การลดขนาดข้อมูลภาพโดยใช้การแปลงแบบแฟรคตอล

โดย

1. นางสาวมรกต ระวีวรรณ
2. นายสัญชัย สุทธิชัชวาลวงศ์
3. นางสาวสุธีรา พึ่งพิศ

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2537

Image Compression Using Fractal Transformation

by

1. Miss Morrakot Raweewan
2. Mr. Sunchai Sutthichatchawanwong
3. Miss Sutira Phangpit

A Special Project Submitted in Partial Fulfillment of the Requirement for
the Degree of Bachelor of Science
Department of Applied Mathematics and Computer Science
Faculty of Science
King Mongkut's Institute of Technology Ladkrabang
1994

ปัญหาพิเศษเรื่อง การลดขนาดข้อมูลภาพโดยใช้การแปลงแบบแฟรคตอล
(Image Compression Using Fractal Transformation)

ชื่อนักศึกษา นางสาวมรกต ระวีวรรณ
นายสัญชัย สุทธิชัชวาลวงศ์
นางสาวสุธีรา พึ่งพิศ

ชื่ออาจารย์ที่ปรึกษา อาจารย์ วีระ บุญจริง
รองศาสตราจารย์ ภัคคินี ชิตสกุล

ปัญหาพิเศษฉบับนี้ กรรมการสอบปัญหาพิเศษ ได้ตรวจพิจารณาแล้ว เห็นชอบ แล้วจึงอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตร์บัณฑิต สาขาวิชาคณิตศาสตร์ประยุกต์ ประจำปีการศึกษา 2537

.....
(รองศาสตราจารย์ ภัคคินี ชิตสกุล)
รักษาการหัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

.....
(รองศาสตราจารย์ อุบลวรรณา เงินวิจิตร)
ประธานกรรมการการสอบปัญหาพิเศษ

.....
(อาจารย์ ศรัณย์ อินทโกสุม)
กรรมการสอบปัญหาพิเศษ

.....
(รองศาสตราจารย์ ภัคคินี ชิตสกุล)
กรรมการและอาจารย์ที่ปรึกษาปัญหาพิเศษ

.....
(อาจารย์ วีระ บุญจริง)
กรรมการและอาจารย์ที่ปรึกษาปัญหาพิเศษ

ปัญหาพิเศษเรื่อง การลดขนาดข้อมูลภาพโดยใช้การแปลงแบบแฟรคตอล
(Image Compression Using Fractal Transformation)

ชื่อนักศึกษา นางสาวมรกต ระวีวรรณ รหัสประจำตัว 34501013
นายสัญชัย สุทธิชัชวาลวงศ์ รหัสประจำตัว 34501022
นางสาวสุธีรา พึ่งพิศ รหัสประจำตัว 34501024

ชื่ออาจารย์ที่ปรึกษา อาจารย์ วีระ บุญจริง
รองศาสตราจารย์ ภัคคินี ชิตสกุล

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

ปีการศึกษา 2537

บทคัดย่อ

ปัญหาพิเศษ ฉบับนี้ มีวัตถุประสงค์ เพื่อศึกษาวิธีการลดขนาดข้อมูลภาพโดยใช้ทฤษฎีเรื่องการแปลงภาพแบบแฟรคตอล, ทฤษฎีฟังก์ชันย้ำเฉพาะที่ (local IFS) , ทฤษฎีภาพปะติดและทฤษฎีการจับคู่แบบย่นย่อ หลังจากการศึกษาทฤษฎีแล้ว จึงนำมาพัฒนาเป็นโปรแกรมลดขนาดข้อมูลภาพด้วยเทอร์โบซี เวอร์ชัน 2

นำโปรแกรมไปทดลองกับภาพแบบเกรย์สเกล 10 รูป เมื่อพิจารณาปัจจัยเรื่องเวลาในการประมวลผลและระดับความพอใจในคุณภาพของภาพแล้วพบว่า ภาพที่มีโครงสร้างซับซ้อนน้อย ควรจะกำหนดขนาดของโดเมนบล็อคให้มีขนาดใหญ่กว่า ภาพที่มีโครงสร้างซับซ้อนมาก จากนั้นคำนวณเปอร์เซ็นต์การลดขนาดในแต่ละภาพ

Special Project Title Image Compression Using Fractal Transformation

Name Miss Morrakot Raweewan Id.34501013
Mr. Sunchai Suttichatchawanwong Id.34501022
Miss Sutira Phangpit Id.34501024

Adviser Mr.Veera Boonjing
Associate Profressor Pakkinee Chitsakul

Department Mathematics and Computer Science

Academic Year 1994

Abstract

The purpose of this project is to implement compression theories based on Fractal Transformation, Local IFS, Collage Theorem and Contraction Mapping. After studing these theories, the implementation was made by using turbo C Version 2.

the ten different grayscale image was tested by the program. Depending on computing time and images' quality, it is shown that domain block of simple images should be larger than of complex images. Finally, percentage of compression ratio of each image is computed.

กิตติกรรมประกาศ

ปัญหาพิเศษนี้สำเร็จได้เพราะความช่วยเหลือของบุคคลต่าง ๆเหล่านี้

1. อาจารย์ วีระ บุญจริง

ท่านช่วยแนะนำแหล่งข้อมูลและที่มาของปัญหาพิเศษนี้,ช่วยให้แนวคิดในการแก้ปัญหา, ให้คำปรึกษาเกี่ยวกับปัญหาพิเศษ รวมทั้งรูปแบบในการจัดทำเอกสารประกอบการสอบปัญหาพิเศษ

2. รองศาสตราจารย์ ภัคคินี ชิตสกุล

ท่านช่วยแนะนำรูปแบบในการจัดทำเอกสารประกอบการสอบปัญหาพิเศษ

3. อาจารย์ ศรัณย์ อินทโกสุม

ท่านช่วยให้คำแนะนำเกี่ยวกับปัญหาในการเขียนโปรแกรม

4. เจ้าหน้าที่ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

ท่านช่วยเอื้อเฟื้ออุปกรณ์และอำนวยความสะดวกต่าง ๆระหว่างการทำปัญหาพิเศษ

5. นายเอกชัย รัตนดิลกชัย

เพื่อนนักศึกษาร่วมชั้นปี ช่วยหาความรู้เพิ่มเติมเกี่ยวกับไฟล์ รูปภาพนามสกุลจุดทีจีเอ และช่วยเสนอแนวคิดในการแก้ปัญหา

ขอขอบคุณในความกรุณาเป็นอย่างสูง

นางสาวมรกต ระวีวรรณ
นายสัญชัย สุทธิชัชวาลวงศ์
นางสาวสุธีรา พึ่งพิศ

ผู้ทำปัญหาพิเศษ

สารบัญรูป

หน้า

หน้า

สารบัญ

หน้า

หน้า

บทที่ 1
บทนำ

ความสำคัญและที่มาของปัญหา

ในปัจจุบันคอมพิวเตอร์กับงานทางด้านภาพมีบทบาทแพร่หลายมาก และการเก็บข้อมูลภาพต้องอาศัยหน่วยความจำ ซึ่งขนาดของข้อมูลขึ้นอยู่กับขนาดของภาพและความละเอียด ยิ่งขนาดและความละเอียดของภาพมากก็ต้องใช้หน่วยความจำในการเก็บข้อมูลมาก และเมื่อนำข้อมูลในหน่วยความจำมาขยายก็จะได้ภาพที่ไม่ชัดเจน ดังนั้นการแก้ปัญหาเหล่านี้จึงนำคณิตศาสตร์แฟรคตอลมาใช้ในการจัดเก็บข้อมูลและขยายภาพที่จัดเก็บมาแสดงในขนาดปกติ

วัตถุประสงค์ของปัญหา

เพื่อจัดทำโปรแกรมลดขนาดข้อมูลโดยอาศัยการแปลงแบบแฟรคตอล

ขอบเขตของปัญหา

1.พัฒนาโปรแกรมลดขนาดข้อมูลเพื่อหาตัวแทนข้อมูลที่จัดเก็บในหน่วยความจำ และนำตัวแทนข้อมูลนั้นผ่านโปรแกรมเพื่อขยายข้อมูลกลับ

2.นำโปรแกรมที่ได้ไปทดลองกับภาพแบบเกรย์สเกลที่มีระดับความซับซ้อนต่างกัน เพื่อหาองค์ประกอบที่เหมาะสมในการลดขนาดข้อมูล และคุณภาพของภาพที่ขยายกลับคืนมีความใกล้เคียงกับภาพเดิม เริ่มจากการนำภาพที่มีระดับความซับซ้อนจากน้อยไปมากมาผ่านสแกนเนอร์เพื่อเป็นอินพุตของโปรแกรมลดขนาดข้อมูล หลังจากได้ตัวแทนข้อมูลภาพแล้วนำตัวแทนที่ได้มาเป็นอินพุตของโปรแกรมขยายข้อมูลกลับ

จากนั้นพิจารณาว่า ภาพที่ผ่านโปรแกรมมีความใกล้เคียงกับภาพเดิมที่ผ่านสแกนเนอร์หรือไม่ ถ้าไม่ใกล้เคียงจะทดลองเปลี่ยนขนาดโดเมนในโปรแกรมลดขนาดข้อมูลแล้วทดลองซ้ำ แต่ถ้าใกล้เคียงแล้วจะบันทึกเป็นเปอร์เซนต์การลดขนาดข้อมูล

ขั้นตอนในการดำเนินงาน

1. ศึกษาคณิตศาสตร์แฟรคตอล การแปลงแบบแฟรคตอล และคอมพิวเตอร์กราฟฟิกส์
2. ออกแบบโปรแกรมสำหรับการลดข้อมูลภาพ
3. เขียนโปรแกรม
4. ทดสอบการทำงานของโปรแกรม และนำภาพเกรย์สเกลที่มีระดับความซับซ้อนต่างกันมาทดลอง
5. สรุป ประเมินผล และ ข้อเสนอแนะ

ประโยชน์ที่คาดว่าจะได้รับ

ผลการศึกษาคาดว่า จะได้โปรแกรมลดขนาดของข้อมูลภาพ และแปลงกลับได้อย่างมีประสิทธิภาพ

บทที่ 2
ทฤษฎีแฟรคตอลและการแปลงแบบแฟรคตอล

แฟรคตอล(Fractal) มีรากศัพท์มาจาก "fragere" ซึ่งเป็นคำกริยาในภาษาลาติน มีความหมายว่า "แบ่งแยก" (breaks) ดังนั้นเมื่อพิจารณาวัตถุใดๆในธรรมชาติแบบแฟรคตอล พบว่าวัตถุที่มีโครงสร้างซับซ้อนจะมีองค์ประกอบย่อยๆที่มีโครงสร้างเหมือนๆกัน(self-similarity)

เรขาคณิตที่ศึกษาคำนวณเป็นเรขาคณิตแบบยูคลิด (Euclidian Geormetry) จะพูดถึงรูปทรง ต่างๆ ของวัตถุ เช่น วงกลม, วงรี, เส้นตรง, ทรงกลม ซึ่งอาณาบริเวณ (space) ของวัตถุเหล่านี้ จะแบ่งออกเป็น 1,2 และ 3 มิติ ตามลำดับ ถ้าในอาณาบริเวณ 1 มิติจะมีแต่ความยาวเพียงอย่างเดียวไม่มีความกว้างและความสูง สิ่งที่อยู่ในบริเวณนี้เช่นเส้นตรง ส่วนโลกที่อาศัยอยู่ในปัจจุบันเป็น 3 มิติ เพราะมีทั้งความยาว ความกว้างและความสูง ต่อมา Dr. Benoit B Mandelbrot ศึกษาค้นคว้าเกี่ยวกับธรรมชาติพบว่า "เมฆไม่ได้เป็นก้อนกลม ภูเขาไม่เป็นรูปกรวย ชายฝั่งไม่ได้เป็นโค้งวงกลม เปลือกไม้ก็ไม่เรียบ ฟ้าก็ไม่ได้ผ่าเป็นเส้นตรง" จะพบว่าแนวคิดแบบเรขาคณิตยูคลิดเป็นแนวคิดแบบอุดมคติเกินไป จึงได้ศึกษาค้นคว้าคณิตศาสตร์ที่สามารถอธิบายวัตถุใดๆในธรรมชาติได้ดีกว่าแบบเดิมเรียกว่า เรขาคณิตแบบแฟรคตอล

2.1 <u>คุณสมบัติของเรขาคณิตแบบแฟรคตอล</u>

ประกอบด้วยคุณสมบัติ 2 ข้อดังต่อไปนี้

1. คุณสมบัติการมีโครงสร้างที่ซ้ำๆกัน ในตัวเองแบบอนันต์ (Self-Similarity)

ความหมายของการคล้ายคลึงในตัวเองคือ ไม่ว่าจะตัดส่วนใด ส่วนหนึ่งของวัตถุออกมาเมื่อนำส่วนนั้นมาขยายและหมุนไปมา พบว่าจะเหมือนกับส่วนอื่นในตัวของวัตถุนั้นเอง ตัวอย่างเช่น สายฟ้า ถ้ามองจากระยะไกลจะเห็นแค่เส้นที่ยึกยักไปมาแต่ถ้าพิจารณาภาพถ่ายของการเกิดฟ้าผ่าใกล้ๆจะเห็นว่ามีเส้นยึกยักแบบเล็กๆแตกออกมาจากเส้นยึกยักเส้นใหญ่ ถ้าใช้แว่นขยายส่องดูจะเห็นเส้นยึกยักย่อยแตกออกมาจากเส้นยึกยักเล็กอีกที

2. คุณสมบัติการมีมิติเป็นเศษส่วน (Fractal Dimension)

มิติของเรขาคณิตแบบแฟรคตอลเป็นเลขเศษส่วนไม่ใช่จำนวนเต็ม ตัวอย่างเช่นกระดาษเรียบมีมิติ 2 มิติ นำมาค่อยๆขยำกระดาษจนเป็นทรงกลม มิติของกระดาษจะเพิ่มขึ้นเรื่อยๆ เช่นเป็น 2.2, 2.5 จนเป็น 3 มิติ เมื่อกระดาษกลายเป็นก้อนกลมพอดี

เมื่อรวมเอาคุณสมบัติทั้งสองเข้าด้วยกันพบว่า วัตถุใดๆที่แบ่งเป็นเศษส่วนที่มีความละเอียดมาก จะมีมิติมากด้วย

2.2 ข้อแตกต่างระหว่างเรขาคณิตแบบยูคลิดกับเรขาคณิตแบบแฟรคตอล

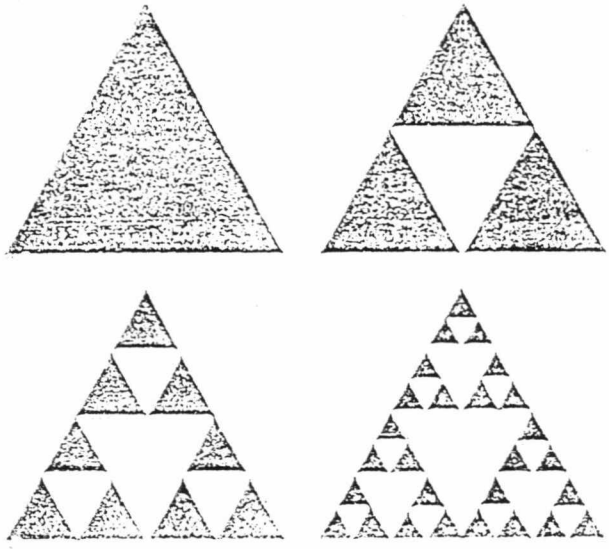
	เรขาคณิตแบบยูคลิด	เรขาคณิตแบบแฟรคตอล
1. การวัดภาพของวัตถุโดยใช้สเกล ในการวัดที่มีหน่วยต่างกัน	ไม่ว่าใช้สเกลใดจะได้ตัวเลขที่แน่นอนเสมอ	ถ้าใช้สเกลในการวัดมีความละเอียดมากขึ้นจะได้ความยาวมากขึ้นด้วย
2. มิติใน Space	เลขที่แสดงมิติเป็นจำนวนเต็มเท่านั้น	เลขที่แสดงมิติเป็นเลขเศษส่วน
3. รายละเอียดของรูป	เมื่อขยายดูรายละเอียดได้รูปร่างแน่นอน ไม่สามารถดูรายละเอียดได้	เมื่อขยายดูรายละเอียดจะได้รูปร่างของโครงสร้างที่มีรายละเอียดมากขึ้นเรื่อยๆ (Infinitely Detailed Shapes)

2.3 มิติแฟรคตอล (Fractional Dimension)

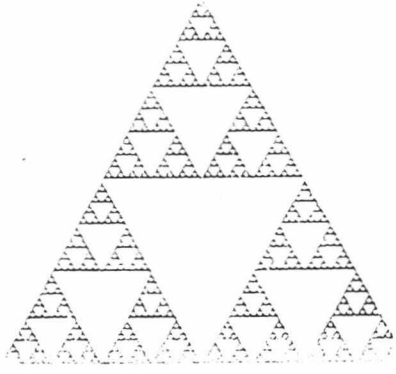
ในเรขาคณิตยูคลิด มิติของวัตถุหรือภาพจะเป็นเลขจำนวนเต็ม เช่น เส้นตรงมี 1 มิติ สี่เหลี่ยมมี 2 มิติ สี่เหลี่ยมลูกบาศก์มี 3 มิติ แต่หากพิจารณาภาพที่เกิดจากการสร้างโดยแฟรคตอล เช่น Sierpinski Triangle การคำนวณระนาบสำหรับสามเหลี่ยมนี้จะเป็นระนาบทิศทางที่มีหลายทิศทาง

พิจารณาเส้น 1 เส้น ที่แบ่งแยกออกเป็น n ส่วน $(n = n^1)$ แต่ละส่วนมีขนาดเท่ากับ $\frac{1}{n}$ แล้วเมื่อขยาย แต่ละส่วนย่อย n เท่า (magnification factor = n) องค์ประกอบเล็กๆ ในเส้นตรงก่อนนั้นยังคงมีลักษณะเช่นเดียวกับเส้นที่ยังไม่ถูกแบ่งแยก และหากแบ่งแยกพื้นที่สี่เหลี่ยมใดๆ โดยพื้นที่สี่เหลี่ยมย่อยมีขนาด $\frac{1}{n}$ เท่าของขนาดเดิม จำนวนพื้นที่สี่เหลี่ยมย่อยเท่ากับ $n = n^2$ ชิ้น (แต่ละด้านของพื้นที่สี่เหลี่ยมแบ่ง n ครั้ง) ในทำนองเดียวกันหากแบ่งสี่เหลี่ยมลูกบาศก์ออกเป็น n^3 ชิ้น แต่ละส่วนมีขนาด $\frac{1}{n}$ เท่าของขนาดเดิมดังแสดงในรูปที่ 2.1ก และ 2.1ข กรรมวิธีในการแบ่งวัตถุเพื่อกำหนดมิติ จะต้องอาศัยค่าลอการิทึมของจำนวนชิ้นส่วนย่อยที่เกิดจากการแบ่ง ดังนี้

1. กรณีที่วัตถุเป็นเส้น log(จำนวนชิ้นส่วนย่อย)= $\log(n^1) = 1 \log n$
2. กรณีที่วัตถุเป็นสี่เหลี่ยม log(จำนวนชิ้นส่วนย่อย) = $\log(n^2) = 2 \log n$
3. กรณีที่วัตถุเป็นสี่เหลี่ยมลูกบาศก์ log(จำนวนชิ้นส่วนย่อย) = $\log(n^3) = 3 \log n$



รูปที่ 2.1 ก ขั้นตอนการสร้างรูปสามเหลี่ยม Sierpinski



รูปที่ 2.1 ข สามเหลี่ยม Sierpinski

จากการแบ่งย่อยเส้น สี่เหลี่ยม และสี่เหลี่ยมลูกบาศก์ ในแบบที่ศึกษาข้างต้น เมื่อขยายแต่ละส่วนย่อย n เท่า สิ่งที่ได้ก็คือวัตถุเดิมก่อนถูกแบ่งแยกนั่นเอง ดังนั้นสำหรับวัตถุที่พิจารณาด้วยแฟรคตอล มิติของวัตถุ (D) มีค่าเท่ากับ

$$D = \frac{\log (\text{number of pieces})}{\log (\text{magnification})}$$

ดังนั้น มิติสำหรับวัตถุที่เป็นเส้นมีค่า $D = \frac{\log n^1}{\log n} = 1$

มิติสำหรับสี่เหลี่ยม $D = \frac{\log n^2}{\log n} = \frac{2 \log n}{\log n} = 2$

มิติสำหรับสี่เหลี่ยมลูกบาศก์ $D = \frac{\log n^3}{\log n} = \frac{3 \log n}{\log n} = 3$

เมื่อได้สมการมิติแฟรคตอลของวัตถุแล้วจะนำมาหามิติของสามเหลี่ยม Sierpinski ซึ่งเป็นภาพที่สร้างแบบแฟรคตอล ในการสร้างภาพสามเหลี่ยม Sierpinski ด้านของสามเหลี่ยมแต่ละด้านแบ่ง 2 ส่วน ดังนั้นภาพในสามเหลี่ยมใหญ่ 1 รูป จะประกอบด้วยสามเหลี่ยมย่อย 3 รูป ดังนั้นจำนวนชิ้นส่วนย่อย = 3 และขนาดขยาย (magnification)= 2 เพราะฉะนั้น มิติของภาพมีค่าเท่ากับ

$$D = \frac{\log (\text{number of pieces})}{\log (\text{magnification})}$$

ซึ่งไม่เป็นเลขจำนวนเต็ม ทดลองหาค่ามิติจากรูปที่เพิ่มความละเอียด ในการแบ่งเมื่อด้านสามเหลี่ยมแต่ละด้านแบ่งแยกออกเป็น 4 ส่วน ภายในสามเหลี่ยมใหญ่ 1 รูป จะประกอบด้วยสามเหลี่ยมย่อย 9 รูป ดังนั้นจำนวนชิ้นส่วนย่อย = 9 และขนาดขยาย (magnificaton) = 4 เพราะฉะนั้นมิติของภาพมีค่าเท่ากับ

$$D = \frac{\log 9}{\log 4} = \frac{\log 3^2}{\log 2^2} = \frac{2 \log 3}{2 \log 2} = \frac{\log 3}{\log 2} = 1.585...$$

ซึ่งมีค่าเท่ากับคำตอบเดิม เพราะฉะนั้นสรุปได้ว่าแฟรคตอลมีมิติแบบเศษส่วน

ข้อสังเกต การหามิติที่กล่าวข้างต้นเป็นวิธีที่ง่าย เนื่องจากขนาดขยายมีค่าคงที่ตลอดการสร้างภาพแต่ในการสร้างภาพแบบแฟรคตอลโดยทั่วไปแล้ว ภายในภาพเดียวกันอาจใช้ขนาดขยายที่มีค่าแตกต่างกัน ดังนั้นการคำนวณหาค่ามิติยากกว่าเดิม และได้มิติที่มีความแตกต่างกัน

2.4 รูปแบบปัญหาสำหรับการลดขนาดข้อมูลภาพด้วยการแปลงแบบแฟรคตอล

ประกอบด้วย 3 รูปแบบดังต่อไปนี้

2.4.1 คณิตศาสตร์สำหรับภาพจริงบนโลก (Formulation of Mathematical Model for Real World Images)

ก. คุณสมบัติของภาพจริงบนโลก (Description and Properties of Real World Images)

ข. รูปแบบคณิตศาสตร์สำหรับภาพจริงบนโลก(Mathematical Model for Real World Images)

ค. การกำหนดรูปแบบข้อมูลเข้าด้วยสแกนเนอร์ (How pictures are measured with the aid of scanner)

2.4.2 แบบของตัวประมาณภาพจริงบนโลกโดยอาศัยการแปลงแบบแฟรคตอล
(The Approximation of Real World Images by Fractal Transform Fractals)

ก. การแปลงแบบแฟรคตอล (Fractal Transform Fractals, FT fractal)

ข. ฟังก์ชันย้ำแบบเฉพาะที่ (Local Iterate Function System)

ค. ทฤษฎีภาพปะติดสำหรับฟังก์ชันย้ำแบบเฉพาะที่ (The Collage Theorem for a Local IFS)

ง. ทฤษฎีการจับคู่แบบย่นย่อ (Contraction Mapping Theorem)

จ. วิธีการลดขนาดข้อมูลภาพแบบแฟรคตอล (General Description of Fractal Image Compression)

2.4.3 การคำนวณบริเวณภาพสำหรับข้อมูลสตริง (A Computationally Tractable Model for The Source of The Data String)

กล่าวถึงคณิตศาสตร์ที่ใช้อธิบายภาพที่อินพุตเข้ามา แล้วนำเอาภาพมาจัดเก็บโดยใช้การแปลงแบบแฟรคตอล จากนั้นจะกำหนดโดเมนและเรนจ์ให้เป็นไปตามทฤษฎีคอล์เลจ(Collage Theorem)

2.5 <u>ทฤษฎีที่ใช้อ้างอิง</u>

สเปซ (space) หมายถึง เซตที่มีโครงสร้างบนตัวเอง

ตัวอย่างเช่น		
	R	เป็น จำนวนจริง
	R^2	เป็น ระนาบยูคลิด
	R^3	เป็น สเปซ 3 มิติ
	$[0,1]$	เป็น ช่วงปิดของ ศูนย์ หนึ่ง
	$\square \subset R^2$	เป็น สี่เหลี่ยมจัตุรัส ตารางหน่วย
	Σ	เป็น รหัสสเปซ

ซึ่งสเปซเหล่านี้จะมีโครงสร้างก็ต่อเมื่อสามารถหาเมทริกซ์ได้

นิยาม เมทริกซ์สเปซ (X,d) เป็นสเปซ หรือเซต เมื่อ X เป็นฟังก์ชันจำนวนจริงและ $d : X \times X \rightarrow R$ เป็นระยะทางระหว่างจุด x และ y จะเรียก d ว่าเป็นเมทริกซ์บนสเปซ X เมื่อ d มีคุณสมบัติดังต่อไปนี้

(i) $d(x,y) = d(y,x) , \forall x,y \in X$

(ii) $0 < d(x,y) < \infty , \forall x,y \in X , x \neq y$

(iii) $d(x,x) = 0 , \forall x \in X$

(iv) $d(x,y) \leq d(x,z) + d(z,y) , \forall x,y,z \in X$

นิยาม เมทริกซ์สเปซ (X,d) จะสมบูรณ์(complete metric space) ถ้าทุกๆลำดับคอร์ชี (Cauchy sequence) $\{x_n\}_{n=1}^{\infty}$ ใน X มีลิมิต $x \in X$

นิยาม ให้ $S \subset X$ เป็นสับเซตของเมทริกซ์สเปซ (X,d) S จะ compact (compact metric space) ถ้าทุกๆลำดับที่ไม่จำกัด $\{x_n\}_{n=1}^{\infty}$ ใน S บรรจุลำดับย่อย(subsequence)ที่มีลิมิต ใน S

นิยาม ลำดับ $\{x_n\}_{n=1}^{\infty}$ ของจุดในเมทริกซ์ สเปซ (X,d) จะเรียกว่า ลำดับคอร์ชี (Cauchy sequence) ถ้ากำหนดจำนวน $\varepsilon > 0$ แล้วพบว่ามีจำนวนเต็ม $N > 0$ ที่ทำให้

$$d(x_n, x_m) < \varepsilon \quad , \quad \forall\ n,m > N$$

Space of Image ~ Hausdorff space ($\mathcal{H}$) จะศึกษาเฉพาะสับเซตของ เมทริกซ์สเปซที่สำคัญ เช่น □ ,Euclidean แต่การลดขนาดข้อมูลภาพจะพิจารณากรณีที่ (X,d) เป็น R^2 และ d เป็น Euclidean metric

ดังนั้นเพื่อความสะดวกในการอ้างอิงจะใช้ เมทริกซ์สเปซ (metric space) $(\mathcal{H}(R^2), d(\text{Euclidean}))$

นิยาม ให้ (X,d) เป็น complete metric space , $x \in X$ และ $B \in \mathcal{H}(X)$ กำหนดให้ $d(x,B) = \min \{ d(x,y) : y \in B \}$ d(x,B) เรียกว่า ระยะทางจากจุด x ไปยังเซต B

นิยาม ให้ (X,d) เป็น complete metric space ให้ A และ B เป็น $\mathcal{H}(X)$ กำหนดให้ $d(A,B) = \max \{ d(x,B) : x \in A \}$ d(A,B) เรียกว่า ระยะทางจากเซต $A \in \mathcal{H}(X)$ ไปยังเซต $B \in \mathcal{H}(X)$

Lemma ให้ (X,d) เป็น complete metric space ถ้า A, B และ C เป็น $\mathcal{H}(X)$ ดังนั้น $d(A \cup B, C) = d(A,C) \vee d(B,C)$; $x \vee y$ หมายถึง ค่าสูงสุดของ x และ y

นิยาม ให้(X,d)เป็นเมทริกซ์สเปซสมบูรณ์ (complete metric space) ดังนั้นระยะทางเฮาส์ดอร์ฟ(Hausdorff distance)ระหว่างจุด A และ B ใน $\mathcal{H}(X)$ถูกนิยามโดย

$$h(A,B) = d(A,B) \cup d(B,A)$$

เรียก h ว่า ***Hausdorff Distance***

ให้ (X,d) เป็น metric space และให้ $(\mathcal{H}(X), h(d))$ แสดง Hausdorff space ด้วย Hausdorff metric h(d) ที่อธิบายมาแล้วจะใช้เครื่องหมาย h(d) เพื่อแสดงว่า d อยู่ ภายใต้เมทริกซ์ สำหรับ Hausdorff metric h

Lemma ให้ ฟังก์ชันการแปลง $f : X \to X$ เป็นการจับคู่แบบย่นย่อ(contraction mapping) บนเมทริกซ์สเปซ (X,d) ถ้าพบว่ามีค่าคงที่ $0 \leq s < 1$ ที่ทำให้

$$d(f(x),f(y)) \leq s \cdot d(x,y) \quad \forall x,y \in X$$

และเรียก s ว่า ตัวประกอบสำหรับการย่อสำหรับฟังก์ชัน เอฟ (contractivity factor for f)

2.5.1 ทฤษฎี Collage

ทฤษฎีนี้เป็นการหา IFS ที่ทำให้ภาพแอทแทรกเตอร์ เหมือนหรือใกล้เคียงกับภาพเดิม จะต้องหาขอบเขตของการแปลงแบบย่นย่อบนสเปซที่เป็นสเปซของภาพเดิม โดยใช้การผูเนียนหรือการปะติดปะต่อ(Collage) ค่าที่ใช้ตัดสินว่าแอทแทรกเตอร์เหมือนกับภาพเดิมหรือไม่จะใช้เฮาส์ดอร์ฟสเปซ

ทฤษฎี (Collage theorem)

ให้ (X,d) เป็น เมทริกซ์สเปซสมบูรณ์ ให้ $T \in \mathcal{H}(X)$ และ $\varepsilon \geq 0$ เลือก IFS $\{X; (w_0),w_1,w_2,...,w_N\}$ เป็นตัวประกอบแบบย่นย่อ $0 \leq s < 1$

$$h(T, \bigcup_{n=1}^{N} w_n(T)) \leq \varepsilon$$

ที่ h(d) เป็น เฮาส์ดอร์ฟเมทริกซ์ เมื่อ

$$h(T,A) \leq \frac{\varepsilon}{1-s}$$

เมื่อ A เป็น แอทแทรกเตอร์ ของ IFS ซึ่งสมมูลกับ

$$h(T,A) \leq (1-s)^{-1} h(T, \bigcup_{n=1}^{N} w_n(T)) \qquad \text{สำหรับทุก } T \in \mathcal{H}(X)$$

2.5.2 ทฤษฎีการจับคู่แบบย่นย่อ (The Contraction Mapping Theorem)

กำหนดให้ $f : X \to X$ เป็นการจับคู่แบบย่นย่อ บน (X,d)ซึ่งเป็น เมทริกซ์สเปซสมบูรณ์ และทำการแปลงด้วยฟังก์ชัน f บนจุดตรึง $x_f \in X$ และสำหรับจุดใด ๆ ที่ไม่ใช่จุดตรึงลำดับ $\{f^{on}(x) : n=0,1,2,...\}$ จะลู่เข้าสู่ x_f

$$\lim_{n \to \infty} f^{on}(x) = x_f \,, \exists x \in X$$

การจับคู่แบบย่นย่อบน สเปซ $\mathcal{H}$ (Contraction mappings on the space $\mathcal{H}$)

Lemma ให้ $w : X \to X$ เป็นการจับคู่แบบย่นย่อบน metric space (X,d) ดังนั้น w จะต่อเนื่อง ให้ $\varepsilon > 0$ ให้ $s > 0$ เป็นตัวประกอบแบบย่อสำหรับ w ดังนั้น

$$d(w(x),w(y)) \leq s\, d(x,y) < \varepsilon$$

เมื่อ $d(x,y) < \delta$, $\delta = \varepsilon/s$

Lemma ให้ (X,d) เป็น เมทริกซ์สเปซ ให้ $\{ w_n : n = 1,2,...,N \}$ เป็นการจับคู่แบบย่นย่อบน $(\mathcal{H}(X),h)$ ให้ตัวประกอบแบบย่นย่อ สำหรับ w_n เป็น s_n สำหรับแต่ละ n กำหนด $W : \mathcal{H}(X) \to \mathcal{H}(X)$ โดย

$$W(B) = w_1(B) \cup w_2(B) \cup ... \cup w_n(B)$$

$$= \bigcup_{n=1}^{N} w_n(B) \quad \text{สำหรับแต่ละ} \quad B \in \mathcal{H}(X)$$

W เป็นการจับคู่แบบย่นย่อ ด้วยตัวประกอบแบบย่อ $s = \max \{ s_n : n = 1,2,...,N \}$

2.5.3 Iterated Function System (IFS)

นิยาม IFS ประกอบด้วย เมทริกซ์สเปซสมบูรณ์ (X,d) กับเซตจำกัดของการจับคู่แบบย่นย่อ $w_n : X \to X$ ที่มีตัวประกอบแบบย่นย่อ s_n สำหรับ $n = 1,2,...,N$ สัญญลักษณ์ที่ใช้แสดง IFS นี้ คือ $\{ X : w_n , n = 1,2,...,N \}$ และ ตัวประกอบแบบย่นย่อ $s = \max \{ s : n = 1,2,...,N \}$

ทฤษฎี IFS ให้ $\{ X ; w_n , n = 1,2,...,N$ เป็น ระบบฟังก์ชันย้ำแบบไฮเปอร์โบลิก มีตัวประกอบแบบย่นย่อ s ดังนั้นการแปลง $W : \mathcal{H}(X) \to \mathcal{H}(X)$ กำหนดโดย

$$W(B) = \bigcup_{n=1}^{N} w_n(B)$$

สำหรับ $B \in \mathcal{H}(X)$, $W(B)$ เป็นการจับคู่แบบย่นย่อ บนเมทริกซ์สเปซสมบูรณ์ $(\mathcal{H}(X),h(d))$ มีตัวประกอบแบบย่นย่อ s นั่นคือ

$$h(W(B),W(C)) \leq s \cdot h(B,C)$$

สำหรับ $B,C \in \mathcal{H}(X)$ มีจุดคงที่เพียงจุดเดียว (unique) ,$A \in \mathcal{H}(X)$ ที่

$$A = W(A) = \bigcup_{n=1}^{N} w_n(A)$$

และ ให้ $A = \lim_{n \to \infty} W^{on}(B)$ $B \in \mathcal{H}(X)$

นิยาม จุดคงที่ที่อธิบายในทฤษฎี IFS เรียกว่า แอทแทรกเตอร์ ของ IFS

2.6 แบบจำลองคณิตศาสตร์สำหรับภาพจริงบนโลก

(Formulation of Mathematical Models for Real World Images)

สำหรับภาพจริงที่เกิดขึ้นบนโลกจะแทนด้วยสเปซ $\mathfrak{R}$ มีคุณสมบัติที่น่าสนใจเช่น ถ้าส่วนใดส่วนหนึ่งของภาพถูกตัดออก(clip) จากภาพจริงบนโลก ส่วนของภาพที่ถูกตัดนั้นจะมีสเปซ $\mathfrak{R}$ สเปซอื่นมารองรับ ดังนั้นจะแบ่งสเปซ $\mathfrak{R}$ ออกเป็น 'world' สมาชิกของภาพจริงบนโลกคือสมาชิกทั้งหมดของสเปซ $\mathfrak{R}$ หากเข้าถึง 'world' ได้ก็จะสามารถเข้าถึงสมาชิกตัวใดตัวหนึ่งใน $\mathfrak{R}$

คำอธิบายและคุณสมบัติของภาพจริงบนโลก

(Description and Properties of Real Worlf Images)

ภาพจริงบนโลก คือ ภาพใดๆที่มองเห็นด้วยตา หรือ เมื่อจินตนาการว่ามีภาพนั้นแล้วสามารถหาดูได้จริง แหล่งกำเนิดภาพเหล่านั้นอาจมาจากระยะไกลเช่น ภาพถ่ายจากอากาศยาน หรือภาพพื้นผิวของดวงจันทร์ หรืออาจเป็นภาพที่มาจากนิตยสาร ซึ่งอาจเป็นทั้งภาพสีหรือขาว-ดำ

กำหนด $\mathfrak{R}$ เป็นเซตของภาพจริงบนโลก ให้ $\mathcal{I}$ เป็นสมาชิกของ $\mathfrak{R}$ ดังนั้น $\mathcal{I}$ มีคุณสมบัติเป็นภาพ ที่มีคุณสมบัติต่อไปนี้

คุณสมบัติข้อ 1

ภาพจริงบนโลกใดๆ ที่ $\mathcal{I}\in\mathfrak{R}$ จะมี สับพอร์ต(สับพอร์ต)และมิติ สับพอร์ตของภาพคือเซต $\square\in R^2$ เมื่อ R^2 แทนระนาบยูคลิด และ $\square$ กำหนดโดย

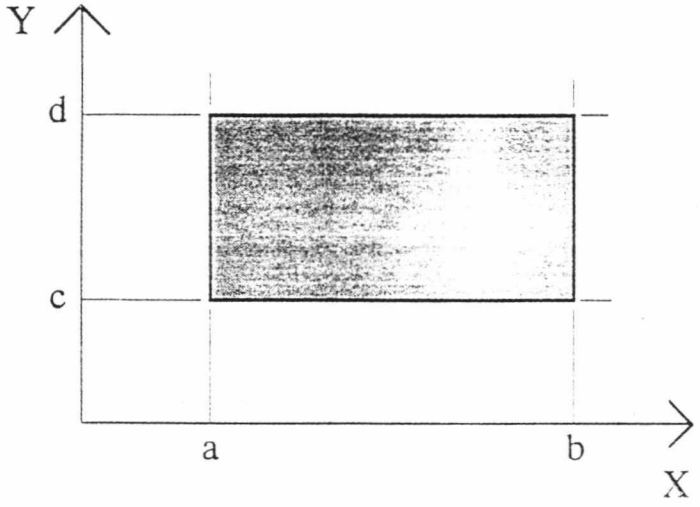
$$\square = \{ (x,y)\in R^2 : a\le x\le b,\ c\le y\le d \}$$

เมื่อ $a<b$ และ $c<d$ เป็นค่าคงที่ที่เป็นจำนวนจริง

ดังนั้น มิติของ $\mathcal{I}$ คือ (b-a) หน่วย และ (d-c) หน่วย หน่วยในการวัดที่ว่านี้มีหน่วยในทางฟิสิกส์เช่น เมตรหรือนิ้ว และกล่าวว่า $\square$ เป็นสับพอร์ตของภาพ $\mathcal{I}$

เพราะฉะนั้นเซต $\square\subset R^2$ จะมีมิติเท่ากับ (b-a) หน่วยและ (d-c)หน่วย ทำให้กำหนดวัตถุใดๆในธรรมชาติได้สอดคล้องกับ $\mathfrak{R}$

สับพอร์ต $\square$ มีคุณสมบัติทางเรขาคณิตและโทโพโลยี รูปที่ 2.2 แสดงสับพอร์ตสำหรับภาพใดๆ



รูปที่ 2.2 แสดงการมีสับพอร์ต $\square$ ของภาพจริงบนโลก $\square = \{(x,y) \in R^2 : a \leq x \leq b, c \leq y \leq d, \}$

จุดที่ปรากฏในภาพเป็นจุดในสับพอร์ตและทุกๆ จุดในสับพอร์ตก็เป็นจุดในภาพนั้นเช่นกัน ระยะทาง d ระหว่างจุด (x_1,y_1) และ (x_2,y_2) หาจากเมตริกยูคลิดซึ่งมีค่า

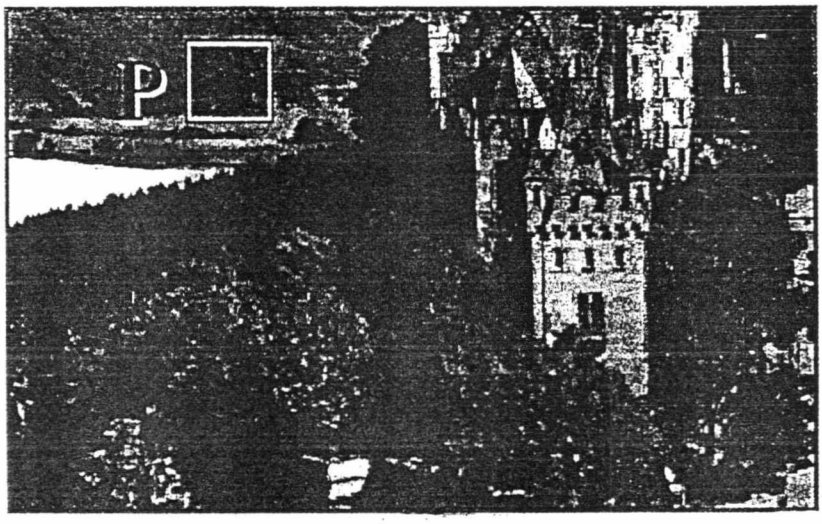
$$d = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

ถ้าจินตนาการว่าสับพอร์ตของภาพเป็นแผ่นกระดาษหรือภาพถ่าย พื้นที่ภายในภาพจะหมายถึงส่วนของภาพที่มีความสัมพันธ์กับพื้นที่สับพอร์ต และสับเซตของภาพหมายถึงส่วนของภาพที่มีความสัมพันธ์กับสับเซตของสับพอร์ต

สับพอร์ตของภาพ $\square \subset R^2$ ในเมตริกยูคลิดเป็นตัวอย่างหนึ่งของเมตริกสเปซโทโพโลยีของสเปซที่อธิบายธรรมชาติของภาพจริงบนโลก คุณสมบัติโทโพโลยีมีความสำคัญสำหรับเมตริกที่เป็นการแปลงของสับพอร์ต คุณสมบัติเหล่านั้นได้แก่ boundary, interior openness, closedness, connectedness และ compectness

คุณสมบัติข้อ 2

กำหนดให้ $\mathcal{I} \in \mathcal{R}$ ดังนั้น $\mathcal{I}$ จะแสดงคุณสมบัติสี (chromatic attributes) ที่ให้รายละเอียดเกี่ยวกับความถี่(สี) และความเข้มแสง ที่มีความสัมพันธ์กับเซตย่อยของภาพ ล่าเหล่านี้จะกำหนดด้วยฟังก์ชันจำนวนจริง เรียกว่า ตัววัดที่เป็นจำนวนจริงของโบเรล (real valued Borel measure) ค่าที่วัดได้จะสนับสนุนสับพอร์ตของภาพ ทุกๆ เซตที่วัดได้ในสับพอร์ตของ $\mathcal{I}$ มีความสัมพันธ์กับกลุ่มตัวเลขที่อธิบายความถี่และความเข้มแสง ดังแสดงในรูปที่ 2.3



รูปที่ 2.3 กำหนดให้ $\mathcal{I} \in \mathcal{R}$ ดังนั้น $\mathcal{I}$ จะดำเนินเกี่ยวกับคุณสมบัติสีซึ่งทุก ๆ เซตการวัดที่วัดจากสับพอร์ตของ $\mathcal{I}$ จะเป็นจำนวนที่แทนสีและความเข้มแสง

หัวข้อ ***คณิตศาสตร์สำหรับภาพจริงบนโลก*** เป็นคณิตศาสตร์ที่ใช้กำหนดคุณสมบัติเม็ดสี แต่จะต้องศึกษารูปแบบเฉพาะต่อไปนี้เสียก่อนเพื่อนำไปอธิบายคุณสมบัติความอิสระของรีโซลูชัน

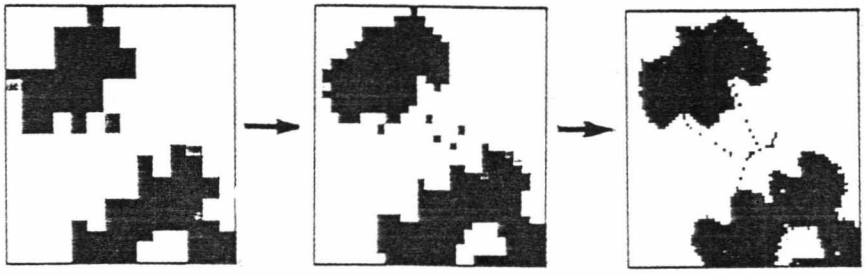
สิ่งที่สำคัญที่สุดในการติดต่อกับอุปกรณ์แบบดิจิตอลคือ การแบ่งสับพอร์ตของภาพออกเป็นอาร์เรย์สี่เหลี่ยมหรือที่เรียกว่า พิกเซล(pixel) แต่ละพิกเซลจะมีคุณสมบัติของเม็ดสีและเป็นสมาชิกใน $\mathcal{R}$ ภายในพิกเซลแต่ละจุดสามารถกำหนดเซตจำกัดของเลขจำนวนจริงและเลขเหล่านี้เป็นตัวแทนของความเข้มแสงเช่น แดง เขียว น้ำเงิน แต่มีภาพบางชนิดมีเฉพาะสีขาว ดำ เท่านั้น เช่น เอกสารแฟกซ์ ภาพถ่ายขาว-ดำ เรียกภาพเหล่านี้ว่า เกรย์สเกล (grayscale)

ถ้าให้ $\mathcal{I}$ เป็นภาพขาว-ดำ แล้วจุดในสับพอร์ต □ จะแทนด้วยเลขศูนย์หรือหนึ่ง ทั้งนี้ขึ้นอยู่กับภาพเป็นสีใด หากจุดนั้นเป็นสีดำจะแทนด้วยศูนย์ และถ้าเป็นสีขาวจะแทนด้วยหนึ่งแต่ถ้า $\mathcal{I}$ เป็นภาพแบบเกรย์สเกล แต่ละจุดใน □ แทนด้วยเลขที่มีความสัมพันธ์กับความเข้มแสง ซึ่งเป็นวิธีที่ยากกว่าการกำหนดภาพแบบขาว-ดำ

อย่างไรก็ตามในการกำหนดคุณสมบัติของสีจะต้องกำหนดพื้นที่และสีของพื้นที่นั้น เช่น กล่าวว่าท้องฟ้ามีสีน้ำเงิน การกำหนดสีที่แน่นอนจะทำให้กำหนดความเข้มของพิกเซลในภาพได้แน่นอนด้วย

คุณสมบัติข้อ 3

ให้ $\mathcal{I} \in \Re$ แล้ว $\mathcal{I}$ เป็นความอิสระของรีโซลูชัน(resolution independent)ตัวเลขในรีโซลูชันจะกำหนดพิกเซลที่ใช้แทนสีและความเข้มสำหรับค่ารีโซลูชันที่มีความละเอียดสูงจะได้ภาพที่ชัดเจนขึ้นดังแสดงในรูปที่ 2.4



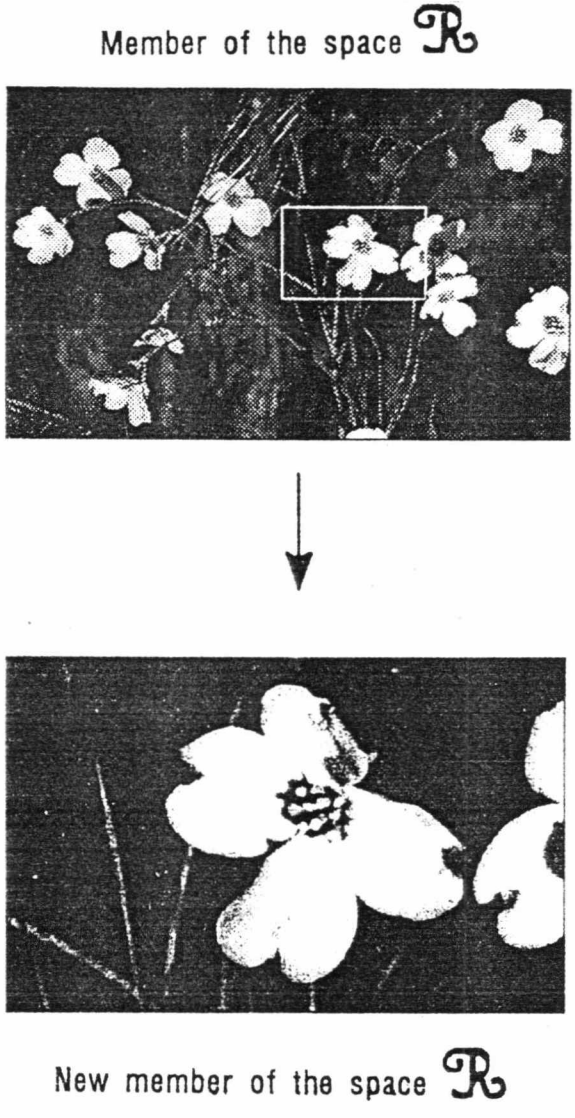
รูปที่ 2.4 แสดงภาพหนึ่งภาพที่ได้จากการใช้ค่ารีโซลูชันที่แตกต่างกัน ดังนั้นภาพ $\mathcal{I}$ ใด ๆ ในเซตของภาพจริงบนโลกจะเป็นอิสระจากรีโซลูชัน (resolution independent)

คุณสมบัติข้อ 4

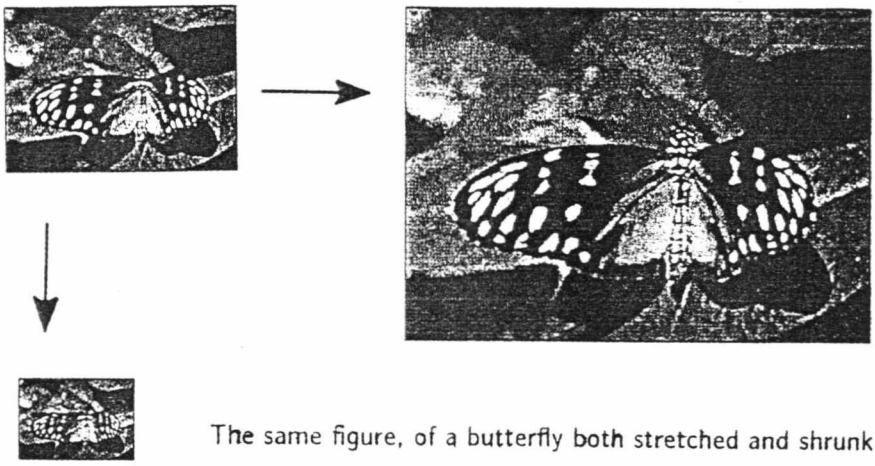
เซต $\Re$ ของภาพจริงบนโลกมีคุณสมบัติปิดภายใต้ตัวดำเนินการคลิป(clipping operation) กำหนดให้ $\mathcal{I} \in \Re$ เลือกพื้นที่สี่เหลี่ยมผืนผ้าที่อยู่ภายในสับพอร์ตของ $\mathcal{I}$ และมีด้านขนานกับสับพอร์ตของ $\mathcal{I}$ จะได้ $\tilde{\mathcal{I}}$ ภาพใหม่แล้ว $\tilde{\mathcal{I}} \in \Re$ ดังแสดงในรูปที่ 2.5 ดังนั้นภายใต้ตัวดำเนินการคลิป ไม่ว่าจะเลือกพื้นที่เล็กเท่าใดภายในพื้นที่นั้นจะยังมีภาพหรือข้อมูลปรากฏอยู่

คุณสมบัติข้อ 5

กำหนดให้ $\mathcal{I} \in \Re$ และให้ $\tilde{\mathcal{I}}$ แทนผลจากการนำภาพ $\mathcal{I}$ ที่เหมือนกันไปย่อหรือขยาย(isotropically stretching หรือ isotropically shrinking) แล้ว $\tilde{\mathcal{I}} \in \Re$ ภาพ $\Re$ จะมีคุณสมบัติปิดภายใต้การย่อและขยายดังแสดงในรูปที่ 2.6 การย่อหรือขยายจะกระทำในทุกทิศทางเท่า ๆ กัน โดยไม่เจาะจงว่าจะย่อหรือขยายส่วนใดส่วนหนึ่งของภาพโดยเฉพาะ



รูปที่ 2.5 เซต $\mathfrak{R}$ เป็นภาพจริงบนโลกซึ่งเป็นเซตปิดภายใต้การคลิป ถ้าเลือกบริเวณภาพจากภาพ $\mathfrak{R}$ ให้เป็นรูปสี่เหลี่ยมผืนผ้า แล้วนำมาขยายภาพที่ได้ พบว่ายังคงเป็นภาพ $\mathfrak{R}$ ซึ่งเป็นภาพจริงบนโลกเช่นกัน

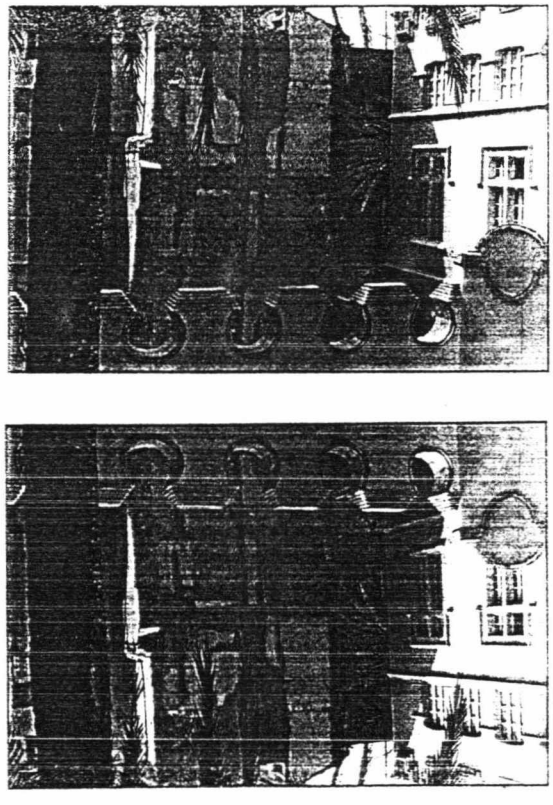


The same figure, of a butterfly both stretched and shrunk.

รูปที่ 2.6 เซต $\mathcal{R}$ เป็นเซตปิดภายใต้การนำภาพเดิมไปขยายหรือย่อ

คุณสมบัติข้อ 6

กำหนดให้ $\mathcal{I} \in \mathcal{R}$ ให้ $\tilde{\mathcal{I}}$ แทนผลที่ได้จากการสะท้อน (reflection) ในแกนที่ขนานกับด้านใดด้านหนึ่งของสับพอร์ต แล้ว $\tilde{\mathcal{I}} \in \mathcal{R}$ นั่นคือ $\mathcal{R}$ มีคุณสมบัติปิดภายใต้การสะท้อนดังแสดงในรูปที่ 2.7

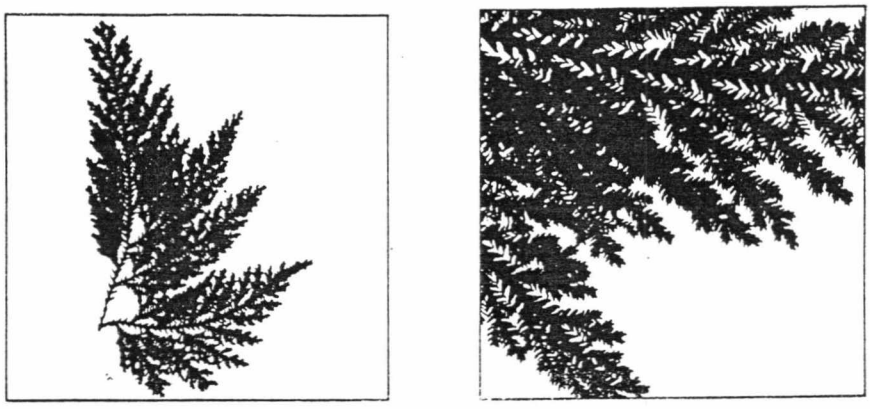


รูปที่ 2.7 ภาพ $\mathcal{R}$ มีคุณสมบัติภายใต้การสะท้อน

คุณสมบัติข้อ 7

กำหนดให้ $\mathcal{I} \in \mathcal{R}$ ให้ $\mathcal{I}^{\sim}$ แทนผลที่ได้จากการกำจัดพื้นที่สี่เหลี่ยมที่ถูกหมุนของ $\mathcal{I}$ (clipping a rotated rectangular region of $\mathcal{I}$) แล้ว $\mathcal{I}^{\sim} \in \mathcal{R}$ นั่นคือ $\mathcal{R}$ มีคุณสมบัติปิดภายใต้การหมุน ดังแสดงในรูปที่ 2.8

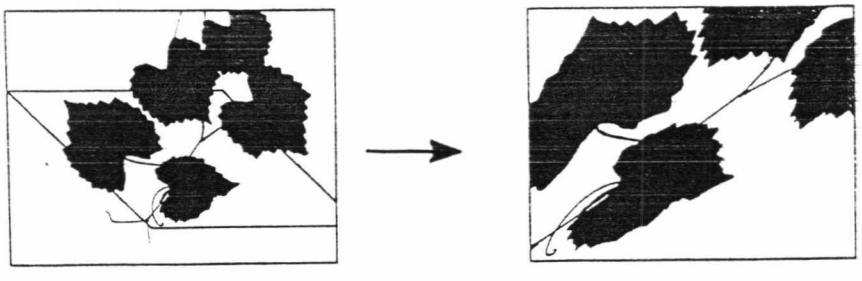
นำคุณสมบัติข้อ 5,6,7 ไปพิจารณาคุณสมบัติข้อ 8 ต่อไป



รูปที่ 2.8 แสดงการสร้างภาพจริงบนโลกที่ได้จากการนำภาพที่ถูกคลิปไปหมุน (rotate)

คุณสมบัติข้อ 8

$\mathcal{R}$ มีคุณสมบัติปิดภายใต้การแปลงผกผันแบบแอฟฟินซึ่งจะกำหนดพื้นที่สำหรับสี่เหลี่ยมผืนผ้าสำหรับรูปที่เกินขอบเขตสี่เหลี่ยม จะถูกกำจัดออกไป(clipping) ดังแสดงในรูปที่ 2.9



รูปที่ 2.9 แสดงภาพที่ได้จากการคลิปเมื่อกำหนดกรอบเริ่มต้นเป็นสี่เหลี่ยมด้านขนาน แล้วแปลงเป็นกรอบสี่เหลี่ยมผืนผ้า โดยการแปลงแบบผกผัน

นิยาม กำหนดให้ $\mathcal{I}$ และ $\mathcal{H}$ เป็นสมาชิกของ $\mathfrak{R}$ แล้ว $\mathcal{H}$ ถูกดีไรฟ์จาก $\mathcal{I}$ ก็ต่อเมื่อ จะสร้าง $\mathcal{H}$ จาก $\mathcal{I}$ โดยวิธีการคลิปและการขยาย(หรือการย่อ) อาศัยคุณสมบัติข้อ 4 และ 5 จะใช้สัญลักษณ์ $\mathcal{H} \prec \mathcal{I}$ แทน $\mathcal{H}$ ที่ถูกดีไรฟ์จาก $\mathcal{I}$

นิยาม กำหนดให้ $S \subset \mathfrak{R}$ แทนเซตของภาพจริงบนโลก กำหนด $\mathcal{W}(S) \subset \mathfrak{R}$ โดย

$$\mathcal{W}(S) = \{\mathcal{H} \in \mathfrak{R} : \mathcal{H} \prec \mathcal{I}, \exists \mathcal{I} \in S\}$$

แล้ว $\mathcal{W}(S)$ เป็นเซตหรือ'world'ของภาพที่สร้างจาก S โดยอาศัยคุณสมบัติข้อ 4 และ 5

นิยาม ภาพ $\mathcal{I}$ และ $\mathcal{H}$ ใน $\mathfrak{R}$ จะเท่ากัน ภายใต้คุณสมบัติข้อ 4 และ 5 ก็ต่อเมื่อ $\mathcal{I} < \mathcal{H}$ และ $\mathcal{H} \prec \mathcal{I}$ ใช้สัญลักษณ์ $\mathcal{H} \sim \mathcal{I}$ แทนภาพ $\mathcal{I}$ เท่ากับ $\mathcal{H}$

จากนิยามการเท่ากันบนสเปซ $\mathfrak{R}$ จะได้ว่า $\mathfrak{R}$ ประกอบด้วยคลาสที่เท่ากัน(equivalence class) แต่ละคลาสที่เท่ากันจะแทนภาพที่เท่ากัน $\varepsilon(\mathcal{I})$ แทนคลาสที่เท่ากัน ที่มี $\mathcal{I} \in \mathfrak{R}$ ให้ $\mathcal{I}\mathcal{H} \in \mathfrak{R}$ แล้ว $\mathcal{I} < \mathcal{H}$ ก็ต่อเมื่อ $\mathcal{P} < \mathcal{Q}$ สำหรับ $\mathcal{P}$ และ $\mathcal{Q}$ ทุกตัวที่ $\mathcal{P} \in \varepsilon(\mathcal{I})$ และ $\mathcal{Q} \in \varepsilon(\mathcal{H})$ ดังนั้นสำหรับภาพที่เกิดจากคลาสที่เท่ากันจะนิยามว่า $\varepsilon(\mathcal{I}) < \varepsilon(\mathcal{H})$ ก็ต่อเมื่อ $\mathcal{I} < \mathcal{H}$

กำหนดให้ $\mathcal{I}\mathcal{H} \in \mathfrak{R}$ และ $\mathcal{W}(\mathcal{I})$ แทนเซตของภาพที่สร้างจาก $\mathcal{I}$

$\varepsilon(\mathcal{H})$ แทนคลาสที่เท่ากันของ $\mathcal{H}$

เมื่อ $\varepsilon(\mathcal{H}) \cap \mathcal{W}(\mathcal{I})$ เป็นเซตว่างหรือเป็นเซตที่เท่ากับ $\varepsilon(\mathcal{H})$ แล้วจะได้

$$\mathcal{W}(\mathcal{I}) = \bigcup_{\mathcal{H} \in T} \varepsilon(\mathcal{H})$$

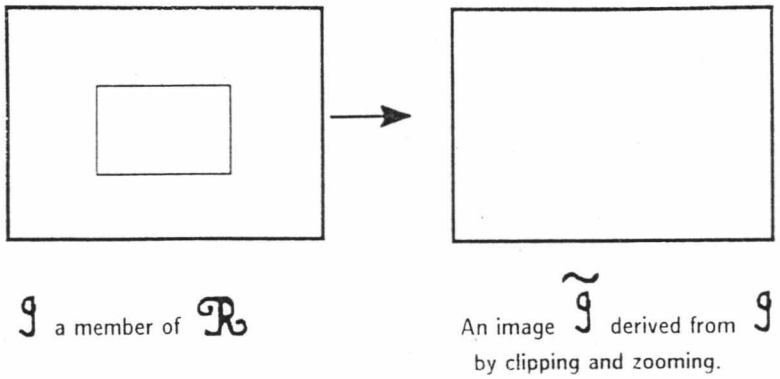
สำหรับบางเซต $T \subset \mathfrak{R}$

กำหนดให้ $|A|$ แทนจำนวนสมาชิกในเซต A สำหรับ $\mathcal{I} \in \mathfrak{R}$ กำหนดให้ $\|\mathcal{W}(\mathcal{I})\|$ แทนจำนวนคลาสที่เท่ากันใน $\mathcal{W}(\mathcal{I})$

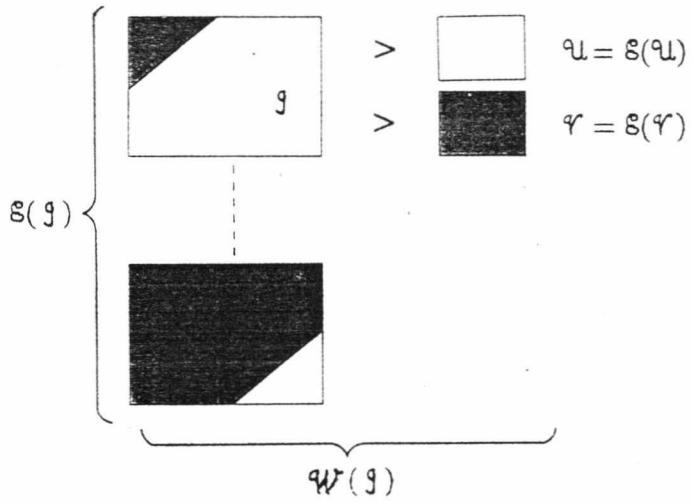
นั่นคือ $\|\mathcal{W}(\mathcal{I})\| = \min \{ |T| : T \subset \mathfrak{R}$ ที่ทำให้ $\mathcal{W}(\mathcal{I}) = \bigcup_{\mathcal{H} \in T} \varepsilon(\mathcal{H}) \}$

ถ้า $\|\mathcal{W}(\mathcal{I})\| \geq 1; \forall \mathcal{I} \in \mathfrak{R}$ แสดงว่าภายในภาพมีเซตที่เท่ากัน ถ้า $\mathcal{U}$ เป็นภาพที่มีรูปแน่นอน ดังแสดงในรูปที่ 2.10 ดังนั้นภาพที่ดีไรฟ์จาก $\mathcal{U}$ จะมีความเหมือนกับ $\mathcal{U}$ แล้ว $\|\mathcal{W}(\mathcal{U})\| = 1$

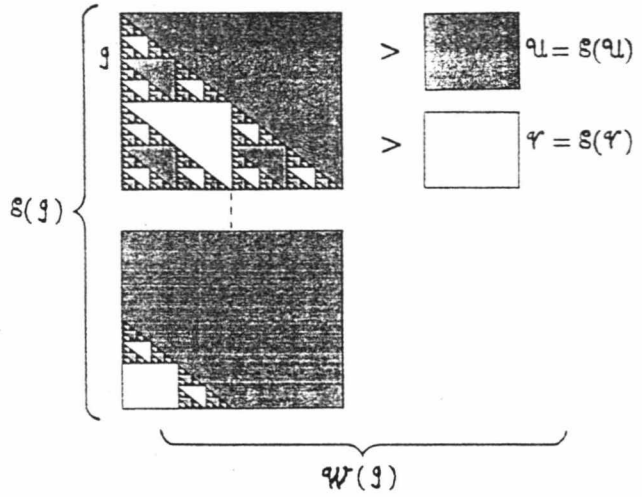
เมื่อ $\|\mathcal{W}(\mathcal{I})\| < \infty$ จะนิยาม $\mathcal{Q} \subset \mathfrak{R}$ โดย $\mathcal{Q} = \{\mathcal{I} \in \mathfrak{R} : \|\mathcal{W}(\mathcal{I})\| < \infty\}$ แล้วสำหรับ $\mathcal{U} \in \mathcal{Q}$ และ $\mathcal{I} \in \mathcal{Q}$ จะกำหนดโครงสร้างของคลาสที่เท่ากันของ $\mathcal{W}(\mathcal{I})$ ภายใต้ภาพที่มีความแตกต่างในขอบเขตของ □ ดังแสดงในรูปที่ 2.11 และ รูปที่ 2.12



รูปที่ 2.10 ภาพใด ๆ ดีไรฟ์จาก $\mathcal{U}$ จะเหมือน $\mathcal{U}$ ดังนั้น $\|\mathcal{W}(\mathcal{U})\| = 1$



รูปที่ 2.11 แสดงองค์ประกอบคลาสที่เท่ากัน (equivalence class) ของ $\mathcal{W}(\mathcal{I})$ เมื่อ $\mathcal{I} \in \mathcal{Q}$ เป็นภาพที่มีสองขอบเขตซึ่งถูกแบ่งโดยเส้นตรงในแนวเฉียง ดังนั้น $\mathcal{W}(\mathcal{I}) = \varepsilon(\mathcal{I}) \cup \varepsilon(\mathcal{U}) \cup \varepsilon(\mathcal{V})$ และ $\|\mathcal{W}(\mathcal{I})\| = 3$ จะสังเกตได้ว่า $\varepsilon(\mathcal{I})$ มีรูปแบบได้ไม่จำกัด



รูปที่ 2.12 แสดงองค์ประกอบคลาสที่เท่ากัน (equivalence class) ของ $\mathcal{W}(\mathcal{I})$ เมื่อ $\mathcal{I}\in\mathcal{Q}$ เป็นภาพที่มีสองขอบเขตซึ่งถูกแบ่งโดยสามเหลี่ยม Sierpinski ดังนั้น $\mathcal{W}(\mathcal{I})=\varepsilon(\mathcal{I})\cup\varepsilon(\mathcal{U})\cup\varepsilon(\mathcal{V})$ และ $\|\mathcal{W}(\mathcal{I})\|=3$ จะสังเกตได้ว่า $\varepsilon(\mathcal{I})$ มีรูปแบบได้ไม่จำกัด

$\mathcal{Q}$ เป็นการรวมกันของภาพที่สามารถพัฒนาทฤษฎีความสมบูรณ์ของภาพที่มีความซับซ้อน และพัฒนาความเป็นอิสระของรีโซลูชันได้ เนื่องจาก $\mathcal{Q}$ มี FT fractals(Fractal Transform Fractals) ดังนั้น $\mathcal{Q}$ มีคุณสมบัติใช้ประมาณ $\mathcal{R}$ ได้

2.7 <u>คณิตศาสตร์สำหรับภาพจริงบนโลก</u> (Mathematical Models for Real World Images)

รูปแบบทางคณิตศาสตร์ที่ใช้กำหนดคุณสมบัติเม็ดสี

โมเดล 1 $\mathcal{R}$ กำหนดโดย $\mathcal{R}_i$ ซึ่งเป็นภาพขาว-ดำ เช่นเอกสารแฟกซ์ ภาพวัตถุดำบนฉากขาว สมาชิก $\mathcal{I}$ ใดๆ ของ $\mathcal{R}_i$ ถูกแทนด้วย สับพอร์ต $\square$ ของ $\mathcal{R}_i$ ซึ่งมีคุณสมบัติ $\chi_A : \square \rightarrow \{0,1\}$ สำหรับสับเซตที่วัดแบบโบเรล $A \subset \square$ (Borel measurable subset) เซต A ที่วาดบน $\square$ แทนการวาดภาพสีดำลงบนฉากขาว ส่วนของภาพที่มีสีขาวแทนด้วยเลขศูนย์และส่วนที่มีสีดำแทนด้วย 1 ดังนั้นฟังก์ชัน $\chi_A(x)$ ที่แสดงคุณสมบัติสี

กำหนดโดย $\chi_a(x)= 1$ เมื่อ $x\in A$

$\chi_a(x)= 0$ เมื่อ $x\notin A$

● ภาพจริงบนโลกสำหรับโมเดลนี้จะสมบูรณ์เมื่อกำหนด สับพอร์ต $\square$ โดยโบเรลสับเซต $A \subset \square$ ในทางกลับกัน สมาชิกของ $\mathcal{R}_i$ กำหนดโดย $\{\square, A\subset\square : A$เป็นโบเรลสับเซตของ $R^2\}$ จากข้างต้นต้องการเซตAที่ทำให้ฟังก์ชัน$\chi_A : \square \rightarrow \{0,1\}$เป็นตัววัดของโบเรล(Borel Measurable) เพื่อที่จะหาค่าความเป็นไปได้บนสับเซตของตัววัดสำหรับภาพใดๆ ดังนั้นจะต้อง

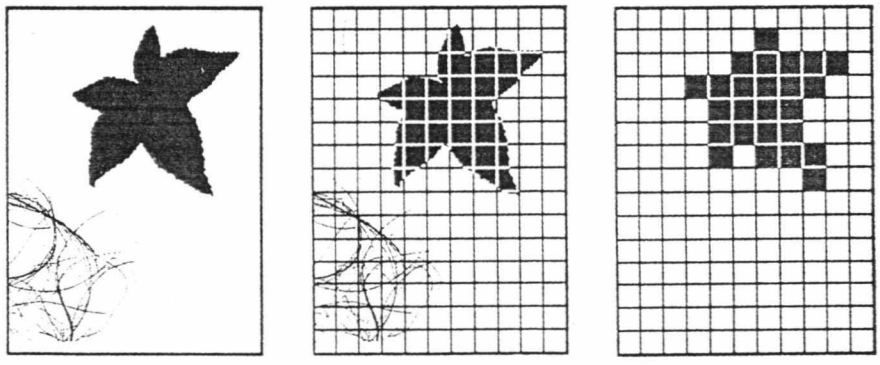
ประมาณค่ารีโซลูชันของ $\mathfrak{R}$ โดยอาศัยคุณสมบัติความเป็นอิสระของรีโซลูชัน(คุณสมบัติข้อ 3) ดังตัวอย่างต่อไปนี้

● สามารถกำหนดรูปแบบการประมาณค่าแบบดิจิตอลของรีโซลูชันที่มีความละเอียดสูง และมีค่าจำกัดสำหรับ $\mathcal{J}$ ได้โดยกำหนดค่า 0 หรือ 1 ให้กับพิกเซล $P \subset \square$ ที่ทำให้

$$\frac{\int_p X_A(x)\,dx}{\int_p dx}$$

มีค่าน้อยกว่า 0.5 หรือ มากกว่าหรือเท่ากับ 0.5

จากรูปที่ 2.13 กำหนดตัวประมาณค่าแบบดิจิตอลสำหรับเซต A แทนด้วย $\tilde{A}$ และให้ $\tilde{A} = \rho_{mxn}(A)$ แทนพิกเซล P ใน $\square$ ซึ่งเป็นอาร์เรย์ที่ประกอบด้วย m พิกเซลในแนวนอนและ n พิกเซลในแนวตั้ง



The set A | Digitization Grid over A | $\tilde{A}$, a Finite Resolution Approximation to A

รูปที่ 2.13 แสดง $\tilde{A}$ ซึ่งเป็นตัวประมาณรีโซลูชันแบบจำกัดที่ได้จากการประมาณ A เมื่อกำหนดอาร์เรย์ของพิกเซลบนเซต A

ดังนั้น $\rho_{mxn} : \mathfrak{R}_I \rightarrow \mathfrak{R}_I$ คือโปรเจคชันโอเปอร์เตอร์ ที่โปรเจคภาพลงในภาพของตัวเอง และสำหรับโบเรลสับเซต A ของ $\square$ ทุกเซตมี

$$\rho_{mxn}(\rho_{mxn}(A)) = \rho_{mxn}(A)$$

และจะหารูปแบบจำกัดของรีโซลูชันได้ $\{ \rho_{mxn}(A);\ m,n=1,2\ldots \}$ เนื่องจากสำหรับเซต A มี $\rho_{m'xn'}(A)$ (เมื่อ m'เป็นจำนวนเต็มที่มีค่ามากกว่า m และ n'เป็นจำนวนเต็มที่มีค่ามากกว่า n) เป็นตัวสร้างตัวประมาณรีโซลูชัน ที่มีความละเอียดสูง แบบดิจิตอล (high-resolution digital approximation) และจะมี $\rho_{m'xn'}(\rho_{mxn}(A))$ เป็นตัวสร้างตัวประมาณค่าสำหรับ $\rho_{mxn}(A)$ ดังนั้นเมื่อกำหนดค่า m และ n ผลที่ได้จากการประมาณคือ

$$d(\rho_{m'xn'}(A),\rho_{m'xn'}(\rho_{mxn}(A))) \rightarrow 0$$

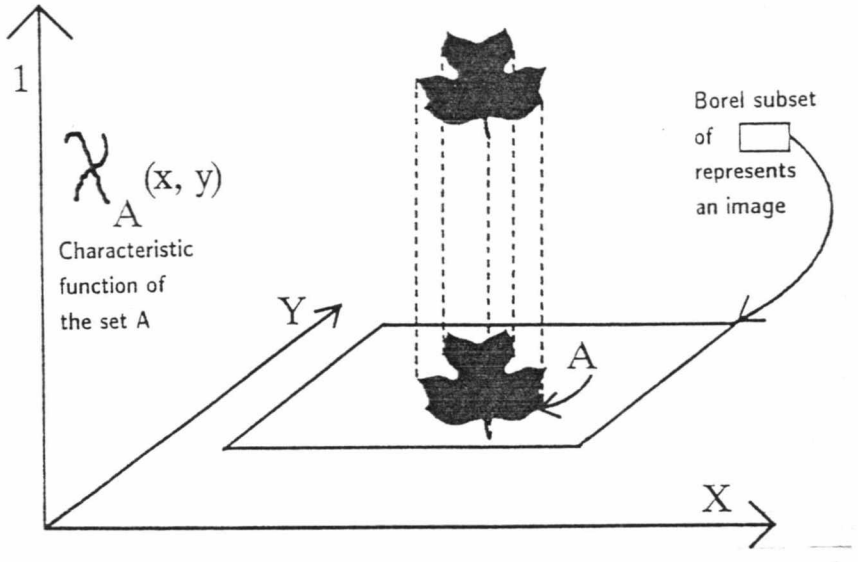
และ ค่า m',n' ลู่เข้าสู่อินฟินิตี

● d แทนเมตริกที่เกิดจากการประมาณบนสเปซ R_j สำหรับการเปรียบเทียบภาพใดๆที่อยู่บน สับพอร์ต เดียวกันเช่น

$$d(A,B) = \int_{\square} |\chi_A(x) - \chi_B(x)|dx$$

● ภาพ $\mathcal{I}$ ใดๆใน $\mathfrak{R}_j$ ที่สอดคล้องกับ A ที่เป็นโบเรลเซต จะนำมาสร้างภาพ $\mathcal{H} \in \mathfrak{R}_j$ ซึ่งทำให้เกิดภาพใหม่ด้วยวิธีการขยาย(ย่อ) สร้างโดยการจับคู่ สับพอร์ต ของ $\mathcal{I}$ ($\square \subset R^2$) ไปยังพื้นที่ $\square \subset R^2$ ซึ่งมีขนาดใหญ่(เล็ก)กว่าพื้นที่เดิมและมีมิติเป็นค่าคงที่ $s>0$ ที่มาก(น้อย)กว่า ในพื้นที่เดิม $\square$

ถ้าจุดกำเนิดของคู่ลำดับอยู่ที่มุมล่างซ้ายของสับพอร์ตแล้ว คุณสมบัติสีของ $\mathcal{H}$ ถูกกำหนด โดย $X(x) = \chi_A(x/s)$ สำหรับ x ทุกตัวที่ $x = (x,y) \in \square$ ภาพประกอบสำหรับ R_j แสดงในรูปที่ 2.14



รูปที่ 2.14 แสดงภาพจริงบนโลกของโมเดล 1 ($\mathfrak{R}_j$) ซึ่งอาศัยโบเรลสับเซตของ $\square$ ในการแทนภาพขาว-ดำ

โมเดล 2 $\mathfrak{R}$ กำหนดโดย $\mathfrak{R}_{jj}$ คุณสมบัติสีของ $\mathfrak{R}_{jj}$ แทนด้วยจำนวนจริงในช่วง $\mathcal{I}$ (เช่น $[0,255] \subset R$) และฟังก์ชัน $f : \square \rightarrow \mathcal{I}$ ฟังก์ชัน f(x,y) ใช้อธิบายความเข้มหรือความสว่างของภาพ ณ จุด(x,y) และเมื่อพิจารณาค่าที่เพิ่ม(integrability) และหรือ คลาสต่อเนื่อง(continuity class) สำหรับ f จะทำให้มองเห็นโมเดลได้ชัดเจนขึ้น

● ภาพของโมเดล $\mathfrak{R}_{jj}$ จะสมบูรณ์เมื่อกำหนด สับพอร์ต $\square$ และฟังก์ชัน $f : \square \rightarrow \mathcal{I}$

ในทางกลับกันฟังก์ชัน $f : \square \rightarrow \mathcal{I}$ ที่ถูกกำหนดด้านบางด้านไว้จะเป็นตัวกำหนดสมาชิกของ $\Re_{ii}$

● สามารถกำหนดรูปแบบการประมาณค่าแบบดิจิตอลของรีโซลูชันที่มีความละเอียดสูง สำหรับ $\mathcal{I} \in \Re_{ii}$ หาได้จากการผ่านฟังก์ชันที่มีค่า

$$f(P) = \frac{\int_p f(x)\, dx}{\int_p dx}$$

ไปยังพิกเซล P , $\exists P \in \square$

● ภาพ $\mathcal{I}$ ใดๆ ใน $\Re_{ii}$ ที่สร้างจาก f จะในไปสร้างภาพ $\mathcal{H} \in \Re_{ii}$ ที่ทำให้เกิดภาพใหม่ด้วยวิธีการขยาย(ย่อ)สร้างโดยการจับคู่ สับพอร์ต ของ $\mathcal{I}$ ($\square \subset R^2$) ไปยังพื้นที่ $\square \subset R^2$ ซึ่งมีขนาดใหญ่(เล็ก)กว่าพื้นที่เดิม และมีมิติเป็นค่าคงที่ s>0 ที่มาก(น้อย)กว่าในพื้นที่เดิม $\square$ คุณสมบัติสีของ $\mathcal{H}$ กำหนดจาก

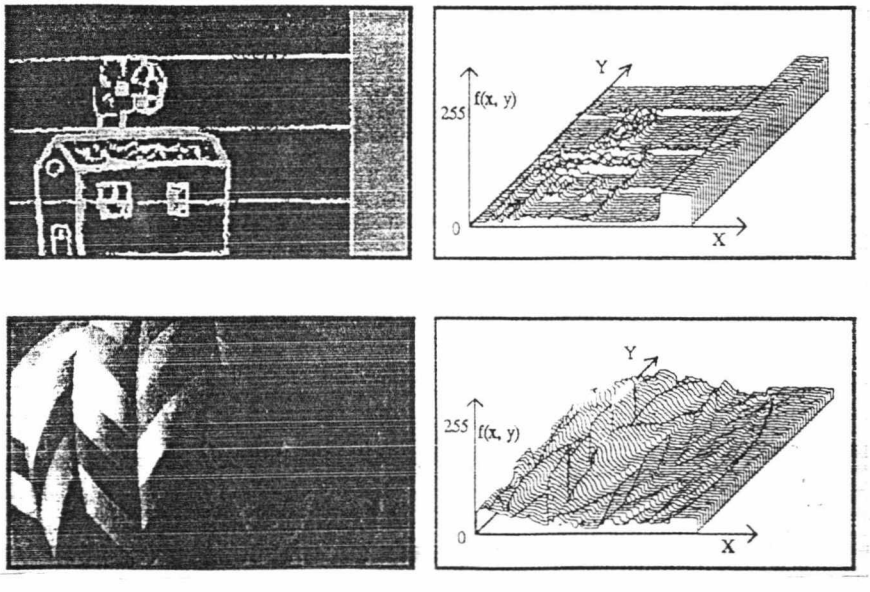
$$g(x,y) = f(x/s, y/s) \qquad ; \forall (x,y) \in$$

● ใน $\Re_{ii}$ ระยะทางระหว่างภาพ $\mathcal{I}$ และ $\mathcal{H}$ ที่อยู่ภายใต้ สับพอร์ต เดียวกันหาได้จาก

$$d(\mathcal{I}\mathcal{H}) = \int_{\square} |f(x) - g(x)|\, dx$$

เมื่อ $f : \square \rightarrow \mathcal{I}$ และ $g : \square \rightarrow \mathcal{I}$ เป็นฟังก์ชันสี

ภาพประกอบสำหรับ $\Re_{ii}$ แสดงในรูปที่ 2.15



รูปที่ 2.15 แสดงภาพจริงบนโลก ของ โมเดล 2 ($\Re_{ii}$) ซึ่งอาศัย ฟังก์ชันจำนวนจริง ($f : \square \rightarrow [0,255]$) ในการแทนภาพแบบเกรย์สเกล

โมเดล 3 $\mathfrak{R}$ กำหนดโดย $\mathfrak{R}_{iii}$ ภาพใน $\mathfrak{R}_{iii}$ แทนด้วย μ (real-valued normalized Borel measure) ที่อยู่บน $\square$ จำนวนแสงทั้งหมดที่แพร่กระจายจาก A เมื่อ A คือสับเซตของ Borel Measurable ($A \subset \square$) มีค่าเท่ากับ $\int_A d\mu$

- ภาพของโมเดล $\mathfrak{R}_{iii}$ จะสมบูรณ์เมื่อกำหนด สับพอร์ต $\square$และ μ ในทางกลับกัน สำหรับ μ ใดๆบน $\square$ $\int_{\square} d\mu=1$ เป็นตัวกำหนดสมาชิกของ $\mathfrak{R}_{iii}$
- สามารถกำหนดรูปแบบการประมาณค่าแบบดิจิตอลของรีโซลูชันที่มีความละเอียดสูง สำหรับ $\mathcal{I} \in \mathfrak{R}_{iii}$ หาได้จากการผ่านพิกเซล P บางตัวที่ $P \in \square$ ไปยัง $f(P)\int_p d\mu$
- ภาพ $\mathcal{I}$ ใดๆ ใน $\mathfrak{R}_{iii}$ ที่สร้างจาก μ จะนำไปสร้างภาพ $\mathcal{H} \in \mathfrak{R}_{iii}$ ที่ทำให้เกิดภาพใหม่ด้วยวิธีการขยาย(ย่อ) สร้างได้โดยการจับคู่ สับพอร์ต ของ $\mathcal{I}$ ($\square \subseteq R^2$) ไปยังพื้นที่ $\subset R^2$ ซึ่งมีขนาดใหญ่(เล็ก)กว่าพื้นที่เดิม และมีมิติเป็นค่าคงที่ $s>0$ ที่มาก(น้อย)กว่าในพื้นที่เดิม $\square$ คุณสมบัติสีของ $\mathcal{H}$ กำหนดจาก ν

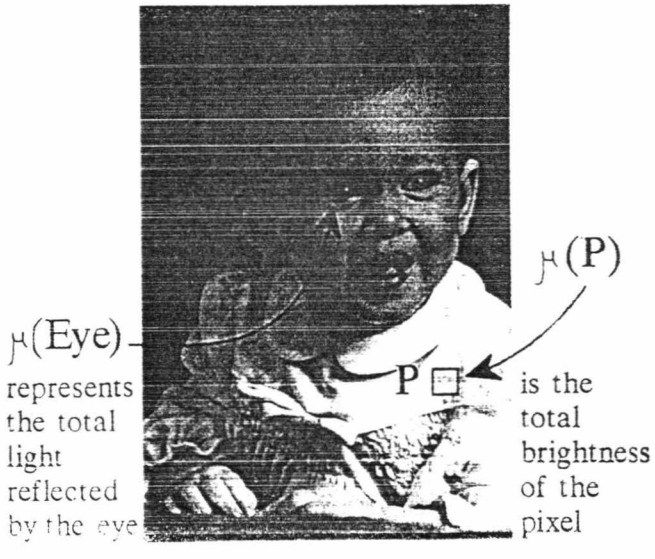
เมื่อ $\nu(B) = \mu(s^{-1}B)$ สำหรับทุกๆ โบเรลสับเซต $B \subset \square$ และ s^{-1}แทนการจับคู่แบบแอฟฟินบน R^2 จากพื้นที่ $\square$ ไปยัง $\square$

- ใน $\mathfrak{R}_{iii}$ ระยะทางระหว่างภาพ $\mathcal{I}$ และ $\mathcal{H}$ ที่อยู่ในสับพอร์ตเดียวกันจะมีค่าตามฟังก์ชันระยะทาง d ที่กำหนดโดย

$$d(\mathcal{I}\mathcal{H}) = \int_{\square} |d\mu(x) - d\nu(x)|$$

เมื่อ μ และ ν คือตัววัด (measure) ที่วัดจากภาพ $\mathcal{I}, \mathcal{H}$

ภาพประกอบสำหรับ $\mathfrak{R}_{iii}$ แสดงในรูปที่ 2.16



รูปที่ 2.16 แสดงภาพจริงบนโลก ของโมเดล 3 ($\mathfrak{R}_{iii}$) ซึ่งอาศัย Normalized Borel Measures (μ) บนสับพอร์ต $\square$ ในการแทนภาพแบบเกรย์สเกล

2.8 การกำหนดรูปแบบข้อมูลนำเข้าด้วยสแกนเนอร์

สแกนเนอร์เป็นอุปกรณ์ที่แปลงข้อมูลแบบแอนะล็อกเป็นข้อมูลแบบดิจิตอล หรือกล่าวว่า สแกนเนอร์แปลงฟังก์ชันของภาพจริงบนโลก ($\Re$) ให้เป็นสเปซของภาพแบบตัวเลข(digital images) วิธีในการแปลงข้อมูลจะใช้อุปกรณ์ที่เรียกว่า ซีซีดี (Charge Coupled Devices)

ภาพที่นำเข้าโดยสแกนเนอร์ถูกแบ่งออกเป็นตารางที่สอดคล้องกับพิกเซล จากนั้นซีซีดีจะแปลงแอตทิบิวลีที่อาจมีขนาด 1 พิกเซลหรือมีขนาดเป็นอาร์เรย์ไปเป็นประจุไฟฟ้า และประจุไฟฟ้านี้ถูกเปลี่ยนเป็นเลขจำนวนจริงโดยตัวแปลงสัญญาณแอนะล็อกเป็นดิจิตอล (analog-to-digital converter) ดังนั้นสรุปว่าข้อมูลภาพนำเข้าสแกนเนอร์เป็นข้อมูลแอนะล็อกและผลข้อมูลภาพที่นำออกจากสแกนเนอร์เป็นข้อมูลดิจิตอล สำหรับภาพ $\mathcal{I}$ เป็นข้อมูลแอนะล็อกมีรูปแบบที่อธิบายได้ด้วย ***คณิตศาสตร์สำหรับภาพจริงบนโลก*** ($\mathcal{I} \in \Re$)

ความผิดพลาดที่เกิดขึ้นจากการใช้สแกนเนอร์ จะทำให้ภาพที่ปรากฏบนจอมีความแตกต่างจากภาพต้นฉบับ มีสาเหตุจาก

1. ความผิดพลาดจากการสุ่ม(sampling error) เป็นความผิดพลาดที่เกิดในขบวนการสุ่มเลขจำกัด (finite number) สำหรับภาพแบบดิจิตอลมาแทนตัวเลขอนันต์(infinite number) ในภาพแบบแอนะล็อก

2. ความผิดพลาดจากการแบ่งพิสัย (quantizaton error) ของข้อมูลสุ่ม (sampled data) ออกเป็นพิสัยย่อยที่ถูกจำกัดค่า คือไม่มีค่าซ้ำซ้อนกัน

หลังจากผ่านขบวนการสุ่ม (sampling) และแยกพิสัย (quantizing) จะได้ข้อมูลที่เป็นอาร์เรย์ของตัวเลข จากนั้นนำอาร์เรย์เหล่านี้ไปสร้างสายข้อมูลตัวเลข (stream of data) เพื่อเป็นข้อมูลเข้า(Input) สำหรับใช้ศึกษาการลดขนาดข้อมูลแบบแฟรคตอลต่อไป

วิธีสร้างสายข้อมูลตัวเลข กระทำได้ 2 แบบคือ

-การอ่านข้อมูลแบบกลุ่ม (a block by block manner)

-การอ่านข้อมูลแบบพิกเซล (a pixel by pixel manner)

โดยทั่วไปมักใช้วิธีแรกมากกว่า สิ่งสำคัญสำหรับการอ่านข้อมูลแบบกลุ่มคือ การกำหนดทิศทางในการกวาดข้อมูล (scanning order) ซึ่งจะทำให้ลดขนาดสายข้อมูลตัวเลขได้และข้อมูลที่ได้จากการกวาดข้อมูลจะต้องมีความแปรผันน้อยที่สุด เมื่อเทียบกับพิกเซลในภาพเดิม

2.9 การแปลงแบบแฟรคตอล (Fractal Transform)

ทฤษฎีการแปลงแฟรคตอล (Fractal Transform Theory) เป็นทฤษฎีของระบบฟังก์ชันซ้ำแบบ local (Local Iterated Function Systems) หรือเรียกย่อ ๆว่า local IFS

local IFS คือ IFS ซึ่งมีข้อจำกัดบนการแปลงโดยที่โดเมนของการแปลงจะเป็น สเปซ (space)ทั้งหมดหรือเป็นสับเซตของสเปซ เช่นการแปลงแบบ local บนสเปซ X เป็นการแปลงที่มีโดเมนเป็นสับเซตของสเปซ X โดยที่การแปลงแบบนี้ไม่จำเป็นต้องกระทำบนจุดทุกจุดในสเปซ การแปลงภาพแบบแฟรคตอลจำเป็นต้องใช้ รหัส IFS (IFS code) ซึ่งเป็นโดเมนของการแปลง การใช้โดเมนแบบ local จะเป็นทฤษฎีที่ค่อนข้างยุ่งยากแต่ง่ายแก่การนำไปใช้ สำหรับรูปแบบการแปลงนั้นจะใช้รูปแบบการแปลงแบบแอฟฟีน (Affine Symmetry Transformation)

2.9.1 ลักษณะทั่วไปของวิธีการลดขนาดข้อมูลภาพแบบแฟรคตอล (Fractal Image Compression)

มีองค์ประกอบพื้นฐานดังนี้

1. มี โมเดล Y สำหรับ สเปซ $\Re$ ของภาพจริงๆบนโลก (Real World Images) จุดใน Y แทนด้วย ภาพจริงๆ บนโลก ซึ่ง สับพอร์ต □ อยู่ มีคุณสมบัติสีและ คุณสมบัติอื่นๆอีก เช่น Y เป็น สเปซของเซต (space of set) เป็น สเปซของฟังก์ชัน (function space) และ สเปซของการวัด (measures space)

2. มีเมทริกซ์ d บน Y ที่ซึ่ง (Y,d) เป็น เมทริกซ์สเปซสมบูรณ์

3. ให้ O เป็นตัวดำเนินการ (Operator) แบบย่นย่อ และกระทำบน (Y,d) โดยมีจำนวนจริง s ที่ $0<s<1$ และ $d(O(\varnothing),O(\psi)) \leq s \cdot d(\varnothing,\psi)$ สำหรับ $\varnothing,\psi \in Y$ Operator O สร้างจากเซตจำกัดของฟังก์ชัน แบบย่นย่อ กระทำภายใต้ สเปซ □ และภายใต้คุณสมบัติสี ตัวอย่าง O คือ ตัวดำเนินการฮัทชิสัน (Hutchison Operator) แทนด้วย W

$$W = \cup w_i$$

ซึ่ง ตัวดำเนินการฮัทชิสัน กระทำบนภาพขาวดำ เมื่อ Y เป็น สเปซ $\Re_1$ และ d เป็น ระยะทางเฮาส์ดอร์ฟ องค์ประกอบ 1,2 และ 3 ที่กล่าวข้างต้นสามารถ สรุปรวมได้เป็น ทฤษฎีและการคาดหมายดังนี้

1. ทฤษฎี (การมี แอทแทรกเตอร์ อยู่จริง) มี O เป็น ตัวดำเนินการ แบบย่นย่อ และ Y เป็น เมทริกซ์สเปซสมบูรณ์ จะมีภาพ 1 ภาพแทนด้วย $\varnothing$ ซึ่ง $\varnothing \in Y$ และทำให้ $O(\varnothing)= \varnothing$

2. การคาดหมาย (คุณสมบัติ fractal ของ แอทแทรกเตอร์s) $\varnothing$ มีคุณสมบัติที่อิสระในการแปลง เนื่องจาก ตัวดำเนินการ O เป็นฟังก์ชันแบบย่นย่อ กล่าวคือ ถ้ารวมภาพที่ย่อขนาดแต่ละส่วนเข้าด้วยกันจะได้ลักษณะของภาพเหมือนกับภาพต้นฉบับเดิม (เกิดจากการลอกเลียนแบบตัวมันเองซ้ำๆกัน แล้วย่อขนาดลง) ลักษณะการย่อจะประกอบด้วยการย่อ พื้นที่ว่าง (spatial contractivity) การย่อความเข้ม การย่อขนาด (measure theoretic) ดังนั้น แอทแทรกเตอร์ $\varnothing$ มีคุณสมบัติแบบแฟรคตอล

3. ทฤษฎี (การคำนวณ แอทแทรกเตอร์) สามารถหาค่า $\varnothing$ ได้ ถ้า $\psi \in Y$ และเมื่อกระทำ ตัวดำเนินการ $\square$ บน ψ ซ้ำกัน ผลลัพธ์ที่ได้ จะลู่เข้าสู่ แอทแทรกเตอร์ $\varnothing$ นั่นคือ

$$\lim_{n\to\infty} O^{\infty} = \varnothing$$

มี C ที่เป็นค่าคงที่ซึ่งเป็นจำนวนจริง ซึ่งทำให้ $d(\varnothing_1,\varnothing_2) < C$ สำหรับ $\varnothing_1,\varnothing_2 \in Y$ จะมีค่าคาดเคลื่อนโดยประมาณ $d(O^{\infty}(\psi),\varnothing) \leq s^n C$

4. ทฤษฎี (การประมาณ ทฤษฎี collage ทั่วๆไป) ระยะทางระหว่าง $\psi \in Y$ และแอทแทรกเตอร์ $\varnothing$ ของ O มีขอบเขตที่ประมาณจาก $d(\psi,\varnothing) \leq \frac{d(\varphi,O(\varphi))}{(1-s)}$

เซต $O(\psi)$ เรียกว่า คอล์เลค (collage) โดยที่ระยะทาง $d(\psi,O(\psi))$ เรียกว่า ค่าผิดพลาดคอล์เลค (collage error) ทฤษฎีนี้กล่าวว่า ถ้าต้องการหาตัวดำเนินการ O ซึ่งมีแอทแทรกเตอร์ ที่เกิดจากการประมาณค่า ψ ควรจะเลือก O ที่กระทำลงบน ψ แล้วทำให้ ψ เปลี่ยนแปลงเล็กน้อย

2.9.2 ระบบฟังก์ชันย้ำแบบโลคอล (Local Iterated Function System)

ทฤษฎี local IFS เป็น ทฤษฎี IFS ที่ใช้สำหรับการแปลงภาพขาวดำ โดยอาศัยนิยามทางคณิตศาสตร์ดังนี้

นิยาม ให้ (X,d) เป็น เมทริกซ์สเปซคอมแพคท์ ให้ R เป็นสับเซตของ X ซึ่งไม่เป็นเซตว่าง ให้ $w : R \to X$ และ มีจำนวนจริง s ที่ $0 \leq s < 1$ ถ้า $d(w(x),w(y)) \leq s \cdot d(x,y)$ สำหรับ x,y ใน R แล้วจะเรียก w ว่า เป็นการจับคู่แบบย่นย่อเฉพาะที่ (local contraction mapping) บน (X,d) s เป็นตัวประกอบการย่อของ w (contractivity factor)

นิยาม ให้ (X,d) เป็น เมทริกซ์สเปซคอมแพคท์ และ ให้ $w_i : R_i \to X$ เป็นการจับคู่แบบ ย่อเฉพาะที่ บน (X,d) มี s_i เป็นตัวประกอบแบบย่อ สำหรับ $i = 1,2,...,N$ เมื่อ N เป็นจำนวนเต็มบวกที่จำกัด แล้ว $\{ w_i : R_i \to X : i = 1,2,...,N \}$
เรียกว่า ระบบฟังก์ชันย้ำแบบ Local IFS โดยที่ $s = \max \{ s_i : i = 1,2,...,N \}$ เป็นตัวประกอบแบบย่อของ local IFS

local IFS จะใช้กำหนดตัวดำเนินการแบบย่อบนสเปซของภาพ พิจารณาตัวอย่างตามรูปแบบต่อไปนี้ ให้ S แทน เซตของสับเซตทั้งหมดของ X จะกำหนด ตัวดำเนินการ $W_{local} : S \to S$ บน $W_{local}(B) = \bigcup_{i=1}^{N} w_i(R_i \cap B)$, $\forall B \in S$

ดังนั้น W_{local} จะเป็นตัวดำเนินการบนเฮาส์ดอร์ฟสเปซ $\mathcal{H}(X)$ ซึ่งประกอบด้วยสับเซตคอมแพคท์ของ X พบว่า W_{local} เป็นการย่อบน สับเซตคอมแพคท์ของ $\mathcal{H}(X)$ ด้วยตัวประกอบแบบย่อ s ที่สอดคล้องกับเมทริกซ์เฮาส์ดอร์ฟ จากองค์ประกอบพื้นฐานสำหรับระบบลดขนาดข้อมูลภาพแฟรคตอล เมื่อ W_{local} ถูกเลือกให้เป็นตัวดำเนินการ O ที่กระทำบนสเปซ $Y=\mathcal{H}(X)$ กล่าวคือ สับเซต A ของ X ซึ่งไม่เป็นเซตว่างเป็น แอทแทรกเตอร์ หรือ เซตที่ไม่เปลี่ยนแปลงของ local IFS ถ้า $W_{local}(A)=A$ local IFSอาจจะไม่มีแอทแทรกเตอร์ หรือ อาจจะมีตั้งแต่ 1แอทแทรกเตอร์ขึ้นไป ถ้า A และ B เป็น แอทแทรกเตอร์ ดังนั้น $A \cup B$ จะเป็น แอทแทรกเตอร์ ที่มีขนาดใหญ่กว่า A และ B โดยที่ $A \cup B$ อยู่ใน W_{local} ด้วย

ให้ $\{W_i : R_i \to X : i = 1,2,...,N\}$ เป็น local IFS ซึ่งเราจะสันนิษฐานว่าเซตของ R_i เป็น เซตคอมแพคท์ ดังนั้นสามารถกำหนดลำดับของสับเซตคอมแพคท์ของ X $\{A_n : n=0,1,2,...\}$ โดย

$$A_0 = X$$

$$A_n = \bigcup_{i=1}^{N} w_i(R_i \cap A_{n-1}) \qquad \text{สำหรับ } n= 1,2,3,..$$

คือ $A_0 \supset A_1 \supset A_2 \supset A_3 \supset ...$

มี $\{A_n : n=0,1,2,3,..\}$ เป็นลำดับที่ลดลงของ เซตคอมแพคท์ โดยเฉพาะมี เซตคอมแพคท์ $A \subset X$ ซึ่ง

$$\lim_{n\to\infty} A_n = A$$

และ

$$A = \bigcup_{i=1}^{N} w_i(R_i \cap A) = W_{local}(A)$$

ถ้า A ไม่เป็นเซตว่าง A จะเป็น แอทแทรกเตอร์ ที่มากที่สุดสำหรับ local IFS คุณสมบัติที่ A เป็นเซตว่างจะไม่เกิดขึ้น ถ้าสามารถหา เซตคอมแพคท์ B ที่ไม่เป็นเซตว่าง ซึ่งทำให้ $W_{local}(B) \supset B$ สิ่งที่จะเกิดขึ้นนี้จะเกิดขึ้น ถ้า $W_i(R_i) \subset R_i$ สำหรับทุก i ดังแสดงในรูปที่ 2.17

2.9.3 ทฤษฎี Collage สำหรับ Local IFS

กำหนดให้ภาพขาวดำแทนด้วยสับเซต $G \subset \square$ เมื่อต้องการสร้าง G ให้เป็น แบบจำลองของแฟรคตอล จะต้องสร้าง $\square$ ด้วยสี่เหลี่ยมเล็ก ดังนั้น G เกิดจากการอินเตอร์เซก D_i ดังแสดงในรูป ที่ 2.18 สำหรับค่า i แต่ละค่าที่ i = 1,2,3,....,N จะหา w_i ที่เป็นการแปลงแบบแอฟฟีนเฉพาะที่ $w_i : R_i \to \square$ ด้วยตัวประกอบแบบย่อ s ที่ซึ่ง

$$w_i(R_i) = D_i$$

และ

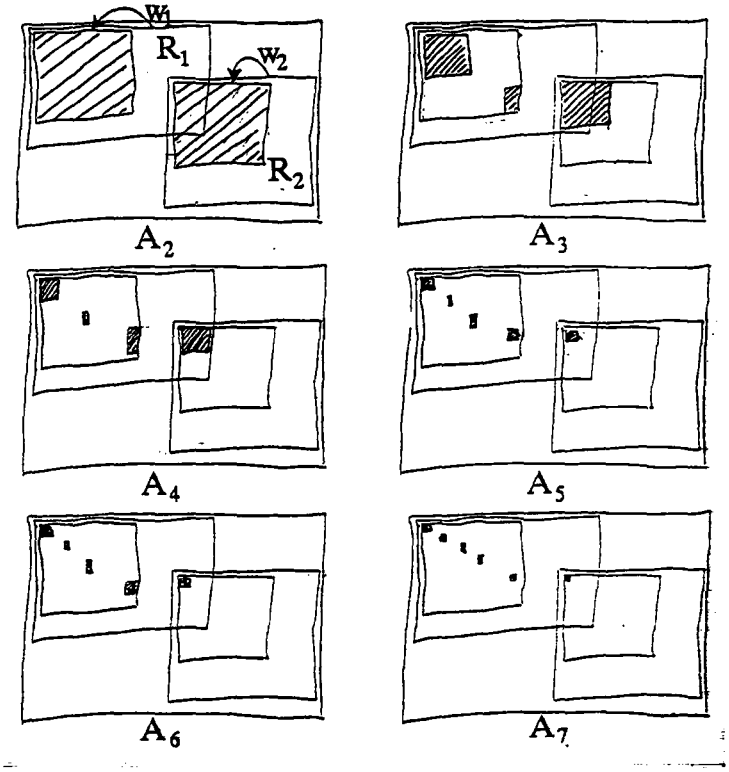
$$w_i(R_i \cap G) \cong D_i \cap G$$

เกณฑ์ในการประมาณความเหมือนจะใช้ ระยะทางเฮาส์ดอร์ฟ ระหว่าง $R_i \cap G_i$ และ $D_i \cap G_i$ ซึ่งจะมีค่าเล็กๆ กล่าวคือ $h(w_i(R_i \cap G), D_i \cap G) < \varepsilon$ สำหรับ i = 1,2,...,N ตัวอย่างการนำทฤษฎีบทนี้ไปใช้สำหรับการแปลง W ที่มีรูปแบบการแปลงดังสมการนี้

$$w(\cdot) = 0.5A \cdot + t$$

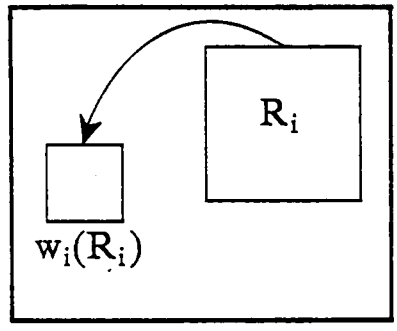
เมื่อ A : □→□ จะมีรูปแบบการแปลงแบบใดแบบหนึ่งดังแสดง ในตารางที่ 2.1

ให้ R_i เป็นบล็อคเล็กๆ ซึ่งมีขนาดเป็น 2 เท่าดังแสดงในรูปที่ 2.19 ถ้าในแต่ละบล็อค D_i มีขนาดเท่ากันสำหรับ i = 1,2,...,N แล้ว local IFS กำหนดด้วยคู่ลำดับ x,y สำหรับแต่ละค่าของ i มุมล่างซ้ายของ D_i แทนด้วย (D_x, D_y) และมุมล่างซ้าย R_i แทนด้วย (R_x, R_y) และ local IFS กำหนดด้วยจำนวนเต็มที่แทนด้วยตัวเลือกสำหรับการแปลงแต่ละแบบ ดังแสดงในตารางที่2.1 จากการกำหนดข้างต้น Local IFS จะแสดงได้ในรูปที่ 2.19 แสดงการรหัส (*encode*) ภาพเต็ม และ แอทแทรกเตอร์

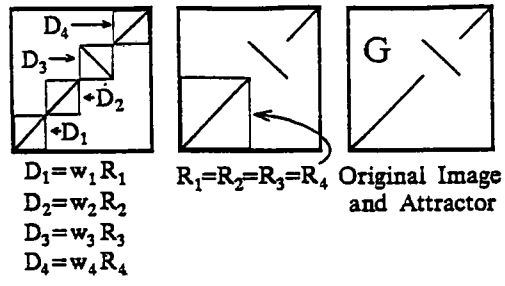


รูปที่ 2.17 การลู่เข้าของลำดับที่ลดลงของเซตไปยังภาพที่ต้องการ (แอทแทรกเตอร์) ของระบบฟังก์ชันย้ำแบบเฉพาะที่ (local IFS)ประกอบด้วยการแปลงแบบแอฟฟีนพื้นฐาน 2 แบบคือ

$w_1 : R_1 \rightarrow$ □ และ $w_2 : R_2 \rightarrow$ □



รูปที่ 2.18 แสดงบริเวณ $R_i \subset \square$ และภาพที่ผ่านการแปลง $w_i(R_i)$ ใน local IFS



Map#	D_x	D_y	R_x	R_y	Symmetry
1	0	0	0	0	0
2	4	4	0	0	0
3	8	8	0	0	2
4	12	12	0	0	0

รูปที่ 2.19 แสดงภาพที่ต้องการซึ่งสอดคล้องกับการแปลงใน local IFS (LIFS) และแสดงการกำหนดบล็อค D_i และ R_i และแอทแทรคเตอร์ (แอทแทรกเตอร์)

Symmetry	Matrix	Description
0	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	identity
1	$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$	reflection in y-axis
2	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	reflection in x-axis
3	$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$	180° rotation
4	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	reflection in line $y = x$
5	$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$	90° rotation
6	$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$	270° rotation
7	$\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$	reflection in line $y = -x$

ตารางที่ 2.1 รูปแบบต่าง ๆ ของการแปลงแอฟฟีนที่ขึ้นอยู่กับรหัส ซิมเมทรี

2.10 การนำทฤษฎีต่าง ๆ ไปใช้สำหรับลดขนาดข้อมูลภาพแบบแฟรคตอล

1. ศึกษาคุณสมบัติของโทโพโลยี เพื่อใช้อธิบายลำดับคอร์ชี ถ้าลำดับคอร์ชีมีการลู่เข้า สามารถหาภาพแบบแฟรคตอลได้
2. การแปลงภาพจะใช้ทฤษฎีดังต่อไปนี้
   - 2.1 การแปลงแบบแอฟฟีน (Affine transformation)
   - 2.2 การจับคู่แบบย่นย่อ (Contraction mapping)
   - 2.3 โครงสร้างการจับคู่แบบย่นย่อบน เฮาส์ดอร์ฟ สเปซเพื่อใช้อธิบาย IFS
3. การลดขนาดข้อมูล (Compression)

   ใช้คอมแพ็คสับเซตบน R^2 เพื่อหาตัวประมาณแฟรคตอล(Fractal approximation) โดยใช้ทฤษฎีภาพปะติด (Collage Theorem)
4. การขยายขนาดข้อมูล (Decompress)

   ให้ตัวประมาณแฟรคตอลเป็น อินพุตโดยผ่านอัลกอริทึมแล้วจะได้เอาท์พุทเป็นภาพแอทแทรกเตอร์

2.11 ตัวอย่างการแปลงภาพขาว-ดำ

ให้ $D = \cup D_i$ กำหนดให้ $f : D \rightarrow \square$

โดย

$$f(x) = W_i^{-1}(x) \quad \forall x \in D_i \, , \, i = 1,2,3,..,N$$

จาก

$$f(D_i) = R_i$$

พบว่า

R_i แทนโดเมนของ W_i

D_i แทนเรนจ์ของ W_i

บริเวณ D_i เรียกว่า โดเมนบล็อค และบริเวณ R_i เรียกว่า เรนจ์บล็อค

ขั้นตอนการแปลงภาพขาว-ดำแบบแฟรคตอล

(พิจารณารูปที่ 2.20 ประกอบ)

1. อินพุตภาพขาว-ดำ G ซึ่งเป็นสับเซตของ $\square \subset R^2$
2. แบ่ง G ออกเป็นโดเมนบล็อค D_i โดยแต่ละ D_i จะต้องไม่เหลื่อมล้ำกัน และผลรวมของ D_i จะต้องคลุม G ทั้งหมด
3. กำหนดเรนจ์บล็อค R_i ซึ่งมีขนาดเป็น 2 เท่า ของ D_i และมีโคออร์ดิเนต (R_x,R_y) ที่มุมล่างซ้ายของแต่ละเรนจ์บล็อคซึ่งอยู่ในเซตจำกัด L กำหนดให้ $\mathcal{T}$ เป็นการย่อแบบ เฉพาะที่โดยใช้รูปแบบการแปลงแบบแอฟฟีนทำการจับคู่จากเรนจ์บล็อคไปยังโดเมนบล็อค นั่นคือสำหรับ $i = 1,2,3,...,N$

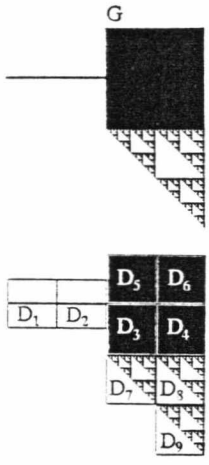
$$\mathcal{T}_i = \{W(D_i,R_x,R_y,J): (R_x,R_y) \in L_i, j=0,1,2,...,7\}$$

มี $W(D_i,R_x,R_y,J)$ เป็นการย่อโดยใช้รูปแบบแอฟฟีนด้วยโดเมน R เรนจ์ D_i ของรูปแบบ

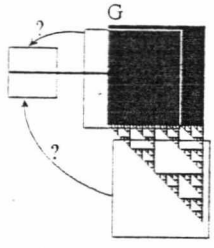
$$0.5A(j)+t$$

และ A(j) เป็นรูปแบบการแปลงขั้นที่ j ดังตารางที่ 2.2

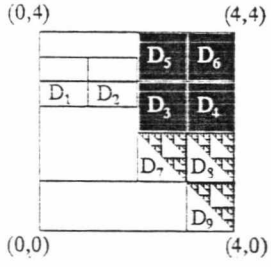
4. แต่ละ i เลือก $W_i \in \mathcal{T}_i$ ที่ทำให้ระยะทางเฮาส์ดอร์ฟน้อยที่สุด โดยพิจารณา D_i แต่ละบล็อคที่มีรูปแบบการแปลงที่เหมาะสม นั่นคือ จะแปลงส่วนของภาพในเรนจ์บล็อค ที่เหมือนกับส่วนของภาพใน D_i เซตของ $W_{local}(G) = \cup W_i(R \cap G)$ เรียกว่าคอล์เลจของภาพ G ที่มี $h(W_i(R \cap G),D_i \cap G)$ เรียกว่าความผิดพลาดคอล์เลจ (Collage Error)
5. บันทึกตำแหน่งของ D_i และ R_i ที่เหมาะสมลงในตารางรหัส IFS แบบโลคอล
6. นำรหัส local IFS ผ่านอัลกอลิทึม Lossless Compressed Data จะได้ รหัส local IFS ที่ลดขนาด(compress)แล้ว จะนำรหัส local IFS ที่ลดขนาดแล้ว มาทำการขยายขนาด (decompress)ต่อไป



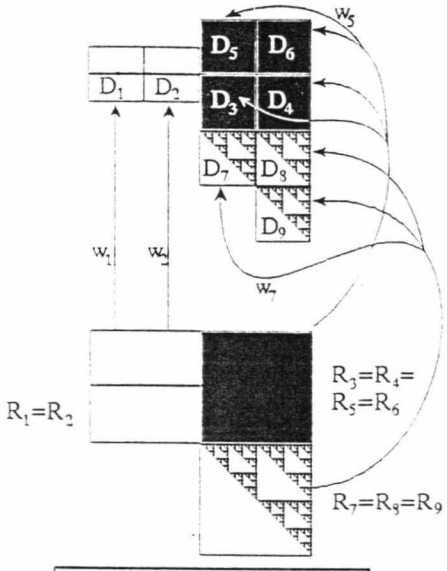
รูปที่ 2.20ก เซต G แทนภาพขาว-ดำที่ถูกแบ่งด้วยโดเมนบล็อคจตุรัส



รูปที่ 2.20ข การแปลงโดยใช้เรนจ์บล็อคเป็นตัวแทนภาพ G

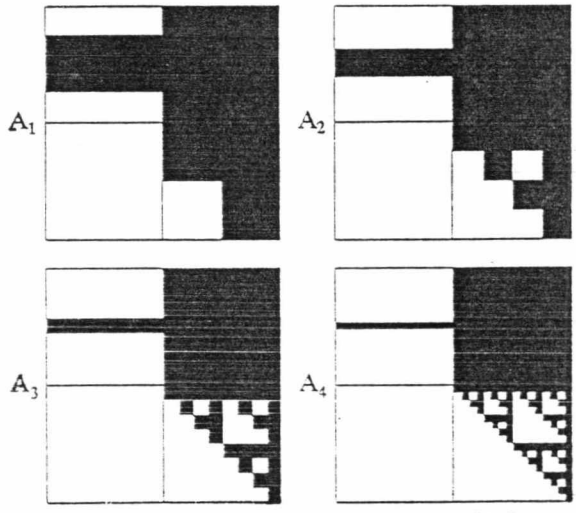


รูปที่ 2.20ค การเลือกโคออร์ดิเนต (x,y) ที่เหมาะสมสำหรับภาพ G



Map#	D_x	D_y	R_x	R_y	Symmetry
1	0	2.5	0	2	0
2	1	2.5	0	2	0
3	2	2	2	2	0
4	3	2	2	2	0
5	2	3	2	2	0
6	3	3	2	2	0
7	2	1	2	0	0
8	3	1	2	0	0
9	3	0	2	0	0

รูปที่ 2.20ง การเลือกโดเมนบล็อคซึ่งมีการแปลง $w_i(R_i)$ ที่เหมาะสมซึ่งทำให้ระยะทางเฮาส์ดอร์ฟมีค่าน้อยที่สุด โดยสร้างเป็นตารางรหัส IFS แบบเฉพาะที่



รูปที่ 2.20จ แสดงการนำรหัส IFS มาขยายกลับ (decompress) เพื่อให้ได้ภาพประมาณแบบแฟรคตอล

2.12 ตัวอย่างการแปลงภาพเกรย์สเกล

การแปลงแบบนี้จะใช้รูปแบบการแปลงแบบแอฟฟีนซึ่งมีรูปแบบดังต่อไปนี้

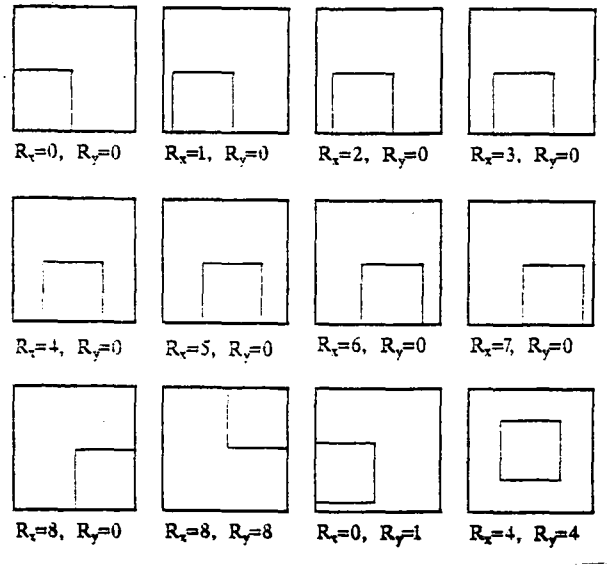
$$W_i = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & \rho \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} R_x \\ R_y \\ R_z \end{bmatrix}$$

มีสัมประสิทธิ์ a_i , b_i , c_i และ d_i เป็นการแปลงในระนาบ xy ตามรูปแบบการแปลง ในตารางที่ 2.1 ด้วยตัวประกอบแบบย่นย่อ 0.5 มีสัมประสิทธิ์ ρ เป็นจำนวนบวกที่กำหนดไว้ คือ $0 \leq \rho \leq s$ และ

$$V_i(z) = P_z + Q_i$$

ต่อไปจะเป็นตัวอย่างง่าย ๆ ของการแสดงว่าระบบการแปลงแบบแฟรกตอลนี้ทำงานอย่างไร โดยเริ่มจากการรวบรวมการแปลงที่เป็นไปตามเรนจ์บล็อคดังแสดงในรูปที่ 2.21 และการแบ่งขนาดโดเมนบล็อคดังแสดงในรูปที่ 2.22

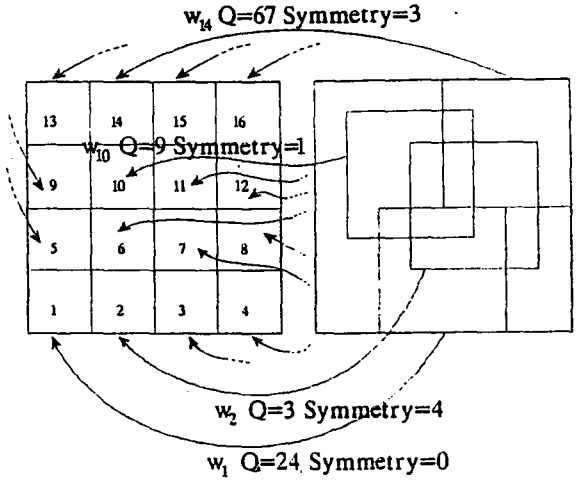
ให้ local IFS และตัวดำเนินการแปลงแบบแฟรกตอล F ซึ่งสอดคล้องกัน แทนด้วย และค่า และรูปแบบการแปลงที่สอดคล้องกับแต่ละโดเมนบล็อคซึ่งแสดงดังรูปที่ 2.23



รูปที่ 2.21 แสดงการกำหนดตำแหน่งเรนจ์บล็อค

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

รูปที่ 2.22 แสดงการแบ่งโดเมนบล็อค



รูปที่ 2.23 แสดงความสัมพันธ์ระหว่างเรนจ์บล็อคและโดเมนบล็อค เพื่อสร้างรหัสการแปลงแบบแฟรคตอล

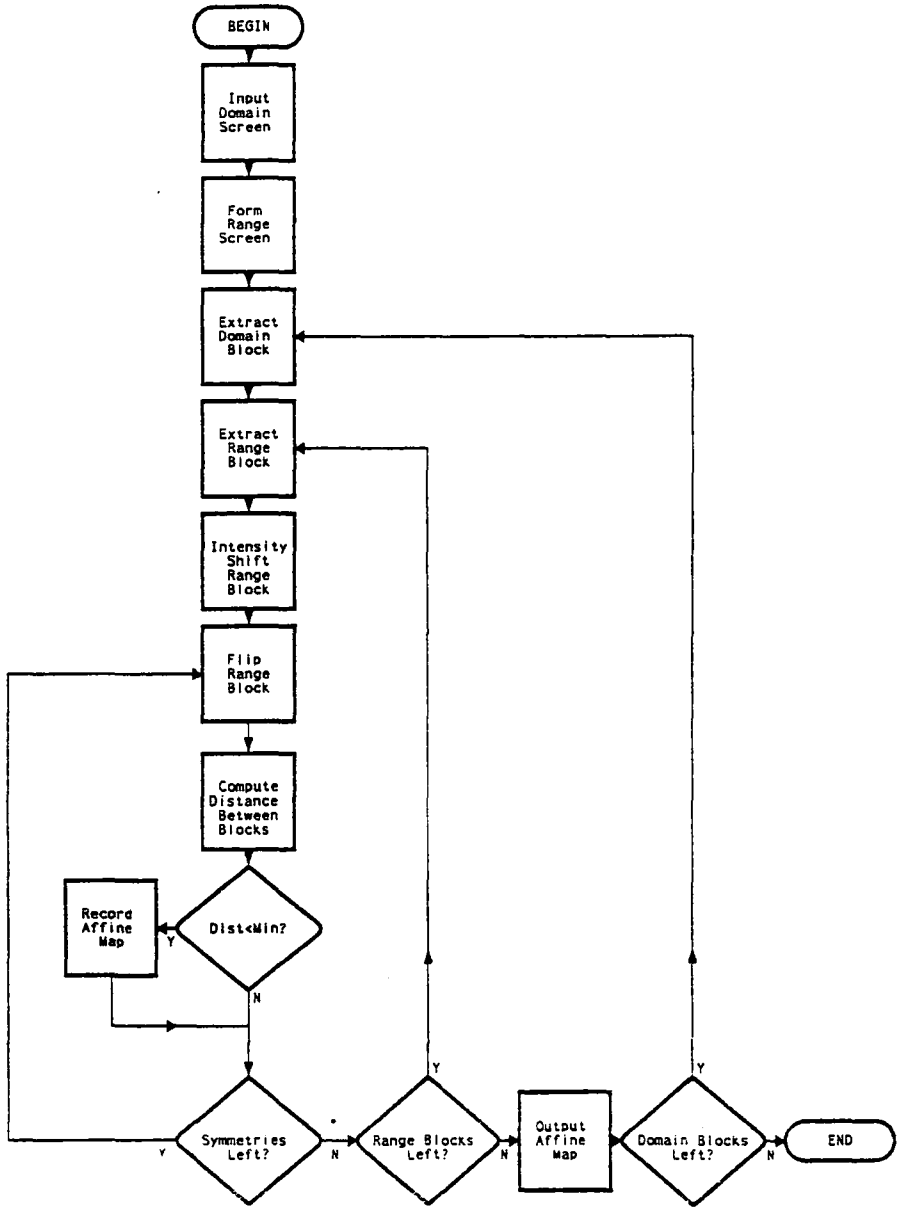
บทที่ 3
การพัฒนาโปรแกรมเพื่อการทดสอบ

3.1 โครงสร้างของโปรแกรม

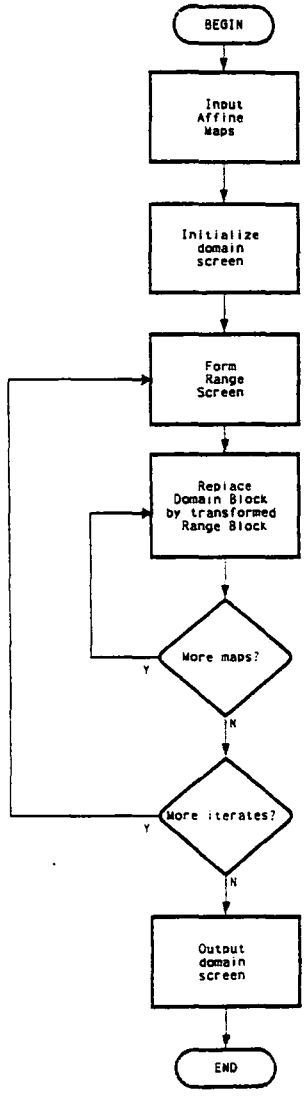
ส่วนประกอบอัลกอริทึมหลักมี 2 ส่วนคือ

1. อัลกอริทึมลดขนาดข้อมูลภาพ (COMPRESSION)
2. อัลกอริทึมขยายขนาดข้อมูลภาพ (DECOMPRESSION)

3.1.1 ผังการทำงานของอัลกอริทึมลดขนาดข้อมูลภาพ (COMPRESSION)



3.1.2 ผังการทำงานของอัลกอริทึมขยายขนาดข้อมูลภาพ (DECOMPRESSION)



3.1.3 รายละเอียดของไฟล์และการเรียกใช้ฟังก์ชันในโปรแกรม

3.1.3.1 ส่วนประกอบไฟล์ต่างๆ ในโปรแกรมดังต่อไปนี้

1. **TGA.H** เป็นการกำหนดโครงสร้างของไฟล์รูปภาพที่มีนามสกุลเป็นจุดทีจีเอ

2. **FRACTAL.H** เป็นไฟล์ที่ใช้กำหนดโครงสร้างของตัวแปร และ ฟังก์ชันที่เรียกใช้ซึ่งประกอบไปด้วย

โครงสร้างของตัวแปร

1) โครงสร้างของ pixel ที่ประกาศเป็น unsigned char
2) โครงสร้างของ symmetry ที่ประกาศเป็น unsigned char
3) โครงสร้างของ rectangle
4) โครงสร้างของ affinemap
5) โครงสร้างของ imageheader

ฟังก์ชันที่เรียกใช้โดยโปรแกรม compress.c และ decompress.c

1) Pixel mean (Rectangle * rectangle)
2) long t_distance (Rectangle *rect1,Rectangle *rect2)
3) void copy_rectangle (Rectangle *src_rect,short src_x,short src_y, Rectangle *dest_rect ,short dest_x,short dest_y,short width,short height)
4) void reduce_image (Rectangle *src_rect, Rectangle *dest_rect)
5) void flip(Rectangle *range_block,Rectangle *transformed_range_block ,Symmetry symmetry)
6) void intensity_shift (Rectangle *rectangle ,short shift)
7) void read_tga_header (FILE *image_file ,ImageHeader *header)
8) void write_tga_header (FILE *image_file ,ImageHeader *header)

3. **UTIL.C** เป็นไฟล์ที่เก็บรายละเอียดของฟังก์ชันที่โปรแกรมหลักเรียกใช้ทั้งหมด ประกอบด้วยรายละเอียดดังต่อไปนี้

1) Pixel mean (Rectangle * rectangle) : ใช้หาตัวแทนภาพที่ดีที่สุด
2) void intensity_shift (Rectangle *rectangle ,short shift) : ปรับระดับความเข้มของแต่ละ pixel
3) long t_distance (Rectangle *rect1,Rectangle *rect2) : ใช้หาระยะทาง
4) void copy_rectangle (Rectangle *src_rect,short src_x,short src_y, Rectangle *dest_rect ,short dest_x,short dest_y,short width ,short height) : สอกเลียนแบบข้อมูลขึ้นมาอีก 1ชุด

5) void reduce_image (Rectangle *src_rect, Rectangle *dest_rect) : ลดขนาดภาพอินพุตให้เหลือ 1/4 เท่าของภาพ

6) void flip (Rectangle *range_block,Rectangle *transformed_range_block, Symmetry symmetry) : การแปลงภาพ (transform) ตามรูปแบบซิมเมทรี

7) void read_tga_header (FILE *image_file ,ImageHeader *header) : ใช้อ่าน header ไฟล์ที่มีนามสกุล . tga

8) void write_tga_header (FILE *image_file ,ImageHeader *header) : ใช้เขียน header ไฟล์ที่มีนามสกุล . tga

3.1.3.2 ฟังก์ชันที่เรียกใช้ในแต่ละขั้นตอนการดำเนินงานโฟล์วชาร์ต

การเรียกใช้ฟังก์ชันของโฟล์วชาร์ตสำหรับการลดขนาดข้อมูลภาพ (COMPRESSION)

1) Input Domain Screen

```
image.width  = header.width;
image.height = header.height;
image.length = header.width * header.height;
image.pixel  = ( Pixel * ) calloc ( reduced_image.length
               ,sizeof(pixel) );
```

2) Form Range Screen

```
reduced_image.width  = header.width/2;
reduced_image.height = header.height/2;
reduced_image.length = header.width * header.height/4;
reduced_image.pixel  = ( Pixel * ) calloc ( reduced_image.length
                       ,sizeof( pixel) );
```

3) Extract Domain Block

```
copy_rectangle (&image,&domain_x,domain_y,&domain_block,0,0,
                DB_SIDE , DB_SIDE );
```

4) Extract Range Block

```
copy_rectangle (&reduced_image,current_range_x,current_range_y,
                &range_block,0,0,DB_SIDE,DB_SIDE );
```

5) Intensity Shift Range Block

```
intensity_shift  ( &range_block,current_shift );
```

6) Flip Range Block

```
flip ( &range_block,&flipped_range_block,current_symmetry );
```

7) Commute Distance Between Blocks

```
current_distance = t_distance (&domain_block,&filpped_range_
                    block );
```

8) Record Affine Map

```
best_map.shift = current_shift;
best_map.symmetry = current_symmetry;
best_map.range_x = current_range_x;
best_map.range_y = current_range_y;
```

9) Output Affine Map

```
fprintf ( fractal_file,"\n%d %d %d %d \n",best_map.range_x,
     best_map.range_y, best_map.symmetry,best_map.shift );
```

3.1.3.3 การเรียกใช้ฟังก์ชันของโฟล์วชาร์ตสำหรับการขยายขนาดข้อมูลภาพ (DECOMPRESSION)

1) Input Affine Maps

```
affine_map_array = ( AffineMap * ) calloc (number_of_maps
                    ,sizeof(AffineMap) );
```

2) Form Range Screen

```
copy_rectangle ( &reduced_image,map_ptr→range_x,map_ptr→
                    range_y,&range_block,0,0,DB_SIDE,DB_SIDE );
```

3) Replace Domain Block by transformed Range Block

```
copy_rectangle ( &transformed_range_block,0,0,&image,
                    domain_x,domain_y, DB_SIDE,DB_SIDE );
```

3.2 วิธีการและขั้นตอนการพัฒนาโปรแกรม

ปัญหาพิเศษเรื่องนี้ทำการพัฒนาโปรแกรม เพื่อการทดสอบอัลกอริทึมที่ใช้ในการลดและ ขยายข้อมูลรูปภาพกลับโดยแปลงอัลกอริทึม เป็นโปรแกรมภาษาซี และใช้คอมไพเลอร์บอร์แลนด์ซี จากโปรแกรมจะพิจารณาวัดประสิทธิภาพของการลดขนาดข้อมูลภาพจากข้อมูลต่อไปนี้

- จำนวนไบต์ของข้อมูลภาพก่อนที่จะมีการลดขนาด
- จำนวนไบต์ของข้อมูลภาพหลังจากที่มีการลดขนาดแล้ว
- จำนวนเปอร์เซ็นต์ของการลดขนาดข้อมูลรูปภาพ โดยคำนวณจากผลต่างระหว่างขนาดข้อมูลภาพ(มีหน่วยเป็นไบต์)ก่อนการลดขนาด (ไฟล์นามสกุล จุดทีจีเอ) กับขนาดของข้อมูลภาพหลังการลดขนาด (ไฟล์นามสกุล จุดเอฟอาร์ซี) แล้วหารด้วย ขนาดข้อมูลภาพก่อนการลดขนาด จากนั้นคูณด้วย 100

- ผลของรูปภาพก่อนที่มีการลดขนาด
- ผลของรูปภาพหลังจากที่มีการลดขนาดแล้ว

3.2.1 วิธีการทดสอบอัลกอริทึม

ในการทดสอบอัลกอริทึมโดยใช้โปรแกรมนั้น มีขั้นตอนการทดสอบดังนี้

1. เตรียมข้อมูลรูปภาพเข้า โดยรูปภาพที่สามารถโหลดได้จากโปรแกรมต้องเป็นไฟล์รูปภาพที่มีนามสกุลเป็นจุด ทีจีเอ (.TGA)

2. ทำการลดขนาดข้อมูลภาพโดยใช้อัลกอริทึมการแปลงแบบแฟรคตอลจะได้ข้อมูลภาพที่ลดขนาดแล้วมีนามสกุลเป็นจุด เอฟอาร์ซี (.FRC)

3. ผลของการลดขนาดข้อมูลภาพสามารถดูประสิทธิภาพของแต่ละภาพได้โดยการคำนวณเปอร์เซนต์การลดขนาดข้อมูลภาพ

4. รูปภาพที่ได้หลังจากการลดขนาดข้อมูลแล้วนำมาผ่านอัลกอริทึมดีคอมเพรส (decompress) จะได้ไฟล์รูปภาพเป็นจุด ทีจีเอ (.TGA)

5. เปรียบเทียบคุณภาพไฟล์ทีจีเอที่เป็นอินพุตในข้อ 1.กับไฟล์ทีจีเอที่เป็นอินพุตในข้อ 4.

<u>หมายเหตุ</u> หลังจากเปรียบเทียบภาพในข้อ 5. แล้ว คุณภาพของภาพไม่เป็นที่พอใจให้กลับไปกำหนดขนาดโดเมนใหม่แล้วทดลองตั้งแต่ข้อ 2 - 5 ซ้ำ

3.2.2 การเรียกใช้โปรแกรมลดขนาดข้อมูลภาพและโปรแกรมขยายข้อมูลภาพ

3.2.2.1 โปรแกรมลดขนาดข้อมูลภาพ

อินพุต: เมื่อต้องการรันโปรแกรมส่วนนี้ต้องพิมพ์

```
compress [domain block size] image_file fractal_file
```

<u>ตัวอย่าง</u> เมื่อต้องการรันไฟล์รูปภาพชื่อ children.tga ต้องพิมพ์คำสั่งดังนี้

```
compress children.tga children.frc
```

โปรแกรมจะกำหนดขนาดโดเมนให้โดยอัตโนมัติ ปกติมีค่าเท่ากับ 8 แต่ถ้าหากผู้ใช้ต้องการกำหนดขนาดโดเมนเองจะต้องเปลี่ยนคำสั่งใหม่

<u>ตัวอย่าง</u> เมื่อต้องการรันไฟล์รูปภาพชื่อ children.tga และกำหนดขนาดโดเมนเท่ากับ 4 ต้องพิมพ์คำสั่งดังนี้

```
compress 4 children.tga children.frc
```

เอาท์พุต: หลังจากผ่านการลดขนาดข้อมูลภาพแล้วจะได้ไฟล์ที่เป็นรหัสย่อของข้อมูล นามสกุล เอฟ อาร์ ซี (.FRC)

3.2.2.2 โปรแกรมขยายขนาดข้อมูลภาพ

อินพุต: เมื่อต้องการรันโปรแกรมส่วนนี้ต้องพิมพ์

```
decomprs [num_iterates] fractal_file image_file
```

ตัวอย่าง เมื่อต้องการรันไฟล์ที่เป็นรหัสย่อของข้อมูล (children.frc) ให้กลับมาเป็นไฟล์รูปภาพที่มีนามสกุลที จี เอ (children1.tga) ต้องพิมพ์คำสั่งดังนี้

```
decomprs children.frc children1.tga
```

โปรแกรมจะกำหนดค่าจำนวนครั้งในการคำนวนซ้ำให้โดยอัตโนมัติ ปกติมีค่าเท่ากับ 256 แต่ถ้าหากผู้ใช้ต้องการความละเอียดในการคำนวนมากขึ้น จะต้องกำหนดค่าเอง

ตัวอย่าง

```
decomprs 32 children.frc children1.tga
```

เอาท์พุต: หลังจากผ่านการขยายขนาดข้อมูลภาพแล้วจะได้ไฟล์รูปภาพ นามสกุลที จี เอ (.TGA) ซึ่งภาพที่ได้จะต้องมีขนาดและลักษณะใกล้เคียงกับรูปต้นฉบับเดิม นั่นคือ ขนาดและลักษณะของไฟล์ children.tga จะต้องใกล้เคียงกับ children1.tga

3.3 ลักษณะข้อมูลไฟล์นามสกุลจุดเอฟอาร์ซี (.FRC)

ไฟล์นามสกุลจุดเอฟอาร์ซีเป็นผลที่เกิดขึ้นจากการพัฒนาโปรแกรม เพื่อลดขนาดข้อมูลภาพ (compression) ข้อมูลภายในไฟล์เป็นตัวเลขที่ประกอบกันเป็นเมตริกซ์ขนาด n แถว คูณ 4 หลัก

ตัวเลขทั้ง 4 ในหลักของเมตริกซ์คือ

หลักที่ 1 คือ best_map.shift แทนระยะการชิฟระหว่างโดเมนกับเรนจ์ของภาพ

หลักที่ 2 คือ best_map.symmetry แทนลักษณะการแปลงของเรนจ์ที่สัมพันธ์กับโดเมน

หลักที่ 3 คือ best_map.range_x แทนตำแหน่งล่างซ้ายของกรอบของเรนจ์ในแนวแกน x

หลักที่ 4 คือ best_map.range_y แทนตำแหน่งล่างซ้ายของกรอบของเรนจ์ในแนวแกน y

และจำนวน n แถวในเมตริกซ์ มีผลจากความสัมพันธ์ของการกำหนดขนาดของโดเมนในโปรแกรมลดขนาดข้อมูลกับขนาดของภาพที่นำมาใช้งาน หากกำหนดขนาดโดเมนให้มีขนาดเล็กจะได้ค่า n ในเมตริกซ์มากกว่าเมื่อกำหนดขนาดโดเมนให้มีขนาดใหญ่

ดังนั้นตัวเลขในเมตริกซ์จะเป็นตัวแทนทั้งหมดของภาพที่นำมาลดขนาดซึ่งก็คือรหัสการแปลงแบบแอฟฟีนเมื่ออ้างอิงถึงทฤษฎีการแปลงแบบแฟรคตอล

ไฟล์นามสกุลจุดเอฟอาร์ซี ก็คือรหัสการแปลงของระบบฟังก์ชั่นย้ำ (IFS code)นั่นเอง และเมื่อต้องการขยายขนาดข้อมูลภาพกลับคืนก็จะนำไฟล์นี้ไปคำนวนในโปรแกรมขยายขนาดข้อมูลภาพ (decompression) ผลจากการคำนวนจะได้ภาพที่มีลักษณะใกล้เคียงกับภาพที่นำมาลดขนาด

<u>หมายเหตุ</u> จำนวนไบต์/เร็คคอร์ด= ขนาดของbest_map.shift + ขนาดของ best_map.symmetry
+ขนาดของbest_map.range_x+ ขนาดของ best_map.range_y
= 2+1+2+2
= 7

บทที่ 4
การทดสอบประสิทธิภาพของโปรแกรม

หลักเกณฑ์ที่ใช้ในการเปรียบเทียบคือเปอร์เซนต์ในการลดขนาดข้อมูลและคุณภาพของข้อมูลภาพจากการทดลองโปรแกรม

4.1 ผลการทดสอบประสิทธิภาพของโปรแกรม

ภาพแบบเกรย์สเกลที่นำมาทดลองจำนวน 10 รูป ทุกรูปมีขนาด 17,170 ไบต์ หลังจากนำไปลดขนาดข้อมูลและพิจารณาคุณภาพของข้อมูลหลังการขยายข้อมูลกลับว่าอยู่ในระดับที่น่าพอใจแล้วจึงบันทึกรายละเอียดได้ดังต่อไปนี้

รูปที่	ขนาดโดเมนบล็อค	ขนาดไฟล์จุดเอฟอาร์ซี	เปอร์เซนต์การลดขนาด
1	8	2,580	84.97
2	8	2,580	84.97
3	8	2,580	84.97
4	8	2,580	84.97
5	8	2,580	84.97
6	8	2,580	84.97
7	4	7,956	53.66
8	4	7,956	53.66
9	4	7,956	53.66
10	4	7,956	53.66

ตัวอย่างการคำนวณเปอร์เซนต์การลดขนาดข้อมูล

จากรูปที่ 1 ขนาดไฟล์เดิมมีค่าเท่ากับ 17,170 ไบต์ เมื่อนำมาผ่านโปรแกรมลดขนาดข้อมูลแล้วพบว่าตัวแทนข้อมูลที่ถูกจัดเก็บมีขนาดไฟล์เพียง 2,580 ไบต์ นำมาคำนวณเปอร์เซนต์การลดขนาดได้ดังนี้

เปอร์เซนต์การลดขนาด = (ขนาดไฟล์จุดทีจีเอ - ขนาดไฟล์จุดเอฟอาร์ซี)/ขนาดไฟล์จุดทีจีเอ * 100

เปอร์เซนต์การลดขนาด = (17,170 - 2,580)/17,170 * 100

= 84.97

หมายเหตุ ไฟล์รูปภาพแบบเกรย์สเกลจะมีส่วนประกอบ 3 ส่วนได้แก่

1) ส่วนหัว (header)
2) ส่วนจำแนกสี (palette)
3) ส่วนข้อมูล (data)

การคำนวณขนาดไฟล์จุดเอฟอาร์ซีอาจทำได้โดย

1) หาจำนวนโดเมนบล็อคทั้งหมดนั่นคือ

จำนวนโดเมนบล็อคทั้งหมด (TD) $= \frac{\text{ความกว้างภาพ}}{\text{ขนาดโดเมน}} * \frac{\text{ความยาวภาพ}}{\text{ขนาดโดเมน}}$

$= \frac{\text{image.width}}{\text{DB\_size}} * \frac{\text{image.height}}{\text{DB\_size}}$

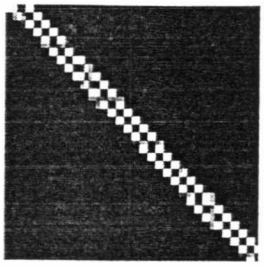
2) หาจำนวนไบต์/เร็คคอร์ดในไฟล์จุดเอฟอาร์ซี (A)
จำนวนไบต์/เร็คคอร์ดในไฟล์จุดเอฟอาร์ซี = 7

3) หาผลรวมของขนาดโดเมนบล็อค, ขนาดของส่วนหัวและขนาดของส่วนจำแนกสี (B)
ขนาดโดเมนบล็อค+(ขนาดของส่วนหัว + ขนาดของส่วนจำแนกสี) = 2+786
= 788

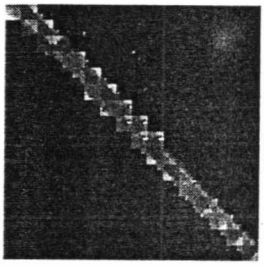
ดังนั้นขนาดไฟล์จุดเอฟอาร์ซี = (TD * A)+ B

ตัวอย่างเช่น ขนาดไฟล์จุดเอฟอาร์ซีของรูปที่ 1 = (16*16*7)+ 788
= 1792 + 788
= 2580

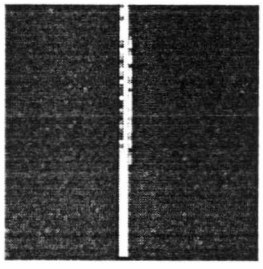
4.2 เปรียบเทียบคุณภาพของข้อมูลภาพเดิมกับข้อมูลภาพที่ผ่านโปรแกรมขยายขนาด



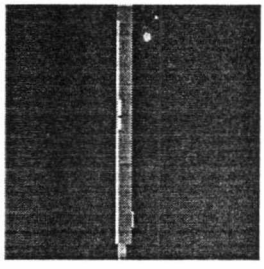
รูปที่ 4.1ก รูปที่1ก่อนการทดลอง



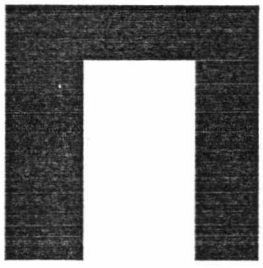
รูปที่ 4.1ข รูปที่1หลังการทดลอง



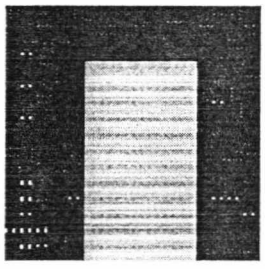
รูปที่ 4.2ก รูปที่2ก่อนการทดลอง



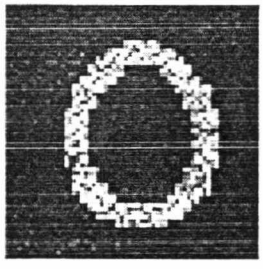
รูปที่ 4.2ข รูปที่2หลังการทดลอง



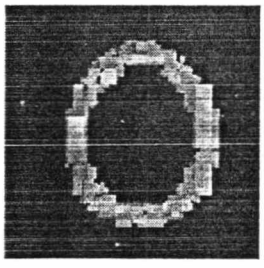
รูปที่ 4.3ก รูปที่3ก่อนการทดลอง



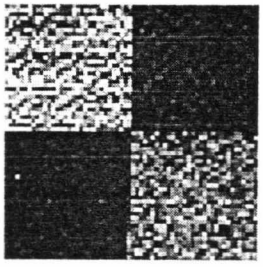
รูปที่ 4.3ข รูปที่3หลังการทดลอง



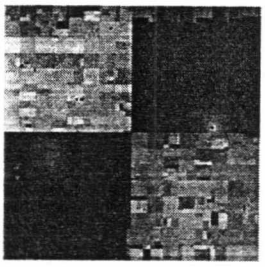
รูปที่ 4.4ก รูปที่4ก่อนการทดลอง



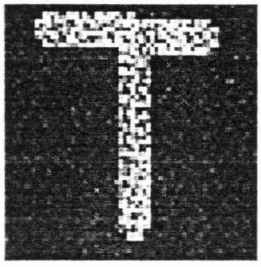
รูปที่ 4.4ข รูปที่4หลังการทดลอง



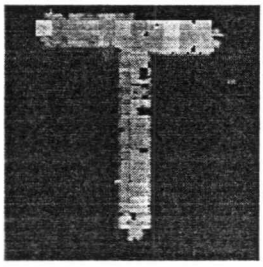
รูปที่ 4.5ก รูปที่5ก่อนการทดลอง



รูปที่ 4.5ข รูปที่5หลังการทดลอง

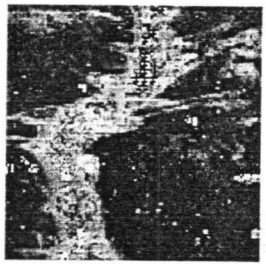


รูปที่ 4.6ก รูปที่6ก่อนการทดลอง



รูปที่ 4.6ข รูปที่6หลังการทดลอง

รูปที่ 4.7ก รูปที่7ก่อนการทดลอง



รูปที่ 4.7ข รูปที่7หลังการทดลอง

รูปที่ 4.8ก รูปที่8ก่อนการทดลอง

รูปที่ 4.8ข รูปที่8หลังการทดลอง

รูปที่ 4.9ก รูปที่9ก่อนการทดลอง

รูปที่ 4.9ข รูปที่9หลังการทดลอง

รูปที่ 4.10ก รูปที่10ก่อนการทดลอง

รูปที่ 4.10ข รูปที่10หลังการทดลอง

4.3 ข้อสังเกตที่ได้จากการทดสอบประสิทธิภาพของโปรแกรม

1. ภาพจากขั้นตอนการขยายข้อมูลภาพกลับคืน (Decompression)มีลักษณะที่ไม่เหมือนข้อมูลเดิม ทั้งนี้เนื่องจากในขั้นตอนการลดข้อมูลภาพ (Compression) ข้อมูลส่วนใหญ่ไม่สามารถหาตัวแทนที่มีค่าเท่ากับข้อมูลเดิมได้ตัวแทนที่หาได้นั้นเป็นเพียงค่าประมาณเท่านั้น หากต้องการให้ภาพที่ได้จากการขยายข้อมูลกลับมีความใกล้เคียงมากขึ้นควรกำหนดขนาดโดเมนให้เล็กลง แต่ทั้งนี้ก็จะทำให้ใช้เวลานานเพิ่มขึ้นเนื่องจากเวลาที่ใช้ในการคำนวณ (Computer time)ขึ้นอยู่กับขนาดของภาพ ในขณะที่ภาพมีขนาดเท่าเดิมแต่ขนาดโดเมนเล็กลง จะทำให้โปรแกรมต้องพิจารณาพื้นที่ย่อย (subregion) ที่มีจำนวนมากขึ้น ดังนั้นจำนวนครั้งในการหาตัวแทนภาพก็เพิ่มขึ้นด้วย

2. ในขั้นตอนการขยายข้อมูลภาพกลับ จำนวนครั้งในการทำซ้ำ (iterations) เพื่อให้ได้ภาพเดิมขึ้นอยู่กับโครงสร้างของภาพ และขอบเขตต่ำสุดของค่าความผิดพลาดที่ยอมให้เกิดขึ้นในขั้นตอนการลดข้อมูลภาพ โดยปกติเวลาที่ใช้ในการขยายข้อมูลกลับจะน้อยกว่าการลดข้อมูล และถ้าหากกำหนดค่าของจำนวนครั้งในการทำซ้ำให้กับพารามิเตอร์ในขั้นตอนการขยายข้อมูลภาพกลับจะทำให้เวลาการคำนวณเพิ่มขึ้น

บทที่ 5
สรุปและข้อเสนอแนะ

สรุป

ปัญหาพิเศษเรื่องการลดขนาดข้อมูลภาพโดยใช้การแปลงแบบแฟรคตอลนี้ เป็นการศึกษาถึงลักษณะการลดขนาดข้อมูลภาพและการขยายข้อมูลภาพกลับโดยใช้ทฤษฎีทางคณิตศาสตร์ ได้แก่ ทฤษฎีการแปลงภาพแบบแฟรคตอล, ทฤษฎีฟังก์ชันซ้ำแบบเฉพาะที่, ทฤษฎีภาพปะติด, ทฤษฎีการจับคู่แบบย่นย่อ ผลจากการศึกษาทฤษฎีต่างๆและนำมาพัฒนาเป็นโปรแกรมโดยใช้ภาษาซีนั้นจะประกอบด้วยโปรแกรมลดขนาดข้อมูลภาพ(Compression)และโปรแกรมขยายข้อมูลภาพ(Decompression) หลังจากนั้นนำโปรแกรมนี้ไปทดลองกับภาพแบบเกรย์สเกลที่มีโครงสร้างของภาพแตกต่างกันจำนวน 10รูป สำหรับผลที่ได้จากการทดลองโปรแกรมสามารถสรุปได้ว่าเพื่อที่จะได้ภาพที่มีความใกล้เคียงกับภาพต้นฉบับมากที่สุด และประหยัดเวลาในการประมวลผลภาพที่มีโครงสร้างไม่ซับซ้อนควรจะกำหนดขนาดของโดเมนบล็อคให้มีขนาดใหญ่กว่าภาพที่มีโครงสร้างซับซ้อน หลังจากนั้นทำการคำนวณเปอร์เซนต์การลดขนาดในแต่ละภาพ

ปัญหาในการใช้งาน

1. หากนำภาพที่มีขนาดใหญ่มาลดขนาดข้อมูลจะต้องใช้เวลาในการประมวลผลนาน
2. เนื่องจากเปอร์เซนต์การลดขนาดข้อมูลมีสาเหตุจากหลายปัจจัย อาทิ
   - ความซับซ้อนของโครงสร้างภาพ
   - ขนาดของโดเมนบล็อคที่ต้องกำหนดให้สัมพันธ์กับขนาดของภาพ
   - มาตรฐานของระดับความพอใจที่แตกต่างกัน

ดังนั้นจึงไม่สามารถกำหนดเปอร์เซนต์เฉลี่ยการลดขนาดข้อมูลสำหรับโปรแกรมนี้ได้

ข้อเสนอแนะ

ปัญหาพิเศษเรื่องการลดขนาดข้อมูลภาพโดยใช้การแปลงภาพแบบแฟรคตอลนี้เป็นการศึกษาลักษณะการลดขนาดข้อมูลภาพและการขยายข้อมูลภาพกลับแบบเกรย์สเกล ซึ่งแท้จริงแล้วเทคนิคการแปลงแบบแฟรคตอลสามารถนำมาใช้กับภาพสีได้และอัลกอริทึมที่ใช้อาจเพิ่มประสิทธิภาพที่ดีกว่าได้ ดังนั้นจึงขอเสนอให้ผู้ที่มีความสนใจในปัญหาพิเศษนี้ ศึกษาและพัฒนาโปรแกรมดังต่อไปนี้

1.เพิ่มประสิทธิภาพอัลกอริทึมที่ใช้ในการลดขนาดข้อมูล

1.1 อาจทำได้โดยเปลี่ยนแปลงรูปร่างของโดเมนที่กำหนดแบบสี่เหลี่ยมจตุรัสให้เป็นรูปเหลี่ยมผืนผ้า หรือรูปทรงที่ไม่ใช่สี่เหลี่ยมเช่น สามเหลี่ยม หรือรูปทรงอื่น ๆ ที่เห็นว่าเหมาะสม

1.2 เพิ่มความละเอียดของตัวเลขที่ใช้ในการคำนวณ

2.ศึกษารายละเอียดของภาพสีและหลักการจับคู่แบบย่นย่อ(Contraction Mapping Principle) เพื่อนำมาพัฒนาโปรแกรมสำหรับลดขนาดข้อมูลภาพสี

ภาคผนวก

ภาคผนวก

รายละเอียดของลักษณะไฟล์แบบทีจีเอ (TGA)

ไฟล์แบบทีจีเอนี้เป็นไฟล์ประเภทบิตแม็พที่กำหนดขึ้นสำหรับการเก็บภาพคุณภาพสูง ลักษณะไฟล์จะประกอบด้วยส่วนหัว (header) และข้อมูลภาพข้อมูลส่วนหัวของแฟ้มมีอยู่ 18 ไบต์(แบ่งได้เป็น 12 ฟิลด์)มีรายละเอียดดังนี้

ออฟเซต (Offset)	ระยะ (Length)	รายละเอียด (Description)
0	1	ID filed length
1	1	Color map type
2	1	Image type
3	2	First color map entry
5	2	Color map entry size
7	1	Color map entry size
8	2	Image X origin
10	2	Image Y origin
12	2	Image width
14	2	Image height
16	1	Bits per pixel
17	1	Image descripter bits

รายละเอียดข้อมูลตั้งแต่ออฟเซ็ต 0 – 17

ID filed length

ใช้เป็นตัวบอกว่าจะมีข้อความอธิบายรายละเอียดของภาพหรือไม่ ถ้าไม่มีจะมีค่าเป็น 0 ถ้ามีค่าตัวเลขของไบต์ (byte) นี้จะเป็นตัวบอกว่าจะมีจำนวนยาวเท่าใด (มากสุดคือ 255 ไบต์)เริ่มนับต่อจาก header คือเริ่มไบต์ที่มีออฟเซต (offset) 18

Color map images

ค่าสีของแต่ละพิกเซล (pixel)ของภาพต้องอ่านผ่านตารางสีที่ติดมากับไฟล์ ก่อนแสดงบนจอภาพ True color images

Image Type

ส่วนนี้จะมีได้อีกค่า พร้อมความหมายดังนี้

รหัส	ความหมาย
0	No Image present
1	Color-mapped,uncompressed
2	True color,uncompressed
3	Black and white ,uncompressed
9	Color-mapped,RLE compressed
10	True color,RLE compressed
11	Black and white,RLE compressed

ในอนาคตอาจมีค่าเพิ่มมากกว่านี้ถ้าใช้วิธีอัดข้อมูลแบบอื่น

Color Images

ค่าสีของแต่ละพิกเซล (pixel)อ่านได้จากข้อมูลในไฟล์โดยตรง ไม่ต้องใช้ตารางสี Black and white Images ไฟล์ที่เก็บเฉพาะค่าสีเทาที่ระดับต่าง ๆของแต่ละพิกเซล (pixel)

Color map Type

ส่วนนี้ถ้ามีค่าเป็น 1 แสดงว่ามีแบบสี (color map)ติดมาด้วย ถ้ามีค่าเป็น 0 แสดงว่าไม่มีฟิลด์ของแบบสีนี้ ปัจจุบันส่วนใหญ่จะกำหนดค่าเป็น 0 ไม่มีความจำเป็นต้องใช้แบบสีอีกต่อไป เนื่องจากการ์ดแสดงผลมีความสามารถมากขึ้น นอกจากไฟล์ Targaรุ่นเก่าจึงมีค่าเป็น 1

First color map entry, Color map length และ Color map entry size จะใช้ร่วมกับ color map ปัจจุบันโอกาสพบน้อยมากไม่กล่าวถึงในที่นี้

Image X origin

ตำแหน่งเริ่มต้นทางแกน x บนจอภาพ

Image y origin

ตำแหน่งเริ่มต้นทางแกน y บนจอภาพ

Image width

ความกว้างของภาพ (หน่วยเป็น pixel)

Image height

ความสูงของภาพ (หน่วยเป็น pixel)

Bits per pixel

จำนวนบิตต่อ pixel ปกติจะมีค่าเป็น 8,16,24 หรือ 32

Image descripter bits

แยกย่อยออกได้อีกหลาย field คือ

4 บิตต่ำ (low four bit) บอกจำนวนลักษณะเฉพาะของบิต(bit attribute) ต่อจุดภาพ (pixel)

2 บิตต่อมาบอกทิศทางการเก็บของภาพ

ถ้ามีค่าเป็น 00 (Hex) หมายถึงเรียงจากซ้ายไปขวา ล่างขึ้นบน

ถ้ามีค่าเป็น 10 (Hex) หมายถึงเรียงจากขวาไปซ้าย ล่างขึ้นบน

ถ้ามีค่าเป็น 20 (Hex) หมายถึงเรียงจากซ้ายไปขวา บนลงล่าง

ถ้ามีค่าเป็น 30 (Hex) หมายถึงเรียงจากขวาไปซ้าย บนลงล่าง

2 บิตต่อมาบอกเกี่ยวกับสแกนไลน์ (scan line) ซึ่งจะสัมพันธ์กับคุณสมบัติของจอภาพ

ถ้ามีค่าเป็น 00 (Hex) หมายถึง noninterleave

ถ้ามีค่าเป็น 40 (Hex) หมายถึง การ scan แบบ two way even-odd interleave

ถ้ามีค่าเป็น 80 (Hex) หมายถึงการ scan แบบ four way interleave

โปรแกรมประกอบ
ปัญหาพิเศษ

FILE TGA.H

```
#define     TGA_GRAYSCALE     1

struct      tga_hdr
            {
              unsigned char     id,cmaptype,imtype,col1,col2,col3,col4,col5;
              short             xorigin,yorigin,width,height;
              unsigned char     depth,descriptor;
            };
```

FILE FRACTAL.H

```
#include      <stdio.h>
#include      "tga.h"

#define       FLIP_X   1
#define       FLIP_Y   2
#define       FLIP_DIAG  4
#define       ARBITRARY_PIXEL_VALUE   128

typedef       unsigned char          Pixel;
typedef       unsigned char          Symmetry;

typedef       struct rectangle
              {
                unsigned long        width,height;
                unsigned long        length;
                Pixel                *pixel;
              } Rectangle;

typedef       struct affinemap
              {
                unsigned short       range_x,range_y;
                short                shift;
                Symmetry             symmetry;
              } AffineMap;

typedef       struct imageheader
              {
                unsigned long        width,height;
              } ImageHeader;
```

```
extern  Pixel   mean(Rectangle *rectangle);
extern  long    t_distance(Rectangle *rect1,Rectangle *rect2);

extern  void    copy_rectangle(Rectangle *src_rect,short src_x,short src_y,
                               Rectangle *dest_rect,short dest_x,short dest_y,
                               short width,short height);

extern  void    reduce_image(Rectangle *src_rect,Rectangle *dest_rect);

extern  void    flip(Rectangle *range_block,Rectangle *transformed_range_block,
                     Symmetry symmetry);

extern  void    intensity_shift(Rectangle *rectangle,short shift);
```

FILE UTIL.C

```
#include    "fractal.h"

Pixel       mean(Rectangle *rectangle)
{
            int     i;
            long    sum=0;

            for (i=0;i<rectangle->length;i++)
              sum+=rectangle->pixel[i];
            return(sum/rectangle->length);
}           /* mean */

void        intensity_shift(Rectangle *rectangle,short shift)
{
            short   i;

            for (i=0;i<rectangle->length;i++)
              rectangle->pixel[i]=rectangle->pixel[i]+shift;
}           /* intensity_shift */

long        t_distance(Rectangle *rect1,Rectangle *rect2)
            /* rect1 and rect2 must have the same length */
{
            long    d,distance=0;
            int     i;

            for (i=0;i<rect1->length;i++)
            {
              d=rect1->pixel[i]-rect2->pixel[i];
              distance+=d*d;
            }
```

```
        return(distance);
}       /* t_distance */

void    copy_rectangle(Rectangle *src_rect,short src_x,short src_y,
                       Rectangle *dest_rect,short dest_x,short dest_y,
                       short width,short height)
{
        int     i,j;

        for (j=0;j<height;j++)
          for (i=0;i<width;i++)
            dest_rect->pixel[i+dest_x+(j+dest_y)*dest_rect->width]=
            src_rect->pixel[(src_x+i)+(src_y+j)*src_rect->width];
}       /* copy_rectangle */

void    reduce_image(Rectangle *src_rect,Rectangle *dest_rect)
{
        int     i,j;

        for (j=0;j<dest_rect->height;j++)
          for (i=0;i<dest_rect->width;i++)
          {
            /* spatial rescale by 2 */

            dest_rect->pixel[i+j*dest_rect->width]=
            (src_rect->pixel[2*i+(2*j)*src_rect->width]+
            src_rect->pixel[2*i+1+(2*j)*src_rect->width]+
            src_rect->pixel[2*i+(2*j+1)*src_rect->width]+
            src_rect->pixel[2*i+1+(2*j+1)*src_rect->width])/4;

            /* intensity rescale by 3/4 */
```

```
                dest_rect->pixel[i+j*dest_rect->width]=
                (dest_rect->pixel[i+j*dest_rect->width]*3)/4;
              }
}             /* reduce_image */

void          flip(Rectangle *range_block,Rectangle *transformed_range_block,
                  Symmetry symmetry)
{
              short   i,j,x,y,t;

              for (j=0;j<range_block->height;j++)
               for (i=0;i<range_block->width;i++)
               {
                if (symmetry & FLIP_X) x=(range_block->width-1)-i;
                else
                 x=i;

                if (symmetry & FLIP_Y) y=(range_block->height-1)-j;
                else
                 y=j;

                /* not allowed unless with=height */

                if (symmetry & FLIP_DIAG)
                {
                 t=y;
                 y=x;
                 x=t;
                }
                transformed_range_block->pixel[x+y*range_block->width]=
                range_block->pixel[i+j*range_block->width];
               }
}             /* flip */
```

FILE COMPRESS.C

```
#include    <math.h>
#include    <stdio.h>
#include    <conio.h>
#include    <alloc.h>
#include    <stdlib.h>
#include    "fractal.h"

main        (int argc,char **argv)
{
            int             i,x,y;
            ImageHeader     header;
            AffineMap       best_map;
            char            *pal=NULL;
            struct tga_hdr  tgaheader;
            Pixel           domain_mean;
            Symmetry        current_symmetry;
            FILE            *image_file,*fractal_file;
            unsigned long   loop,count=0,temp,nsyms=8,loop1,loop2,
                            current_distance,minimum_distance,infinity;
            short           current_range_x,current_range_y,domain_x,domain_y,
                            current_shift,arg_offset=0,db_side=8;
            Rectangle       image,domain_block,range_block,flipped_range_block,
                            reduced_image;


            if ((argc<3)||(argc>4))
            {
              fprintf(stderr,"\nUsage : compress [db_side] image_file
                    fractal_file.\n");
              exit(1);
            }
```

```
if (argc==4)
{
  db_side=atoi(argv[1]);
  arg_offset=1;
}
/* Step 0 : Allocate memory for blocks */

domain_block.width=range_block.width=flipped_range_block.width=
db_side;
domain_block.height=range_block.height=flipped_range_block.height=
db_side;
domain_block.length=range_block.length=flipped_range_block.length=
db_side*db_side;

domain_block.pixel=(Pixel *) calloc(db_side*db_side,sizeof(Pixel));
range_block.pixel=(Pixel *) calloc(db_side*db_side,sizeof(Pixel));
flipped_range_block.pixel=
(Pixel *)calloc(db_side*db_side,sizeof(Pixel));
infinity=255*255*db_side*db_side;

/* Step 1 : Input Image */

if (NULL==(image_file=fopen(argv[arg_offset+1],"rb")))
{
  fprintf(stderr,"\nUnable to open file %s.\n",argv[arg_offset+1]);
  exit(1);
}

if (NULL==(fractal_file=fopen(argv[arg_offset+2],"wb")))
{
  fprintf(stderr,"\nUnable to open file %s.\n",argv[arg_offset+2]);
  exit(1);
}
```

```
pal=(char *) calloc(768,sizeof(char));
if (1!=fread(&tgaheader,sizeof(struct tga_hdr),1,image_file))
{
  fprintf(stderr,"\nError reading targa header.\n");
  exit(1);
}

if ((tgaheader.imtype!=TGA_GRAYSCALE)||(tgaheader.depth!=8))
{
  fprintf(stderr,"\nInvalid targa file.\n");
  exit(1);
}

header.width=tgaheader.width;
header.height=tgaheader.height;
fwrite(&tgaheader,sizeof(struct tga_hdr),1,fractal_file);

if (1!=fread(pal,768,1,image_file))
{
  fprintf(stderr,"\nError reading targa palatte.\n");
  exit(1);
}
fwrite(pal,768,1,fractal_file);
fwrite(&db_side,2,1,fractal_file);

fprintf(stderr,"\nWidth %ld Height %ld\n",header.width,header.height);

image.width=header.width;
image.height=header.height;
image.length=header.width*header.height;
image.pixel=(Pixel *) calloc(image.length,sizeof(Pixel));
```

```
reduced_image.width=header.width/2;
reduced_image.height=header.height/2;
reduced_image.length=header.width*header.height/4;
reduced_image.pixel=
(Pixel *)calloc(reduced_image.length,sizeof(Pixel));

if (NULL==image.pixel)
{
 fprintf(stderr,"\nUnable to allocate %ld bytes for image
         buffer.\n",image.length);
 exit(2);
}

if (image.length!=fread(image.pixel,sizeof(Pixel),
    image.length,image_file))
{
 fprintf(stderr,"\nError reading header from image file %s.\n",argv[1]);
 exit(1);
}
fclose(image_file);

/* rescale(contract)image in spatial and */
/* intensity directions */

reduce_image(&image,&reduced_image);

/* MAIN LOOP */

loop1=(image.height/db_side)*(image.width/db_side);
loop2=(reduced_image.height+1-db_side)*(reduced_image.width+1-
        db_side);
loop=loop1*loop2*nsyms;
temp=loop/40;
```

```
printf("\n              0              50              100 %\n");
printf("              ");
x=wherex();
y=wherey();
textcolor(1);
for (i=0;i<40;i++)
putch(219);
gotoxy(x,y);
textcolor(14);

for (domain_y=0;domain_y<image.height;domain_y+=db_side)
 for (domain_x=0;domain_x<image.width;domain_x+=db_side)
 {
  /* Step 2 : Get Domain Block */

  minimum_distance=infinity;
  copy_rectangle(&image,domain_x,domain_y,&domain_block,
                 0,0,db_side,db_side);
  domain_mean=mean(&domain_block);
  for (current_range_y=0;current_range_y<=reduced_image.height-
       db_side;current_range_y++)
   for (current_range_x=0;current_range_x<=reduced_image.width-
        db_side;current_range_x++)
   {
    /* Step 3 : Get Range Block */

    copy_rectangle(&reduced_image,current_range_x,
                   current_range_y,&range_block,0,0,db_side,
                   db_side);

    /* best mean square fit is given by shifting */
    /* means to be equivalent */
```

```
        current_shift=((short) domain_mean)-
                ((short)mean(&range_block));
        intensity_shift(&range_block,current_shift);

        /* Step 3 : Loop Over Symmetries */

        for (current_symmetry=0;current_symmetry<nsyms;
             current_symmetry++)
        {
          flip(&range_block,&flipped_range_block,current_symmetry);
          current_distance=t_distance(&domain_block,
                                      &flipped_range_block);
          if (current_distance<minimum_distance)
          {
            minimum_distance=current_distance;
            best_map.shift=current_shift;
            best_map.symmetry=current_symmetry;
            best_map.range_x=current_range_x;
            best_map.range_y=current_range_y;
          }
          count++;
          if ((count%temp)==0) putch(219);
        }
      }
      fwrite(&best_map,sizeof(AffineMap),1,fractal_file);
  }
printf("\n\n                                          complete
       compression\n");
fclose(fractal_file);
free(pal);
free(image.pixel);
```

```
        free(reduced_image.pixel);
        free(domain_block.pixel);
        free(range_block.pixel);
        free(flipped_range_block.pixel);
        return(0);
}
```

FILE DECOMPRS.C

```
#include    <math.h>
#include    <stdio.h>
#include    <conio.h>
#include    <stdlib.h>
#include    <alloc.h>
#include    "fractal.h"

main        (int argc,char **argv)
{
            int             j,x,y;
            struct tga_hdr  header;
            char            *pal=NULL;
            long            number_of_maps,i;
            FILE            *image_file,*fractal_file;
            AffineMap       *affine_map_array,*map_ptr;
            short           domain_x,domain_y,arg_offset=0,db_side;
            unsigned long   loop,count=0,temp,iterate,iterates=256,loop1,loop2;
            Rectangle       image,reduced_image,range_block,
                            transformed_range_block;


            if ((argc<3)||(argc>4))
            {
             fprintf(stderr,
                    "\nUsage : decompress [num_iterates] fractal_file
                    image_file.\n");
             exit(1);
            }
```

```
if (argc==4)
{
  iterates=atoi(argv[1]);
  arg_offset=1;
}

/* Read in affine maps and header information. */

if (NULL==(fractal_file=fopen(argv[arg_offset+1],"rb")))
{
  fprintf(stderr,"\nUnable to open fractal file
          %s.\n",argv[arg_offset+1]);
  exit(1);
}

if (NULL==(image_file=fopen(argv[arg_offset+2],"wb")))
{
  fprintf(stderr,"\nUnable to open image file
          %s.\n",argv[arg_offset+2]);
  exit(1);
}

pal=(char *) calloc(768,sizeof(char));
if (1!=fread(&header,sizeof(struct tga_hdr),1,fractal_file))
{
  fprintf(stderr,"\nError reading header from fractal file %s.\n",
           argv[arg_offset+1]);
  exit(1);
}
```

```
if (1!=fread(pal,768,1,fractal_file))
{
 fprintf(stderr,"\nError reading palatte from fractal file %s.\n",
        argv[arg_offset+1]);
 exit(1);
}

if (1!=fread(&db_side,2,1,fractal_file))
{
 fprintf(stderr,"\nError reading domain block from fractal file %s.\n",
         argv[arg_offset+1]);
 exit(1);
}

if (1!=fwrite(&header,sizeof(struct tga_hdr),1,image_file))
{
 fprintf(stderr,"\nError writing header Targa header.\n");
 exit(1);
}

if (1!=fwrite(pal,768,1,image_file))
{
 fprintf(stderr,"\nError writing palatte Targa header.\n");
 exit(1);
}

number_of_maps=header.width*header.height/(db_side*db_side);
affine_map_array=
(AffineMap *)calloc(number_of_maps,sizeof(AffineMap));

image.width=header.width;
image.height=header.height;
```

```
image.length=(long)header.width*(long)header.height;
image.pixel=(Pixel *) calloc(image.length,sizeof(Pixel));

reduced_image.width=header.width/2;
reduced_image.height=header.height/2;
reduced_image.length=(long)header.width*(long)header.height/4;
reduced_image.pixel=
(Pixel *)calloc(reduced_image.length,sizeof(Pixel));

range_block.width=db_side;
range_block.height=db_side;
range_block.length=(long)db_side*(long)db_side;
range_block.pixel=(Pixel *) calloc(range_block.length,sizeof(Pixel));

transformed_range_block.width=db_side;
transformed_range_block.height=db_side;
transformed_range_block.length=(long)db_side*(long)db_side;
transformed_range_block.pixel=
(Pixel *) calloc(transformed_range_block.length,sizeof(Pixel));

for (i=0;i<image.length;i++)
  image.pixel[i]=ARBITRARY_PIXEL_VALUE;

/* Loop over domain blocks. */

loop1 =image.height/db_side;
loop2=image.width/db_side;
loop=iterates*loop1 *loop2;
temp=loop/40;
printf("\n              0              50              100 %\n");
printf("               ");
x=wherex();
```

```
y=wherey();
textcolor(1);
for (j=0;j<40;j++)
  putch(219);
gotoxy(x,y);
textcolor(14);

for (domain_y=0,map_ptr=affine_map_array;domain_y<image.height;
       domain_y+=db_side)
 for (domain_x=0;domain_x<image.width;domain_x+=db_side,
        map_ptr++)
   fread(map_ptr,sizeof(AffineMap),1,fractal_file);

fclose(fractal_file);

/* Loop for a prescried number of iterations. */

for (iterate=0;iterate<iterates;iterate++)
{
  reduce_image(&image,&reduced_image);

  /* Loop over domain blocks. */

  for (domain_y=0,map_ptr=affine_map_array;domain_y<image.height;
        domain_y+=db_side)
    for (domain_x=0;domain_x<image.width;domain_x+=db_side,
          map_ptr++)
    {
      /* Extract range block. */
```

```
        copy_rectangle(&reduced_image,map_ptr->range_x,
                       map_ptr->range_y,&range_block,0,0,db_side,
                       db_side);
        intensity_shift(&range_block,map_ptr->shift);

        /* Apply indicated symmetry. */

        flip(&range_block,&transformed_range_block,map_ptr->symmetry);

        /* Insert transformed block into image. */

        copy_rectangle(&transformed_range_block,0,0,&image,domain_x,
                       domain_y,db_side,db_side);
        count++;
        if ((count%temp)==0) putch(219);
      }
    }

    printf("\n\n                                    complete
          decompression\n");
    if (image.length!=fwrite(image.pixel,sizeof(Pixel),image.length,
        image_file))
    {
     fprintf(stderr,"\nError writing data to %s.\n",argv[arg_offset+2]);
     exit(1);
    }

    free(pal);
    free(affine_map_array);
    free(image.pixel);
    free(reduced_image.pixel);
    free(range_block.pixel);
    free(transformed_range_block.pixel);
    fclose(image_file);
    return(0);
}
```

บรรณานุกรม

A. Jacquin, "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations." *IEEE Transactions on Image Processing*. 1. (1992): 18-30.

Barnsley,M. *Fractals Everywhere*. Academic Press, Boston,1988.

Barnsley,Michael,and Lyman P. Hurd. *Fractal Image Compression*. A.K. Peters Ltd, 1993.

J.M. Beaumont, "Image data compression using fractal techniques." *BT Technology Journal*. 9. (1991): 93-109.

J. Waite, "A Review of Iterated Function System Theory for Image Compression." preprint. British Telecom Research Laboratories, Martlesham Heath, U.K. (1992).

Mark Finlay, and Keith A. Blanton. *Real World Fractals*. M&T Books, NewYork, 1993.

Michael F. Barnsley,and Alan D. Sloan. "A Betterway to compress Images." *Byte*. Vol.13, 1. (January,1988): 215-221.

Peitgen,H.-O.,and Richter,P. *The Beautly of Fractals*. Springer Verlag, Newyork, 1986.

Rhys lewis, *Practical Digital Image Processing*. Ellis Honwood, NewYork, 1990.

Robert L. Devanc. Chaos, *Fractals and Dynamics*. Addison-Wesley Publishing, Boston, 1989.