX = HX + 1;
}
```

Code Segment 1: Ordered-dither matrix algorithm

3.0 ROTATING ORDERED-DITHERING MATRIX

From the function of ordered-dither matrix technique shown in the Code Segment 1 above, it clearly shows that the comparison value between picture location and the integer number stored in the matrix is constant on the entire picture. Therefore, we make a modification to this value in order to produce a better result. The modification is done by rotating the ordered-dither matrix along every scan line.

The process of rotating the matrix can be done either in row- or column-wise. The row or column can also be determined to get various patterns. What we did is as shown in Fig. 3. Fig. 4 shows all the possible matrices that could be produced from the matrix rotation process in Fig. 3.

80	130
180	255

Fig. 3: Direction of rotation in 2X2 matrix

As in the dither-matrix technique, the rotated matrix is then mapped onto the picture as shown in Fig. 5. We can see in each scan line (consists of HY1 and HY2) the matrix is rotated and we can see different values in each comparison.

80	130	80	130	80	130	80	130	...
180	255	180	255	180	255	180	255	...
80	130	80	130	80	130	80	130	...
180	255	180	255	180	255	180	255	...
80	130	80	130	80	130	80	130	...
.	.	.	.	.	.	.	.	...
.	.	.	.	.	.	.	.	...
.	.	.	.	.	.	.	.	...

Fig 2: Superimposed of 2X2 matrix above onto picture

80	130		130	255		255	180		180	80
180	255		80	180		130	80		255	130
(a)			**(b)**			**(c)**			**(d)**	

Fig 4: Possible matrices after rotation

	HX1	HX2	HX1	HX2	HX1	HX2	HX1	HX2	. . .
HY1	80	130	80	130	80	130	80	130	...
HY2	130	255	130	255	130	255	130	255	...
HY1	255	180	255	180	255	180	255	180	...
HY2	130	80	130	80	130	80	130	80	...
HY1	80	130	80	130	80	130	80	130	...
.	.	.	.	.	.	.	.	.	...
.	.	.		.	.	.	.	.	...
.	.	.	.	.	.	.	.	.	...

Fig. 5: Mapping of rotated ordered-dither onto picture

This whole process of rotating the ordered-dither matrix could be represented by code segment 2 below.

```
double X;                    // frame buffer row index
double HX, HY;               //first row and column indexes
double HXX, HYY;//second row and column indexes
TColor intensity;
if (X1 == X2) return;
HX = fmod (X1, Half_X);
HY = fmod (Y,Half_Y);
for (X = X1; X <= X2; ++X) {
     if (HX == 0 && HY == 0) {
//4 cycles refers to the rotation

         if (cycle == 1) { HXX = HX; HYY = HY;} else
          if (cycle == 2) { HXX = 0; HYY = 1;} else
          if (cycle == 3) { HXX = 1; HYY = 1;} else
          if (cycle == 4) { HXX = 1; HYY = 0)
     }else if (HX == 1 && HY == 0) {
          if (cycle == 1) { HXX = HX; HYY = HY;} else
          if (cycle == 2) { HXX = 0; HYY = 0;} else
         if (cycle == 3) { HXX = 0; HYY = 1;} else
         if (cycle == 4) { HXX = 1; HYY = 1)
     }else if (HX == 1 && HY == 1) {
        if (cycle == 1) { HXX = HX; HYY = HY;} else
        if (cycle == 2) { HXX = 1; HYY = 0;} else
        if (cycle == 3) { HXX = 0; HYY = 0;} else
        if (cycle == 4) { HXX = 0; HYY = 1)
    }else if (HX == 0 && HY == 1) {
        if (cycle == 1) { HXX = HX; HYY = HY;} else
        if (cycle == 2) { HXX = 1; HYY = 1;} else
        if (cycle == 3) { HXX = 1; HYY = 0;} else
        if (cycle == 4) { HXX = 0; HYY = 0}
    }
    if (intensity > pH->HalfToneCell[HXX][HYY])
        FrameBuffer [X][Y] = ON
    else
        FrameBuffer [X][Y] = OFF;
    if (HX == Half_X) HX = 0
    else
         ++HX;}
```

Code Segment 2: Rotated ordered-dither matrix algorithm

The frame buffer is set to 1 or ON if for any given scaled picture intensity, the integer number stored in the matrix is smaller than or equal to the scaled picture intensity. Otherwise, the pixel is turned OFF.

4.0 IMPLEMENTATION

All of the halftoning techniques in RCP were developed successfully using Borland C++ version 4.0 and executed under windows-based programming platform. As we know, windows application have several advantages like multi-tasking, 32-bit code, graphics, toolbars, and others. All these can be easily included using Borland C++ and Object Windows Class Library. Both software really help to develop RCP in a reliable time frame. They employ an object-oriented programming [5] approach that is known to be easy to maintain. This approach increases the system productivity and reliability [5].

RCP incorporates windows graphical user interface [4] in its interface. New users become competent quickly with it since many applications are currently also adopting the same user interface. Besides, Borland Resource Workshop is also used to change the usual windows application into an attractive graphics presentation by including the icon, cursor, menu, dialog boxes and keyboard. These objects are associated in the Graphics Device Interface (GDI) object class with other objects like pen, brush, and colour palette.

RCP can be executed in any IBM compatible personal computer of 8Mb RAM with VGA monitor of 640X480 resolution. We have chosen .BMP files to be an input picture for RCP so far. This file contains an image that is scanned by the scanned in or edited from Paintbrush in the windows application. After the halftoning process of any selected techniques, the image produced is still in the similar format. It only consists of two colours that is black and white. Black means there is ink while white means no ink.

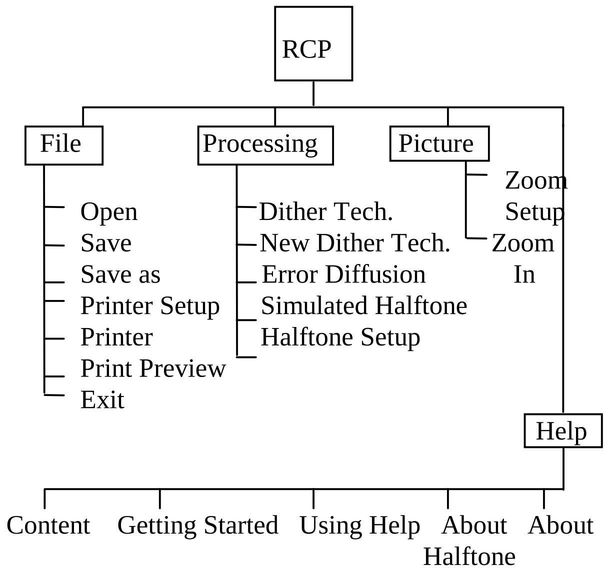

Fig. 6: RCP System Main Menu

Basically, RCP consists of four major options, namely File, Processing, Picture and Help as shown in Fig. 6. File option is concerned with few basic file-management function such as Open, Save, Save as, Print and also Exit. In the processing option, four techniques of halftoning are available i.e. dithering, new dithering (rotated ordered-dither), error diffusion, and simulated halftoning technique. Users can compare the outputs of a picture from those techniques by examining the printed output. User are also allowed to set the parameters involved before the halftoning process. At the same time, users are also allowed to zoom in and out on certain portion of the picture. At any time, on-line help is always available to guide users if they encounter any problem or if they need any explanation.

5.0 CONCLUSION

Considering the output produced by all the techniques of the choice in RCP, the error-diffusion technique yields a better result for .BMP files. However, if we compare between the ordered-dithering matrix technique with the rotated ordered-dithering matrix technique, the latter produced better output for the same picture of the same file format. On the other hand, there are other techniques that could produce better output for digital halftoning such as neural network and simulated annealing [6].

The result produced by RCP can be summarised in a line graph as shown in Fig. 7. The subjective measurement used was by looking to the sharpness of image detail and the smoothness of the grayscale simulation [6,7].

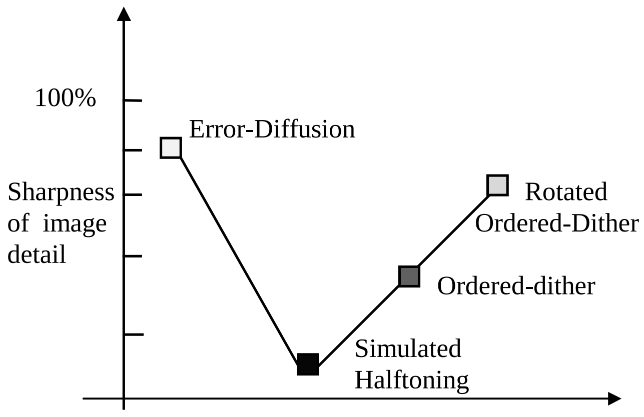

Fig. 7: Performance of output produced by halftoning techniques in RCP

Since RCP is a preliminary study on this matter, it shall also be used in other type of graphics file formats such as GIF, TIFF, PCX, IMG and Targa for future research.

ACKNOWLEDGMENT

The author is pleased to acknowledge the vital contributions of Mr. Khairi Abu Bakar in the development of RCP.

REFERENCES

[1] D. Hearn, M. P. Baker, *Computer Graphics*, 2nd Ed., Prentice-Hall, 1995.

[2] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, *Computer Graphics: Principles and Practice*, 2nd Ed., Reading, Mass., Addison-Wesley, 1990.

[3] J. R. Levine, *Programming for graphics files in C & C++*. John Wiley & Sons, Inc., 1994.

[4] D. Hix, H. R. Hartson, *Developing User Interfaces Ensuring Usability Through Product & Process*. John Wiley, 1993.

[5] S. R. Davis, *Hands on Turbo C++*. Addison-Wesley, 1991.

[6] R. Geist, R. Reynolds and D. Suggs, “A Markovian framework for digital halftoning”, *ACM Transaction on Graphics*, Vol. 12 No. 2, April 1993, pp. 136-159.

[7] C. Jer-Sen, "A comparative study of digital halftoning techniques", in *Proceeding of the IEEE 1992 National Aerospace and Electronics Conference*, Dayton, Ohio May 1992, Vol. 3, pp. 1139-1145.

[8] W. W. Ping, "Inverse halftoning and kernel estimation for error diffusion", *IEEE Transaction on Image Processing*, Vol. 4 No. 4, April 1995, pp. 486-498.

BIOGRAPHY

Zarinah Mohd Kasirun obtained her Master of Computer Science from Universiti Kebangsaan Malaysia in 1993. Currently, she is a lecturer at the Faculty of Computer Science and Information Technology, University of Malaya. Her research areas include computer graphics, computer-assisted learning and information technology.