การบีบอัดข้อมูลภาพแบบแฟรกตอลสำหรับภาพสี

Fractal Image Compression for Color Image

ผิน ฉัตรแก้วมณี* ● สมนึก คีรีโต**

บทคัดย่อ

การบีบอัดข้อมูลภาพแบบแฟรกตอลสำหรับภาพสี มีพื้นฐานการทำงานทั่วไปแบบ Jacquian's Algorithm คือ หาส่วนของโดเมนที่มีความสัมพันธ์เหมาะสมกับเรนจ์ภายในภาพ ใช้วิธีการแบ่งภาพแบบต้นไม้สี่กิ่ง โดยแต่ละส่วนไม่ซ้อนทับกัน และรวมเอาข้อมูลโดเมนในแต่ละช่องทางสี เป็นกลุ่มข้อมูลชนิดเดียวกัน ส่วนแบบจำลองสีที่ใช้แทนค่าสีในภาพ ก็เป็นปัจจัยหนึ่ง ที่ส่งผลต่อประสิทธิภาพการทำงานด้วยเช่นกัน ในเอกสารฉบับนี้มีการเปรียบเทียบผลการทำงานในแบบจำลองสีต่าง ๆ เพื่อหาข้อดีข้อเสียในแต่ละแบบ สุดท้ายจะแสดงผลการทำงานในข้อมูลนำเข้าที่แตกต่างกัน และวิเคราะห์ผลการทำงานด้วยกราฟความสัมพันธ์ของค่าที่เกี่ยวข้องต่าง ๆ

Abstract

Fractal image compression for color image uses the same concept as those of "Jacquian's Algorithm". It searches for portion of domain block which has relationship with range block within the color image. It then divides the color image using the quad-tree partition with non-overlap approach and collects data from the domain block. The model of the color image is a factor which has impact on the efficiency of this algorithm. In order to find any benefit or drawback of the fractal's approach, there are comparisons, graphical representation and conclusion based on various models of color imge in this paper.

* *อาจารย์ประจำภาควิชาวิศวกรรมคอมพิวเตอร์ คณะสารสนเทศศาสตร์ มหาวิทยาลัยศรีปทุม*

** *ผู้ช่วยศาสตราจารย์ ดร. , รองผู้อำนวยการสถาบันวิจัยและพัฒนาแห่งมหาวิทยาลัยเกษตรศาสตร์ มหาวิทยาลัยเกษตรศาสตร์*

1. บทนำ

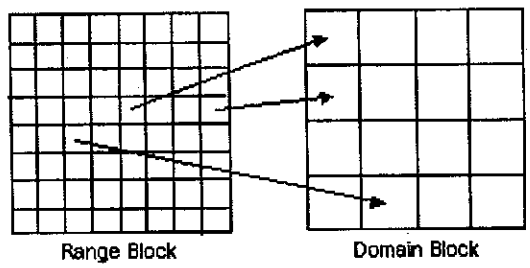
การบีบอัดข้อมูล (Data Compression) ที่นำมาใช้สำหรับรูปภาพในปัจจุบันแบ่งได้ 2 ชนิด คือ การบีบอัดข้อมูลที่ไม่มีข้อมูลสูญเสีย (Lossless Compression) โดยภาพที่ผ่านขั้นตอนวิธีแล้ว จะได้ภาพที่เหมือนภาพต้นฉบับทุกประการ อีกชนิดหนึ่งคือการบีบอัดข้อมูลที่มีข้อมูลสูญเสีย (Lossy Compression) ในวิธีการนี้ภาพที่ได้หลังจากการบีบอัดส่วนหนึ่งจะมีความผิดพลาด ซึ่งเป็นผลให้ภาพที่ได้ผิดเพี้ยนไปบ้างเล็กน้อยจากเดิม แต่จะมีตัวควบคุมความผิดพลาดให้มีค่าน้อยที่สุด สำหรับการบีบอัดข้อมูลภาพแบบแฟรกตอลจัดเป็นการทำงานในแบบที่มีข้อมูลสูญเสีย โดยในการทดลองครั้งนี้มีวัตถุประสงค์ เพื่อศึกษาการทำงานการบีบอัดข้อมูลภาพแบบแฟรกตอลสำหรับภาพเกรย์สเกล พร้อมทั้งนำมาพัฒนา เป็นการบีบอัดข้อมูลภาพแบบแฟรกตอลสำหรับภาพสีและพิจารณาในส่วนของการแทนค่าสีในแบบจำลองสี (Color Model) แบบต่าง ๆ ที่มีใช้ในปัจจุบัน จึงแบ่งได้เป็นส่วนสำคัญต่าง ๆ คือ มีส่วนของทฤษฎีพื้นฐานที่เกี่ยวข้องต่าง ๆ เช่น ข้อมูลพื้นฐานการทำงานแบบภาพเกรย์สเกล ข้อมูลเกี่ยวกับแบบจำลองสี เป็นต้น ส่วนของการทำงานของการบีบอัดข้อมูลสำหรับภาพสี จะกล่าวถึงเกี่ยวกับข้อมูลเพิ่มเติมที่ใช้ในการทำงาน การวัดค่าสำหรับภาพสี และส่วนของผลการทดลองและสรุปผลการทดลอง ซึ่งรวบรวมข้อมูลที่ได้จากการทำงานของโปรแกรมทดสอบที่สร้างขึ้น เพื่อเก็บค่าที่วัดได้จากการทำงาน ผลการวิเคราะห์การทำงานและแนวทางการพัฒนาประสิทธิภาพในการทำงาน

2. ทฤษฎีและหลักการ

การบีบอัดข้อมูลภาพแบบแฟรกตอลสำหรับภาพสี เป็นการรวมการทำงานพื้นฐานของ Jacquain's Algorithm และการใช้แบบจำลองสีแบบต่าง ๆ ที่ใช้ในปัจจุบัน ซึ่งจะต้องมีช่องทางสี 3 ช่องทางสีเหมือนกัน เพื่อวัดประสิทธิภาพในการทำงาน จึงมีส่วนต่าง ๆ ที่เป็นพื้นฐานในการทำงาน 3 ส่วน ได้แก่ การทำงานพื้นฐานของ Jacquain's Algorithm การแทนค่าแบบจำลองสี และการวัดประสิทธิภาพการทำงาน

2.1 Jacquain's Algorithm

การบีบอัดข้อมูลภาพแบบแฟรกตอลที่ใช้มีพื้นฐานการทำงานทั่วไปแบบ Jacquian's Algorithm [1-2] สำหรับข้อมูลภาพเกรย์สเกล มีการทำงานหลัก ๆ ด้วยกัน 3 ขั้นตอน ได้แก่ ขั้นตอนการแบ่งภาพออกเป็นพื้นที่สี่เหลี่ยมจัตุรัสที่มี 2 ชนิด คือ โดเมน และ เรนจ์ โดยโดเมนมีความกว้างแต่ละด้านเป็นสองเท่าของเรนจ์ พร้อมทั้งเก็บข้อมูลในแต่ละส่วนไว้ ขั้นตอนต่อไปเป็นการค้นหาว่าโดเมนใด ที่มีความสัมพันธ์เหมาะสมกับเรนจ์ [3] ซึ่งความสัมพันธ์นั้นเรียกว่า Affine Transformation ขั้นตอนสุดท้ายเป็นการเก็บข้อมูลในรูปแบบบิต โดยใช้เทคนิคการแบ่งภาพแบบต้นไม้สี่กิ่งที่สามารถปรับระดับความลึกได้ แสดงในรูปที่ 1 เป็นลักษณะของความสัมพันธ์ระหว่างโดเมน และเรนจ์ ส่วนการแปลงลักษณะของการจัดเรียงข้อมูลค่าสีในพื้นที่ และระดับค่าสีโดยรวมของภาพภายในพื้นที่ที่กำหนด สมการความสัมพันธ์แสดงในสมการที่ 1



รูปที่ 1 การทำงานพื้นฐานของ Jacquian

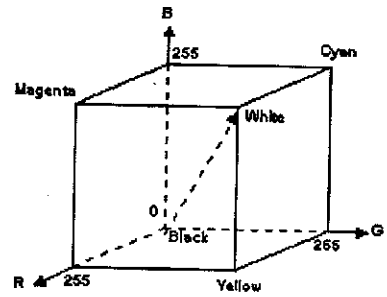
$$\begin{bmatrix} x_{new} \\ y_{new} \\ c_{new} \end{bmatrix} = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & s \end{bmatrix} \cdot \begin{bmatrix} x_{old} \\ y_{old} \\ c_{old} \end{bmatrix} + \begin{bmatrix} r_x \\ r_y \\ r_o \end{bmatrix} \qquad (1)$$

จากสมการจะประกอบด้วยส่วนที่ทำหน้าที่จัดการเกี่ยวกับการแปลงลักษณะของภาพ ซึ่งในสมการได้แก่ค่าของ a, b, c และ d โดยมี rx และ ry เป็นตัวบอกตำแหน่งของเรนจ์ อีกส่วนหนึ่งจะทำหน้าที่เป็นตัวจัดการเกี่ยวกับ ลักษณะของสีที่เปลี่ยนจากค่าสีในโดเมนเป็นค่าสีในเรนจ์ คือ ค่าของ s และ o ในบางครั้งเราจะเรียกว่าค่า Scale และ Offset ตามลำดับ แล้วจึงนำค่าต่าง ๆ ที่ใช้ เก็บลงในแฟ้มข้อมูลในรูปแบบบิต

2.2 แบบจำลองสี

แบบจำลองสีในปัจจุบัน มีหลายแบบด้วยกัน แต่ส่วนมากจะเป็นแบบจำลองสีที่มีช่องทางสี 3 ช่องทางสี [4] โดยแบบจำลองสีพื้นฐานที่ใช้กันในระบบคอมพิวเตอร์ คือ RGB (Red, Green, Blue) ส่วนแบบจำลองสีอื่น ๆ ที่ได้เกิดจากการเปลี่ยนแบบจำลองสีจาก RGB ไปเป็นจำลองนั้น ๆ เพื่อง่ายต่อการใช้งานเป็นเรื่อง ๆ ไป เช่น CMY (Cyan, Magenta, Yellow) นำไปใช้งานในเครื่องพิมพ์ เป็นต้น RGB เป็นแบบจำลองสีที่มีวิธีการรวมสีแบบเสริมกันของแม่สี (Additive Primaries) มีแม่สีหลักด้วยกัน 3 สี ได้แก่ สีแดง, สีเขียว และสีน้ำเงิน ในการแทนค่าสีจะมีลักษณะการอ้างถึงระดับของค่าแม่สีทั้งสาม ซึ่งจะอยู่ในช่วงของ 0 ถึง 255 มีด้วยกัน 256 ระดับ เช่น ถ้าค่าสีเป็นสีแดง ก็จะแทนด้วยตำแหน่งของ (R, G, B) เป็น (255, 0, 0) เป็นต้น ลักษณะของแบบจำลองสี RGB แสดงในรูปที่ 2

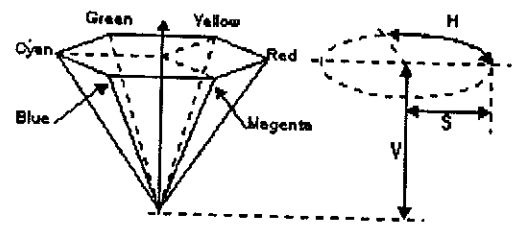
CMY มีลักษณะคล้ายกับ RGB แต่เป็นการรวมสีแบบหักล้างของค่าแม่สี (Subtractive Primaries) มีแม่สีหลัก 3 สี ได้แก่ สีฟ้า สีม่วง และสีเหลือง มีสมการการเปลี่ยนแบบจำลองสีระหว่าง RGB เป็น CMY ดังสมการที่ 2



รูปที่ 2 แบบจำลองสี RGB

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 255 \\ 255 \\ 255 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (2)$$

HSV (Hue, Satuation, Value) เป็นอีกแบบจำลองหนึ่งที่ใช้ในระบบคอมพิวเตอร์ปัจจุบันทางด้านรูปภาพ มีลักษณะการแทนค่าสีในแบบจำลองเหมือนกับกรวยหกเหลี่ยม (Hexcone) โดยในส่วนของแกนกลางเป็นค่าของ V ที่บ่งบอกถึงระดับความเข้มแสง ส่วนค่าของมุมที่ได้ คือค่า H เป็นตัวบ่งบอกถึงเฉดสี ส่วนสุดท้ายคือค่าที่หาจากจุดกึ่งกลางของกรวย คือค่า S บอกถึงความเข้มของเฉดสีที่เลือก ถ้าเราใช้ค่าที่มีเฉพาะค่า V (H = 0, S = 0) ภาพที่ได้จะเป็นภาพแบบเกรย์สเกล ซึ่งลักษณะแบบจำลองสี HSV แสดงให้เห็นในรูปที่ 3



รูปที่ 3 แบบจำลองสี HSV

YUV เป็นแบบจำลองสีที่มีใช้ในระบบการแพร่ภาพสัญญาณโทรทัศน์ การใช้แบบจำลอง YUV นี้ ต้องเปลี่ยนแบบจำลองมาจาก RGB เป็นหลัก ซึ่งการเปลี่ยนจะใช้คุณสมบัติของเมตริกซ์มาใช้ โดยสมการในการเปลี่ยนเป็นไปตามสมการที่ 3

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.275 & 0.504 & 0.098 \\ 0.439 & -0.368 & -0.0711 \\ -0.148 & -0.291 & 0.439 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \quad (3)$$

2.3 ประสิทธิภาพการทำงาน

ประสิทธิภาพการบีบอัดข้อมูลภาพวัดได้จากค่าต่าง ๆ ที่นิยมนำมาเป็นค่าอ้างอิงถึงเสมอ ได้แก่ อัตราการบีบอัดข้อมูล (Compression Ratio) จำนวนบิตต่อจุดสี (Bit Per Pixel) ในบางครั้งเรียกว่า Color Bit Rate และอัตราส่วนระหว่างข้อมูลกับสิ่งรบกวน เรียกย่อ ๆ ว่า PSNR (Peak-to-Peak Signal-to-Noise Ratio) ซึ่งแต่ละค่าที่กล่าวมามีความหมายและสมการในการคำนวณดังต่อไปนี้

อัตราการบีบอัดข้อมูล เป็นค่าพื้นฐานที่บ่งบอกถึงประสิทธิภาพในการบีบอัดข้อมูล โดยค่าที่นำมาใช้ในการคำนวณจะพิจารณาถึงจำนวนของข้อมูลก่อนและหลังการบีบอัดข้อมูล สำหรับข้อมูลก่อนการบีบอัดก็พิจารณาจากผลคูณของจำนวนของจุดสีที่มีภาพ ในภาพกับจำนวนบิตที่นำมาแทนค่าสีในแต่ละจุดสี ส่วนข้อมูลหลังจากการบีบอัดจะประกอบด้วยข้อมูล ส่วนหัว (Header File) ซึ่งเป็นตัวบอกคุณลักษณะของภาพและส่วนของข้อมูลที่เก็บข้อมูลแต่ละความสัมพันธ์ มีสมการในการคำนวณตามสมการที่ 4 เมื่อ CR เป็นอัตราการบีบอัดข้อมูล, C เป็นจำนวนบิตสี, W_{Img} และ H_{Img} เป็นความกว้างและความยาวของภาพตามลำดับ และ N_C เป็นขนาดของข้อมูลที่ได้หลังการบีบอัดข้อมูลและค่า CR ที่ได้จากการคำนวณอยู่ระหว่าง 0 ถึง 1 เท่านั้น

$$CR = \frac{C.W_{Img}.H_{Img}}{N_c} \quad (4)$$

Color Bit Rate เป็นค่าเฉลี่ยในการหาว่าจำนวนบิตที่ใช้ในการแทนค่า ในแต่ละจุดสีเป็นเท่าไร ภาพเกรย์สเกลใช้ 8 บิตสำหรับแทนค่าสีในแต่ละจุดสี ส่วนภาพสีจริงใช้ 24 บิต [5] สำหรับแทนค่าสีในแต่ละจุด การบีบอัดข้อมูลภาพสำหรับภาพสีต้องมีค่า Colot Bit Rate อยู่ระหว่าง 0 ถึง 24 ถ้ามีค่า เข้าใกล้ค่า 0 เท่าใด ก็หมายความว่า มีประสิทธิภาพการบีบอัดดีขึ้นส่วนหนึ่งเพราะการพิจารณาประสิทธิภาพ ในการบีบอัดข้อมูลภาพ จะต้องดูที่คุณภาพของภาพด้วยสมการในการคำนวณหา Color Bit Rate [6] เป็นไปตามสมการที่ 5

$$CBR = \frac{1}{CR} = \frac{N_C}{C.W_{Img}.H_{Img}} \quad (5)$$

PSNR เป็นค่าที่ใช้วัดระหว่างสัญญาณของข้อมูลที่ใช้เทียบกับสัญญาณที่มีความผิดพลาดของข้อมูล ถ้าค่าของ PSNR มีค่ามาก จะแสดงให้เห็นว่าสัญญาณที่เป็นส่วนของข้อมูลที่ถูกต้องมีมากเช่นกัน ในทางกลับกัน ถ้าค่า PSNR มีค่าน้อยหมายถึงสัญญาณที่เป็นส่วนของข้อมูลที่ผิดพลาดมีเพิ่มขึ้นเช่นกัน [7] ซึ่งสมการการคำนวณหาค่า PSNR เป็นไปตามสมการที่ 6 เมื่อ n คือ จำนวนจุดภายในภาพ ส่วน $P_{new,i}$ และ $P_{old,i}$ คือ ค่าสีในจุดที่ i ภายในภาพหลังและก่อนการบีบอัดข้อมูลตามลำดับ

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{n}\sum_{i=0}^{n-1}\left|p_{new,i} - p_{old,i}\right|} \quad (6)$$

3. การทำงานสำหรับการบีบอัดภาพสี

สำหรับวิธีการบีบอัดข้อมูลแบบแฟรกตอลสำหรับภาพสี จะมีบางส่วนในการทำงานเหมือนกับในภาพเกรย์สเกล รวมทั้งการวัดประสิทธิภาพในการทำงานด้วยบางส่วน ขั้นตอนพื้นฐานที่ใช้แบ่งการทำงานออกเป็น 2 ส่วนด้วยกัน ได้แก่ การเปลี่ยนแบบจำลองสีและการเข้ารหัสแบบแฟรกตอล ก็จะได้แฟ้มที่เป็นข้อมูลที่ได้หลังการบีบอัด ส่วนการถอดรหัสข้อมูลเป็นการสร้างภาพจากข้อมูลที่ได้จากการเข้ารหัส

3.1 การเปลี่ยนแบบจำลองสี

การเปลี่ยนแบบจำลองสีเป็นการเปลี่ยนวิธีในการแทนค่าสี ซึ่งแบบจำลองสีที่ใช้ในการทำงานจะมีด้วยกัน 4 แบบได้แก่ RGB, CMY, HSV และ YUV ซึ่งในแต่ละแบบ จะมีลักษณะในการแทนค่าสีที่แตกต่างกันไป รวมทั้งวิธีการเปลี่ยนแบบจำลองสีจาก RGB ที่มีลักษณะเฉพาะด้วยเช่น ภาพที่แทนค่าสีแบบ CMY คือภาพกลับสีของแบบ RGB สำหรับ YUV มีวิธีการเปลี่ยนแบบจำลองสีจากแบบ RGB โดยใช้เมตริกซ์ เป็นต้น ซึ่งได้กล่าวมาข้างต้น

3.2 การเก็บข้อมูลของโดเมน

การหาข้อมูลของโดเมนภายในรูปภาพทำได้โดย การแบ่งภาพออกเป็นสี่เหลี่ยมจัตุรัสตามระดับความลึกของการแบ่ง ซึ่งมีความลึกของการแบ่งเริ่มจาก 0 ซึ่งความกว้างแต่ละด้านในระดับความลึกนี้ มีค่าเท่ากับความกว้างของภาพในด้านยาวที่สุดที่สามารถถอดค่าได้โดย Logarithm ฐานสองลงตัวและในระดับความลึกที่เพิ่มขึ้น 1 ระดับ จะส่งผลให้ความกว้างของส่วนที่แบ่งเป็นครึ่งหนึ่งของระดับความลึกที่ต่ำกว่า ซึ่งภาพที่นำมาทดสอบการทำงานนี้จะมีขนาดเท่ากับ 25×256 จุด ดังนั้น เมื่อนำมาเขียนเป็นตารางการแบ่งภาพเพื่อหาโดเมน จะมีความสัมพันธ์ระหว่างความลึกและขนาดของโดเมน รวมถึงจำนวนของโดเมนในระดับความลึกต่าง แสดงในตารางที่ 1

ตารางที่ 1 ความสัมพันธ์ระหว่างความลึกกับความกว้าง และจำนวนโดเมน

Depth	Domain Width	Number of Domain
0	256	1
1	128	4
2	64	16
3	32	64
4	16	256
5	8	1024
6	4	4096

เมื่อนำมาเขียนเป็นสมการการคำนวณ โดยมีการหาความกว้างที่เหมาะสมของภาพ ซึ่งเป็นการปรับแต่งขนาดของภาพก่อนการบีบอัดข้อมูล แสดงให้เห็นในสมการที่ 7 เมื่อ S_{Img} คือความกว้างของภาพที่เหมาะสม ส่วนการคำนวณเพื่อหาความกว้างของโดเมนและจำนวนของโดเมน ในระดับความลึก D แสดงให้เห็นในสมการที่ 8 และ 9 เมื่อ $W_{Domain,D}$ เป็นความกว้างของโดเมน และ $N_{Domain,D}$ เป็นจำนวนโดเมนทั้งหมดในระดับความลึก D

$$S_{Img} = 2^{\log_2 \left| \min\left(W_{Img}, H_{Img}\right) \right|} \tag{7}$$

$$W_{Domain,D} = \frac{S_{Img}}{2^D} \tag{8}$$

$$N_{Domain,D} = \left(\frac{S_{Img}}{W_{Domain,D}} \right)^2 \tag{9}$$

เนื่องจากเรนจ์มีขนาดเล็กที่สุดเท่ากับ 2×2 จุด ทำให้โดเมนที่มีขนาดเล็กสุดเท่ากับ 4×4 จุดด้วย ดังนั้น สำหรับความลึกมากสุด $(D_{\max})$ ที่เหมาะสมกับภาพในขนาดต่าง ๆ สามารถคำนวณหาได้จากสมการที่ 10

$$D_{\max} = \log_2\left(\frac{S_{Img}}{4} \right) \tag{10}$$

3.3 การเข้ารหัสโดยใช้วิธีการต้นไม้สี่กิ่ง

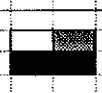
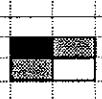
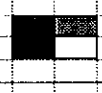
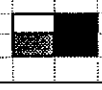
การเข้ารหัสโดยวิธีการต้นไม้สี่กิ่งเป็นการนำวิธีการแบ่งภาพแบบต้นไม้สี่กิ่งมาใช้ ส่วนของภาพที่ได้หลังจากการแบ่ง คือ เรนจ์ เช่นเดียวกันกับการแบ่งภาพเพื่อหาโดเมน คือ มีการระบุระดับชั้นของเรนจ์โดยเริ่มจากระดับความลึก 0 และเพิ่มขึ้นเรื่อย ๆ จนกระทั่งถึงระดับชั้นที่ทำให้ความกว้างของเรนจ์เท่ากับ 2×2 จุด เมื่อนำมาเขียนเป็นสมการความสัมพันธ์

ระหว่างความลึก (D) กับ ความกว้างของเรนจ์ (W_{Range}, D) ในระดับความลึก D และ ความกว้างของภาพที่เหมาะสม (S_{Img}) จะได้สมการที่ 11

$$W_{Range,D} = \frac{S_{Img}}{2^{(D+1)}} \qquad (11)$$

การแบ่งจะเป็นแบบไม่ตายตัว คือ ภายในภาพจะประกอบด้วยเรนจ์ที่มีขนาดไม่เท่ากัน ขึ้นอยู่กับค่าความผิดพลาด ที่นำมาเปรียบเทียบระหว่างโดเมนกับเรนจ์ ที่มีค่าระดับความลึกเดียวกัน ถ้าค่าความผิดพลาดมากกว่าค่าที่ยอมรับได้ ก็จะแบ่งส่วนของภาพในเรนจ์นั้นออกเป็น 4 ส่วน และมีค่าระดับความลึกเพิ่มขึ้น 1 ระดับ แล้วทำเช่นเดิมจนกระทั่งมีค่าความผิดพลาดน้อยกว่าค่าที่ยอมรับได้ หรือจนกระทั่งถึงระดับชั้นที่กำหนดไว้

ตารางที่ 2 การแปลงลักษณะที่ประกอบด้วยการหมุนและการกลับภาพ

Symmetry	Old Block	New Block
Rotate 0° (Identity)		
Rotate 90°		
Rotate 180°		
Rotate 270°		
Herizontal Reflection		
Vertical Reflection		
Reflection in Y=X		
Reflection in Y=-X		

สำหรับการเปรียบเทียบระหว่างโดเมน และเรนจ์ จะนำวิธีการแปลงลักษณะภาพมาใช้ เพื่อเพิ่มประสิทธิภาพในการหาโดเมนที่เหมาะสม การแปลงลักษณะประกอบด้วยการหมุนภาพและการกลับภาพ แสดงให้เห็นในตารางที่ 2 โดยข้อมูลที่ได้เรียกว่า ข้อมูลการแปลงภาพที่ประกอบด้วย ชนิดของการแปลงภาพ ตำแหน่งของโดเมน ตำแหน่งของเรนจ์ และข้อมูลเกี่ยวกับลักษณะการแปลงค่าสี ซึ่งข้อมูลเกี่ยวกับตำแหน่งของโดเมนและเรนจ์ จะมีส่วนของข้อมูลช่องทางสีรวมอยู่ด้วย

3.4 การขยายข้อมูลที่ถูกบีบอัด

การสร้างภาพจากข้อมูลการแปลงภาพเป็นขั้นตอนที่มีการทำงานคล้าย ๆ กับการเข้ารหัสแบบต้นไม้สี่กิ่ง แต่แทนที่จะเปรียบเทียบหาโดเมนและเรนจ์ที่เหมาะสม กลับเป็นการคำนวณเพื่อหาค่าภายในเรนจ์ที่ต้องการ โดยใช้ข้อมูลการแปลงภาพเป็นการทำงานที่เรียกว่า การทำงานซ้ำเดิม ดังนั้นในการสร้างภาพจำเป็นต้องมีพื้นที่หน่วยความจำเป็นสองเท่าของภาพเดิม เพื่อให้สามารถนำข้อมูลของภาพเดิม ในแต่ละรอบการทำงานมาสร้างเป็นข้อมูลภาพในรอบใหม่ได้ และจำนวนรอบการทำงานก็เป็นสิ่งสำคัญในการสร้างภาพด้วย คือถ้ามีจำนวนรอบในการทำงานมาก จะทำให้ได้ภาพที่มีลักษณะคล้ายกับภาพเดิมมากที่สุด สำหรับการคำนวณเพื่อหาจำนวนรอบที่เหมาะสม ได้จากขนาดของภาพนั้นเอง โดยคำนวณตามสมการที่ 12 เมื่อ $N_{Iteration}$ คือจำนวนรอบการทำซ้ำที่เหมาะสม

$$N_{Iteration} \geq \log_2 s_{Img} \qquad (12)$$

3.5 การวัดความผิดพลาดเกรย์สเกล

การวัดค่าสำหรับภาพสีนั้น ถ้าพิจารณาในแต่ละจุดสีโดยตรงจะเห็นว่าเป็นการยากในการวัดค่าความผิดพลาด เนื่องจากในแต่ละจุดสีจะมีพื้นที่

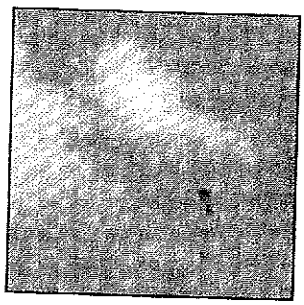
ในการเก็บข้อมูลสี 24 บิต ซึ่งเก็บข้อมูลช่องทางสี 3 ช่องทางได้แก่ R, G และ B ที่ใช้ในระบบคอมพิวเตอร์ทั่วไป ในกรณีที่มีความผิดพลาดในช่องทางใดช่องทางหนึ่งก็มิได้หมายความว่าข้อมูลสีในช่องทางอื่น ๆ จะมีความพลาดไปด้วย เพราะฉะนั้นในการวัดความความผิดพลาดก็ควรพิจารณาในแต่ละช่องทางสีแยกจากกัน เช่น ถ้าพิจารณาจุดสีหนึ่งที่มีข้อมูลเดิม เป็น 452F1216 หลังจากการแปลงแบบจำลองสี, การบีบอัดข้อมูล, การขยายข้อมูลและการแปลงแบบจำลองสีกลับเรียบร้อยแล้วได้ค่าข้อมูลเป็น 462F1216 และ 452F1316 ถ้าพิจารณาเป็นค่าสีเดียวแล้วจะมีความผิดพลาดเป็น 1000016 และ 116 หรือ 6553610 และ 110 ตามลำดับ เมื่อนำไปเทียบกับช่วงข้อมูลทั้งหมด 1677721610 ทำให้ได้ความผิดพลาดที่ได้เป็น 0.390625 % และ 0.000006 % จะเห็นได้ว่าในหลักความจริงแล้วค่าที่ได้ทั้งสองค่านั้นมีค่าความผิดพลาดที่เท่ากัน คือ 110 แต่ผิดพลาดในช่องทางสีที่ต่างกัน เพราะฉะนั้นค่าความผิดพลาดก็ควรมีค่าเท่ากันด้วย และเนื่องจากมีการแบ่งออกเป็นช่องทางสีจะเห็นว่าในแต่ละช่องทางสีจะมีค่าช่วงข้อมูลเท่ากันคือ 256 ระดับ เช่นเดียวกับการแทนค่าแบบเกรย์สเกล จึงเรียกค่าที่ได้นี้ว่า ค่าความผิดพลาดเกรย์สเกล แสดงให้เห็นในสมการที่ 13 และเมื่อคำนวณตามสมการดังกล่าว จะได้ค่าความผิดพลาดเกรย์สเกลเป็น 130208 % เท่ากัน โดยพิจารณาเพียงจุดเดียว

$$GE = \frac{\sum_{i=0}^{3}\sum_{j=0}^{n-1}\left|p_{new,i,j} - p_{old,i,j}\right|}{3.256.n} \qquad (13)$$

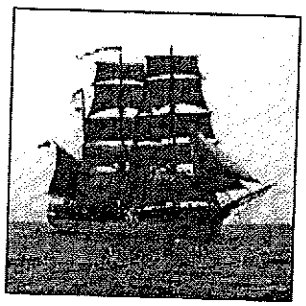
ผลการทดลอง

ภาพทดสอบที่ใช้ในการทดสอบโปรแกรม ดังแสดงให้เห็นในภาพที่ 4 -7

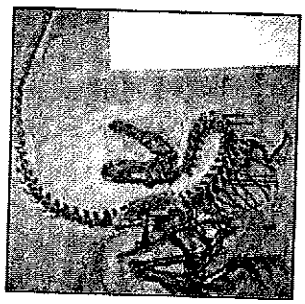
รูปที่ 4 ภาพทดสอบที่ 1 (ภาพงานศิลปะ)



รูปที่ 5 ภาพทดสอบที่ 2 (ภาพกลุ่มเมฆ)



รูปที่ 6 ภาพทดสอบที่ 3 (ภาพเรือสำเภา)



รูปที่ 7 ภาพทดสอบที่ 4 (ภาพ Fossil)

สำหรับการทดลอง จะมีการกำหนดค่าตายตัวที่ใช้ในการทำงาน ได้แก่ ค่าของความลึกมากสุดเป็น 6 และค่าความลึกน้อยสุดเป็น 0 เพื่อให้ภายในประกอบด้วยโดเมนที่มีขนาดต่าง ๆ โดยความลึกดังกล่าวเหมาะสำหรับภาพที่มีขนาด 256×256 จุดสีเท่านั้น ค่าจำนวนบิตที่ใช้แทนค่า s และ o ตามสมการที่ 1 กำหนดเป็น 5 และ 7 บิตตามลำดับ และค่าความผิดพลาดที่ยอมรับได้ในแต่ละช่องทางสี ซึ่งกำหนดให้มีค่าเท่ากันในทุกช่องทางสี ในการทดลองแต่ละครั้ง

ค่าที่บันทึกได้แก่ค่าของอัตราการบีบอัดข้อมูลและค่า PSNR โดยการประมาณค่าข้อมูลเพื่อหาค่า PSNR ณ จุดที่มีอัตราการบีบอัดข้อมูลเป็น 2, 4, 8 และ 16 เท่า ตามลำดับ ความผิดพลาดส่วนหนึ่งจะเกิดขึ้นในขั้นตอนการเปลี่ยนแบบจำลองสี แสดงให้เห็นในตารางที่ 3

ตารางที่ 3 ความผิดพลาดที่เกิดจากการเปลี่ยนแบบจำลองสี

Figure	Model	Convert Model Error	
		Level per Pixel	Percent
1	HSV	0.5043	0.20
	YUV	1.3197	0.52
2	HSV	0.2916	0.11
	YUV	1.5385	0.60
3	HSV	0.2846	0.11
	YUV	1.6783	0.66
4	HSV	0.3629	0.14
	YUV	2.0788	0.81

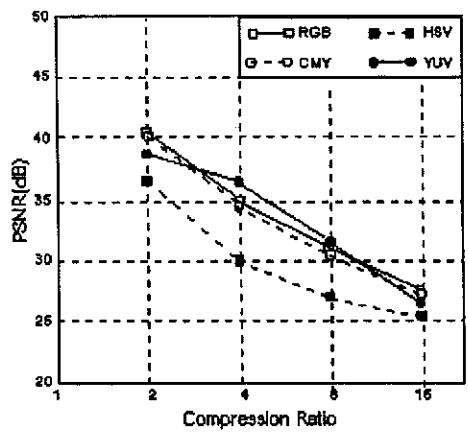
แบบจำลองสี RGB และ CMY มีค่าความผิดพลาดเป็น 0 จึงไม่ได้นำมาแสดงในตารางที่ 3 ข้างต้น สำหรับผลการทดลองที่ได้จากภาพทดสอบเป็นไปตามตารางที่ 4 โดยค่าที่บันทึกอยู่ในตารางเป็นค่าของ PSNR

ตารางที่ 4 ผลการทดลองภาพตัวอย่าง

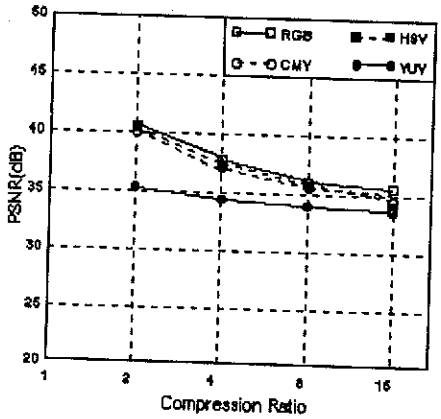
Image No.	Model	Compression Ratio			
		2	4	8	16
1	RGB	39.8572	34.4856	30.6519	27.3714
	CMY	39.8708	34.4902	30.6612	27.3793
	HSV	36.7949	29.2885	26.9837	24.7478
	YUV	38.6355	35.0394	30.9498	26.9830
2	RGB	41.2751	38.4331	37.6235	37.0420
	CMY	40.7409	38.2270	37.5931	37.0280
	HSV	41.2127	38.3842	36.9205	36.1357
	YUV	36.9856	36.1911	36.0369	35.6227
3	RGB	42.6909	36.5856	31.1672	26.0429
	CMY	42.7762	36.6474	31.1950	26.0272
	HSV	42.8326	37.6403	31.5351	25.6090
	YUV	39.1759	38.6907	34.0158	31.3088
4	RGB	39.5407	31.6402	26.2374	22.0434
	CMY	39.5541	31.6905	26.0374	21.9985
	HSV	41.0701	31.5483	23.7835	18.0297
	YUV	38.0775	35.2790	29.2515	24.9376

5. สรุป

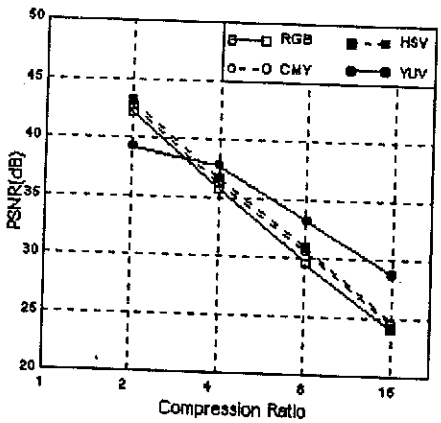
จากผลการทดลองข้างต้น เมื่อนำมาเขียนเป็นเส้นความผิดเพี้ยน (Distortion Curve) ของแต่ละภาพทดสอบ แสดงในรูปที่ 8 - 11



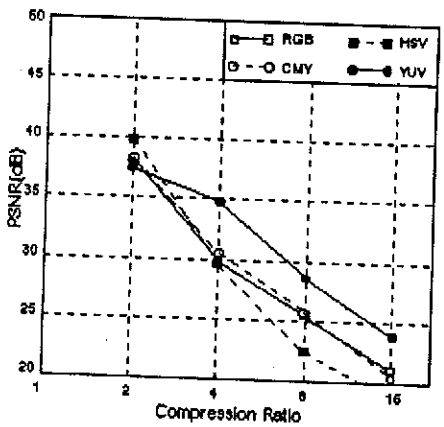
รูปที่ 8 เส้นความผิดเพี้ยนสำหรับภาพทดสอบที่ 1



รูปที่ 9 เส้นความผิดเพี้ยนสำหรับภาพทดสอบที่ 2



รูปที่ 10 เส้นความผิดเพี้ยนสำหรับภาพทดสอบที่ 3



รูปที่ 11 เส้นความผิดเพี้ยนสำหรับภาพทดสอบที่ 4

จากเส้นความผิดเพี้ยนจะเห็นได้ว่าภาพที่ใช้แบบจำลองสีแบบ RGB และ CMY จะมีประสิทธิภาพในการบีบอัดข้อมูลที่พอกัน หรือเรียกได้ว่าเป็นแบบจำลองพื้นฐานนั่นเอง ประกอบกับทั้งสองแบบจำลองเมื่อมีการเปลี่ยนแบบจำลองสีจากภาพบิตแมป แล้วไม่มีส่วนของความผิดพลาดเลย สำหรับ HSV และ YUV ควรจะเป็นแบบจำลองที่เป็นทางเลือกในการบีบอัดข้อมูลแบบแฟรกตอล โดยเมื่อสังเกตแบบจำลองทั้งสองจะพบว่าแบบจำลอง YUV จะมีประสิทธิภาพในการบีบอัดข้อมูลดีกว่าแบบจำลอง HSV สาเหตุหนึ่งเนื่องจากแบบจำลอง YUV มีการใช้เมตริกซ์ในการเปลี่ยนแบบจำลองสีจาก RGB ทำให้องค์ประกอบสีทั้งสามของ RGB ผสมผสานไปในทุกช่องทางสีของ YUV ดังนั้นเมื่อมีการเปลี่ยนแบบจำลองสีกลับเป็น RGB จึงมีความผิดพลาดของข้อมูลน้อยกว่าในแบบของ HSV ในทางกลับกันแบบจำลอง HSV เป็นเพียงการเปลี่ยนวิธีการแทนค่าสีเท่านั้น

แต่อย่างไรก็ตาม มิได้หมายความว่า YUV จะเป็นแบบจำลองที่มีประสิทธิภาพดีที่สุด เนื่องจากในการทำงานของโปรแกรมสามารถตั้งค่าความผิดพลาดที่ยอมรับได้ สำหรับแต่ละช่องทางสีได้อย่างอิสระ จึงสามารถปรับเปลี่ยนค่าดังกล่าวไปตามความเหมาะสมกับชนิดของภาพและค่าความผิดพลาดในการเปลี่ยนแบบจำลองสี

ดังนั้นแบบจำลองที่เหมาะสมกับการบีบอัดข้อมูล ภาพแบบแฟรกตอลสำหรับภาพสี ควรมีค่าความผิดพลาดในการเปลี่ยนแบบจำลองสีจาก RGB น้อยและในแบบจำลองสีดังกล่าว ควรมีการผสมผสานค่าสีจากทุกช่องทางของ RGB ที่เหมาะสมไปยังทุกช่องทางสีในแบบจำลองสีนั้น ๆ ด้วย จึงจะทำให้การบีบอัดข้อมูลภาพในวิธีการนี้ มีประสิทธิภาพการทำงานมากขึ้น ❐

เอกสารอ้างอิง

1. Michael, F. and Lyman, P. (1993). **Fractal Image Compression.** Massachusetts, United States of America : AK Peters Ltd.
2. Gharavi-Alkhansari, M. and Huang, T. (1997). **Fractal-Base Image and Video Coding.** University of Illinois at Urbana-Champaign, Urbana, Illinois.
3. Zhang, Y. and Po, L.M. (1996). **Fractal Color Image Compression Using Vector Distortion Measure.** University of Hong Kong. Kowloon Tong, Hong Kong.
4. Foley, J. and Dam, A. (1994). **Introduction to Computer Graphic.** United States of America : Addison-Wesley Publishing Company.
5. Fernandez, J. (1997). **GIFs, JPEGs & BMPs : Handing Internet Graphics.** New York, United States of America : MIS press.
6. Saupe, D., Hamzaoui, R. and Hartersten, H. (1996). **Fractal Image Compression An Introduction Overciew.** Freiburg University. Freiburg, Germany.
7. Fisher, Y. (1997). **A Comparison of Fractal Methods with DCT and Wavelets.** Institute for Nonlinear Science. California, United States of America.

บทคัดย่อ

บทค
ด้วยการเรียน
(Web-Based
และเวิลด์วาย
มีความหมาย
กันและกัน ที่
การสอนและก

บทคว
ออกแบบเว็บก
ผลการเรียนที่
การสอนต่อไป

Abstract

This
Teaching Eff
on Web-base
and World
Instructors a
learning. This
in the new

The s
Process of le
for developin

* รักษาการผู้อ