บทที่ 2
ทฤษฎีเบื้องต้น

วิธีที่ใช้ในโปรแกรมคือการค้นหาภาพโดยใช้องค์ประกอบพื้นฐานของภาพ (Content-Based - Image Retrieval) ซึ่งจะเป็นการนำรูปตัวอย่างมาวิเคราะห์ แล้วนำเปรียบเทียบกับภาพตัวอย่างในฐานข้อมูล ซึ่งในที่นี้โปรแกรมนี้จะทำการวิเคราะห์ สี และพื้นผิว (Color and Texture) โดยจะใช้ทฤษฎีเกี่ยวกับ สีและพื้นผิว ซึ่งทฤษฎีต่างๆนั้นจะกล่าวดังต่อไปนี้

2.1 การค้นหาภาพโดยใช้องค์ประกอบพื้นฐานของภาพ(Content-Based Image Retrieval)

ในที่นี้จะนำรูปภาพผ่านคุณสมบัติพื้นฐาน (low-level features) จากนั้นจะใช้วิธีวิเคราะห์รูปภาพ 2 แบบคือ สี (Color) และพื้นผิว (Texture) เมื่อได้ ดัชนี(index) แล้วก็จะนำดัชนีที่ได้ไปเปรียบเทียบกับภาพในฐานข้อมูล แล้วคำนวณหาค่าความแตกต่าง (Distance) เพื่อนำค่านี้ไปหาภาพที่ต้องการในฐานข้อมูล

ซึ่งในส่วนของการวิเคราะห์นั้น ได้ทำการวิเคราะห์ 2 วิธีคือ สี (Color) และ พื้นผิว (Texture) โดยทฤษฎีที่เกี่ยวข้องกับกับ 2 วิธีนี้มีดังนี้

1. สี (Color) วิเคราะห์ด้วยทฤษฎีกราฟแสดงความถี่ความเข้มสี (Color Histrogram)
2. พื้นผิว (Texture) วิเคราะห์ด้วยทฤษฎีกาบอร์เวฟเลต (Gabor Wavelet)

ส่วนของการค้นหาภาพแบบย้อนกลับจะใช้ทฤษฎีการป้อนกลับ (Relevance Feedback)

2.2 ส่วนของสี (Color)

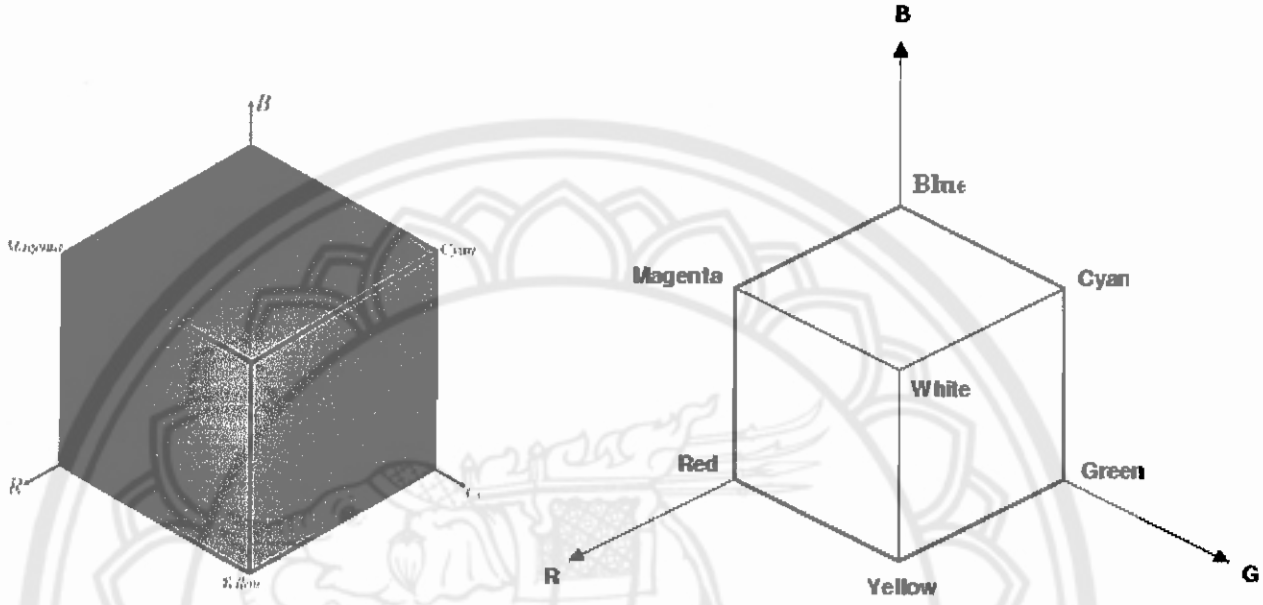
2.2.1 กราฟแสดงความถี่ความเข้มสี (Color Histrogram)

มาตรฐานของสี

ในปัจจุบันนี้มีการใช้มาตรฐานสีอยู่หลายระบบในแต่ละระบบก็จะมีการกำหนดแตกต่างกันไป แต่โดยทั่วไปก็จะมีการแบ่งที่คล้ายกันอยู่คือการแทนสีด้วยจุด โดยจะแทนจุดในสเปส 3 มิติ มีแกนอ้างอิงแต่ละแกน อย่างเช่น แกนของสี(สีแดง, สีเขียว, สีน้ำเงิน) แกนของความสว่าง (Lightness) แกนของความบริสุทธิ์ของสี (Saturation) ซึ่งแกนแต่ละแกนนี้จะเป็นอิสระต่อกัน เราจะยกตัวอย่างของระบบที่นิยมใช้ ระบบอาร์จีบี (RGB) และเอชเอสวี (HSV) ทั้ง 2 ระบบนี้ เป็นระบบของมาตรฐานของสีที่มีความคล้ายคลึงกัน

ระบบสี RGB

ระบบนี้จะเป็นการรวมกันของแสงสี ซึ่งมีแสงสีแดง แสงสีเขียว และ แสงสีน้ำเงิน จะแบ่งแต่ละสีออกจะเป็นแกน3แกน และ มีการไล่ความเข้มสี จากค่า 0→1 โดยค่า 0 ก็คือความเข้มสีมาก ค่า 1 คือความเข้มสีน้อย หรือแสงสีขาว ดังรูป



รูปที่ 2.1 รูป ระบบของ RGB

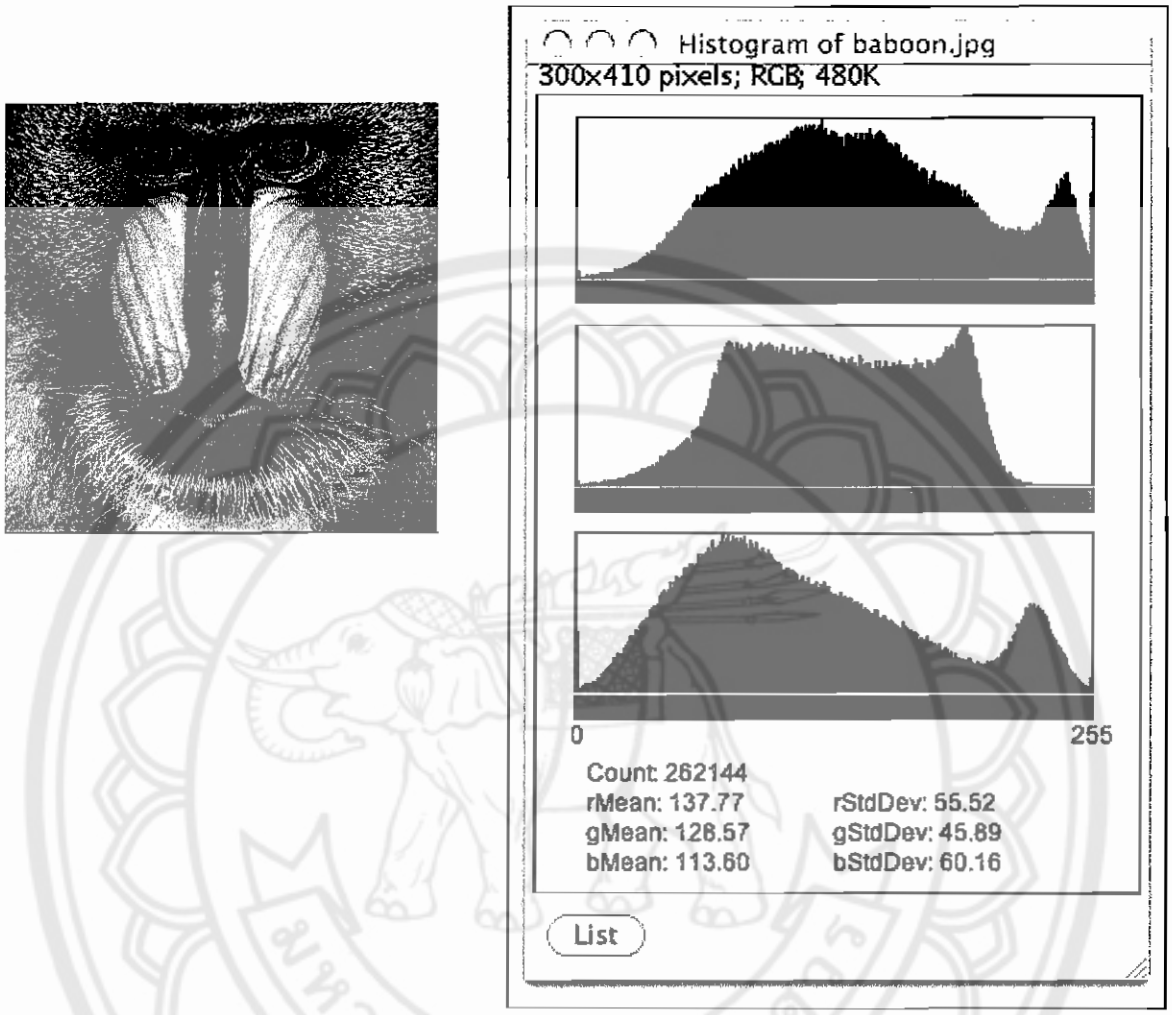
2.2.2 การอธิบายภาพโดยใช้กราฟแสดงความถี่ความเข้มสี (Color Histogram)

ภาพ ๆ หนึ่งจะประกอบด้วยจุดของเม็ดสีต่างๆ หลายจุดด้วยกัน ซึ่งถ้ามองด้วยตาเปล่าจะเป็นเม็ดสีรวมกันเป็นภาพโดยแต่ละเม็ดสีจะมีค่าของสีที่แตกต่างกันออกไป ซึ่งถ้าเป็นระบบสี RGB ก็จะประกอบด้วยสี แดง เขียว น้ำเงินดังภาพ

R	G	B	
0	0	0	black
255	0	0	red
0	255	0	green
0	0	255	blue
0	255	255	cyan
255	0	255	magenta
255	255	0	yellow
255	255	255	white

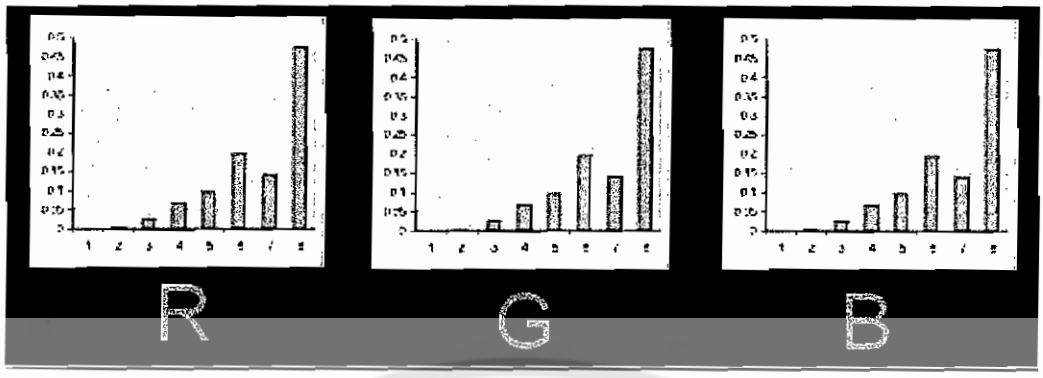
รูปที่ 2.2 รูปการเก็บค่าสี RGB

แล้วจะนำค่าสีแดง เขียว และ น้ำเงินของแต่ละจุดมาพล็อตกราฟได้ดังภาพ



รูปที่ 2.3 รูปแสดงคัลเลอร์ฮิสโตรแกรม (Color Histogram)

ซึ่งค่าที่ได้ของแต่ละสี จะมีค่าตั้งแต่ 0 ถึง 255 เราจะแบ่งออกเป็น บิน (Bin) เพื่อให้ง่ายต่อการเก็บลงฐานข้อมูลโดยจำนวนบินนั้นขึ้นอยู่กับ ความละเอียดของภาพที่เราต้องการ ยิ่งจำนวนบินมากเท่าไหร่ในการค้นหาภาพก็จะมีความถูกต้องมากขึ้นเท่านั้น แต่ทั้งนั้นฐานข้อมูลก็ต้องเก็บข้อมูลที่มากด้วยเช่นกัน

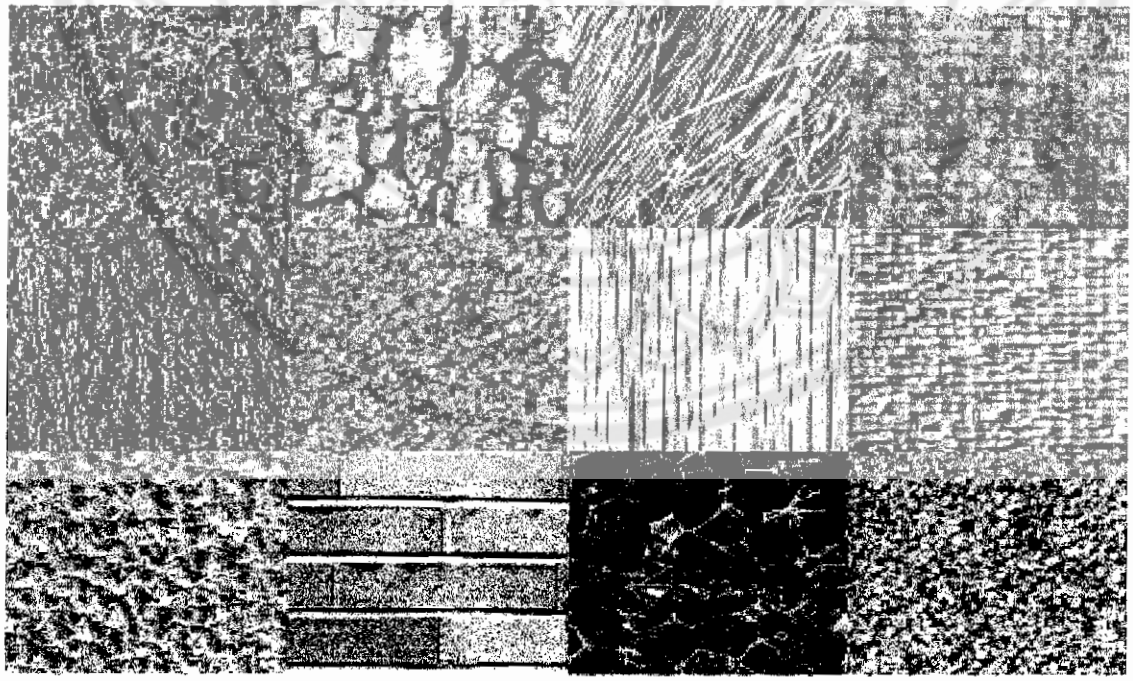


รูปที่ 2.4 รูปแสดงการแบ่งคัลเลอร์ฮิสโตรแกรม(Color Histrogram)
ออกเป็นบิน (Bin)

โดยเราจะเก็บค่าสีลงในฐานข้อมูลตามความถี่ของแต่ละบิน

2.3 ส่วนของพื้นผิว (Texture)

พื้นผิวในที่นี้จะหมายถึงส่วนที่แสดงถึงความลึกความตื้นของรูปภาพ โดยพื้นผิวนี้จะทำให้รูปภาพดูมีมิติ รวมถึงทั้งรายละเอียดที่รวมอยู่ในรูปภาพหรือสิ่งที่แสดงในรูปภาพนั้น ซึ่งรายละเอียดเหล่านี้จะประกอบรวมกันขึ้นทำให้รูปมีความลึกหรือมีมิตินั้นเอง



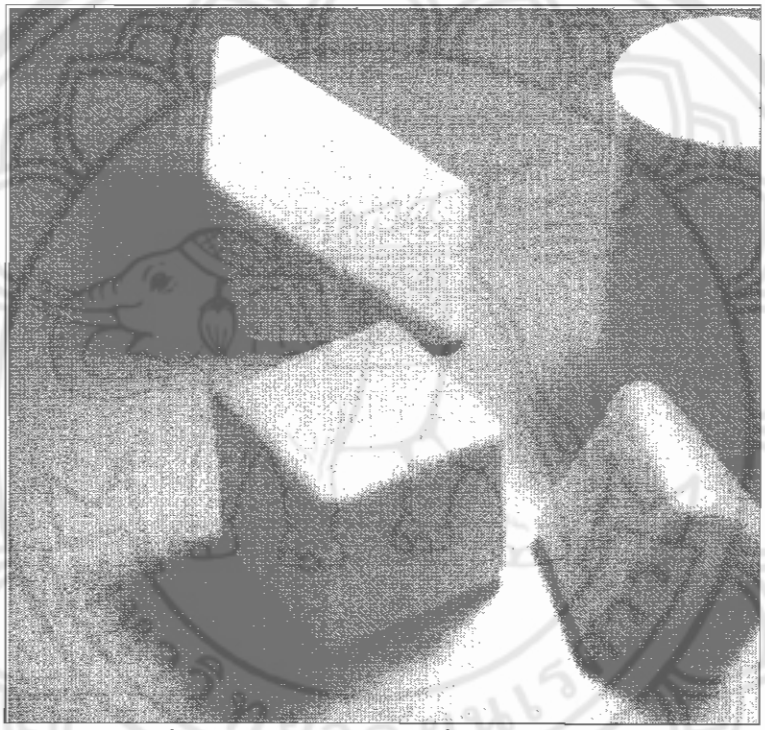
รูปที่ 2.5 รูปตัวอย่างของ พื้นผิว (Texture)

ในการวิเคราะห์ส่วนของพื้นผิว (Texture) นั้นเราจะใช้ทฤษฎีที่เกี่ยวข้องคือ Gabor Wavelet ซึ่งทฤษฎีจะมีดังนี้

2.3.1 กาบอร์เวฟเลต (Gabor Wavelet)

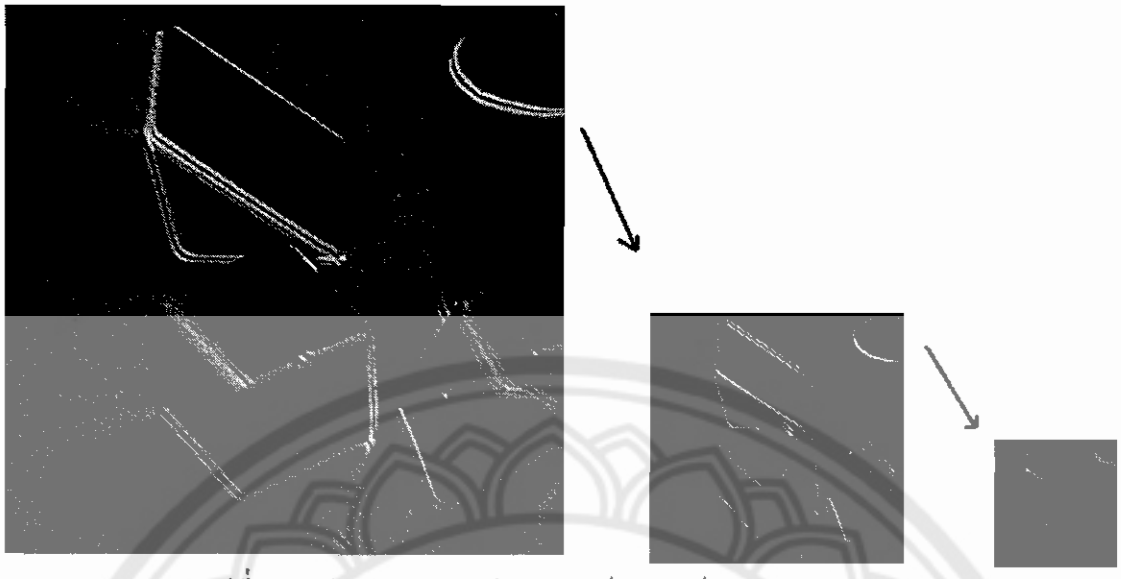
ทฤษฎีกาบอร์เวฟเลต (Gabor Wavelet) จะใช้ฟังก์ชั่นของกาบอร์เวฟเลต (Gabor Wavelet) มาเป็นฟิลเตอร์หรือตัวกรอง โดยจะนำรูปมาผ่านฟิลเตอร์เพื่อให้ได้ความถี่หรือข้อมูลที่เราต้องการ ให้อยู่ในรูปแบบข้อมูลของรูปภาพ

โดยเมื่อทำการนำรูปมาผ่านฟิลเตอร์หรือฟังก์ชั่นของกาบอร์เวฟเลตแล้วจะได้ผลดังตัวอย่างนี้



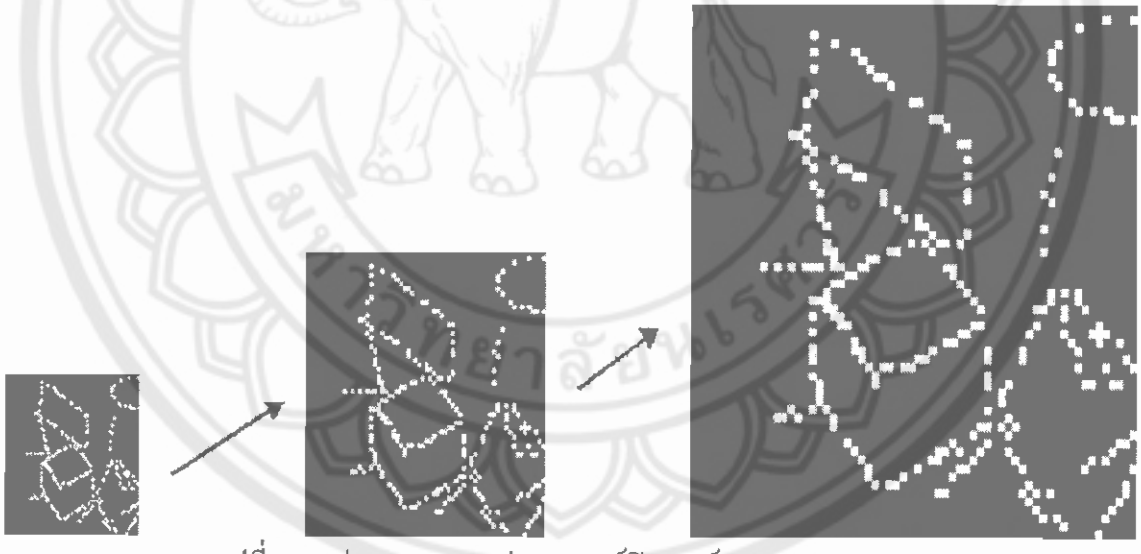
รูปที่ 2.6 รูปตัวอย่างที่ทำการทดลองผ่านฟังก์ชั่นกาบอร์เวฟเลต (Gabor Wavelet)

ซึ่งเมื่อทำการนำรูปผ่านเข้าฟิลเตอร์หรือฟังก์ชั่นกาบอร์เวฟเลตแล้ว รูปก็จะเปลี่ยนแปลงโดยจะเปลี่ยนรูปให้ไปเป็นระบบภาพขาว-ดำ (Gray Scale) เพื่อให้เห็นความแตกต่างของความลึกและมิติ จากนั้นทำการลดขนาดของรูปลงจนสามารถแยกส่วนที่สามารถแสดงให้เห็นถึงพื้นผิวที่แตกต่าง ดังรูป



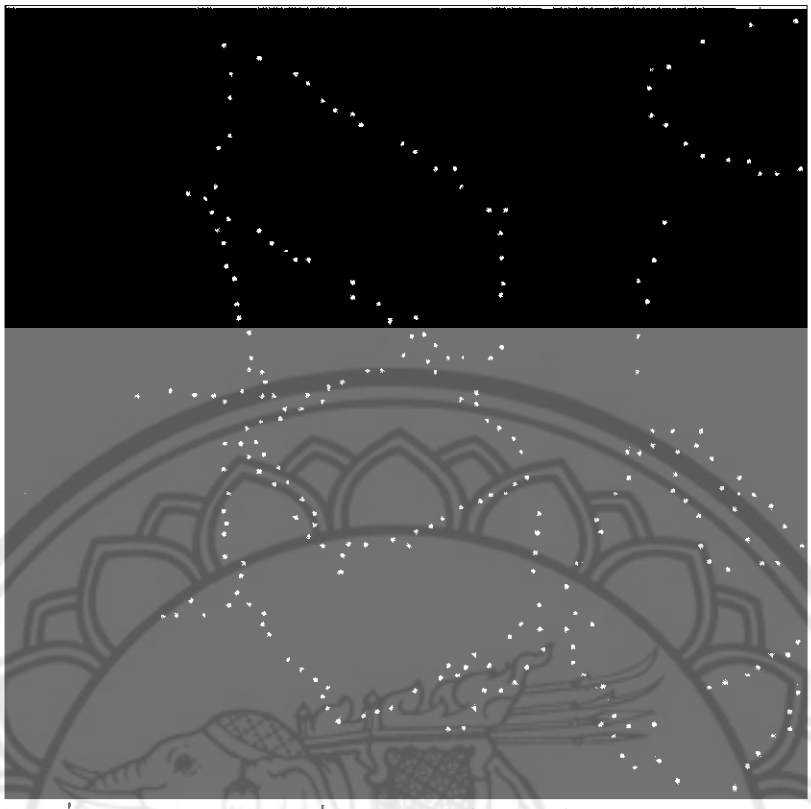
รูปที่ 2.7 รูปกระบวนการผ่านกาบอร์ฟิลเตอร์ (Garbor filter) (1)

จากนั้นเมื่อทำการกำหนดส่วนที่แบ่งส่วนของพื้นผิวได้แล้ว ก็ทำการกำหนดจุดไว้ดังรูปและทำการขยายภาพให้กลับมาขนาดปกติเพื่อแสดงให้เห็นส่วนที่แสดงไว้



รูปที่ 2.8 รูปกระบวนการผ่านกาบอร์ฟิลเตอร์ (Garbor filter) (2)

เมื่อผ่านกระบวนการกาบอร์เวฟเลต แล้วจะได้รูปดังนี้



รูปที่ 2.9 รูปผลการแสดงเมื่อผ่านกาบอร์ฟิลเตอร์ (Garbor filter) แล้ว

2.3.2 ทฤษฎีกาบอร์เวฟเลต (Garbor Wavelet)

ในทางการวิเคราะห์พื้นผิว (Texture) นั้น เราใช้ กาบอร์เวฟเลต (Gabor Wavelet) มาวิเคราะห์ ซึ่งมีการใช้กาบอร์ฟิลเตอร์ (Garbor filter) มาช่วยในการกรองภาพที่เราต้องการ โดยที่กาบอร์ฟิลเตอร์จะมีฟังก์ชั่นการทำงานเป็นสมการทางคณิตศาสตร์ ซึ่งกาบอร์เวฟเลตจะวิเคราะห์พื้นผิว (texture) และนำมาเปรียบเทียบความคล้ายคลึงกันซึ่งจะใช้สมการที่คอนโวลูชั่น (Convolution) ดังนี้

$$G_{mn}(x,y) = \sum_{x}\sum_{t} I(x-s, y-t)\psi_{mn}(s,t) \qquad (2.1)$$

s,t เป็นขนาดของฟิลเตอร์

ψ_{mn} เป็นส่วนของฟิลเตอร์

I คือ Image(x,y) ซึ่งมีขนาด $P \times Q$

สมการนี้หมายถึงกระบวนการที่นำรูป (Image) ซึ่งในที่นี้คือตัว I นำมาคอนโวลูชั่น (Convolution) กับฟังก์ชั่นของกาบอร์เวฟเลต (Gabor Wavelet) ซึ่งฟังก์ชั่นของกาบอร์เวฟเลต ในที่นี้หมายถึงตัว

ψ_{mn} หมายถึง ฟิลเตอร์ซึ่งมีขนาด (s,t) จากนั้นเมื่อคอนโวลูชั่นกับรูปภาพแล้วก็จะได้ $G_{mn}(x,y)$ ซึ่งเป็นตัวที่ผ่านกระบวนการ กาบอร์เวฟเลตแล้ว

Gabor Function (Mother wavelet)

$$\psi(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\exp[-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})]\bullet\exp(j2\pi Wx) \qquad (2.2)$$

โดย W คือ Modulation Frequency

และ $\exp[-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})]$ คือ ฟังก์ชั่นเกาส์เซี้ยน

ส่วนนี้จะอธิบายถึงตัวฟิลเตอร์ของฟังก์ชั่นกาบอร์เวฟเลต (Gabor Wavelet) ซึ่งจะเป็นสมการทางคณิตศาสตร์ ประกอบขึ้นจากการนำ Modulation Frequency และฟังก์ชั่นเกาส์เซี่ยนมาประกอบกัน

โดยเมื่อได้ $G_{mn}(x,y)$ แล้ว จะทำการหาค่าเฉลี่ย (μ_{mn}) และส่วนเบี่ยงเบนมาตรฐาน (σ_{mn}) จากสมการ

$$E(m,n) = \sum_x \sum_y |G_{mn}(x,y)|, m = 0,1,...,M-1; n = 0,1,...,N-1 \qquad (2.3)$$

$$\mu_{mn} = \frac{E(m,n)}{PxQ} \qquad (2.4)$$

$$\sigma_{mn} = \frac{\sqrt{\sum_x \sum_y (|G_{mn}(x,y)| - \mu_{mn})^2}}{PxQ} \qquad (2.5)$$

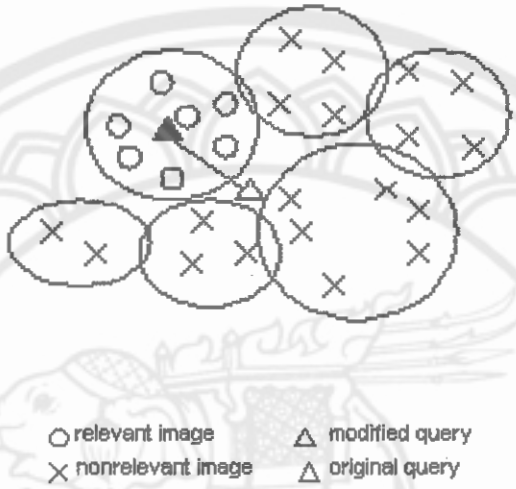
ซึ่งเมื่อได้ ค่าเฉลี่ย (Mean) และส่วนเบี่ยงเบนมาตรฐาน (Standard Divation) แล้วจะนำไปเก็บลงในฐานข้อมูลเพื่อใช้ในการเปรียบเทียบต่อไป

2.4 ระบบการป้อนกลับ (Relevance Feedback)

ระบบป้อนกลับ คือการสื่อสาร (Interactive) กันระหว่างผู้ใช้กับโปรแกรม โดยผู้ใช้จะทำการป้อนข้อมูลที่เป็นความต้องการของผู้ใช้กลับไป เพื่อให้โปรแกรมทำการเรียนรู้จากสิ่งที่ผู้ใช้ต้องการแล้วแสดงผลลัพธ์ที่ได้จากการป้อนกลับออกมา ซึ่งในการป้อนกลับแต่ละครั้งจะทำให้ผลลัพธ์ที่ได้นั้น

เป็นไปตามความต้องการของผู้ใช้ ยิ่งผู้ใช้ทำการป้อนกลับหลายครั้ง ก็ยิ่งทำให้ผลลัพธ์เป็นไปตามความต้องการของผู้ใช้มากขึ้น

โดย การป้อนกลับ (Relevance Feedback)จะมีหลักการทำงาน คือ จะเลือกแบ่งกลุ่มภาพที่ค้นหาได้ในรอบแรกออกเป็นกลุ่มที่ตรงกับภาพต้นแบบกับกลุ่มที่ไม่ตรง แล้วทำการย้ายจุดศูนย์กลางจากจุดศูนย์กลางของกลุ่มเก่า ไปจุดศูนย์กลางของกลุ่มใหม่ที่ตรงกับภาพต้นแบบมากที่สุด ดังในภาพ



รูปที่ 2.10 รูปการทำงานของระบบการป้อนกลับ(Relevance Feedback)

O	relevant image	=	ภาพที่ตรงตามที่ต้องการ
X	nonrelevant image	=	ภาพที่ไม่ตรงกับที่ต้องการ
▲	modified query	=	ค่าที่เรียงลำดับ (query) ที่ผ่านการคำนวณใหม่แล้ว
△	original query	=	ค่าที่เรียงลำดับ (query) จากภาพเดิม

จากรูปที่ 2.10 การทำงานของระบบการป้อนกลับ(Relevance Feedback) เมื่อมีการค้นหารูปออกมาแล้ว ภาพที่ได้อาจจะมีทั้งภาพที่ตรง(วงกลม)และภาพที่ไม่ตรงตามความต้องการ(กากบาท) ซึ่งเมื่อทำการป้อนกลับ แล้วระบบจะทำการย้ายจุดศูนย์กลางในการค้นหาจากที่เดิม(สามเหลี่ยมสีขาว) ซึ่งเป็นศูนย์กลางของภาพทั้งหมดไปไว้ที่จุดศูนย์กลางของภาพกลุ่มที่เราต้องการ(สามเหลี่ยมสีดำ) ดังรูป แล้วนำภาพในกลุ่มนั้นมาแสดง ซึ่งจะได้ภาพที่ตรงความต้องการมากขึ้นจากเดิม

ซึ่งสมการที่ใช้ในการย้ายจุดศูนย์กลางไปยังกลุ่มภาพที่ตรงความต้องการ คือ

$$Z = Z' + \alpha_R(\bar{X}' - Z') - \alpha_N(\bar{X}'' - Z') \quad (2.6)$$

α_N, α_R = ค่าคงที่ค่าหนึ่ง
$\bar{x}'$ = ค่าเฉลี่ยของกลุ่มภาพที่เราเลือก
$\bar{x}''$ = ค่าเฉลี่ยของกลุ่มภาพที่เราไม่ได้เลือก
Z' = ค่าดัชนี(Index) ของภาพต้นแบบ
Z = ค่าดัชนี(Index) ค่าใหม่ที่จะนำไปใช้ค้นหาในครั้งต่อไป

2.5 ดอทเน็ต แพลทฟอร์ม (.NET Platform) โดย C#

C# เป็นภาษาที่ถูกสร้างขึ้นมาเพื่อทำงานบนดอทเน็ต แพลทฟอร์ม (.NET Platform) สร้างและทำงานในลักษณะของการเขียนโปรแกรมเชิงวัตถุ Object Oriented Programming (OOP) ได้อย่างสมบูรณ์ ซึ่งต่างกับ C++ ที่ยังทำงานในลักษณะของการเขียนโปรแกรมเชิงวัตถุได้บางส่วน ไลบรารีของ C# ถูกสร้างขึ้นเพื่อให้ทำงานได้ครอบคลุมตั้งแต่การสร้างรูปแบบการติดต่อแบบ GUI ไปจนถึงการแอ็คเซสฐานข้อมูลผ่านอินเตอร์เน็ตหรือแม้แต่การทำงานร่วมกับ XML เพื่อทำให้การแลกเปลี่ยนข้อมูลระหว่างแอพพลิเคชั่นทำได้อย่างสมบูรณ์ไม่ว่าข้อมูลนั้นจะอยู่บนแพลทฟอร์มใดก็ตาม

เมื่อเปรียบเทียบกับ C++ แล้ว การสร้างแอพพลิเคชั่นจะทำได้ง่ายกว่ามาก เนื่องจาก C# ถูกออกแบบมาเพื่อการสร้างแอพพลิเคชั่นให้ทำงานบนอินเตอร์เน็ตโดยตรง (.NET Framework) นอกจากนี้ C# เป็น Object Oriented Programming (OOP) อย่างสมบูรณ์ ไม่ว่าจะเป็น

- Encapsulation การรวมกลุ่มฟังก์ชั่นการทำงานของออบเจ็กต์ต่างๆ (Object Blueprint, Class) เพื่อให้โค้ดถูกเขียนขึ้นมาอย่างเป็นระเบียบ
- Polymorphism (Inheritance, Interfacing และ Overloading) การนำโค้ดที่เขียนขึ้นมาแล้วนั้นมาใช้ในงานอื่นได้อีก

.NET Framework คือ กรอบการทำงานของการเขียนโปรแกรมที่ไมโครซอฟท์คิดขึ้นมาเพื่อรับรองการติดต่อสื่อสาร เพื่อแลกเปลี่ยนข้อมูล (Exchange Data) ระหว่างกันหรือแลกเปลี่ยนระหว่างแพลตฟอร์ม(Platform) ให้มีความสมบูรณ์ยิ่งขึ้น

แน่นอน การเขียนโปรแกรมบน Visual Basic ทำได้ง่ายกว่าแต่ประสิทธิภาพของโปรแกรมจะด้อยกว่าโปรแกรมที่เขียนขึ้นมาจาก C++ ในบางกรณี อย่างเช่นโปรแกรมที่ต้องติดต่อกับฮาร์ดแวร์ จะเลือกใช้ C++ แต่ถ้าต้องการความง่ายในการเขียนโปรแกรม โดยไม่ต้องคำนึงถึงประสิทธิภาพการทำงานมากนัก จะเลือกใช้ Visual Basic, C# จะรวมเอาลักษณะการเขียนโปรแกรมจากภาษาทั้งสอง เข้ามาไว้ เช่น C# จะไม่มี Overhead มากนัก เมื่อเทียบกับ Visual Basic

จุดเด่นของภาษา C# ที่ถูกพัฒนาขึ้น

ภาษา C# สนับสนุนการเขียนโปรแกรมแบบคอมโพเนนท์ (Component-Oriented Programming) ซึ่งประกอบด้วย พร็อพเพอร์ตี้ อีเวนต์ เดลอิเกต (Delegate) และ แอตทริบิวต์ (Attribute) โดยที่แอตทริบิวต์จะนำมาใช้เพื่อเพิ่มชนิดข้อมูลเมต้าดาต้า (Meta-Data) ก่อนออบเจ็กต์ใดๆ ส่วนในเวอร์ชั่นถัดไปจะสนับสนุนด้านการเขียนโปรแกรมแบบ Generic (คล้ายกับการเขียนโปรแกรม C++ ที่ใช้เทมเพลต ตัวอย่างเช่น Standard Template Library)

ภาษา C# ยังเพิ่มคำสั่งสำหรับการทำเอกสาร XML (eXtensible Markup Language) โดยตัวคอมไพล์ C# จะสร้างเอกสารโดยตรงจากซอร์สโค้ดโปรแกรม

ภาษา C# ยังปรับปรุงในเรื่องเกี่ยวกับหน่วยความจำที่จะช่วยลดเวลาในการแก้บักได้มาก การปรับปรุงนี้ก็คือการใช้ข้อมูลแบบ Type Safety และ Garbage Collection ซึ่งจะคอยปกป้องการรั่วไหลหน่วยความจำเมื่อมีการสร้างอินสแตนซ์ หรือ อ๊อบเจ็คต์ใหม่ โดยใช้โอเปอเรเตอร์ new และถ้าพบว่าอ๊อบเจ็คต์ไม่มีการใช้อีกต่อไป มันก็จะลบออกจากหน่วยความจำโดยอัตโนมัติ ภายใต้สภาปัตยกรรม .NET เราเรียกว่าโค้ดแบบถูกจัดการ (Managed code) ด้วยเหตุผลข้างต้นนี้ ภาษา C# จึงไม่มีพอยน์เตอร์และดีสตรัคเตอร์(แต่สามารถเพิ่มได้ และจะเป็นโค้ดที่ไม่ปลอดภัย หรือ Unsafety)

นอกจากนี้แล้วภาษา C# ยังสามารถใช้ร่วมกับโค้ดเดิมที่มีอยู่แล้ว ไม่ว่าจะเป็น C++ หรือ Basic ซึ่งเป็นโค้ดที่ไม่ถูกจัดการ (Unmanaged or Native Code) เหตุผลหนึ่งที่ต้องสนับสนุนโค้ดดั้งเดิมเพราะไม่ต้องเสียเวลามาเขียนโค้ดไลบรารีใหม่ และบางหน่วยงานยังใช้โค้ดแบบเดิมๆอยู่ รวมทั้งไม่ต้องเสียค่าใช้จ่ายมาก

ผู้สร้างภาษา C#

ภาษา C# โดยรับการพัฒนาโดยนาย Andrers Hejlsberg ซึ่งเป็นผู้ที่มีชื่อเสียงคนหนึ่งในการเขียนคอมไพเลอร์ Turbo Pascal ในช่วงศตวรรษที่ 1980 นอกจากนั้นเขายังเป็นหัวหน้าทีมออกแบบแอพพลิเคชั่น Borland Delphi