# CHAPTER 3

# EXPERIMENT

## 3.1 Materials

| | | |
|---|---|---|
| 3.1.1 | Printing substrates | : HP Based Semi Gloss 6783A Ink Jet paper, 90 $g/m^2$ |
| | | : Crown Vantage High Weight Strength Map paper, 105 $g/m^2$ |
| 3.1.2 | Graphic arts film | : Kodak GEN 5 |
| 3.1.3 | Printing plate | : Positive Euro Plate EP 6 |
| 3.1.4 | Printing inks | : CMYK Litho Rex Maxi Nouveau |
| 3.1.5 | Developer | : Kodak RA 2000 |
| 3.1.6 | Fixer | : Kodak 3000 |

## 3.2 Apparatus

| | | |
|---|---|---|
| 3.2.1 | Offset press | : Heidelberg Speed Master 74 |
| 3.2.2. | Image setter | : Linotype-Hell Herkules PRO |
| 3.2.3 | Film processor | : Linotype-Hell Herkules Multiline 860 |
| 3.2.4 | Plate exposure | : Helioprint PD 290 |
| 3.2.5 | Plate processor | : InterPlater 88 |

| | | |
|---|---|---|
| 3.2.6 | Personal computers | : Pentium III 733 MHz 128 MB RAM |
| | | : AMD 475 MHz 32 MB RAM |
| 3.2.7 | Compiler software | : Visual C++ |
| 3.2.8 | Application software | : Adobe PhotoShop 5.5 |
| | | : Abobe Page Maker 6.5 |
| 3.2.9 | Digital color printers | : EPSON STYLUS PHOTO 700 |
| | | : HEWLET PACKARD 750 C+ |
| 3.2.10 | Densitospectrophotometer | : X-rite 530 |

3.2.11 Digital map images in RGB data contained in CD-ROM

## 3.3 Procedure

### 3.3.1 Preparation of Digital Map Images

Digital map images data used in this research are taken from the Automate Mapping office of the Royal Thai Survey Department (RTSD). This data is in 24 bits RGB Color System at the resolution of 600, 300, 250, and 200 dpi with Raw format.

### 3.3.2 RTSDZIP Program Creation

The compression - decompression program, RTSDZIP, is created. The compression function starts from the conversion of image's color values in RGB

Color System to index numbers in Indexed Color System, writing data with Optimized Run-Length Coding. With Huffman Coding, the probabilities of occurrence are calculated to assign the Huffman codes. The decompression function starts from reading data stream, comparing the data codes with Huffman and Indexed Tables, and converting back to RGB. The run of each index is read in order to assign the number of adjacent pixels that have its index.

### 3.3.3 Image Compression Processing

When a map image data is put into the RTSDZIP, it is passed through compression algorithms that are divided into 3 parts as follows:

#### 3.3.3.1 Indexed Table Creation

The first pixel or the first byte of R, G, and B color values of images are read. Next, these first RGB values are put into a table by assigning the first index 0, for the RGB values. The next pixel or byte is then read and compared with the previous RGB values in the table to find out whether they are the same values or not. If not, the next index will be assigned to those values. This process will be repeated until the last pixel or byte of RGB values have been read.

When all data has been converted from an RGB to Index Codes, we will have a set of image data that has replaced each pixel of the RGB values with an Index Code. Now, the first index 0 in data is the index of the first pixel of that image. The image file size is smaller than the original by 1/3 because it doesn't have to store each pixel's color values in 3 channels or 3 bytes (R 0-255, G 0-255, B 0-255). The

Indexed Color System uses only 1 byte for storing each pixel's color values (index 0-255).

3.3.3.2 Optimized Run-Length Coding

Run-Length Coding is a standard of the CCITT (Consultative Committee of the International Telephone and Telegraph)[20]. Theoretically, RLC stores the code in couple (index, run). It is called packet[21] that requires minimum 2 bytes of memory for each group of index codes. RLC is useable if an image has large regions that have the same colors as:

000000011111222 ⟶ (0,7)(1,5)(2,3)

The required memory is reduced from 15 to only 6 bytes. But the disadvantages of RLC is that it does not compensate when being used with a continuous tone image because its adjacent pixels have short runs of the same color as:

05486237 ⟶ (0,1)(5,1)(4,1)(8,1)(6,1)(2,1)(3,1)(7,1)

If an image has color information like this, the required storage memory will be increased from 8 to 16 bytes. Hence, an Optimized Run-Length Coding (ORLC) is developed to modify the basic RLC algorithm by using linkage between packet and non-packet storage among color information. ORLC selects packet storage when the adjacent pixels have the same color of at lease 3 pixels and non-packet when the adjacent pixels have the same color, less than 3 pixels.

One problem occurs, that is how to know which number in code is an index or run number. Hence, the numbers 0, 1, and 2 are assigned to the flag values[22]. The flag value is the number that designates the value or byte before it becomes the index number and the value or byte(s) after it is the run number. The flag values are assigned into 3 levels:

A. If the runs are 3 – 256 pixels, the flag value is 0 and is followed by the run that uses 1 byte of memory.

B. If the runs are 257 – 65 536 pixels, the flag value is 1 and is followed by the run that uses 2 bytes of memory.

C. If the runs are 65 537 – 4 294 967 296 pixels, the flag value is 2 and is followed by the run that uses 4 bytes of memory.

| | | |
|---|---|---|
| Index | 0 | runs3–256 |

| | | | |
|---|---|---|---|
| Index | 1 | runs 257–65 536 | |

| | | | | | |
|---|---|---|---|---|---|
| Index | 2 | runs 65 536–4 294 967 296 | | | |

- 1 box is 1 byte of memory

It should be noted that the flag values 0, 1, and 2 are similar to those index numbers that store the RGB values in the indexed table. Thus, the indexed table must be redesigned by adding 3 to each index.

0 0 0 0 1 1 0 0 2 2 2 3 3 3 3 3 3 3 3 3 3 3 1 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6

↓ +3 ↓ +3 ↓ +3

3 3 3 3 4 4 3 3 5 5 5 6 6 6 6 6 6 6 6 6 6 6 4 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 9 9

Thus, the ORLC storage becomes:

| | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 0 | 4 | 4 | 4 | 3 | 3 | 5 | 0 | 3 | 6 | 0 | 11 | 4 | 7 | 0 | 4 | 8 | 0 | 16 | 9 | 0 | 2 |

To store the above ORLC file, it must allocate the memory at the head of the file for storing the RGB color values of that image. This will link between the index numbers and RGB color values. 768 bytes are used and divided into 3 sections, the first 256 bytes for R, the second 256 bytes for G, and the last 256 bytes for B. As the index numbers 0 – 2 are used for flag values, the first 3 bytes of each section are designed to have no color information. Hence, the color information will start from the $4^{th}$ byte of each section.

### 3.3.3.3 Huffman Coding

Huffman Coding is the last process of compression techniques. It changes the code from fixed length codes to variable length codes. Huffman starts by looking at the data in each byte. With the histogram, the probabilities of occurrences are computed and rearranged in descending order. The shortest code will be assigned to the most frequency value, and the least frequency value will be assigned with the longest code.

The last compressed image file is saved into a new file format, RTSD.

### 3.3.4 Image Decompression Processing

The RTSD file that is the compressed image file is put into the RTSDZIP and then Huffman Table is recreated in order to translate Huffman code in each byte. The fist 768 bytes that stored R, G, and B values are taken to create the Indexed Table firstly. It starts at the bytes 1st, 257th, and 513th that stored the RGB value of the index 0. Their Huffman codes are converted back to RGB values. Because the numbers 0, 1 and 2 are flag values, so the bytes 1st-3rd, 257th-259th, and 523rd-515th will not have RGB color values. This computation is completed when all codes of the first 768 bytes are read and converted back to values.

Next, byte 769th which stored the index of the first pixel in the image is read. It is converted from the Huffman code to index number by comparison to the Huffman Table. The next byte is read, if it is not 0, 1 or 2, the next step is to convert that index number to RGB color value by comparison to Indexed Table. The R, G and B color values of that pixel are then stored in the 3 files separately, each file for each channel.

But if the next byte is flag value 0, 1 or 2, it will be divided into 3 ways:

A. If the flag is 0, the next 1 byte will be read for run value.

B. If the flag is 1, the next 2 byte will be read for run value.

C. If the flag is 2, the next 4 byte will be read for run value.

When the run is translated, the processor will back to read the index number of that run which is located in the previous byte of flag-value byte. Last, the RGB values of that pixel will be stored like in the same manner of the data that did not have the flag value.

It completes when the last pixel is reached. The R, G and B color values are stored in 3 files separately, each file for each channel. Last, all 3 files are connected together by the R value file firstly, G value, and B value. This data is then saved in Raw file format that is the same as the original image file format.

### 3.3.5 Image Viewing and Print Out

The original RGB image data and the output image data obtained from decompress processor are retrieved by application software such as PhotoShop and Page Maker in order to print out through a digital printer and to make color separation for an imagesetter. The latter is aimed at offset printing. Thus, plate-making process is necessary.

### 3.3.6 Results Analysis

The compressed data obtained from the RTSDZIP and Winzip will be compared with the original image data in terms of compression ratio and compression-decompression time. The map images obtained from digital printing and offset printing are then compared to consider print quality in terms of tone reproduction and color change.