# การจดจำรูปแบบใบหน้ามนุษย์

# Pattern Recognition of Human Faces


พลโทวาภิรมย์ มนัสรังษี

Lt. Gen Vapirom Manasrangsi

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2540

ISBN 974-622-075-6

# PATTERN RECOGNITION OF HUMAN FACES

VAPIROM MANASRANGSI

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE IN COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

1997

ISBN 974-622-075-6

| | |
|---|---|
| **หัวข้อวิทยานิพนธ์** | การจดจำรูปแบบใบหน้ามนุษย์ |
| **นักศึกษา** | พลโทวาภิรมย์ มนัสรังษี |
| **อาจารย์ผู้ควบคุมวิทยานิพนธ์** | รศ. ดร. ครรชิต ไมตรี |
| **อาจารย์ผู้ควบคุมวิทยานิพนธ์ร่วม** | รศ. วิเชียร ศรีเสือชาม |
| **ระดับการศึกษา** | วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง |
| **พ.ศ.** | 2540 |

## บทคัดย่อ

มนุษย์สามารถจดจำมนุษย์ด้วยกันจากใบหน้าของมนุษย์ และแยกแยะออกได้ว่าเป็นใคร มีประวัติย่ออย่างไร แต่ในปัจจุบันนี้สถาปัตยกรรมของคอมพิวเตอร์ที่ประกอบด้วยโครงข่ายประสาทเทียม สามารถที่จะนำมาใช้เป็นเครื่องช่วยในการจดจำใบหน้ามนุษย์ เมื่อได้รับการป้อนข้อมูลไว้ในโปรแกรมด้วยกรรมวิธีเรียนรู้ ซึ่งจะทำให้คอมพิวเตอร์สามารถที่จะจดจำและแยกประเภทใบหน้ามนุษย์ที่ข้อมูลของใบหน้าได้รับการป้อนเข้าสู่ระบบ แม้ว่าภาพนั้นจะมีความแตกต่างกันด้วยท่าทาง ด้วยขนาด ด้วยองค์ประกอบอื่นๆ ด้วยมุมถ่าย และด้วยความชัดเจนของภาพ ฯลฯ วิทยานิพนธ์ฉบับนี้เสนอระบบที่ออกแบบระบบโครงข่ายประสาทเทียมสำหรับใช้ในการค้นหาภาพใบหน้าที่มีความคล้ายที่สุด 5 อันดับในฐานข้อมูลขนาดใหญ่ที่เก็บภาพด้านหน้าเพียงภาพเดียวต่อคน โดยมีความเร็วกว่าระบบเดิมที่เป็นการเปรียบเทียบภาพทีละภาพ โดยได้ออกแบบให้ใช้โครงข่ายประสาทเทียมรู้จำแบบเป็นกลุ่ม แทนการรู้จำแบบระบุบุคคล เพื่อแก้ปัญหาเรื่องความซ้ำซ้อนกัน ของข้อมูลภาพ ทำให้โครงข่ายประสาทเทียมเที่ยงตรงในการรู้จำเกือบถึง 100% ภายใต้เงื่อนไขที่กำหนดและมีความสามารถพิเศษคือสามารถป้อนภาพที่ไม่เคยผ่านการฝึกมาก่อนเพื่อหา 5 อันดับความคล้ายได้ และช่วยในการสร้างระบบการจดจำรูปแบบใบหน้ามนุษย์ เพื่อเป็นการประหยัดเวลา เกิดความถูกต้องมากที่สุด และสามารถที่จะพัฒนาต่อไปในอนาคต

| | |
|---|---|
| **Thesis Title** | Pattern Recognition of Human Faces |
| **Student** | Lt. Gen Vapirom Manasrangsi |
| **Thesis Advisor** | Assoc. Prof. Dr. Kanchit Mitree |
| **Thesis Co-advisor** | Assoc. Prof. Vichein Srisuakam |
| **Level of Study** | Master of Science in Computer Science and Information Technology<br>King Mongkut's Institute of Technology Ladkrabang |
| **Year** | 1997 |

## Abstract

Human being are extremely efficient at image recognition of human faces and can easily distinguish between individual faces. But today architecture of computer with being trained on the type of images based on the learning process. The processing can be carried out by designing proper system to solve problems of details, different perspectives and orientations and noised images. This thesis presents a design and an implementation of a neural network based system for searching an receiving the most five similar pictures of human faces from a database. This database record the pictures of the front views of individual human faces. The system employs a neural network for a group-based classification rather than individual-based classification. This technique makes this system more efficient than the ones that use the later approach. It also solves the problem of the information redundancy, and, hence, within defined condition the accuracy of recognition is approach to 100%. In addition, this system can recognize face pictures which have not been learned before to find the most 5 similar faces and constructing human face recognition system. This system will make more comfortable in memorizing by saving time, make more reliable, and can be developed in the future.

# กิตติกรรมประกาศ

ข้าพเจ้าขอกราบขอบพระคุณ รศ. ดร. ครรชิต ไมตรี รศ. วิเชียร ศรีเสือขาม และ รศ. ดร. ชม กิ้มปาน เป็นอย่างสูงที่ได้ให้การประสิทธิ์ประสาทวิชา ให้คำแนะนำ คำปรึกษา และเป็นกำลังใจในเรื่องต่างๆ มาโดยตลอด จนทำให้วิทยานิพนธ์นี้สามารถสำเร็จลุล่วงได้

ขอขอบคุณเพื่อนๆ น้องๆ ทุกๆ คน ที่เป็นกำลังใจ ให้คำแนะนำ และให้ความช่วยเหลือจนทำให้การจัดพิมพ์วิทยานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดีและสวยงามกว่าที่ตั้งใจไว้อย่างมาก

วาภิรมย์ มนัสรังษี

# สารบัญ

หน้า

# สารบัญ (ต่อ)

# สารบัญ (ต่อ)

# สารบัญตาราง

**ตารางที่** **หน้า**

# สารบัญภาพ

หน้า

# สารบัญภาพ (ต่อ)

หน้า

# บทที่ 1
# บทนำ

## 1.1 คำนำ

ปัจจุบันคอมพิวเตอร์และวิธีการต่างๆ ในการประมวลผลภาพมีความสามารถสูงขึ้นมาก โดยทางด้านการรู้จำต่างๆ ไม่ว่าจะเป็นการรู้จำวัตถุ รู้จำตัวอักษร หรือ รู้จำเสียงพูด ก็มีโครงข่ายประสาทเทียมที่ถอดแบบมาจากการทำงานของระบบสมองของมนุษย์ เป็นวิธีการยุคใหม่ที่ช่วยให้ลดความยุ่งยากซับซ้อนลงไปได้อย่างมาก ดังนั้นในการรู้จำใบหน้ามนุษย์ด้วยโครงข่ายประสาทเทียมซึ่งประเทศไทยยังมีการศึกษากันน้อยมากจึงเป็นเรื่องท้าทายอย่างมากอีกแขนงหนึ่ง

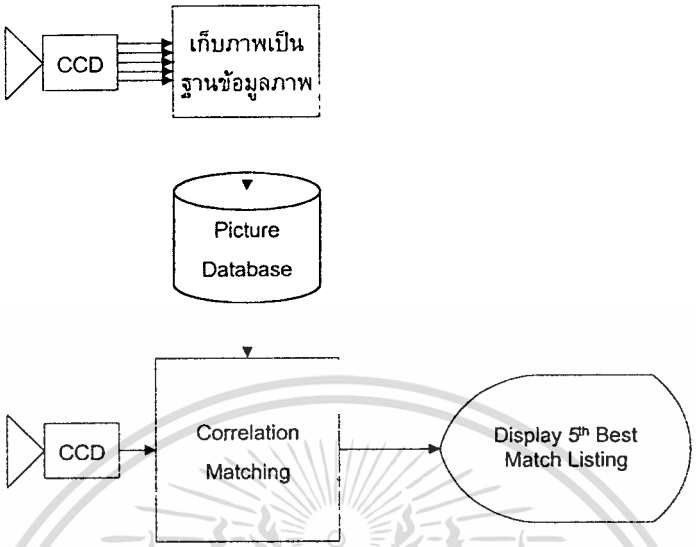
## 1.2 วัตถุประสงค์ของการวิจัย

เป็นการนำเสนอวิธีการหนึ่งในการรู้จำภาพใบหน้าจากภาพถ่ายโดยเป็นการรู้จำแบบกลุ่มที่ใช้โครงข่ายประสาทเทียมในการรู้จำ โดยเป็นเทคนิคเฉพาะที่ทำการปรับปรุงและทดลองมาจากแนวความคิดของตนเองซึ่งเกิดขึ้นมาจากการค้นคว้าศึกษาจากวิธีการต่างๆ ในการรู้จำใบหน้าในปัจจุบันที่มีอยู่ในเอกสารทางวิชาการต่างๆ ซึ่งแนวทางของวิธีการที่นำเสนอนี้ยังจะสามารถพัฒนาต่อเพื่อเพิ่มความสามารถได้อีกมากในอนาคต

## 1.3 ลำดับที่มาของแนวความคิดและหลักการที่นำเสนอในวิทยานิพนธ์นี้

จากเรื่องราวต่างๆ ของการรู้จำใบหน้าที่ได้สำรวจบทความต่างๆ นั้นได้พบเห็นอย่างชัดเจนถึงประโยชน์ที่จะได้รับจากการศึกษา แนวทางที่ต่างวิ่งไปสู่ อุปสรรคและความสำเร็จต่างๆ ที่เกิดขึ้นในแนวทางของตน โดยแรกเริ่มของการวิจัยนี้ได้ใช้ระบบที่ใช้การเปรียบเทียบภาพโดยการใช้สหสัมพันธ์เทียบภาพในฐานข้อมูลทั้งหมดโดยตรงทีละภาพ ดังภาพที่ 1.1 ในการตรวจสอบภาพจะใช้วิธีการสหสัมพันธ์ (Correlation) ในการเทียบภาพกับทุกๆ ภาพในฐานข้อมูล โดยมีการเรียงลำดับจากภาพที่คล้ายที่สุดเรียงกันไปตามลำดับจำนวน 5 ภาพ ปัญหาของระบบนี้คือถ้าฐานข้อมูลมีขนาดใหญ่จะใช้เวลาในการประมวลผลมากทีเดียว

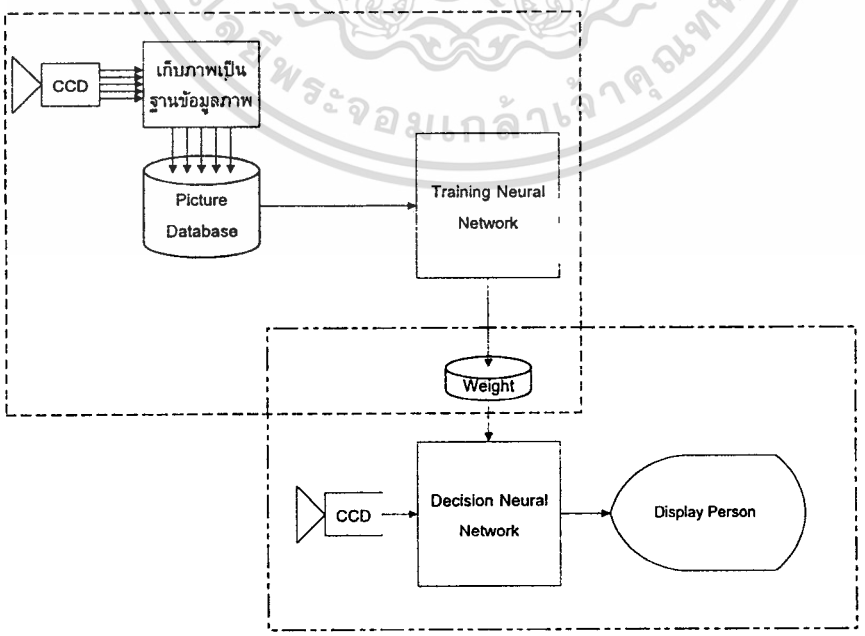
ภาพที่ 1.1




ระบบในการเทียบภาพจากฐานข้อมูลทีละภาพ

ดังนั้นจึงได้ทดลองใช้ระบบที่ประกอบด้วยโครงข่ายประสาทเทียม โดยได้ทำการฝึกสอนให้มีการรู้จำจากข้อมูลภาพระดับสีเทาโดยตรง ดังแสดงดังภาพที่ 1.2

ภาพที่ 1.2



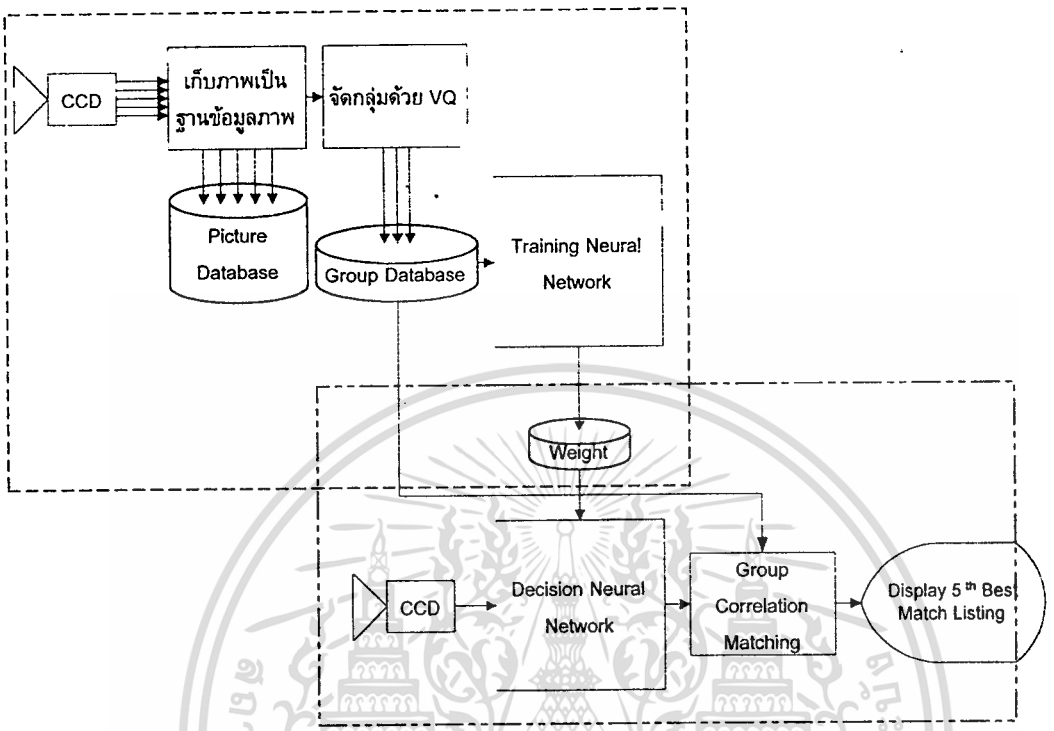

ระบบในการรู้จำแบบระบุบุคคลโดยใช้โครงข่ายประสาทเทียม

และเพื่อหาธรรมชาติของโครงข่ายประสาทเทียมด้วยการกำหนดชั้นฮิดเดนตายตัวที่ 30 หน่วยพบปัญหาว่าโครงข่ายประสาทเทียมใช้เวลาในฝึกสอน (Training) ที่แปรผันโดยตรงกับขนาดของฐานข้อมูลภาพใบหน้า ซึ่งเมื่อเพิ่มขนาดฐานข้อมูลใหญ่ขึ้น (จำนวนสมาชิกมากขึ้น) อีกเพียงเล็กน้อยเท่านั้น เวลาที่ใช้ในการฝึกสอนต้องใช้เพิ่มขึ้นอีกอย่างมาก และหากเพิ่มขนาดให้ใหญ่ขึ้นไปอีกเรื่อยๆ พบว่าโครงข่ายประสาทเทียมไม่สามารถเรียนรู้ (Learning) ได้อีกต่อไปไม่ว่าจะฝึกสอนนานเท่าใดก็ตาม และแม้ในครั้งที่ฝึกสอนได้ก็ตามในส่วนของการตัดสินใจของโครงข่ายประสาทเทียมในการรู้จำภาพใบหน้านั้นรู้จำระบุบุคคลผิดพลาดเป็นจำนวนมากครั้ง เพราะข้อมูลภาพใบหน้าที่เข้ามามีการแปรเปลี่ยนไปจากภาพใบหน้าที่เคยรู้จักถึงแม้ผู้วิจัยพยายามปรับข้อมูลภาพที่เข้ามาตรวจสอบรู้จำด้วยการประมวลผลก่อน (Preprocessing) ต่างๆ แล้วก็ตาม และพบอีกว่าการฝึกสอนโครงข่ายประสาทเทียมเพื่อให้สามารถรับกับการแปรเปลี่ยนของภาพ เช่น การเปลี่ยนแปลงของ การให้แสง (Lighting) การวางตัวของใบหน้า (Orientation) ขนาดของใบหน้า (Scaling) เป็นต้น นั่นต้องมีภาพบุคคลเดียวกันจำนวนหลายภาพ มาทำการฝึกสอนซึ่งก็จะทำให้เกิดความยุ่งยากและใช้เวลามากขึ้นไปอีก นั่นคือถ้าฐานข้อมูลมีขนาดใหญ่มากๆ ไม่มีโอกาสที่จะใช้วิธีการดังกล่าวได้อย่างมีประสิทธิภาพได้เลย และในการที่จะนำไปใช้ในทางปฏิบัติเมื่อมีการเปลี่ยนแปลงฐานข้อมูลภาพ ซึ่งอาจจะเนื่องมาจากการเพิ่มจำนวนสมาชิกใหม่เข้ามาในระบบ หรือการเปลี่ยนแปลงภาพใบหน้าของสมาชิกเดิม จะต้องทำการฝึกฝนให้โครงข่ายประสาทเทียมรู้จำใหม่ทุกครั้ง

จึงทำให้เกิดคำถามขึ้นมาสองข้อคือหนึ่งจะต้องคอยเป็นเวลาเท่าใดหลังการเปลี่ยนแปลงฐานข้อมูลแล้วจึงจะสามารถเปิดระบบใช้งานได้อีกครั้ง และอีกคำถามหนึ่งคือจะต้องใช้เครื่องคอมพิวเตอร์ที่มีความสามารถสูงเท่าใดในการฝึกโครงข่ายประสาทเทียมให้รู้จำได้อย่างรวดเร็ว ซึ่งในทางปฏิบัติเป็นไปได้ยากและอาจจะไม่คุ้มราคา และคงทำให้ไม่สามารถประยุกต์ไปใช้งานในด้านต่างๆ ให้แพร่หลายได้

จากการศึกษาค้นคว้าและพัฒนาแก้ปัญหาเป็นลำดับ จนพบหนทางแก้ปัญหาดังกล่าวได้ โดยถ้าหากเราสามารถลดความซ้ำซ้อนกำกวมของข้อมูลภาพใบหน้า โดยจัดให้ภาพใบหน้าที่มีความคล้ายคลึงกันให้เป็นกลุ่มเดียวกันเสีย แล้วให้โครงข่ายประสาทเทียมเรียนรู้กลุ่มภาพดังกล่าวเป็นสิ่งเดียวกัน ดังแสดงระบบที่นำเสนอ ดังภาพที่ 1.3

ภาพที่ 1.3




ระบบในแนวความคิดโดยรวมที่นำเสนอในวิทยานิพนธ์
ที่ใช้โครงข่ายประสาทเทียมในการรู้จำกลุ่ม

เมื่อมีภาพใดเข้ามาตรวจสอบในภายหลังโครงข่ายประสาทเทียมจะสามารถตัดสินใจในการระบุกลุ่มที่ได้รู้จำไว้แล้วนั้นได้อย่างมีประสิทธิภาพและง่ายดาย ซึ่งวิธีการหรือแนวความคิดที่เสนอนี้ยังไม่เคยมีการกล่าวถึงกันมาก่อน ซึ่งหากเป็นไปตามที่ตั้งใจไว้มันจะสามารถเพิ่มความน่าเชื่อถือให้แก่ระบบได้อย่างมาก อีกทั้งสามารถนำแนวความคิดการรู้จำกลุ่มดังกล่าวไปประยุกต์กับระบบรู้จำใบหน้าอื่นๆ ที่ได้มีการศึกษากันอยู่แล้วมากมายได้อย่างมีประสิทธิภาพเช่นกัน

ระบบการรู้จำใบหน้าที่นำเสนอนี้จะเน้นการรู้จำโดยใช้โครงข่ายประสาทเทียมที่รู้จำในลักษณะกลุ่มใบหน้า และมีการเปรียบเทียบภาพโดยการใช้วิธีการสหสัมพันธ์ในการเทียบภาพเฉพาะภาพสมาชิกที่อยู่ในกลุ่ม ซึ่งจะเป็นระบบที่มีความเร็วสูง

การทำงานแบ่งเป็น 2 ขั้น คือขั้นเตรียมการ และขั้นของการตรวจสอบใบหน้า

1) ในขั้นเตรียมการของระบบนั้นเริ่มจากการเก็บภาพทั้งหมดเป็นฐานข้อมูลภาพ จากนั้นทำการจัดกลุ่มภาพใบหน้าด้วยเวกเตอร์ควอนไตซ์ แล้วนำข้อมูลกลุ่มดังกล่าวมาทำการฝึก

ให้โครงข่ายประสาทเทียม โดยผลจากการเรียนรู้จะได้เป็นไฟล์ค่าถ่วงน้ำหนัก (Weight File) เพื่อใช้ในการตัดสินใจของโครงข่ายประสาทเทียม

2) ในการตรวจสอบภาพใบหน้าเริ่มการทำงานจากการเก็บภาพที่ต้องการเข้ามา จากนั้นโดยการนำไฟล์ค่าถ่วงน้ำหนักมาใช้ โครงข่ายประสาทเทียมจะตัดสินใจว่าภาพใบหน้าที่เข้ามาเป็นภาพของบุคคลที่เป็นสมาชิกอยู่ในกลุ่มใด เอาต์พุตที่ได้จะเป็นตัวชี้กลุ่ม ซึ่งจะเป็นตัวบอกกระบวนการในการเทียบภาพด้วยวิธีการสหสัมพันธ์ในกระบวนการถัดไปว่าจะต้องทำการตรวจสอบภาพที่เข้ามาเทียบกับสมาชิกคนใดบ้างในฐานข้อมูลกลุ่ม การเลือกกระทำเฉพาะสมาชิกในกลุ่มเช่นนี้เพื่อเป็นการเพิ่มความเร็วในการเปรียบเทียบภาพและอีกทั้งหลังจากการตรวจสอบครบทุกสมาชิกในกลุ่มแล้วจะนำมาเรียง 5 อันดับที่มีความคล้ายมากที่สุดตามลำดับได้อีกด้วย ผลจากภาพ 5 ภาพที่ค้นพบและผ่านการเรียงลำดับความคล้ายจะนำมาแสดงบนหน้าจอตามลำดับ

รายละเอียดทั้งหมดอย่างละเอียดในการทำงานของระบบอธิบายไว้ในบทที่ 4 ต่อไป

**1.4 สมมติฐานของการศึกษา**

สิ่งที่น่าจะได้มาจากแนวทางของวิธีการที่นำเสนอนี้ น่าจะได้มาซึ่งสิ่งสำคัญที่สุดที่เราต้องการคือ

1) บุคคลเดียวใช้ภาพถ่าย 1 ภาพในการป้อนให้โครงข่ายประสาทเทียมรู้จำ ทั้งนี้เพราะในการรู้จำแบบรู้จำกลุ่มนั้น ในแต่ละกลุ่ม ก็มีจำนวนสมาชิกที่ทำให้เกิดการแปรเปลี่ยนได้เป็นอย่างดีอยู่แล้ว ถ้าภาพบุคคลต่างๆ เข้ามาตรวจสอบนั้นมีการแปรเปลี่ยนไปบ้าง ก็มีผลต่อการรู้จำน้อยกว่าการรู้จำแบบระบุบุคคลแบบที่กล่าวมาตอนต้นนั้นแน่นอน

2) การฝึกสอนโครงข่ายประสาทเทียมที่ต้องใช้เวลานานนั้น ไม่ จำเป็นจะต้องนำภาพบุคคลจำนวนมากถึง 1000 คน หรือ 10000 คนมาทำการฝึกสอนทั้งหมด โดยแค่สุ่มเลือกมาจำนวนหนึ่งเท่านั้น ทั้งนี้เพราะการรู้จำกลุ่มเป็นเพียงทางด่วนในการระบุกลุ่มเพื่อการลดปริมาณการประมวลผลเท่านั้น เมื่อเราสามารถรู้กลุ่มประเภทของใบหน้าได้แล้ว จะมีการทำการค้นหาเฉพาะในกลุ่มดังกล่าว หรือเทียบกันภายในกลุ่มเดียวกันเท่านั้น ซึ่งสิ่งนี้จะทำให้ลดเวลาในการฝึกสอนได้อย่างมาก

3) ถ้ามีการเปลี่ยนแปลงฐานข้อมูล เช่น มีสมาชิกใหม่ หรือเปลี่ยนแปลงภาพสมาชิกเก่า ก็ไม่จำเป็นจะต้องทำการฝึกสอนโครงข่ายใหม่อีก โดยโครงข่ายประสาทเทียมจะสามารถให้คำตอบกับภาพที่เข้ามาตรวจสอบได้ทุกครั้งว่าจัดอยู่ในกลุ่มใด ดังนั้นถ้ามีการเปลี่ยนแปลงฐานข้อมูลก็เพียงนำภาพทั้งหมดในฐานข้อมูลมาให้โครงข่ายประสาทเทียมตรวจหากลุ่มใหม่

แล้วจัดแยกกลุ่มใหม่อีกครั้งแทนการต้องเรียนรู้ใหม่ ทำให้สามารถนำไปใช้ในทางปฏิบัติ ได้แทบทุกอย่างของการประยุกต์รู้จำใบหน้าในงานต่างๆ เพราะมีความเร็วสูง

**1.5 ขอบเขตของงานวิจัย**

ในวิทยานิพนธ์นี้ได้เสนอวิธีการรู้จำกลุ่มภาพใบหน้าโดยใช้โครงข่ายประสาทเทียมในการรู้จำกลุ่ม โดยมุ่งเน้น 4 ประเด็นหลัก ดังนี้

1) การตรวจหาส่วนที่เป็นใบหน้าในภาพถ่าย

2) การลดรายละเอียด ของข้อมูล โดยการใช้การแปลง เวฟเลต

3) การจัดกลุ่มข้อมูลเพื่อรวมกลุ่มข้อมูลซ้ำซ้อนเข้าด้วยกัน โดยการใช้เวกเตอร์ควอนไตซ์

4) การตัดสินใจ ในการรู้จำกลุ่มโดยใช้โครงข่ายประสาทเทียม

**1.6 รายละเอียดของวิทยานิพนธ์**

เนื้อหาของวิทยานิพนธ์ได้แยกออกเป็น 6 บท เพื่อแสดงถึง วิธีการที่เคยใช้กันในการรู้จำใบหน้า ทฤษฎีที่นำมาใช้ในวิทยานิพนธ์ วิธีการรู้จำใบหน้าที่นำเสนอ ผลการทดลอง สรุป และแนวทางในการพัฒนาต่อไป ดังมีรายละเอียดในบทต่างๆ ดังต่อไปนี้

**บทที่ 1 บทนำ**

กล่าวถึงวัตถุประสงค์ของการวิจัย สมมติฐานของการศึกษา ขอบเขตของการวิจัย และเนื้อหาของวิทยานิพนธ์

**บทที่ 2 การประมวลผลภาพและการรู้จำภาพใบหน้า**

กล่าวถึงการที่มนุษย์มีความสามารถในการรู้จำใบหน้า ความยุ่งยากในการรู้จำภาพใบหน้าด้วยเครื่องคอมพิวเตอร์ ประโยชน์ในการนำไปประยุกต์ใช้งาน และแนะนำตัวอย่างวิธีการบางอย่างที่ได้มีการศึกษา และทำกันอยู่จนถึงปัจจุบันเพื่อให้คอมพิวเตอร์มีความสามารถในการรู้จำใบหน้า

**บทที่ 3 ทฤษฎีพื้นฐานที่เกี่ยวข้อง**

กล่าวถึงทฤษฎีต่างๆ ที่วิทยานิพนธ์นี้ได้นำมาใช้ประกอบ คือ การกำจัดสัญญาณรบกวนด้วยตัวกรองเลือกค่ากลาง การหาขอบภาพด้วยโซเบล การเทียบภาพโดยวิธีการสห

สัมพันธ์ การค้นหาใบหน้าโดยการใช้สหสัมพันธ์ การค้นหาใบหน้าโดยวิธีการสหสัมพันธ์กับการย่อภาพสามระดับ การแปลงเวฟเลต การจัดกลุ่มด้วยเวกเตอร์ควอนไตซ์ และโครงข่ายประสาทเทียมในการรู้จำ

**บทที่ 4 ระบบการรู้จำภาพใบหน้าด้วยโครงข่ายประสาทเทียม**

กล่าวถึง แนวความคิดที่วิทยานิพนธ์นี้นำเสนอในการรู้จำภาพใบหน้า และกระบวนการในขั้นตอนต่างๆ อย่างละเอียด

**บทที่ 5 การทดลองและผลการทดลอง**

กล่าวถึงการทดลองเพื่อทดสอบแนวความคิดที่นำเสนอ

**บทที่ 6 สรุปผลการวิจัยและแนวทางในการพัฒนา**

กล่าวถึงผลสรุปจากการวิจัย และข้อเสนอแนะ สำหรับแนวทางในการพัฒนาต่อไป

# บทที่ 2
# การประมวลผลภาพและการรู้จำภาพใบหน้า

การรู้จำใบหน้าในปัจจุบันนี้ สามารถนำไปใช้ได้จริง และมีความเร็วเป็นแบบเวลาจริง (Real Time) ซึ่งทำให้เป็นเรื่องท้าทายอย่างมากที่เราจะสามารถทำให้เครื่องคอมพิวเตอร์มีความสามารถดังกล่าวได้ ซึ่งทั้งนี้เหตุผลหนึ่งที่สามารถบรรลุเป้าหมายดังกล่าวได้ ก็เนื่องจากการเลียนแบบการมองเห็นของมนุษย์ ด้วยวิธีการใหม่คือโครงข่ายประสาทเทียม ที่พัฒนาความสามารถขึ้นเป็นลำดับ ในการรู้จำใบหน้านั้น ประกอบด้วย 2 ส่วนที่น่าท้าทาย คือ การตรวจจับเพื่อค้นหาส่วนที่เป็นภาพใบหน้า และการรู้จำภาพใบหน้าว่าเป็นภาพของบุคคลคนใด ได้เคยรู้จักมาก่อนหรือเปล่า

## 2.1 มนุษย์เรามองและรู้จำภาพใบหน้าได้อย่างไร

สมองของมนุษย์เป็นโมเดลในการทำงานที่ดีมากจึงเหมาะที่จะนำมาเลียนแบบให้กับคอมพิวเตอร์ในส่วนของการตัดสินใจหรือรู้จำสิ่งต่างๆ บางอย่างที่รู้จักกันว่ารหัสแฟคตอเรียล (Factorial Code) อาจจะเป็นสิ่งที่ถูกใช้ระหว่างการประมวลผลข้อมูลในการมองเห็นเพื่อนำมารวบรวมและหาค่าว่าสิ่งที่ตาของเรามองเห็นเป็นอะไร ดังนั้นคำถามอันหนึ่งก็คืออะไรคือตัวแปลงที่มีความสามารถในการแปลงภาพที่เรามองเห็นนั้นให้ออกมาเป็นรหัสแฟกตอเรียลนี้ได้

หลักฐานสนับสนุนรหัสของแฟคตอเรียลมาจากการทดลองของทาง Neurophysiological โดยศึกษาเส้นทางเดินของสมองในส่วนของการมองเห็น พบว่ากระบวนการมองเห็นนั้นเริ่มต้นมาจากเรตินา (Retina) ตรงบริเวณตำแหน่งด้านหลังที่เป็นผนังของดวงตาที่ซึ่งแสงถูกส่งผ่านไปยัง Neural Impulses ผ่าน Optic Nerve ไปยังส่วนที่กว้างใหญ่และเป็นพื้นที่ที่เป็นชั้นด้านหลังของสมองที่รู้จักกันในชื่อว่า Visual Cortex ซึ่งจากการศึกษาถึงข้อมูลทั้งหลายที่แพร่ไปยังชั้นที่ลึกที่สุดและไปสิ้นสุดที่พื้นที่พิเศษอื่นๆ ที่เป็นส่วนที่เกี่ยวข้องกับการรู้จำวัตถุและใบหน้า จากการวิเคราะห์ผลการทดลองการตอบสนองทางไฟฟ้าเมื่อมีการกระตุ้นการมองเห็น โดยได้สอดขั้วไฟฟ้าจำนวนมากเข้าไปใกล้ๆ นิวรอลที่อยู่ตามรายทางของทางเดินของการมองเห็น แล้วบันทึกการตอบสนองของนิวรอล ในการทดลองพบว่ามีการแทนวัตถุนั้นออกมาเป็นรหัสต่างๆ ดังนั้นนักวิจัยได้ให้ความสนใจเป็นพิเศษในรูปแบบวัตถุอินพุตกับรูปแบบที่คายไฟฟ้าออกมาจากสมองเพื่อลงความเห็นว่ารูปแบบภาพถูกหาออกมาได้อย่างไรซึ่ง

ตอนนี้ก็กำลังศึกษากันอย่างขมักเขม่นอยู่ คงต้องรอกันต่อไปที่จะได้รู้และได้บรรลุสิ่งน่าทึ่งภายในมนุษย์ที่เป็นกลไกของการประมวลผลการมองเห็นของสมอง

**2.2 ความซับซ้อนและยุ่งยากในการรู้จำใบหน้าด้วยคอมพิวเตอร์**

ในการรู้จำใบหน้านั้นมนุษย์ดูเหมือนจะทำได้อย่างรวดเร็วและง่ายดาย เหมือนกับไม่ต้องใช้ความพยายามสักเท่าไหร่เลย แต่ก็ไม่ได้หมายความว่าการจะให้คอมพิวเตอร์รู้จำได้นั้นจะเป็นเรื่องง่ายๆ ไปด้วย

ความยากอันดับแรกคือ ในการประยุกต์ใช้งานในโลกแห่งความเป็นจริงนั้นต้องการระบบที่มีความถูกต้องสูงมาก เช่น ระบบรักษาความปลอดภัย และ ATM Cash Machine ที่ใช้การรู้จำภาพใบหน้าในการระบุบุคคลนั้นจะต้องมีความเร็วสูงแบบเวลาจริงและต้องไม่มีการผิดพลาดได้เลย ในการประยุกต์ทางด้านตรวจจับคนที่เจ้าหน้าที่ทางกฎหมายต้องการจับตัวและผู้ลี้ภัยนั้น ระบบที่ติดตั้งที่สถานที่สาธารณะต่างๆ และสนามบินนั้นระบบจะต้องสามารถเลือกสรรค์คัดบุคคลผิดกฎหมายที่ต้องการนั้นออกมาให้ได้ในขณะที่ทุกคนต่างก็กำลังเดินหรือเคลื่อนไหวกันอยู่ ซึ่งเป็นเรื่องที่ยากมาก ดังนั้นระบบส่วนใหญ่ที่ทำกันได้ หรือบทความที่วิจัยกันอยู่ต่างก็มีขอบเขตความสามารถและมีขีดจำกัดในการประยุกต์ด้วยกันทั้งหมดทั้งสิ้น ขึ้นอยู่กับจะบอกให้ผู้อื่นรับรู้หรือไม่เท่านั้นเอง

ความยากอันที่สอง คือ ในการรู้จำภาพใบหน้าที่มีประสิทธิภาพจริงๆ นั้นไม่สามารถใช้การประมวลผลภาพแบบเก่าๆ เช่น การเทียบภาพ (Image Matching) ได้ ซึ่งส่วนที่ท้าทายนั้นแบ่งเป็นสองส่วน คือ ส่วนแรกเป็นการตรวจจับใบหน้า ที่จะต้องทำการตัดสินใจว่าในภาพนั้นมีภาพใบหน้าอยู่หรือไม่ และอยู่ตรงตำแหน่งใดในภาพ และส่วนที่สองการรู้จำใบหน้าที่จะต้องระบุได้ว่าเป็นบุคคลใด จากฐานข้อมูลที่มีอยู่ ซึ่งใบหน้านั้นยากในการที่จะตรวจจับและรู้จำทั้งนี้เพราะข้อเท็จจริงที่ว่ามันมีรูปแบบไม่ตายตัว ซึ่งใบหน้าอาจเปลี่ยนแปลง ตำแหน่ง ขนาด พื้นหลังต่างๆ การส่องสว่าง การแสดงออกทางอารมณ์ การวางตัวของศีรษะ ทรงผม คิ้ว หนวด ใส่แว่น สวมหมวก ซึ่งสิ่งเหล่านี้เป็นเรื่องน่ากลัวมาก อีกทั้งถ้าเป็นภาพจากกล้องวีดีโอจะต้องแก้ปัญหา เรื่องการเคลื่อนที่ของภาพด้วย

**2.3 การนำระบบรู้จำใบหน้าไปใช้จริงในปัจจุบัน** มี 4 ประเภทดังต่อไปนี้

**2.3.1 เฝ้าดูบุคคลแปลกหน้า**

เป็นการทำงานอัตโนมัติ ในการเฝ้าดูบุคคลที่ไม่มีอยู่ในฐานข้อมูล มันสามารถส่งเสียงเตือน เมื่อเห็นคนแปลกหน้า และสามารถเก็บบันทึกภาพใบหน้านั้นได้ด้วย ซึ่งทำให้เรา

กลับมาดูภาพใบหน้าคนแปลกหน้าทั้งหมดได้เมื่อต้องการ หรือในภายหลังสิ้นสุดวันวันนั้น ทั้งยังมีการบันทึกเวลาที่บุคคลนั้นเข้ามาให้เห็นด้วย ซึ่งจะสามารถดูได้ง่ายกว่าการดูจากวีดีโอ เพราะมีการเก็บเฉพาะภาพที่เป็นใบหน้าบุคคลแปลกหน้าเท่านั้น ไม่ได้บันทึกแบบต่อเนื่องซึ่งต้องใช้เวลาในการดูและค้นหาอย่างมาก

**2.3.2 เฝ้าดูบุคคลที่รู้จัก**

เป็นการติดตามบุคคลในครอบครัวหรือบุคคลในฐานข้อมูล ที่เข้ามาและออกไป ตัวอย่างเช่น ถ้าตำรวจต้องการเฝ้าดูบุคคลต่างๆ ที่เข้าและออกอาคาร ก็สามารถใช้ระบบดังกล่าวในการบันทึกเวลาและดูความบ่อยเพียงใดที่เขาและเธอเข้ามาให้ระบบตรวจจับมองเห็นได้ การทำงานลักษณะเช่นนี้สามารถบันทึกข้อมูลวันเวลาและจำนวนครั้งที่บุคคลคนนั้นเข้ามาให้เห็นได้ด้วย

**2.3.3 ล็อคหน้าจอ (Screen Lock)**

เป็นการทำงานที่เหมือนกับการล็อคหน้าจอแบบปกติที่ใช้การป้อนรหัสผ่านเพื่อเข้าสู่ระบบคอมพิวเตอร์หรือระบบเครือข่ายนั่นเอง เว้นแต่ว่ามันจะมีการปลดล็อคอัตโนมัติเมื่อผู้ใช้กลับมาและมองไปที่กล้องจับภาพ หมายความว่าจะไม่ยอมให้บุคคลอื่นๆ เข้ามาใช้เครื่องหรือเข้าสู่ระบบได้ ระบบนี้สามารถออกแบบให้ปลดล็อคเมื่อผู้สั่งล็อคเพียงคนเดียวที่ยินยอมนั้นกลับมาอีกครั้ง หรือสามารถออกแบบให้ปลดล็อคเมื่อกลุ่มใบหน้าที่อยู่ในฐานข้อมูลคนใดคนหนึ่งกลับมาก็ได้

**2.3.4 ค้นหาในฐานข้อมูลภาพ**

การทำงานนี้สามารถค้นหาภาพใบหน้าในฐานข้อมูลภาพขนาดใหญ่ หรือเป็นฐานข้อมูลภาพเคลื่อนไหวที่เป็นวีดีโอขนาดใหญ่ก็ได้ โดยจะมีการกวาดหาอย่างอัตโนมัติในทุกๆ ภาพหรือทุกๆ เฟรมของวีดีโอ ที่มีภาพบุคคลที่เราต้องการหาซึ่งอาจจะต้องการหาหลายคนในคราวเดียว ตัวอย่างเช่น การค้นหาภาพจากแถบบันทึกวีดีโอ ซึ่งเป็นสิ่งที่เป็นเรื่องที่น่าเบื่อ และเป็นกระบวนการที่ต้องใช้เวลามากคือการค้นหาภาพบุคคลที่ต้องการ ซึ่งหากวีดีโอมีเป็นจำนวนมากๆ หรือต้องทำบ่อยๆ กลับไปกลับมา จะเป็นเรื่องน่าเบื่อมาก แต่หากใช้โปรแกรมรู้จำใบหน้าทำการค้นหาแทน มันจะสามารถทำได้อย่างไม่รู้จักเหน็ดเหนื่อยในการค้นหาต่อเนื่องกันเป็นชั่วโมงๆ โดยไม่มีการหยุดพัก หรือถ้าเป็นการค้นหาในฐานข้อมูลภาพขนาดใหญ่ เช่นในฐานข้อมูลกรมตำรวจเป็นต้น มันจะเป็นการหาบุคคลที่มีใบหน้าคล้ายกับภาพที่เราต้อง

การค้นหา โดยจะแสดงรายชื่อประวัติและภาพออกมาตามลำดับความคล้าย อาจจะเป็น 5 หรือ 25 อันดับแรก เป็นต้น

**โดยสามารถสรุปอีกครั้งได้สองแนวทางใหญ่ๆ คือ**

1) **กรณีที่ต้องการค้นหาบุคคลในฐานข้อมูลใบหน้าขนาดใหญ่** (เช่น ฐานข้อมูลของกรมตำรวจ) ระบบเช่นนี้จะแสดงภาพและประวัติของบุคคลที่มีความคล้ายออกมา ส่วนใหญ่แนวทางนี้จะใช้ภาพ 1 ภาพ ต่อ 1 คนในระบบฐานข้อมูล และเป็นระบบที่ไม่จำเป็นต้องมีความเร็วแบบเวลาจริงในการทำงาน (Non Real Time)

2) **กรณีที่ต้องการชี้ระบุตัวบุคคลแบบเวลาจริง** (เช่น ในระบบเฝ้าดูเพื่อรักษาความปลอดภัย (Security Monitoring System และ ระบบติดตามตำแหน่ง (Location Tracking System) เป็นต้น หรือ กรณีที่ต้องการอนุญาตให้คนบางกลุ่มเท่านั้นให้ผ่านเข้าสู่ระบบได้ โดยไม่ยินยอมให้บุคคลอื่นๆ ผ่านเข้าไปได้เลย (เช่น การเข้าสู่อาคาร หรือ คอมพิวเตอร์ เป็นต้น) แนวทางเช่นนี้ส่วนใหญ่ใช้ภาพบุคคลๆ ละหลายๆ ภาพสำหรับการทำการฝึก (Training) ให้แก่ระบบ และจะต้องมีการทำงานในแบบเวลาจริง จึงจะสามารถนำไปใช้ได้ในทางปฏิบัติ

**2.4 สิ่งที่เปลี่ยนแปลงได้บนภาพใบหน้าในการรู้จำใบหน้า**

**2.4.1 สภาพแวดล้อม**

การส่องสว่าง

พื้นหลังต่างๆ

**2.4.2 การแสดงออกทางอารมณ์บนใบหน้า**

การขมวดคิ้ว

ลักษณะของปาก (เช่น ยิ้ม)

กระพริบตา

กระดิกหนวด

**2.4.3 อื่นๆ**

ใส่แว่น

สวมหมวก

ทรงผม

**ขนาดศีรษะ เนื่องจากระยะใกล้ไกลจากกล้อง (Scaling)**

ตำแหน่งและการวางตัวศีรษะ (Orientation Rotation Translation)

การเคลื่อนที่ของภาพในการเดินผ่านกล้อง

**2.5 ระบบฐานข้อมูลและภาพที่เข้ามาตรวจสอบแตกต่างกันเพียงใดนั้น มีอยู่ 4 กรณี**

**2.5.1 ฐานข้อมูลภาพเดียว กับภาพที่เข้ามาตรวจสอบ**

ระบบรู้จำที่ใช้ฐานข้อมูลภาพแบบนี้ออกแบบได้ง่ายที่สุด แต่มีวงจำกัดในการประยุกต์ใช้งานได้บางอย่างเท่านั้น เช่น ใช้ภาพจากบัตรในการเข้าสู่อาคารสำนักงานและค้นหาภาพที่ใกล้เคียงกันจากฐานข้อมูลภาพ เป็นต้น

**2.5.2 ฐานข้อมูล 1 ภาพต่อคน ถ่ายต่างสถานที่ต่างเวลากับภาพที่เข้ามาตรวจสอบ**

กรณีนี้ต้องใช้กระบวนการประมวลผลระดับสูงในการรู้จำ เพราะภาพทั้งสองที่นำมาเปรียบนั้นต่างกัน หลายกรณี ถึงแม้เป็นคนๆ เดียวกันก็ตาม แต่กรณีนี้สามารถนำประยุกต์ใช้ประโยชน์ได้มากที่สุด และฐานข้อมูลจะสร้างขึ้นมาได้ง่ายที่สุด

**2.5.3 ฐานข้อมูลหลายภาพต่อ 1 คน ด้านหน้าทั้งหมด เปลี่ยนท่าทาง ต่างสถานที่และเวลา**

กรณีนี้มักใช้ในกระบวนการประมวลผลที่ใช้โครงข่ายประสาทเทียมในการรู้จำ เพื่อให้สามารถเรียนรู้ได้ในหลายๆ ท่าทาง

**2.5.4 ฐานข้อมูลหลายภาพต่อ 1 คน ด้านหน้า และ ด้านข้าง ถ่ายต่างสถานที่ และเวลา**

กรณีนี้มักใช้ในกระบวนการประมวลผลที่ใช้ลักษณะทางเรขาคณิตในการรู้จำ เพื่อให้สามารถแก้ข้อผิดพลาดในการหาจุดเด่นจากภาพด้านหน้าภาพเดียว

**2.6 วิธีการในการรู้จำภาพใบหน้าที่มีการวิจัยกันมาจนถึงปัจจุบัน** ได้มีแนวความคิดในด้านการพิจารณามองเห็นใบหน้า และวิธีการในการตัดสินใจรู้จำใบหน้าที่แตกต่างกัน ซึ่งสามารถแบ่งได้เป็น 6 กลุ่มการวิจัย คือ

2.6.1 การเทียบเทมเพล็ต (Template Matching)

2.6.2 ลักษณะทางเรขาคณิต (Geometrical Features)

2.6.3 การเทียบกราฟ (Graph Matching)

2.6.4 ใบหน้าไอเกน (Eigen Face)

2.6.5 โครงข่ายประสาทเทียม (Neural Network)

2.6.6 ภาพ 3 มิติ

**2.6.1 การเทียบเทมเพล็ต (Template Matching)**

วิธีการการเทียบเทมเพล็ต [1] นี้ทำงานโดยการหาสหสัมพันธ์ของภาพ 2 ภาพโดยตรง ให้ผลอย่างมีประสิทธิภาพโดยมีอัตราในการรู้จำเป็น **100%** เมื่อภาพมีขนาดเดียวกัน มีการวางอยู่ตรงกัน และมีการส่องสว่างของแสงเดียวกัน ถ้าภาพที่เข้ามาทดสอบมีการเปลี่ยนแปลงไปจากเดิม ต้องมีการประมวลผลก่อน (Preprocessing) เพื่อทำให้ภาพเข้าสู่ตำแหน่งหรือการส่องสว่างใกล้เคียงกัน

**2.6.2 ลักษณะทางเรขาคณิต (Gepmetrical Features)**

มีนักวิจัยจำนวนมากที่เลือกรู้จักใบหน้าในลักษณะทางเรขาคณิต คืออาศัยอัตราส่วนหรือสัดส่วนของระยะทางจากตาซ้ายตาขวา จากตาไปจมูก จากจมูกไปปาก รูปร่างของปาก รูปร่างของตา และจากรูปร่างของคาง ซึ่งเป็นลักษณะทางกายภาพของใบหน้า ดังมีรายละเอียดของการวิจัยดังนี้

**Kanade [2]** ได้นำเสนอวิธีการแยกลักษณะเด่นอัตโนมัติ (Automatic Feature Extraction) โดยใช้ อัตราของระยะทาง (Ratios of Distance) โดยมีผลการรู้จำอยู่ระหว่าง **45-75%** ของฐานข้อมูลภาพ 20 คน

**Brunelli และ Poggio [1]** ได้คำนวณเซตของลักษณะทางเรขาคณิต เช่นความกว้างของจมูก ตำแหน่งของปากบนใบหน้า และรูปร่างของคาง โดยมีอัตราการรู้จำ 90% ของฐานข้อมูล 47 คน อย่างไรก็ตามพวกเขาได้แสดงให้เห็นว่าเมื่อพวกเขาใช้วิธีการเทียบเทมเพล็ตอย่างง่ายนั้นให้ผลในการรู้จำที่ 100% โดยใช้ฐานข้อมูลเดียวกัน

**Cox และคณะ [3]** เมื่อไม่นานมานี้เองได้นำเสนอเทคนิคระยะทางผสม (Mixture-Distance) ที่ทำให้ได้อัตราการรู้จำสูงมากถึง **95%** โดยใช้ฐานข้อมูลภาพที่สุ่มมาทดสอบทีละ 96 ภาพจากทั้งหมด 685 ภาพ ซึ่งใบหน้าของแต่ละภาพนั้นจะถูกแทนด้วยระยะทางต่างๆ ถึง 30 อย่าง

ระบบการรู้จำใบหน้าแบบที่มีการพิจารณาลักษณะทางเรขาคณิตนี้ จะสามารถนำไปใช้เป็นระบบที่มีประโยชน์และเป็นไปได้ในทางปฏิบัติในการค้นหาในฐานข้อมูลขนาดใหญ่ ได้ถ้ามีการวัดระยะทางจากจุดที่เป็นลักษณะเด่นได้อย่างถูกต้องเที่ยงตรง ทั้งนี้เพราะข้อมูลที่

ได้จะมีจำนวนน้อยมากเมื่อเทียบกับการพิจารณาแบบอื่น จึงใช้เวลาน้อยกว่าแบบอื่นๆ ในการค้นหา ดังนั้นจึงมีการสร้างระบบฐานข้อมูลขึ้นมาสนับสนุนวิธีการแบบนี้ ซึ่งฐานข้อมูลดังกล่าวเรียกว่าฐานข้อมูลภาพหลายด้าน (Mugshot Database) ทั้งนี้เพราะถ้าเรามีภาพด้านข้างด้วยจะเป็นการช่วยหาจุดที่เป็นลักษณะเด่นได้เที่ยงตรงแม่นยำมากขึ้น แต่ทั้งนี้ ความเที่ยงตรงในการวัดยังขึ้นอยู่กับอัลกอริทึมที่ใช้ด้วย ดังนั้นในปัจจุบันการทำงานอย่างอัตโนมัติก็ยังไม่ค่อยมีความเที่ยงตรงนัก จึงต้องรอกันต่อไป สำหรับวิธีการแบบนี้

**2.6.3 การเทียบกราฟ (Graph Matching)**

เป็นการมองใบหน้าในลักษณะที่เป็นเวกเตอร์ของกราฟซึ่งมีจุดและเส้นในการเชื่อมต่อเป็นโครงร่างที่ฟิตไปบนใบหน้าบนลักษณะเด่นที่พิจารณา เช่น รูปร่างกราฟของตา ปาก คางกับหู และคิ้ว เป็นต้น

**Lader และคณะ [4]** เสนอสถาปัตยกรรมการเชื่อมต่อแบบยืดหยุ่น (Dynamic Link Architecture) สำหรับการรู้จำภาพเป้าหมายแบบไม่ขึ้นกับ ความเพี้ยน โดยใช้การเทียบกราฟแบบยืดหดได้ (Elastic Graph Matching) เพื่อหากราฟในฐานข้อมูลที่ใกล้เคียงที่สุด วัตถุเป้าหมายนั้นจะถูกแทนด้วยกราฟห่างๆ กันที่มีจำนวนไม่มากนัก (Sparse Graph) ซึ่งมุมจะถูกติดป้าย (Label) แทนด้วยฟังก์ชันความโค้งด้วยวิธีการของเขา และขอบต่างๆ ก็ถูกติดป้าย (Label) แทนด้วยระยะทางทางเรขาคณิต เมื่อป้อนภาพเข้ามา 1 ภาพ และค้นหาในฐานข้อมูลจำนวน 87 คน และภาพทดสอบที่มีหลายๆ ลักษณะรวมทั้งมีการหมุน 15 องศาด้วย ใช้เวลาในการทำแมชชิง 25 วินาที โดยใช้ทรานสพิวเตอร์ 23 ตัว ในการประมวลผล แต่บทความดังกล่าวไม่ได้บอกอัตราการรู้จำไว้เลย และเมื่อ Wiskott และคณะ [45] ได้มีการปรับปรุงกระบวนการดังกล่าวนั้นใหม่ โดยทดสอบฐานข้อมูล 300 คน และโดยป้อน 300 ภาพที่แตกต่างกันของบุคคลต่างๆ จากฐานข้อมูล FERET ปรากฏว่ามีผลในการรู้จำสูงมากถึง **97.3%** แต่ไม่ได้บอกเวลาที่ใช้ไว้ในบทความดังกล่าว และตัวประมวลผลที่ใช้เลย ดังนั้นคงเป็นไปได้ว่าระบบดังกล่าวคงใช้ตัวประมวลผลประสิทธิภาพและความเร็วสูงมาก และคงใช้เวลานานในการประมวลผล จึงไม่ต้องการบันทึกไว้ให้เป็นข้อด้อยของระบบของพวกเขา

**2.6.4 ใบหน้าไอเกน (Eigen Face)**

**Turk และ Pentland [5]** จาก MIT ได้ริเริ่มและเสนอวิธีการในการรู้จำใบหน้าโดยทำการฉาย (Projection) ภาพใบหน้าไปยังองค์ประกอบหลัก (Principal Components) โดยเรียกภาพใบหน้าดังกล่าวนั้นว่าใบหน้าไอเกน (Eigenface) โดยนำใบหน้าไอเกนดังกล่าวไปทำการ

ค้นหาในฐานข้อมูลใบหน้าไอเกน โดยพวกเขาได้แสดงให้เห็นว่าโดยการทดสอบกับฐานข้อมูลใบหน้าไอเกน 16 คนที่มีการเลื่อน การวางตำแหน่งศีรษะหลายๆ ลักษณะ มีการย่อขยายและการเปลี่ยนแปลงของแสง โดยการแทนด้วยใบหน้าไอเกนที่ใช้เทคนิคของพวกเขาแล้วจะมีการเปลี่ยนแปลงเพียงเล็กน้อย โดยระบบยังคงมีความสามารถในการรู้จำ เมื่อมีการเปลี่ยนแปลงของแสงมีความสามารถในการรู้จำ **96%** มีการวางตำแหน่งศีรษะ เปลี่ยนแปลงจะมีความสามารถในการรู้จำ **85%** และ มีการย่อขยายจะภาพมีความสามารถในการรู้จำ **64%** โดยในการย่อขยายนั้นพวกเขาใช้อัลกอริทึมในการปรับขนาดศีรษะให้มีขนาดเดียวกับขนาดใบหน้าไอเกนโดยใช้การประมาณขนาดของศีรษะ และนำส่วนกลางของใบหน้ามาใช้เท่านั้นเพื่อลดผลจากการเปลี่ยนแปลงของทรงผม และฉากด้านหลังที่อาจเกิดการเปลี่ยนแปลงได้

**Pentland และคณะ [6,7]** มีผลการรู้จำที่ดีในฐานข้อมูลขนาดใหญ่ โดยมีอัตราการรู้จำ **95%** ในฐานข้อมูล 200 คนที่สุ่มมาจากฐานข้อมูลขนาด 3000 คน

**Mohadam and Pentland [8]** มีผลการ รู้จำที่ดีมาก โดยทดลองกับฐานข้อมูล FERET โดยมีข้อผิดพลาดเพียงตรงที่ใช้ภาพด้านหน้าเพียง 150 ภาพเท่านั้น ระบบมีการทำการประมวลผลก่อน (Preprocessing) เพื่อหาตำแหน่งของศีรษะ ลักษณะเด่น และทำการนอร์มอลไลซ์ ทางเรขาคณิตของใบหน้า โดยการเลื่อนตำแหน่ง การหมุน การย่อขยายภาพ ปรับแสง และปรับคอนทราส ของภาพ

**Swets และ Weng [9]** เสนอวิธีของ Selecting Discriminant Eigenfeature โดยใช้ Multidimensional Linear Discriminant Analysis พวกเขาเสนอวิธีการในการหา Most Expression Feature (MEF) และ Most Discriminatory Feature (MDF) แต่ไม่มีผลในการรู้จำที่เทียบกับใบหน้าไอเคนแบบเดิมให้ดูในบทความของพวกเขา

โดยสรุปแล้ววิธีการใช้ใบหน้าไอเกนนั้น เป็นวิธีที่รวดเร็วและง่าย สามารถใช้ได้ในทางปฏิบัติ แต่อย่างไรก็ตามขีดจำกัดจะอยู่ที่มันต้องการความคล้ายกันมากของจุดภาพความสว่างของภาพในฐานข้อมูลกับภาพที่เข้ามาทดสอบ ซึ่งต้องใช้กระบวนการประมวลผลก่อน (Preprocessing) ที่ดีในการทำนอร์มอลไลซ์ภาพให้ใกล้เคียงกัน

### 2.6.5 โครงข่ายประสาทเทียม (Neural Network)

จนถึงปัจจุบันมีการเสนอบทความที่ใช้โครงข่ายประสาทเทียม ในการรู้จำใบหน้ามาก มาย แต่ส่วนใหญ่แล้วน่าเสียดายที่วิธีการต่างๆ ที่ เสนอนั้น มีการทดสอบกับฐานข้อมูลขนาดเล็กๆ ได้เท่านั้น (ต่ำกว่า 20 คน) จะมีที่ใช้ฐานข้อมูลขนาดใหญ่มีไม่มากบทความนัก

**Demers [10]** ใช้สัมประสิทธิ์ 50 ตัวแรกจากส่วนประกอบหลัก (Principal Component) ที่ดึงมาจากภาพ แล้วนำมาใช้โครงข่ายประสาทเทียมแบบมาตรฐานของเปอเซบตรอนหลายชั้น (Standard Multilayer Perceptron) ในการรู้จำ ผลการรู้จำค่อนข้างดี แต่ฐานข้อมูลที่ใช้ค่อนข้างง่ายเกินไป และภาพไม่มีการเปลี่ยนแปลงของแสง ไม่มีการเปลี่ยนแปลงจากการหมุน โดยใช้ฐานข้อมูลแค่ 20 คนเท่านั้น

**Lawrence และคณะ [11]** ได้ใช้โครงข่ายประสาทเทียมแบบผสม (Hybrid Neural Network) โดยใช้การผสมของโครงข่ายประสาทเทียมแบบ Self-Organizing Map Neural Network (SOM) ในการลดมิติ (Quantization) ของข้อมูลภาพใบหน้า ไปในโทโปโลยีสเปส (Topological Space) และโครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolution Neural Network : CNN) ในการรู้จำ ซึ่ง Lawrence และคณะได้เปรียบเทียบผลการรู้จำของพวกเขา ได้ดังนี้

1) SOM ในการลดมิติข้อมูล + CNN ในการรู้จำ มีความถูกต้องในการรู้จำ **96.2%**

2) การแปลงแบบคาร์ฮูเนนเลิฟ (KL Transform) ในการลดข้อมูล + CNN ในการลดข้อมูลมีความถูกต้อง **94.7%**

3) SOM ในการลดมิติข้อมูล+โครงข่ายประสาทเทียมแบบเปอเซบตรอนหลายชั้น (Multilayer Perceptron) ในการรู้จำ มีความถูกต้อง **60%**

4) ใบหน้าไอเกนมีความถูกต้อง **89.5%**

โดยฐานข้อมูลที่ Lawrence และคณะ ใช้ทดสอบระบบเป็นภาพบุคคลที่มีท่าทางแตกต่างกันคนละ 10 ภาพเป็นจำนวน 40 คน ซึ่งภาพนี้ถ่ายมาต่างเวลากัน ตั้งแต่เดือนเมษายน ปีคศ. 1992 ถึงปีคศ. 1994 เดือนเดียวกัน ภาพมีการเปลี่ยนแปลงของการแสดงออก เช่น การเปิดปิดตา ยิ้มหรือไม่ยิ้ม และ มีแว่นหรือไม่มีพื้นฉากหลังเป็นสีดำเนื้อเดียวกันทั้งหมด และศีรษะวางอยู่ประมาณตรงกลางภาพถ่าย ซึ่งบางภาพอาจมีการไปหมุนประมาณ 20 องศา และบางภาพอาจมีการเปลี่ยนแปลงขนาดประมาณ 10% รูปแบบของภาพนั้นเป็นภาพระดับสีเทาขนาด 92x112 จุดภาพ

### 2.6.6 ภาพ 3 มิติ

การมองภาพใบหน้าเป็น 3 มิติ เป็นวิธีการชั้นสูงและเป็นวิธีที่ดีมากในปัจจุบัน และมีการทำเป็นธุรกิจผลิตซอร์ฟแวร์เพื่อเป็นการค้าเรียบร้อยแล้ว

**FaceIt** คือชื่อซอร์ฟแวร์ทางการค้าของบริษัท Visionics Corporation ในการรู้จำภาพใบหน้าในรูปแบบ 3 มิติแบบเวลาจริงที่ไม่ต้องการฮาร์ดแวร์เพิ่มพิเศษในโลกแห่งความเป็น

จริง ที่เป็นผลงานวิจัยที่ใช้เวลานับสิบปีโดยกลุ่มของ Professor Joseph Atick ของ Computer Nero-Science Laboratory ที่ Rockefeller University และของ US Army Research Laboratory รัฐนิวยอร์ค ประเทศสหรัฐอเมริกา โดยเป็นผู้ชนะในการแข่งขันชิงชัยระหว่างระบบรู้จำใบหน้าด้วยกันซึ่งจัดโดยรัฐบาลสหรัฐ เมื่อเร็วๆ นี้ และได้รางวัลซอร์ฟแวร์ดีเด่นแห่งปีจากนิตยสารไบต์ในปีเดียวกัน โดยเป็นโปรแกรมการให้ทุนจาก US Defense Advanced Research Project โดยการแข่งขันได้ทดสอบโดยใช้ฐานข้อมูลมาตรฐานคือ FERET database ที่เป็นฐานข้อมูลภาพถ่าย 1 ภาพต่อ 1 คน ที่ประกอบด้วยคนต่างวัย ต่างสีผิว และต่างท่าทาง โปรแกรม Faceit ในยุคแรกทำงานบน Silicon Graphics และพัฒนามาเป็นการค้าบนเครื่องเพนเทียมระบบปฏิบัติการ Windows 95 และ Windows NT ที่มีฟังก์ชันการทำงานครบทั้งในระบบรักษาความปลอดภัย และค้นหาภาพในฐานข้อมูลที่ส่งภาพที่คล้ายจำนวน 25 ภาพออกมาเรียงตามลำดับ หลักการที่ใช้เป็นการแก้ข้อเสียเปรียบต่างๆ ของใบหน้าไอเกนได้ทั้งหมด โดยเขาไม่มองเป็นลักษณะกว้าง (Global) แต่มองเป็นจุดๆ (Local)

ภาพที่ 2.1


หน้าตาของโปรแกรม FaceIt V1.0

FaceIt ใช้การมองภาพในรูปแบบสเตอริโอของภาพสามมิติที่ประกอบด้วยบล็อคความสูง โดยเขาเรียกว่าหัวไอเกน (Eigen Head) [12] ที่เป็นตัวแสดงภาพสามมิติของศีรษะ ที่ได้มาจากข้อมูลเชดสี (Shading Information) ของภาพสองมิติ ซึ่งหัวไอเกนนี้จะไม่ขึ้นอยู่กับแสงของการส่องสว่าง และท่าทางการวางศีรษะ และมีการส่งภาพ 3 มิติที่มีการนอร์มอลไลต์แล้วแปลงไปเป็นรหัสด้วยการวิเคราะห์ลักษณะเด่นแบบท้องถิ่น (Local Feature Analysis) [13] เช่น จมูก ปาก แก้ม กระดูก และแนวขากรรไกร ซึ่งจะสร้างสิ่งที่เป็นเอกลักษณ์ของบุคคลคนเดียวกันที่เรียกว่าพิมพ์ของใบหน้า (Faceprint) พิมพ์ของใบหน้านี้สามารถนำไปค้นหาในฐานข้อมูลได้แบบเวลาจริง (Real Time) และมีความรวดเร็วในการประมวลผลสูงมากจนกระทั่ง

สามารถทำกับภาพที่เป็นภาพนิ่งและกับภาพจากวีดีโอก็ได้ ซึ่งความสามารถจริงของระบบอยู่ที่ความเร็ว และการแก้ปัญหาเรื่องการขมวดคิ้ว การยิ้ม การกระพริบตา การใส่แว่น ปัญหาของแสง การย่อขยาย การวางตำแหน่งศีรษะ การเอียง ได้ทั้งหมด อีกทั้งยังเป็นแนวทางที่ใช้ลักษณะสามมิติที่เป็นลักษณะเด่นแบบท้องถิ่นคือ จมูก ปาก แก้ม กระดูก และแนวขากรรไกร ที่สามารถจะพัฒนาเป็นการประมาณใบหน้าเมื่ออายุเพิ่มมากขึ้นหรือลดลงได้อีกด้วย จึงเป็นสุดยอดแนวทางของการรู้จำใบหน้าอย่างแท้จริงที่สามารถใช้ในการค้นหาประวัติของเด็กหลงทางที่ไม่สามารถบอกบ้านหรือพ่อแม่ได้ หรือค้นหาคนที่หายไปหลายๆ ปีได้ว่าจะมีหน้าตาปัจจุบันเป็นอย่างไร

# บทที่ 3
# ทฤษฎีพื้นฐานที่เกี่ยวข้อง

## 3.1 การกำจัดสัญญาณรบกวนโดยใช้ตัวกรองแบบเลือกค่ากลาง (Median Filter)

Median Filter [14] เป็นการกำจัด Noise ที่มีความแตกต่างจากจุดอื่นมากๆ เช่น Impulse Noise ซึ่งจะทำให้เกิดเป็นจุดดำหรือจุดขาวในภาพ วิธีการคือการนำหน้ากาก ทาบลงไปบนจุดที่สนใจ เช่นใช้หน้ากากขนาด 3x3 โดยจุดที่สนใจอยู่ตรงกลางหน้ากากนั้น จากนั้นทำการเรียงค่าระดับสีเทาของจุดภาพต่างๆ ในหน้ากาก จากน้อยไปมาก แล้วทำการเลือกเอาค่าที่อยู่กึ่งกลาง หรือค่า Median มาแทนลง ณ จุดที่สนใจ ดังนั้น ด้วยวิธีการนี้ Impulse Noise จึงสามารถถูกกำจัดไปได้

ตัวอย่างเช่น ถ้าหน้ากากสองมิติขนาด 3x3 ทาบลงไปบนข้อมูลที่สนใจ โดยมีค่าระดับสีเทาดังนี้

$$\begin{bmatrix} 59 & 65 & 63 \\ 60 & \underline{255} & 80 \\ 120 & 110 & 95 \end{bmatrix}$$

ทำการเรียงข้อมูลในหน้ากาก จากน้อยไปหามาก คือ

59,60,63,65,<u>80</u>,95,110,120,255

จะพบว่าค่า Median ที่ได้คือ 80 ดังนั้น จุดเดิม ซึ่งมีค่าเป็น 255 ซึ่งถือว่าเป็น Impulse Noise จะถูกกำจัดไปโดยจะถูกแทนที่ด้วยค่า 80 ซึ่งจะได้ข้อมูลหลังการกรองเป็นดังนี้

$$\begin{bmatrix} 59 & 65 & 63 \\ 60 & \underline{80} & 80 \\ 120 & 110 & 95 \end{bmatrix}$$

**3.2 การหาขอบภาพโดยใช้วิธีโซเบล (Sobel Method)**

ในการหาขอบภาพด้วย Sobel Operator [14] ซึ่งเป็น Mask แบบมีทิศทาง สามารถหาความรุนแรงในการเปลี่ยนแปลงของขอบซึ่งเรียกว่าเกรเดียนท์ (Gradient) ได้ดังนี้

$$\nabla(x,y) = \sqrt{\nabla_x(x,y)^2 + \nabla_y(x,y)^2} \tag{3.1}$$

โดยที่

$$\nabla_x(x,y) = \sum_{i=1}^{3}\sum_{j=1}^{3}\left[f(x-2+i, y-2+j) * M_x(i,j)\right] \tag{3.2}$$

$$\nabla_y(x,y) = \sum_{i=1}^{3}\sum_{j=1}^{3}\left[f(x-2+i, y-2+j) * M_y(i,j)\right] \tag{3.3}$$

เมื่อ

$$M_x(i,j) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad M_y(i,j) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.4}$$

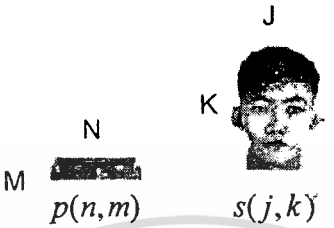
**3.3 การตรวจหาภาพใบหน้า (Face Detection) ด้วยการเทียบภาพด้วยสหสัมพันธ์และการย่อภาพสามระดับ**

ในการค้นหาบริเวณที่ต้องการโดยเฉพาะภาพใบหน้านั้นไม่สามารถหาขอบวงปิดของภาพ (Contour) ได้ ทำให้ไม่สามารถนำหลายๆ วิธีที่มีความเร็วสูงและมีประสิทธิภาพจากการรู้จำรูปร่างขอบวงปิดของภาพที่สนใจนั้นด้วยวิธีการทางการประมวลผลภาพวิธีอื่นๆ ได้ ดังนั้นจึงจำเป็นจะต้องใช้วิธีการเทียบภาพเพื่อหาใบหน้า ซึ่งวิธีนี้เป็นวิธีการเก่าแก่และมีการทำงานค่อนข้างช้ามาปรับปรุงแนวความคิดใหม่และเพิ่มเทคนิคต่างๆ เข้าไปเพื่อให้เหมาะสมกับงานและเพื่อให้มีความเร็วสูงขึ้นจนสามารถใช้งานได้จริงในทางปฏิบัติโดยเฉพาะกับการค้นหาใบหน้า ดังที่จะได้เสนอต่อไปนี้

การทำสหสัมพันธ์ (Correlation) [15] นั้นเป็นการหาความสัมพันธ์ของข้อมูลภาพใดๆ ของภาพสองภาพ ซึ่งทำให้ทราบได้ว่าข้อมูลหรือวัตถุของทั้งสองชิ้นนั้นเป็นชนิดเดียวกันหรือเป็นพื้นที่เดียวกันหรือไม่ ซึ่งสามารถนำมาใช้หาบริเวณที่ภาพทั้งสองซ้อนทับกันได้ดีที่สุด ภาพที่ 3.1 แสดงภาพสองภาพ ในที่นี้ภาพเล็ก 3.1(a) เป็นบริเวณหรือวัตถุที่สนใจนั้นขอเรียกว่า

"ภาพอ้างอิง" และภาพอีกภาพหนึ่งที่เป็นภาพใหญ่กว่า 3.1(b) นั้น เป็นภาพที่มีส่วนของภาพเล็กประกอบอยู่ภายในว่า "ภาพที่ต้องการค้นหา"

ภาพที่ 3.1




(a) ภาพอ้างอิง (b) ภาพที่ต้องการค้นหา

วิธีการทำสหสัมพันธ์ที่ใช้ในที่นี้ [15] เป็นวิธีการเปรียบเทียบค่าความแตกต่างระหว่างจุดภาพต่อจุดภาพของทั้งสอง ซึ่งเรียกว่า Mean normalized correlation ซึ่งไม่ขึ้นกับความแตกต่างของแสงของภาพทั้งสอง แต่จะขึ้นกับลักษณะหรือโครงสร้างของภาพ ดังสมการสหสัมพันธ์แบบนอร์มอลไลซ์ดังสมการที่ 3.5

$$R(j,k)=\frac{\sum_{n=1}^{N}\sum_{m=1}^{M}(p(n,m)-\bar{p})(s(j+n-1,k+m-1)-\bar{s}(j,k))}{\left(\sqrt{\sum_{n=1}^{N}\sum_{m=1}^{M}(p(n,m)-\bar{p})^2}\sqrt{\sum_{n=1}^{N}\sum_{m=1}^{M}(s(j+n-1,k+m-1)-\bar{s}(j,k))^2}\right)} \qquad (3.5)$$

เมื่อภาพอ้างอิงคือ $p$ และภาพที่ต้องการค้นหา คือ $s$ โดยค่าเฉลี่ยของภาพอ้างอิง คือ

$$\bar{p}=\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}p(n,m) \qquad (3.6)$$

ค่าเฉลี่ยของภาพที่ต้องการค้นหาตำแหน่งพิกัดที่ตรงกันกับภาพอ้างอิง คือ

$$\bar{s}(j,k)=\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}s(j+n-1,k+m-1) \qquad (3.7)$$

เมื่อ $R(j,k)$ คือค่าสัมประสิทธ์ของการทำสหสัมพันธ์ (correlation coefficient) ที่ตำแหน่งพิกัดภาพที่ทำการค้นหา

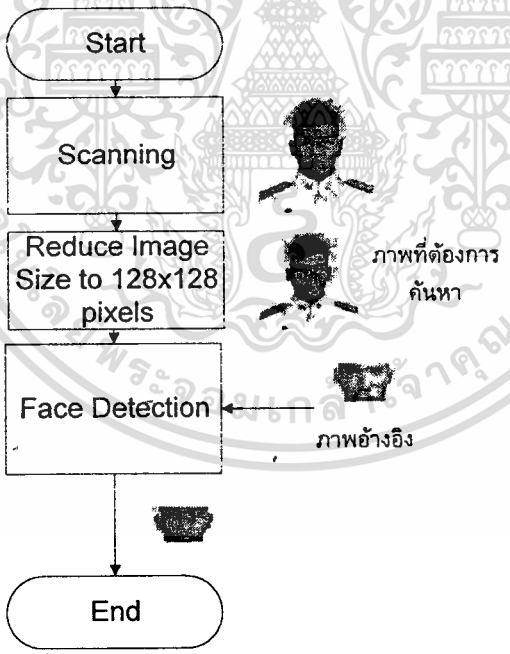
$p(n,m)$ คือค่าความสว่างของจุดภาพในภาพย่อยที่พิกัด $n,m$

$s(j,k)$ คือค่าความเข้มของจุดภาพในภาพใหญ่ ที่ต้องการค้นหาที่พิกัด $j,k$

ค่าสัมประสิทธิ์ของสหสัมพันธ์ $R(j,k)$ จะมีค่าเปลี่ยนแปลงในช่วง $[-1, 1]$ และเมื่อเราได้ผลลัพธ์ $R(j,k) = 1$ ก็จะเป็นค่าดีที่สุดในการซ้อนทับ นั่นคือหมายความว่าภาพทั้งสองซ้อนทับกันได้สนิทพอดี ในทางปฏิบัติมักไม่พบค่าที่ได้ความคล้ายสูงสุดเป็น 1 ยกเว้นจะเป็นภาพเดียวกัน ดังนั้นจึงต้องทำการหาค่า $R(j,k)$ ในตำแหน่ง $j,k$ ต่างๆ แล้วเลือกค่าที่มากที่สุดออกมาเป็นพิกัดของภาพในตำแหน่งที่ซ้อนทับกันได้ดีกว่าที่อื่น

ตัวอย่างการนำวิธีการสหสัมพันธ์มาใช้ในการค้นหาภาพใบหน้าแสดงดังภาพที่ 3.2

ภาพที่ 3.2




กระบวนการในการทำสหสัมพันธ์เพื่อการเทียบภาพ

ตัวอย่างภาพผลลัพธ์จากวิธีการสหสัมพันธ์แสดงดังภาพที่ 3.3 โดยภาพต้นแบบคือภาพที่ 3.3 (a) เป็นภาพที่มีขนาด 128x128 ซึ่งจะเห็นว่าภาพมีขนาดของใบหน้าไม่เท่ากัน ดังนั้นต้องมีการปรับขนาดใบหน้าให้เท่ากันหรือใกล้เคียงกันก่อน ดังแสดงในภาพที่ 3.3(b) ซึ่ง

ภาพดังกล่าวได้มาจากวิธีการหาระยะทางจากหูซ้ายไปหูขวา เพื่อใช้กำหนดอัตราส่วนที่จำเป็นต้องใช้การย่อภาพ เพื่อให้มีการลดขนาดของภาพลง โดยให้มีความกว้างจากหูซ้ายไปหูขวาเป็น 32 จุดภาพเท่ากันทุกภาพโดยความสูงลดลงมาด้วยอัตราส่วนการย่อเดียวกันแต่อาจไม่ใช้สูงเป็น 32 จุดภาพก็ได้ สุ่มเลือกภาพอ้างอิงดังภาพ 3.3(c) แล้วเมื่อทำการค้นหาในภาพต้นแบบต่างๆ จะได้ภาพใบหน้าดัง 3.3(d) และอีกตัวอย่างหนึ่งกรณีที่ต้องการค้นหาเฉพาะบริเวณตาแสดงภาพอ้างอิงที่สุ่มเลือกขึ้นมาได้ดังภาพ 3.4(a) แล้วเมื่อทำการค้นหาในภาพต้นแบบต่างๆ จะได้ภาพบริเวณตาดังภาพ 3.4(b)

ตัวอย่างที่ 1 ย่อภาพให้ได้ 128x128 จุดภาพ และย่อภาพอีกครั้งให้ได้ระยะทางจากหูซ้ายไปหูขวาเท่ากับ 32 จุดภาพ โดยมีความสูงตามสัดส่วน

ภาพที่ 3.3

(a) ภาพต้นแบบที่มีขนาด 128 x 128

(b) ภาพที่นำมาปรับขนาดของใบหน้าและจะนำมาเป็นภาพใช้ในการค้นหา

(c) ภาพอ้างอิง

(d) ภาพผลลัพธ์ที่ได้จากการค้นหา

ภาพตัวอย่างจากการค้นหาใบหน้า จากภาพต้นแบบที่มีการปรับขนาดของใบหน้าให้มีขนาดใกล้เคียงกัน

ตัวอย่างที่ 2 เฉพาะบริเวณตา

ภาพที่ 3.4

(a) ภาพอ้างอิง

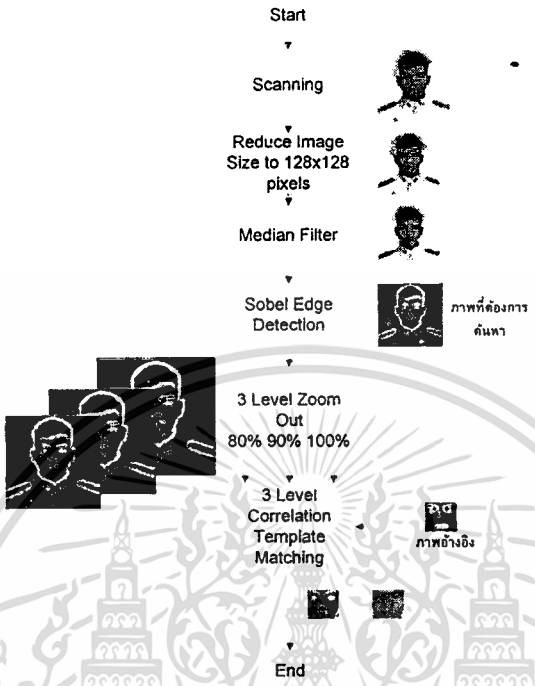
(b) ภาพผลลัพธ์ที่ได้จากการค้นหา

ภาพตัวอย่างจากการค้นหาบริเวณตา จากภาพต้นแบบที่มีการปรับขนาดของใบหน้าให้มีขนาดใกล้เคียงกัน

ด้วยวิธีการทำสหสัมพันธ์ (Correlation) เพื่อเทียบภาพแบบดั้งเดิมนั้นไม่สามารถใช้ได้ถ้าวัตถุมีการเปลี่ยนแปลงขนาด และการในการปรับขนาดของใบหน้าให้มีขนาดใกล้เคียงกัน ดังตัวอย่างในภาพ 3.3(b) โดยการค้นหาจุดเริ่มต้นของหูซ้ายและหูขวาเพื่อวัดระยะทางเพื่อใช้ในการกำหนดอัตราการย่อนั้นจะถูกรบกวนได้ง่าย เช่น ถ้าพื้นด้านหลังเป็นภาพรายละเอียดต่างๆ อยู่ด้วย หรือไม่ใช้สีพื้นขาว หรือมีลายเส้นอยู่ การหาจุดเริ่มต้นของหูดังกล่าวจะเป็นการยาก และจากการทดลองกับภาพทั้งหมดที่มีคือจำนวน 163 ภาพ และภาพโดยทั่วไปอีกจำนวนหนึ่ง ผลปรากฏว่าไม่สามารถปรับขนาดของใบหน้าด้วยวิธีการดังกล่าวได้เป็นจำนวนมาก ตัวอย่างเช่นภาพมีผมมากจนปิดบริเวณหูจนหมดเป็นต้น อีกทั้งถ้ามีการพิมพ์หมึกสัญลักษณ์ขององค์กร การค้นหาบริเวณหูก็มีโอกาสผิดพลาดเยอะมาก

ดังนั้นจึงได้ทำการปรับปรุงวิธีการใหม่โดยนำการย่อภาพสามระดับเข้ามาประยุกต์เพื่อให้สามารถทำการเทียบภาพได้แม้วัตถุมีการเปลี่ยนแปลงขนาด และผลจากการเทียบภาพท้ายสุดจะได้ภาพที่มีขนาดใกล้เคียงกันออกมาอย่างอัตโนมัติ โดยมีความไวต่อการรบกวนจากสิ่งแวดล้อมต่างๆ ในภาพไวน้อยกว่ามาก จากกระบวนการและแนวความคิดที่ใช้ จะเป็นการเฝ้ามองหาเพียงว่า ส่วนไหนในภาพที่ต้องการค้นหาคือส่วนที่คล้ายกับภาพอ้างอิงที่สุด ถึงแม้มีการพิมพ์หมึกตราสัญลักษณ์ของหน่วยงานถ้าตราประทับไม่คล้ายภาพใบหน้ามากกว่าตัวใบหน้าเองก็จะไม่มีผลกระทบแต่อย่างใดเลย ดังการทำงานในภาพที่ 3.5

ภาพที่ 3.5




การค้นหาภาพที่ใช้การย่อภาพสามระดับกับวิธีสหสัมพันธ์

การทำงานเริ่มจากนำภาพขนาดใดๆ (จากการสแกนมีหน่วยเป็น dpi) มาย่อให้เล็กลงเป็นขนาด 128x128 จุดภาพ เพื่อลดขนาดของข้อมูลซึ่งจะทำให้คำนวณในขั้นตอนต่างๆ ได้รวดเร็วขึ้น จากนั้นจะทำการกำจัดสัญญาณรบกวนในภาพโดยใช้ตัวกรองแบบเลือกค่ากลาง (Median Filter) และนำมาหาขอบภาพโดยใช้วิธีการโซเบล (Sobel Method) แล้วเตรียมภาพขอบที่มีการย่อเป็น 3 ระดับ คือ ระดับปกติ ระดับที่ย่อเหลือ 90 % และ 80% เมื่อได้ภาพดังกล่าวเราจะนำมาเทียบกับภาพขอบอ้างอิงด้วยวิธีการสหสัมพันธ์ โดยจะเทียบหาบริเวณที่คล้ายที่สุดในภาพขอบขนาดปกติ บันทึกค่าความคล้ายและพิกัด x,y ที่พบภาพดังกล่าวไว้ แล้วหาบริเวณที่คล้าย ที่สุดในภาพขอบขนาด 90% บันทึกค่าความคล้ายและพิกัด x,y ที่พบภาพดังกล่าวไว้ และหาบริเวณที่คล้ายที่สุดในภาพขอบขนาด 80% บันทึกค่าความคล้ายและพิกัด x,y ที่พบภาพดังกล่าวไว้ สุดท้ายก็นำค่าความคล้ายทั้งสามมาเปรียบเทียบกันว่าภาพใดได้ความคล้ายสูงสุด เราจะได้คำตอบเป็นภาพนั้น แต่นั่นไม่ใช่คำตอบที่เราต้องการ สิ่งที่เราต้องการคือภาพบริเวณใบหน้าที่ค้นพบ ดังนั้นในกระบวนการที่ผ่านมาเราได้มีการบันทึกพิกัดที่ค้นพบภาพเอาไว้ด้วย เราจึงสามารถไปดึงภาพใบหน้าจากภาพการย่อที่มีความคล้ายภาพอ้างอิง

มากที่สุดนั้นออกมาเป็นผลลัพธ์ที่แท้จริง โดยไปดึงมาจากทั้งภาพระดับสีเทาและภาพขอบที่มีการย่อด้วยอัตราส่วนดังกล่าว เพื่อไปเลือกใช้งานต่อไป

สาเหตุที่มีการใช้ภาพขอบมาทำสหสัมพันธ์นั้นเนื่องจากผลจากการทดลองพบว่าการทำ สหสัมพันธ์ในการใช้ภาพระดับสีเทาเป็นภาพอ้างอิงและทำการเทียบภาพกับภาพต่างๆ ที่เข้ามานั้น บ่อยครั้งที่ค้นพบผิดพลาดไปจากส่วนที่เป็นใบหน้า แต่ไปเจอส่วนที่คล้ายๆ กันได้บ่อย เช่น กระดุมคอ กับดวงตา ดังนั้นจึงได้ทดลองนำภาพขอบมาทำสหสัมพันธ์ ก็พบว่ามีความถูกต้องมากกว่ามาก จะผิดพลาดกรณีเดียวเท่านั้น คือส่วนที่เป็น กระดุมที่ปกเสื้อ กับกระดุมที่หน้าอก ด้านหน้า ที่รวมกันได้เป็น ตา และจมูก ปาก เท่านั้น เอง ซึ่งก็สามารถแก้ปัญหาได้ โดยง่าย เพียง ไม่กวาดค้นหาให้พ้นไปจากครึ่งหนึ่งของภาพเท่านั้น ก็สามารถแก้ปัญหาดังกล่าวได้ ดังนั้น ถึงแม้การหาของภาพจะทำให้การทำงานช้าลงไปบ้างก็ตามแต่ก็ได้ความถูกต้องแลกมาได้อย่างคุ้มค่าทีเดียว ถ้าจะสรุปเหตุผลที่การใช้ภาพขอบทำไมถึงดีกว่าการใช้ภาพระดับสีเทาในการเทียบภาพด้วยสหสัมพันธ์ ก็คือ เนื่องจากต้นกำเนิดในการทำสหสัมพันธ์ในกระบวนการประมวลผลภาพนั้นเริ่มต้นใช้ในการหาจุดตัดของถนน หรือสิ่งปลูกสร้างที่เป็นเส้นเด่นขึ้นมาอย่างชัดเจน เพื่อกำหนดเป็นจุดควบคุมบนพื้นดิน (Ground Control Point) เพื่อเป็นจุดในการแก้ภาพที่รูปร่างผิดปกติทางตรีโกณมิติ (Geometric Correction) และใช้ในการต่อภาพถ่ายหลายๆ ภาพ (Moseic) ที่มีบริเวณที่ซ้อนทับกันได้ (Overlap) ให้เป็นภาพเดียวกัน ซึ่งต่างต้องการหาจุดที่ตรงกันในภาพสองภาพ ซึ่งจะใช้สหสัมพันธ์เพื่อการเทียบภาพหาจุดที่สอดคล้องกันดังกล่าว ซึ่งส่วนใหญ่จะใช้จุดตัดของถนนหรือสิ่งปลูกสร้างที่ชัดเจนมากๆ ซึ่งหมายความว่าการทำสหสัมพันธ์จะมีการมองถึงความคล้ายกันในทางโครงสร้างมากกว่าจะมองความสว่างของภาพเพื่อมาเทียบกัน ดัง นั้น ภาพขอบที่เรานำมาใช้จะเป็นตัวกำหนดโครงสร้างคล้ายจุดตัดของถนนต่างๆ ได้ดีกว่าภาพระดับสีเทานั่นเอง

ผลจากการปรับปรุงวิธีการ พบว่าสามารถทำให้การค้นหาภาพได้ดียิ่งขึ้นมาก ถึงแม้ภาพที่ป้อนเข้ามาจะมีการเปลี่ยนแปลงขนาดใบหน้าก็ตาม ตัวอย่างภาพที่ถูกสแกนมาด้วยความละเอียด 100 dpi แสดงดังภาพที่ 3.6(a) จะมีขนาดภาพแตกต่างกันเนื่องจากขนาดของรูปถ่ายไม่เท่ากัน ซึ่งจะถูกนำมาปรับให้มีขนาด 128x128 จุดภาพ ดังภาพที่ 3.6(b) แล้วนำมาทำการกำจัดสัญญาณรบกวนโดยใช้ Median Filter ได้ดังภาพ 3.6(c) และนำมาทำการหาขอบภาพด้วยโซเบลได้ดังภาพที่ 3.6(d) และในส่วนของการย่อภาพ 3 ระดับแสดงดังภาพที่ 3.6(e) สำหรับภาพอ้างอิงได้มีการเตรียมไว้ล่วงหน้าแล้วแสดงดังภาพที่ 3.6(f) เมื่อนำภาพมาทำสหสัมพันธ์และได้ภาพที่มีความคล้ายที่สุดออกมา ได้ดังภาพที่ 3.6(g) และ 3.6(h) โดยที่ภาพที่

3.6(i) นั้นได้แสดงให้เห็นได้อย่างชัดเจนถึงส่วนที่ถูกตัดออกไปได้ถึงแม้ภาพต้นแบบที่เข้ามาจะมีขนาดใบหน้าต่างกันก็ตาม

ภาพที่ 3.6

(a) ภาพที่ถูกสแกนมาด้วยความละเอียด 100 dpi

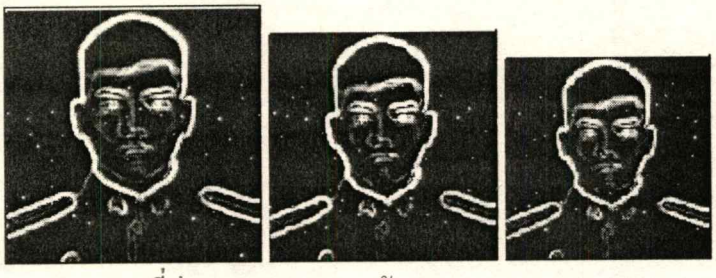
(b) ภาพที่มีการย่อให้มีจำนวนจุดภาพน้อยลงเป็น 128x128 pixels

(c) ภาพที่กำจัดสัญญาณรบกวนด้วย ตัวกรองแบบเลือกค่ากลาง (Median Filter)

(d) ภาพขอบจากวิธีการของโซเบล

ภาพในขั้นตอนต่างๆ จากการใช้ วิธี การ สหสัมพันธ์ กับการย่อภาพ 3 ระดับ

ภาพที่ 3.6 (ต่อ)



(e) ภาพที่ผ่านการขยาย 3 ระดับ 100% 90% และ 80%

(f) ภาพอ้างอิงที่สุ่มเลือกมาและส่วนที่ตัดมาใช้ในการค้นหา (ผ่านการลบขอบแก้มทิ้งไป) นำไปทำสหสัมพันธ์

(g) ภาพขอบของใบหน้าที่ค้นพบหลังการทำสหสัมพันธ์

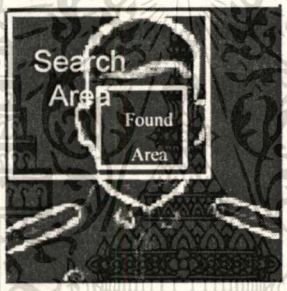
(h) ภาพระดับสีเทาของใบหน้าในบริเวณที่ตรงกับภาพขอบ (g)

(i) ภาพที่แสดงให้เห็นอย่างชัดเจนถึงส่วนที่ถูกตัดไปจากขั้นตอนการค้นหาจากภาพต้นแบบใบหน้าขนาดต่างๆ ที่สามารถตัดออกไปได้ในที่สุด

**ภาพในขั้นตอนต่างๆ จากการใช้ วิธี การ สหสัมพันธ์ กับการย่อภาพ 3 ระดับ**

เนื่องจากในการค้นหาบริเวณที่สนใจทั้วทุกบริเวณทั้งภาพ ด้วยวิธีการสหสัมพันธ์นั้นต้องใช้เวลาในมาก ดังนั้นจึงต้องเลือกกำหนดให้ค้นหาเฉพาะในบริเวณที่คาดว่าจะมีภาพดังกล่าวอยู่เท่านั้น ดังแสดงในภาพที่ 3.7 ยังผลให้สามารถลดระยะเวลาในการค้นหาได้อย่างมาก และภาพทั้งหมดที่นำมาใช้นั้นเป็นภาพจากบัตรซึ่งพบภาพใบหน้าในบริเวณใกล้ตรงกลางภาพถ่ายเสมอ ดังนั้นในที่นี้จึงเริ่มทำการค้นหาในส่วนบนด้านซ้ายของภาพถ่ายลงมาจนถึงตรงกลางภาพถ่ายเท่านั้น และจากการทดลองก็ประสบความสำเร็จในการค้นหาด้วยดีซึ่งมีความเร็วในการค้นหาใบหน้า 0.9 วินาทีต่อภาพ (เวลานี้ได้รวมทั้งการบันทึกภาพในขั้นตอนต่างๆ ลง harddisk ด้วย)

ภาพที่ 3.7




บริเวณที่ได้ทำการค้นหาเพื่อเพิ่มความเร็วในการทำงาน

**3.4 การแปลงเวฟเลต (Wavelet Transform)**

การแปลงเวฟเลต (Wavelet Trasform :WLT) [16-18] ในการประมวลผลภาพเป็นการแยกองค์ประกอบความถี่สูงและความถี่ต่ำของภาพซึ่งมีหลักการคล้ายกับการแปลงฟูเรียร์ (Fourier Transform) ในการนำไปใช้ประโยชน์จะเป็นการกำจัดความถี่ที่ไม่ต้องการในโดเมนของเวฟเลตทิ้งไป แล้วทำการสร้างภาพขึ้นมาใหม่จากองค์ประกอบของความถี่ที่เหลืออยู่ ซึ่งเวฟเลตมีประโยชน์มากมายทางการประมวลผลภาพ เช่นใช้ในเรื่องการบีบอัดข้อมูลแบบที่มีการสูญเสีย (Lossy Data Compression) เป็นต้น

การแปลงเวฟเลตแบ่งเป็นสองขั้นตอน คือ การแยกองค์ประกอบย่อย (Wavelet Decomposition) หรือการแปลงเวฟเลตและการสร้างขึ้นมาใหม่ (Wavelet Reconstruction) หรือการแปลงกลับเวฟเลต

เวฟเลตฟังก์ชันที่นำมาใช้ในการแปลงเวฟเลต มีอยู่หลายชนิด ซึ่งเหมาะสมในการใช้งานที่แตกต่างกันไป ในที่นี้ใช้ฮาร์เวฟเลต (Haar Wavelet) ซึ่งเป็นเวฟเลต ที่ง่ายและเก่าแก่ที่สุดในบรรดา เวฟเลตทั้งหลายๆ ที่มีอยู่ มันถูกใช้ในการวิเคราะห์ภาพโดยรู้จักกันในชื่อการแปลงฮาร์ (Haar Transform)(Pratt 1991) โดยฮาร์เวฟเลตเป็นสเต็บฟังก์ชัน ที่มีค่าระหว่าง +1 และ -1 ดังนี้

$$\psi_H(t) = \begin{cases} +1 & 0 < t < 0.5 \\ -1 & 0.5 \le t < 1 \\ 0 & otherwise \end{cases} \tag{3.8}$$

เวฟเลตจะมีคุณสมบัติในการย่อขยาย (Scale) และการเลื่อน (Translation) การย่อขยายจะใช้ในรูปของเลขสองยกกำลัง ตัวอย่างเช่น การย่อขยายของ $\psi_H(t)$ คือ $\psi_j(t) = \psi_H(2^j t)$ และการเลื่อนตำแหน่ง $\psi_H(t)$ คือ $\psi_j(t) = \psi_H(t-k)$ ดังนั้นเราสามารถเขียนฮาร์เวฟเลตได้ดังนี้

$$\psi_{kj}(t) = \psi_H(2^j t - k) \tag{3.9}$$

กระบวนการเวฟเลตคล้ายกับการนำ อนุกรมฟูเรียร์ (Fourier Series) มาประมาณฟังก์ชันต่างๆ ที่เป็นคาบ (Periodic) ว่าประกอบด้วยส่วนประกอบพื้นฐานที่เป็นฟังก์ชันสัญญาณไซด์ (Sine) และโคไซด์ (Cosine) ได้ ดังนั้นในการประมาณแบบองค์ประกอบย่อยด้วยฮาร์เวฟเลตเขียนได้ดังนี้

$$f(t) = \sum_{j=-\infty}^{\infty} A_j \sum_{k=-\infty}^{\infty} B_{jk} \psi_H(2^j t - k) \tag{3.10}$$

ซึ่งมีสัมประสิทธิ์คือ $A_j$ และ $B_{jk}$ ซึ่งสามารถรวมกันได้เป็นสัมประสิทธิ์ $C_{jk}$

โดยการนำมาใช้กับการประมวลผลสัญญาณฟังก์ชัน $f$ เป็นข้อมูลที่สุ่มมาแบบไม่ต่อเนื่อง และเมื่อนำมาเขียนเป็นสมการการแปลงเวฟเลตที่ใช้พื้นฐานของฮาร์ ก็คือการหาสัมประสิทธิ์ $C_{jk}$ ต่างๆ นั่นเอง ดังสมการต่อไปนี้

$$C_{jk} = 2_j \int_{2^{-jk}}^{2^{-j(k+1)}} f(t)\psi_{jk}(t)dt \tag{3.11}$$

**3.4.1 การคำนวณการแปลงของเวฟเลต**

ด้วยการอธิบายของ Lancaster (1996) อนุกรมฟังก์ชันในสมการ 3.10 สามารถเขียนได้เป็น

$$f(t) = \sum_{j=-\infty}^{-1} \sum_{k=-\infty}^{\infty} c_{jk} \psi_{jk}(t) \tag{3.12}$$

โดยการสมมติว่าส่วนประกอบความถี่สูงที่สุดจะอยู่ที่ใน $2^{-1}$ ซึ่งเป็นการทำการสเกลด้วย 2 ขั้นตอนแรกในการหาสัมประสิทธิ์ $C_{jk}$ เป็นการแยกส่วนความถี่สูงออกมาก่อน ซึ่งมันอยู่ที่ชั้น $j = -1$ ดังนั้นผลรวมจะกลายเป็น

$$\begin{aligned} f(t) &= \sum_{k=-\infty}^{\infty} c_{-1,k} \psi_{-1,k}(t) + \sum_{j=-\infty}^{-2} \sum_{k=-\infty}^{\infty} c_{jk} \psi_{jk}(t) \\ &= W_{-1}(t) + f_{-1}(t) \end{aligned} \tag{3.13}$$

โดยที่ผลรวมที่ถูกแทนด้วยเทอม $W_{-1}(t)$ เป็นส่วนความถี่สูง และ $f_{-1}(t)$ เป็นส่วนที่เหลือทั้งหมด สัมประสิทธิ์ของชั้นความถี่ที่สูงที่สุดมีค่า $j = -1$ โดยการแก้สมการ 3.10 โดยใช้ฮาร์เวฟเลตจะได้

$$c_{-1,k} = \frac{1}{2} \int_{2k}^{2(k+1)} f(t)\psi_{-1,k}(t)dt = \frac{(f(2k) - f(2k+1))}{2} \tag{3.14}$$

ในการอินทีเกรตนี้สามารถทำได้ด้วยมือหรือโดยการใช้โปรแกรมสำเร็จรูปก็ได้มาทำการหาคำตอบ ซึ่งเราจะได้ค่า, $c_{-1,k}$ ออกมาได้จากข้อมูล $f$ โดยตรง

และในการคำนวณในชั้นความถี่ที่สูงขึ้นไปคือ $W_{-2}(t)$ สามารถทำได้โดยการแยกองค์ประกอบที่เหลืออยู่คือ $f_{-1}(t)$ ด้วยวิธีการที่คล้ายกับที่กล่าวมา โดยเราจะได้

$$W_{-2}(t) = \sum_{k=-\infty}^{\infty} c_{-2,k}\psi_{-2,k}(t)$$
$$f_{-2}(t) = \sum_{j=-\infty}^{-3}\sum_{k=-\infty}^{\infty} c_{jk}\psi_{jk}(t) \qquad (3.15)$$

โดยการใช้สมการ 3.15 และทำการหาคำตอบของการอินทีเกรต สัมประสิทธิ์ ในชั้นถัดไปจะได้เป็นดังนี้

$$c_{-2,k} = \frac{1}{4}(f(4k) + f(4k+1) - f(4k+2) - f(4k+3)) \qquad (3.16)$$

ในแต่ละชั้นสามารถคำนวณได้โดยวิธีการเช่นนี้ไปเรื่อยๆ ซึ่งจะมีจำนวนเทอมเพิ่มขึ้นเป็นสองเท่าทุกๆ ชั้น และ โดยการสรุป ความสัมพันธ์ ของสัมประสิทธิ์ในแต่ละชั้นที่เกี่ยวพันธ์กับชั้นก่อนหน้า (Preview Layer) เขียนความสัมพันธ์ได้ดังนี้

$$c_{j,k} = \frac{1}{2}(a_{j+1,2k} - a_{j+1,2k+1})$$
$$a_{j,k} = \frac{1}{2}(a_{j+1,2k} + a_{j+1,2k+1}) \qquad (3.17)$$

โดยเหตุที่ $f$ เป็นกลุ่มข้อมูลที่ได้จากจากการสุ่มมา ค่าของ $a_{0,k}$ ก็คือค่าของข้อมูล $f(k)$ นั่นเอง และเราสามารถคำนวณหาสัมประสิทธิ์ $c_{j,k}$ ต่างๆ ได้ โดยเริ่มต้นจาก $a_{0,k}$ แล้วค่าก็จะเปลี่ยนไปเป็น $a_{j,k}$ ในชั้นต่างๆ ของการแปลง

สมมติ $f$ เป็น ข้อมูล สุ่มจำนวน 16 ตัว มีค่าดังนี้

$$0\,1\,2\,2\,4\,4\,2\,2\,0\,0\,2\,2\,0\,0\,0\,0 \qquad (3.18)$$

จากสมการที่ 3.17 นำมาทำการหาสัมประสิทธิ์ $c_{-1,k}$ ได้ดังนี้

-0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 (3.19)

และสัมประสิทธิ์ $a_{-1,k}$ ที่เปลี่ยนไปคือ

0.5 2.0 4.0 2.0 0.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 (3.20)

ชั้นที่เหลืออยู่ก็สามารถคำนวณได้ด้วยวิธีการเช่นเดียวกันได้ออกมาเป็นลำดับ โดยเซตของสัมประสิทธิ์ที่ได้มาที่มีค่าไม่เท่ากับศูนย์นั้นจะมีจำนวนลดลงเรื่อยๆ ซึ่งจะลดลงเหลือครึ่งหนึ่งของชั้นก่อนหน้า ดังนั้นจากตัวอย่างที่มีข้อมูลสุ่ม 16 ตัว เมื่อคำนวณเพียงสี่ชั้นก็จะเหลือสัมประสิทธิ์ที่มีค่าไม่เป็นศูนย์เพียงตัวเดียวดังแสดงในตารางที่ 3.1

ตารางที่ 3.1

| k | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| data | 0 | 1 | 2 | 2 | 4 | 4 | 2 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 |
| $c_{-1}$ | -0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $a_{-1}$ | 0.5 | 2.0 | 4.0 | 2.0 | 0.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $c_{-2}$ | -.75 | 1.0 | -1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $a_{-2}$ | 1.25 | 3.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $c_{-3}$ | -.875 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $a_{-3}$ | 2.12 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $c_{-4}$ | 0.81 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $a_{-4}$ | 1.31 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

แสดงข้อมูลจากการแปลงเวฟเลตจำนวน 4 ชั้น

จากนั้นในกระบวนการสร้างข้อมูลกลับขึ้นมาใหม่ โดยใช้ทุกสัมประสิทธิ์มาสร้างข้อมูลกลับขึ้นมาใหม่หรือการแปลงกลับเวฟเลตก็จะได้ข้อมูลที่เหมือนเดิมทุกประการ ซึ่งการคำนวณจะทำการย้อนกลับโดยเริ่มจากส่วนของความถี่ที่ต่ำที่สุดแล้วย้อนไปสู่ความถี่ที่สูงขึ้นเรื่อยๆ ด้วยวิธีการนี้รูปร่างของข้อมูลจะค่อยๆปรากฏขึ้นมาและส่วนที่เป็นรายละเอียดต่างๆ ที่เป็นความถี่สูงก็จะออกมาในการคำนวณชั้นต่างๆ ตามลำดับ จากความถี่ต่ำสุดที่สัมประสิทธิ์ $c_{-4,k}$ ในการสร้างข้อมูลกลับขึ้นมาใหม่ หรือการแปลงกลับเวฟเลตทำได้ดังนี้

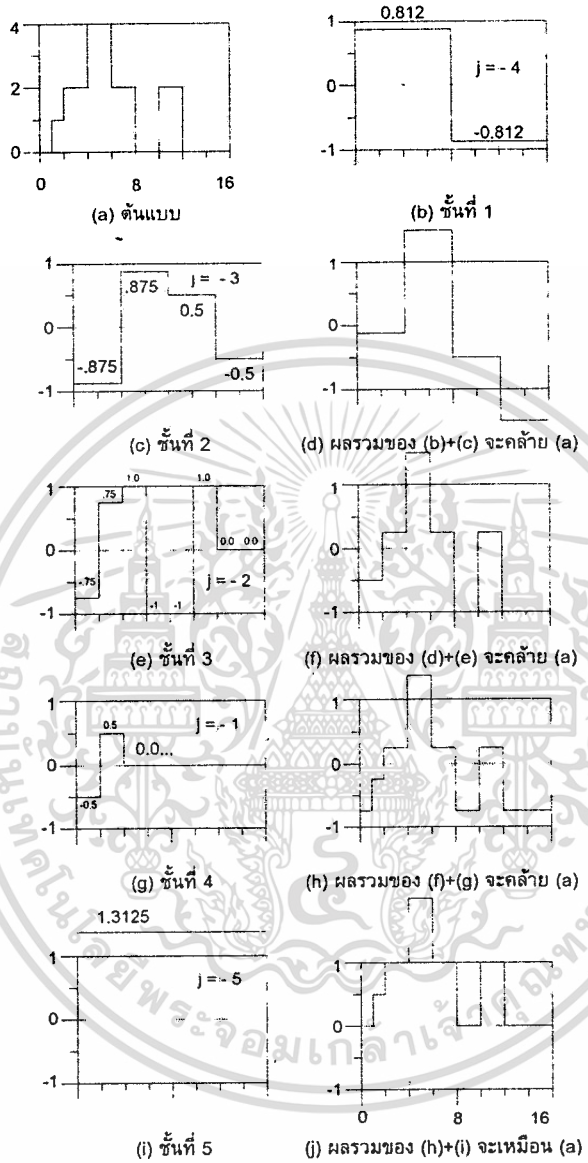
$$f_4(t) = \sum_{k=0}^{0} c_{-4,k}\psi_{-4,k}(t) = c_{-4,0}\psi_{-4,0}(t) \qquad (3.21)$$

ดังแสดงในภาพที่ 3.8(b) และคำนวณในชั้นความถี่ถัดไป เขียนได้เป็น

$$f_3(t) = c_{-3.0}\psi_{-3.0}(t) + c_{-3.1}\psi_{-3.1}(t) \tag{3.22}$$

แสดงดังภาพที่ 3.8(c) ซึ่งจะเห็นได้ว่ามีความละเอียดของข้อมูลมากขึ้น ภาพที่ 3.8(d) แสดงผลรวมของทั้งสององค์ประกอบความถี่ ซึ่งจะมีความใกล้เคียงข้อมูลที่เป็นสัญญาณต้นแบบมากขึ้น ภาพถัดไปแสดงระดับที่สูงขึ้นและแสดงผลรวมในภาพถัดไป จนกระทั่งทำครบทั้งสี่ชั้น เราจะได้ข้อมูลที่เกิดจากการรวมของแต่ละความถี่ในรูปร่างที่เหมือนข้อมูลเดิมแล้ว แต่ข้อมูลมีความสูงลดลงทุกๆ จุด นั่นคือขาดไปเท่ากับ 1.312 หรือเท่ากับค่าที่เป็นส่วนที่เป็น $a_{-4.0}(=c_{-5.0})$ คือต้องมีสัมประสิทธิ์ในชั้นที่ -5 ที่ได้มาจากข้อมูล $a$ ในชั้นที่ -4 ดังแสดงในภาพที่ 3.8(i) ที่ต้องเพิ่มเข้ามาที่เป็นส่วนประกอบที่เป็นความถี่ต่ำสุด (DC) ที่ต้องรวมเข้าไปกับผลรวมความถี่อื่นๆ ด้วยก็จะได้ข้อมูลหลังการแปลงกลับที่เหมือนข้อมูลเดิมทุกประการดังแสดงในภาพที่ 3.8(j)

ภาพที่ 3.8




แสดงการแปลงกลับข้อมูลจากสัมประสิทธิ์การแปลงของเวฟเลต

(a) ข้อมูลต้นแบบ (b) ระดับสัมประสิทธิ์ -4 ที่ให้สัญญาณสองระดับ

(c) ระดับสัมประสิทธิ์ -3 ที่สัญญาณ 4 ระดับ (d) ผลรวมของระดับ -4 กับ -3 ที่ให้สัญญาณย่อยกลับขึ้นมา (e) ระดับสัมประสิทธิ์ -2 (8 ระดับ) (f) ผลรวมของสามระดับ

(g) ระดับสัมประสิทธิ์ -1 (16 ระดับ) (h) ผลรวมของ 4 ระดับ (i) ค่าคงที่ในระดับ -5

(j) สัญญาณที่สร้างกลับขึ้นมาใหม่ ซึ่งเป็นผลรวมของทั้ง 5 ระดับ

### 3.4.2 การแปลงเวฟเลตแบบเร็ว (Fast Wavelet Transform :FWLT)

จากการคำนวณหาค่าสัมประสิทธิ์ $C_{j,k}$ และค่า $a_{j,k}$ ค่าต่างๆ นั้น ผลการแปลงจะได้ข้อมูลสองมิติบนระนาบ $j,k$ ซึ่งจำนวนสัมประสิทธิ์จะเท่ากับจำนวนข้อมูลต้นแบบ เพราะว่าแต่ละชั้นจะมีส่วนประกอบที่มีจำนวนลดลงเป็นครึ่งหนึ่งของชั้นก่อนหน้า

ลองพิจารณาฮาร์เวฟเลตที่เริ่มจากชั้นที่ $j=-1$ จากข้อมูลคือ $f$ ที่ถูกเก็บไว้ในแอเรย์หน่วยความจำ ที่มีการหาคำนวณหาค่าสัมประสิทธิ์ตามสมการ 3.17 แล้วนำผลลัพธ์คือ $a_{j,k}$ เก็บทับลงไปบนแอเรย์ข้อมูล $f$ เดิมนั้นโดยทับลงไปบนตำแหน่งแรก ($n/2(=8)$) และนำ $c_{j,k}$ ทับลงบนส่วนหลังของตำแหน่ง $n/2$

แล้วทำการคำนวณต่อไปในชั้น $j=-2$ โดยเก็บ $a_{-2,k}$ ลงใน $n/4$ แรก และเก็บ $c_{-2,k}$ ลงใน ตำแหน่ง $n/4$ แล้วทำเช่นนี้ไปเรื่อยๆ จนได้สัมประสิทธิ์ครบทั้งหมด และก็ทับข้อมูลใน แอเรย์ $f$ ได้ทั้งหมดแล้วด้วย ที่เราสามารถทำเช่นนี้ได้ก็เพราะในแต่ละชั้นที่เราคำนวณจำนวนผลลัพธ์จะลดลงจากชั้นก่อนหน้าเหลือจำนวนเพียงครึ่งหนึ่งเรื่อยๆ และอีกอย่างหนึ่งก็คือเนื่องจากในชั้นถัดไปเราเพียงต้องการ $a_{j,k}$ เท่านั้นในการคำนวณ $a_{j-1,k}$

พิจารณาข้อมูลจากตัวอย่างในตารางที่ 3.1 ค่าของ $a_{j,k}$ และ $c_{j,k}$ สำหรับ $j=-1$ เมื่อทำกระบวนการแปลงเวฟเลตแบบเร็วจะได้ดังนี้

0.5 2.0 4.0 2.0 0.0 2.0 0.0 0.0 -0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 (3.23)

ค่าของ $a_{-1,k}$ ค่าของ $c_{-1,k}$

จากการคำนวณต่อไปในชั้นที่ $j=-2$ ของการแปลงเวฟเลตแบบเร็วจะมีการเปลี่ยน 8 ตัวแรกของข้อมูลดังนี้

1.25 3.0 1.0 0.0 -0.75 1.0 -1.0 0.0 -0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 (3.24)

$a_{-2,k}$ $c_{-2,k}$ $c_{-1,k}$

สุดท้ายหลังจากผ่านไปหลายรอบของการแปลง จะได้

1.3125 0.8125 -0.875 0.5 -0.75 1.0 -1.0 0.0 -0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 (3.25)

สังเกตว่าข้อมูลที่เป็นความถี่ต่ำจะอยู่ก่อน (ซ้ายมือสุด) และตามมาด้วยความถี่ที่สูงขึ้น ซึ่งจากการจัดเรียงดังกล่าวทำให้เราสามารถทำการกรองความถี่ (Filtering by Frequency) ได้อย่างสะดวกมาก ในส่วนของการแปลงกลับเวฟเลตแบบเร็วก็ทำได้ในทิศทางตรงกันข้ามกับที่

กล่าวมา อันดับแรกค่าคงที่ $c_{-5,k}$ จะนำมาบวกเข้ากับชั้นถัดมาคือ $c_{-4,k}$ นั่นคือส่วนประกอบ 8 ตัวแรกในการสร้างกลับข้อมูลขึ้นมาใหม่คือ $c_{-5,k}+c_{-4,k}$ และอีก 8 ตัวหลังคือ $c_{-5,k}-c_{-4,k}$ แต่อย่างไรก็ตามเราไม่มีความจำเป็นต้องใช้ตำแหน่งในการเก็บผลลัพธ์ครบทั้ง 8 ตัว และจะมีการเก็บทับลงไปใน แอเรย์หน่วยความจำที่เก็บข้อมูลสัมประสิทธิ์ต้นแบบ โดยหลังจากคำนวณชั้นแรกเราจะได้ดังนี้

2.125 0.5 -0.875 0.5 -0.75 1.0 -1.0 0.0 -0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 (3.26)

ขั้นที่ 1 ในการแปลงกลับ ข้อมูลสัมประสิทธิ์ต้นแบบ

ชั้นถัดไป ทำให้เกิดค่าใหม่ขึ้น4 ค่า ดังการเปลี่ยนแปลงดังนี้

| $(c_{-5,k}+c_{-4,k}+c_{-3,k})$ | $(c_{-5,k}+c_{-4,k}-c_{-3,k})$ | $(c_{-5,k}-c_{-4,k}+c_{-3,k})$ | $(c_{-5,k}-c_{-4,k}-c_{-3,k})$ |
|---|---|---|---|
| 2.145+(-0.875) | 2.145-(-0.875) | 0.5+0.5 | 0.5-0.5 |
| 1.25 | 3.00 | 1.0 | 0.0 |

(3.27)

ซึ่งจะนำไปเก็บใน 4 ตัวแรกของแอเรย์ข้อมูล ดังนี้

1.25 3.0 1.0 0.0 -0.75 1.0 -1.0 0.0 -0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0

ขั้นที่ 2 ข้อมูลสัมประสิทธิ์ต้นแบบ

ในการแปลงกลับ (3.28)

กระบวนการนี้จะทำต่อไปเรื่อยๆ จนครบทุกตัว ซึ่งจะได้แอเรย์ของข้อมูลกลับมาเหมือนกับข้อมูลแรกสุดก่อนการแปลงเวฟเลตและถือเป็นการเสร็จสิ้นกระบวนการแปลงกลับเวฟเลต

### 3.4.3 การแปลงเวฟเลตแบบสองมิติแบบเร็ว (Fast 2D Wavelet Transform)

ในการแปลงเวฟเลตแบบที่เป็นของมูล 2 มิติสำหรับใช้ในการประมวลผลภาพนั้น สามารถนำการแปลงแบบ 1 มิติมาทำการแปลงในแนว Row แล้วนำข้อมูลมาแปลงต่ออีกครั้งในแนว Column ก็จะได้การแปลงเวฟเลตแบบสองมิติตามต้องการ สำหรับในขั้นตอนการกรองความถี่นั้น เนื่องจากการจัดเรียงความถี่จากเวฟเลตแบบเร็ว นั้นมีการจัดเรียงจากความถี่ต่ำอยู่ซ้ายสุดใน Row และความถี่สูงอยู่ขวาใน Row แต่เมื่อเป็นเวฟเลตแบบเร็วสองมิติ ดังนั้นในแนว Column ก็เช่นเดียวกัน ข้อมูลความถี่ต่ำจะอยู่ด้านบน ในขณะที่ความถี่สูงอยู่ในตำแหน่งด้านล่าง ดังนั้นจึงเป็นการง่ายที่จะทำการกรองความถี่ในโดเมนของเวฟเลต ซึ่งการ

กรองความถี่ต่ำผ่านทำได้โดยเลือกบล็อกนับจากมุมบนด้านซ้ายขนาด NxN ที่ต้องการให้ผ่านการกรอง แล้วกำหนดให้พื้นที่นอกเหนือจากพื้นที่ดังกล่าวให้มีค่าเป็นศูนย์ทั้งหมด จากนั้นทำการแปลงกลับเวฟเลตออกมาก็จะได้ผลของการกรองความถี่ต่ำผ่านตามต้องการ ซึ่งหากมีสัมประสิทธิ์ที่เหลืออยู่น้อย คุณภาพหรือรายละเอียดของภาพภายหลังการแปลงก็จะลดลงไปตามต้องการ

ในวิทยานิพนธ์นี้ใช้การแปลงเวฟเลตในการลดรายละเอียด (Information) ของข้อมูลจำนวนมากลงเพื่อประโยชน์ในการจัดกลุ่มภาพใบหน้า แต่เอาต์พุตที่ได้จากเวฟเลตนี้ จะมีขนาดภาพเท่าขนาดเดิมทั้งนี้เพราะต้องการผลของการลดรายละเอียดของภาพมาใช้เท่านั้น ซึ่งก็สามารถพัฒนาต่อไปได้อีกโดยเลือกกระบวนการทำเวฟเลตที่สามารถลดขนาด (Size) ของภาพได้ด้วยซึ่งจะเป็นการลดจำนวนโหนด อินพุตของโครงข่ายประสาทเทียมเพื่อเพิ่มความเร็วในการเรียนรู้ของโครงข่ายประสาทเทียมได้อย่างมาก ตัวอย่างการใช้เวฟเลตในการลดรายละเอียดของภาพใบหน้า แสดงได้ดังภาพที่ 3.9

ภาพที่ 3.9

(a) (b) (c) (d) (e)

การแปลงเวฟเลตที่ใช้กับภาพใบหน้าเพื่อลดรายละเอียดของข้อมูล
(a) ภาพต้นแบบ (b) ใช้ 16x16 ข้อมูลแรกในการแปลงกลับ
(c) ใช้ 8x8 ข้อมูลแรกในการแปลงกลับ (d) ใช้ 4x4 ข้อมูลแรกในการแปลงกลับ
(e) ใช้ 2x2 ข้อมูลแรกในการแปลงกลับ

### 3.5 เวกเตอร์ควอนไตซ์ในการจัดกลุ่มภาพใบหน้า

วิธีการเวกเตอร์ควอนไตซ์ [19-21] (Vector Quantization , VQ) เป็นอัลกอริทึมที่ส่วนใหญ่จะถูกนำไปใช้ในการบีบอัดข้อมูล (Data Compression) ซึ่งเป็นการบีบอัดแบบที่มีการสูญเสียวิธีหนึ่ง (Lossy Compression) แต่ทั้งนี้เนื่องจากคุณสมบัติที่สามารถนำมาใช้ในการจัดกลุ่มของข้อมูล (Classification) ได้ ดังนั้นในที่นี้จึงนำมาใช้ในการจัดกลุ่มของภาพใบหน้าออกเป็นกลุ่มต่างๆ โดยเรากำหนดเพียงตัวเริ่มต้นกลุ่ม (Initializing) แต่ละกลุ่มให้เท่านั้น ด้วยวิธีการเวกเตอร์ควอนไตซ์จะทำการปรับตัวและจัดกลุ่มต่างๆ ให้อีกครั้งหนึ่ง ซึ่งนิยามของเวกเตอร์ควอนไตซ์คือการสร้างความสัมพันธ์ ($Q$) ระหว่างข้อมูลของ สเปซยูคลิเดียน $K$ มิติ ($R^K$) ลงในเซตย่อย $Y$ ของ $R^K$ ดังนี้

$$Q \;:\; R^K \rightarrow Y \tag{3.29}$$

ซึ่ง $Y = (\hat{x}_i; i = 1,2,\ldots N)$ คือ เซตของเวกเตอร์ที่ถูกสร้างกลับขึ้นมาใหม่ และ $N$ คือจำนวนของเวกเตอร์ใน $Y$

ในการทำเวกเตอร์ควอนไตซ์เพื่อจัดกลุ่มจะประกอบด้วย 2 ส่วนคือ การสร้างตัวเก็บรหัส (Codebook) และการเข้ารหัส (Encoder)

#### 3.5.1 การสร้างตัวเก็บรหัส (Codebook)

เทคนิคที่นิยมใช้กันมากในการสร้างตัวเก็บรหัส คือใช้อัลกอริทึม LBG (Linde-Buzo-Gray) [4] ซึ่งเราจะนำวิธีการนี้มาสร้างตัวเก็บรหัส (Codebook) ที่เป็นข้อมูลของการจัดกลุ่มใบหน้า

ในที่นี้ เพื่อให้สามารถอธิบายอัลกอริทึมของการเรียนรู้ของเวกเตอร์ควอนไตซ์ได้อย่างชัดเจนขอใช้ตัวอย่างดังอยู่ภายในตารางที่ 3.2 ที่เก่าแก่อันหนึ่ง [20] ในการอธิบาย โดยเริ่มต้นจากที่มีข้อมูลจำนวน 12 ตัว $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{12}$ (มีค่า $n = 12$) ซึ่งข้อมูลแต่ละตัวจะเป็นเวกเตอร์ที่มีสมาชิกสองตัว เช่น $\tilde{x}_1 = (-0.37449, 0.98719)$ (มีค่า $k = 2$) กำหนดค่าความเพี้ยนที่ยอมรับได้ (Distortion Threshold) เป็น 0.1% (มีค่า $\varepsilon = 0.001$) และต้องการแบ่งกลุ่มของข้อมูลออกเป็น 4 กลุ่ม (มีค่า $N = 4$) โดยมีกระบวนการทั้งหมดดังต่อไปนี้

**ขั้นตอนที่ 1** การกำหนดค่าเริ่มต้น (Initialization) โดยการกำหนด $N$ คือจำนวนของระดับ (Level) หรือจำนวนกลุ่มที่ต้องการจัด $k$ คือความยาวของบล็อคหรือขนาดของเวกเตอร์ $\varepsilon$ คือค่าความเพี้ยนที่ยอมรับได้ ข้อมูลที่นำมาฝึกคือ $x_j; j = 0,1,\ldots,n-1$ ให้ $\hat{A}_0$ เป็นตัวเก็บรหัสที่ถูกกำหนดให้มีค่าเริ่มต้นแบบ Uniform Initialization ตัวอย่างเช่นกำหนดให้มีค่าเริ่มต้นที่ (-2,-2), (-2,2), (2,-2) และ (2,2) ซึ่ง เป็นจุดตรงกลางของควอดเรนต์ทั้ง 4 บนระนาบสอง

มิติ ซึ่งหมายความว่าในการจัดกลุ่มจะเริ่มต้นจากจุดทั้งสี่นี้ นอกจากนั้นมีการกำหนดค่าที่เก็บจำนวนรอบในการสอน (Training) คือกำหนดให้ $m=0$ และให้ค่าความเพี้ยนเริ่มต้น $D_{-1}=\infty$

**ขั้นตอนที่ 2** ให้ $\hat{A}_m=\{y_i;i=1,2,...,N\}$ หาความเพี้ยนน้อยที่สุด (Minimum Distortion) $P(\hat{A}_m)=\{S_i;i=1,2,...,N\}$ ตัวอย่างเช่นสำหรับแต่ละ $j=0,1,...,n-1$ คำนวณ $d(x_i,y_i)$ สำหรับ $i=1,2,...N$ ถ้า $d(x_i,y_i)\le d(x_j,y_m)$ สำหรับ $m$ ทั้งหมด แล้วหมายความว่า $x_j\in s_i$ แล้วคำนวณค่าความเพี้ยนเฉลี่ย ดังนี้

$$D_m=D(\{\hat{A}_m\})=n^{-1}\sum_{j=0}^{n-1}\min_{y\in A_m} d(x_j,y) \tag{3.30}$$

**ขั้นตอนที่ 3** ถ้าการเปลี่ยนแปลงของค่าความเพี้ยนของตัวปัจจุบันกับตัวก่อนหน้านี้มีค่าน้อยคือ $\frac{D_{m-1}-D_m}{D_m}\le 0.001$ คือมีความเสถียร (Stable) แล้ว ก็จะได้คำตอบที่เป็นตัวเก็บรหัส $\hat{A}_m$ และสิ้นสุดกระบวนในการสร้างตัวเก็บรหัส แต่ถ้าค่าการเปลี่ยนแปลงความเพี้ยนยังคงสูงอยู่ก็จะต้องไปทำกระบวนการในขั้นตอนที่ 4 ต่อไป

**ขั้นตอนที่ 4** ทำการปรับปรุงตัวเก็บรหัส คือ นำข้อมูลของตัวเก็บรหัสต่างๆ มาทำการหาค่าเฉลี่ย โดย $\hat{x}(P(\hat{A}_m))=\{\hat{x}(S_i);i=1,2,...,N\}$ สำหรับตัวเก็บรหัส $P(\hat{A}_m)$ ซึ่ง $\hat{x}(S_i)$ คือ Euclidean Center of Gravity หรือ Centroid ดังนี้

$$x(S_i)=\frac{1}{\|S_i\|}\sum_{j:x_j\in S_i}x_j \tag{3.31}$$

โดย $\|S_i\|$ แทนจำนวนของเวกเตอร์ที่อยู่ในกลุ่มนั้นๆ เช่น ในกลุ่ม $S_1$ มีจำนวนเวกเตอร์อยู่ 4 ตัว กลุ่ม $S_2$ มีอยู่ 2 ตัว เป็นต้น ถ้า $\|S_i\|=0$ คือไม่มีสมาชิกอยู่ในกลุ่มนั้นเลย ให้กำหนด $\hat{x}(S_i)=y_i$ หรือเป็นตัวเดิม กำหนด $\hat{A}_{m+1}=\hat{x}(P(A_m))$ เป็นการเปลี่ยนตัวเก็บรหัสเป็นค่าใหม่ที่หาได้มา แล้วเพิ่มตัวแปรนับจำนวนรอบการทำงาน m ขึ้นอีกหนึ่ง แล้วกลับไปทำงานตามขั้นตอนที่ 2 ใหม่อีกครั้ง

ตารางที่ 3.2

Step 1 **Initialization**: $N = 4,\ k = 2,\ \varepsilon = 0.001.\ n = 12$

Training Sequence:

| | |
|---|---|
| $\tilde{x}_1 = (-0.37449,\ 0.98719)$ | $\tilde{x}_7 = (-0.59161,\ 0.17968)$ |
| $\tilde{x}_2 = (0.63919,\ -0.11875)$ | $\tilde{x}_8 = (0.14093,\ 1.76413)$ |
| $\tilde{x}_3 = (-0.83293,\ 0.60645)$ | $\tilde{x}_9 = (0.70898,\ -0.35017)$ |
| $\tilde{x}_4 = (-0.70534,\ -1.21856)$ | $\tilde{x}_{10} = (0.30038,\ 0.79836)$ |
| $\tilde{x}_5 = (-0.28952,\ -0.94821)$ | $\tilde{x}_{11} = (0.30165,\ 1.06552)$ |
| $\tilde{x}_6 = (1.09924,\ 0.516)$ | $\tilde{x}_{12} = (0.37801,\ -0.32708)$ |

$$\hat{A}_0 = \{(2,2),(2,-2),(-2,2),(-2,-2)\}$$
$$= \{\tilde{y}_1, \tilde{y}_2, \tilde{y}_3, \tilde{y}_4\}$$

$D_{-1} = 9.99E+62 \quad (\infty)$

Set $m = 0$

$\underline{m = 0}$ Step 2 Find $P(\hat{A}_0) = \{S_1, S_2, S_3, S_4\}$

$\tilde{x}_j \in S_i \quad if \quad d(x_j, y_i) \le d(x_j, y_m) \quad ;all\ m \ne i$

$S_1 = \{\tilde{x}_6, \tilde{x}_8, \tilde{x}_{10}, \tilde{x}_{11}\}$

$S_2 = \{\tilde{x}_2, \tilde{x}_9\}$

$S_3 = \{\tilde{x}_1, \tilde{x}_3, \tilde{x}_7\}$

$S_4 = \{\tilde{x}_4, \tilde{x}_5, \tilde{x}_{12}\}$

Compute $D_0$:

$$D_0 = \frac{1}{12}\sum_{j=1}^{12} \min_{\tilde{y} \in \hat{A}_0} d(\tilde{x}_j, \tilde{y}) = 2.0172$$

Step 3 $(D_{-1} - D_0) / D_0 > 0.001,\ continue$

Step 4 Find the optimal reproduction alphabet $\hat{A}_1 = \hat{\tilde{x}}(P(\hat{A}_0)) = \{\hat{\tilde{x}}(S_i), i = 1,2,..,4\}$:

**ตัวอย่างการคำนวณเพื่อสร้างตัวเก็บรหัส**

**ตารางที่ 3.2 (ต่อ)**

$\hat{\tilde{x}}(S_1) = \{\tilde{x}_6, \tilde{x}_8, \tilde{x}_{10}, \tilde{x}_{11}\} / 4 = (0.46055,\ \ 1.036)$

$\hat{\tilde{x}}(S_2) = \{\tilde{x}_2, \tilde{x}_9\} / 2 = (0.674085,\ \ -0.23446)$

$\hat{\tilde{x}}(S_3) = \{\tilde{x}_1, \tilde{x}_3, \tilde{x}_7\} / 3 = (-0.599676,\ \ 0.591106)$

$\hat{\tilde{x}}(S_4) = \{\tilde{x}_4, \tilde{x}_5, \tilde{x}_{12}\} / 3 = (-0.457623,\ \ -0.831283)$

Set $m = 1$ goto Step 2

$m = 1$ Step 2 Find $P(\hat{A}_1)$:

Evaluating distortion shows $P(\hat{A}_1) = P(\hat{A}_0)$ (Not change in partition)

Compute $D_1$:

$$D_1 = \frac{1}{12} \sum_{j=1}^{12} \min_{\tilde{y} \in \hat{A}_m} d(\tilde{x}_j, \tilde{y}) = 0.0997308$$

Step 3 $(D_0 - D_1) / D_1 \cong 19 > 0.001$

Step 4 $\hat{A}_2 \overset{\Delta}{=} \hat{\tilde{x}}(P(\hat{A}_1)) = \hat{A}_1$ Since $P(\hat{A}_1) = P(\hat{A}_0)$ and hence

$\hat{\tilde{x}}(P(\hat{A}_1)) = \hat{\tilde{x}}(P(\hat{A}_0)) = \hat{A}_1$ Thus $\hat{A}_1$ is a fixed point. Set $m = 2$. Goto Step 2

$m = 2$ Step 2 $P(\hat{A}_1) = P(\hat{A}_0)$ and hence $D_2 = D_1$

and hence $(D_1 - D_2) / D_2 = 0 < 0.001$

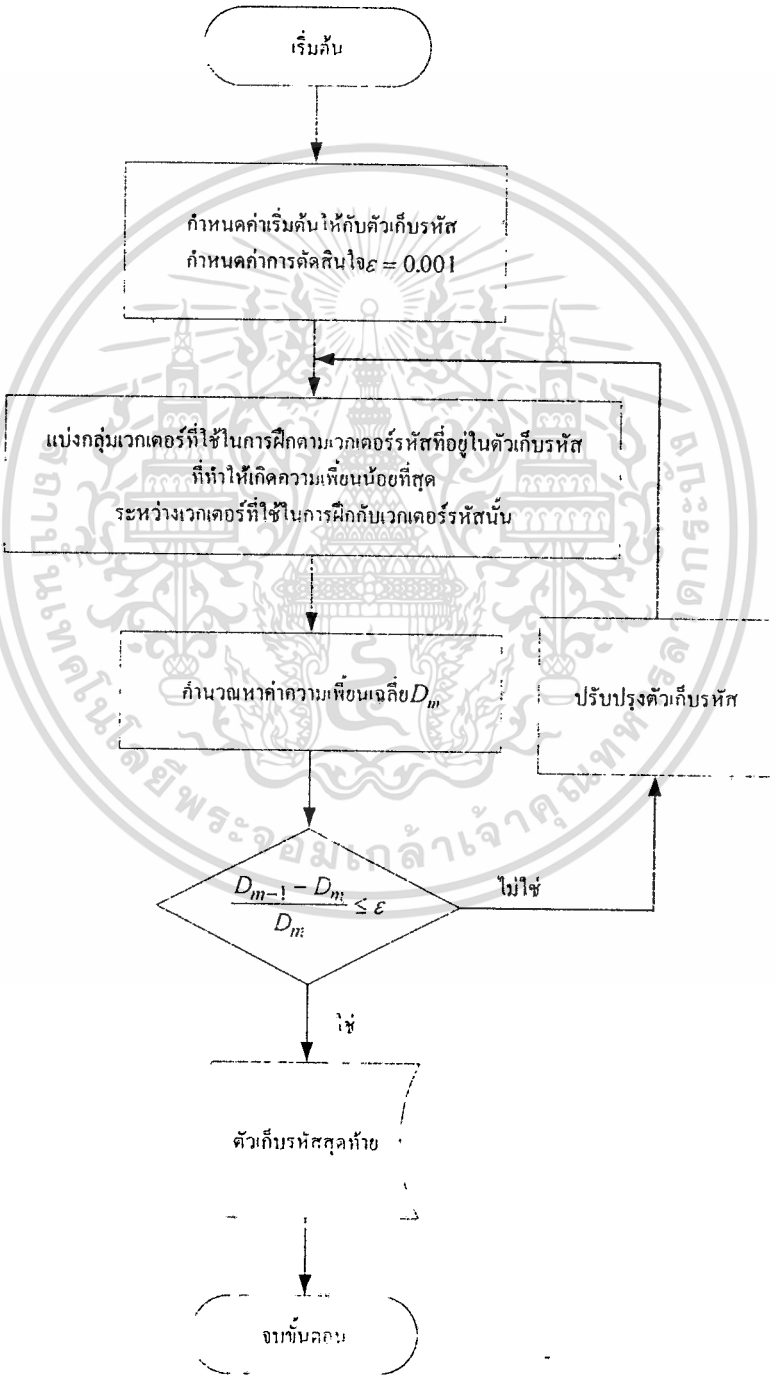
Halt with final quantizer described by $\{\hat{A}_1,\ P(\hat{A}_1)\}$.

ตัวอย่างการคำนวณเพื่อสร้างตัวเก็บรหัส

จากตัวอย่างในตารางที่ 3.2 เราสามารถเขียนในรูปแบบโฟล์ตชาร์ตอัลกอริทึม LBG ได้อีกครั้งดังภาพที่ 3.10 ซึ่งการทำงานเริ่มต้นจากการกำหนดค่าเริ่มต้นให้แก่ตัวเก็บระหัส จากนั้นกำหนดค่าการตัดสินใจที่เรายอมรับได้ที่จะเป็นตัวกำหนดการสิ้นสุดโปรแกรม โปรแกรมจะหยุดเมื่อเสถียร หรือค่าผลต่างของความเพี้ยนไม่เปลี่ยนแปลงแล้ว จากนั้นทำการจัดกลุ่มที่ให้ความเพี้ยนน้อยที่สุด และทำการคำนวณหาความเพี้ยนเฉลี่ย และทำการคำนวณค่าการเปลี่ยนแปลงความเพี้ยน ถ้าระบบเสถียรหมายว่าจะได้ผลต่างน้อยกว่าค่าการตัดสินใจ $\varepsilon$ จึงสิ้นสุดกระบวนการสร้างตัวเก็บรหัส แต่ถ้ายังมีผลต่างสูงอยู่ให้ทำการปรับปรุงตัวเก็บรหัสและเริ่มกระบวนการใหม่ ตัวเก็บรหัสจะเปลี่ยนแปลงไปจนกระทั่งความเพี้ยนเฉลี่ยจะน้อยที่

และเริ่มกระบวนการใหม่ ตัวเก็บรหัสจะเปลี่ยนแปลงไปจนกระทั่งความเพี้ยนเฉลี่ยจะน้อยที่สุด และไม่สามารถปรับปรุงได้อีก ค่าการเปลี่ยนแปลงความเพี้ยนก็จะไม่เปลี่ยนอีกต่อไปถือว่าสิ้นสุดกระบวนการ

แผนภูมิที่ 3.10




อัลกอริทึม LBG ในการสร้างตัวเก็บรหัส

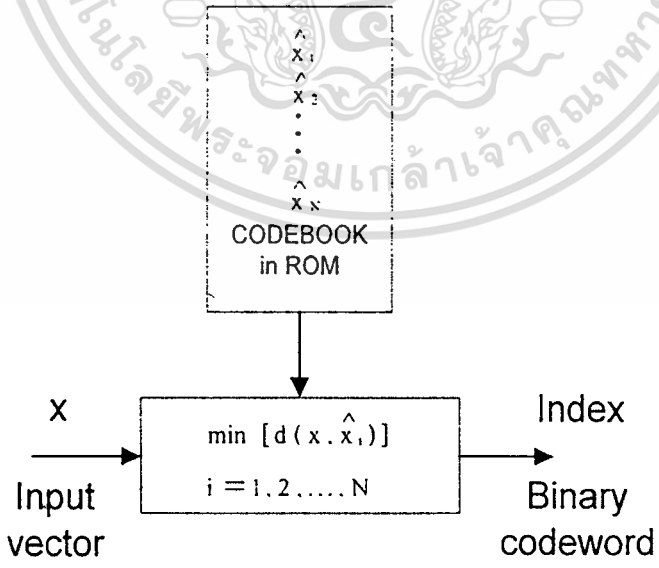
### 3.5.2 การเข้ารหัส (Encoder)

ทำหน้าที่เปรียบเทียบอินพุทเวกเตอร์กับตัวเก็บรหัส (Codebook) ซึ่งประกอบด้วยกลุ่มของเวกเตอร์ที่แทนเวกเตอร์ที่เป็นไปได้ทั้งหมด เราเรียกเวกเตอร์เหล่านี้ว่า Codework โดยการเปรียบเทียบนี้จะได้ผลลัพธ์คือดัชนีที่แทน Codeword ที่มีความเพี้ยนน้อยที่สุด (Minimum Distortion) กับอินพุทเวกเตอร์ ดังแสดงในแผนภูมิที่ 3.11 ซึ่งเราจะนำส่วนนี้มาหาว่าภาพใบหน้าภาพต่าง ๆ ที่เข้ามานั้นควรจัดอยู่ในกลุ่มหมายเลขใด การหาความเพี้ยนนี้หาได้จาก

$$d(x,\hat{x}_i)=\frac{1}{K}\sum_{m=1}^{K}[x(m)-\hat{x}_i(m)]^2 \qquad (3.32)$$

จากแผนภูมิที่ 3.11 การทำงานเริ่มต้นจากการป้อนข้อมูลเวกเตอร์ที่ต้องการตรวจ เข้ามาและทำการหาความเพี้ยนกับทุก ๆ กลุ่ม จากนั้นนำมาหาความเพี้ยนน้อยที่สุดเพื่อเลือกว่าภาพใบหน้าของเรา ที่ป้อนเข้ามาตรวจสอบนั้นจัดอยู่ในกลุ่มใด

แผนภูมิที่ 3.11




**บล็อกของส่วนเข้ารหัส**

**3.6 โครงข่ายประสาทเทียม**

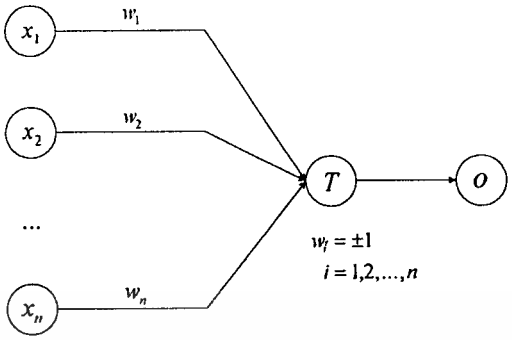
โครงข่ายประสาทเทียม [22-24] เป็นวิธีหนึ่งที่ช่วยให้คอมพิวเตอร์มีความสามารถมากขึ้น โดยเฉพาะการประมวลผลข่าวสารที่มีความยุ่งยากซับซ้อน เช่น การทำให้คอมพิวเตอร์สามารถรู้จำวัตถุ รู้จำตัวอักษร รู้จำใบหน้า หรือ รู้จำเสียงพูด ซึ่งจะต้องมีการตัดสินใจที่ต้องอาศัยความรู้และประสบการณ์ ซึ่งถ้าใช้เทคนิคทางคณิตศาสตร์ธรรมดาแก้ปัญหาก็จะทำให้ระบบมีความซับซ้อนมาก แต่ถ้าใช้ระบบโครงข่ายประสาทเทียม ก็จะช่วยลดความยุ่งยากลงได้มากทีเดียว โดยระบบที่สร้างขึ้นมาได้ถูกเรียกว่าระบบแบบจำลองโครงข่ายประสาทเทียม (Artificial Neural Network System : ANNS) ที่ทำให้คอมพิวเตอร์มีความสามารถทางด้านที่มันสามารถเรียนรู้ได้ และสามารถตัดสินใจให้ระบบได้ โดยมันได้ถอดแบบมาจากการทำงานของระบบสมองของมนุษย์

แบบจำลองระบบประสาทที่ใช้ในการประมวลผลโดยเครื่องคอมพิวเตอร์เพื่อนำไปใช้ควบคุม (รักษา สมดุล) ต่างๆ นั้นระบบแรกถูกนำเสนอโดย McCulloch และ Pitt ในปีค.ศ. 1943 ซึ่งเป็นแบบจำลองของเซลล์ประสาทดังภาพที่ 3.12 อินพุต $x_i$ (สำหรับ $i = 1,2,\ldots,n$) จะมีค่าเป็น {0, 1} ซึ่งจะขึ้นอยู่กับสัญญาณอินพุตจากเซลล์อื่นในขณะนั้นว่าจะมีหรือไม่มีสัญญาณ ส่วนสัญญาณที่จะส่งต่อไปยังเซลล์ถัดไปซึ่งเป็นเซลล์ผลลัพธ์ (จะแทนด้วย o) และ Firing Level ของแบบจำลองนี้ถูกกำหนดโดย

$$o^{k+1} = \begin{cases} 1 & if \ \sum_{i=1}^{n} w_i x_i^k \geq T \\ 0 & otherwise \end{cases} \tag{3.33}$$

โดยที่ $k = 0,1,2,\ldots$ เป็นช่วงเวลาแบบไม่ต่อเนื่อง $w_i$ เป็นค่า ถ่วงน้ำหนัก ที่เชื่อมต่อกับอินพุตที่ $i$ ซึ่งถ้า $w_i = +1$ แสดงถึงการกระตุ้นของซิแนปส์ และถ้า $w_i = -1$ ซิแนปส์จะมีการยับยั้งการส่งผ่านสัญญาณ และ $T$ เป็นค่าความต่างศักดิ์เทรสโฮลด์หรือขีดเริ่มเปลี่ยนซึ่งถ้าค่าผลรวมของผลคูณระหว่างค่าถ่วงน้ำหนักกับสัญญาณอินพุตจะต้องมากกว่า $T$ จึงจะมีสัญญาณผ่านไปยังเซลล์อื่นได้

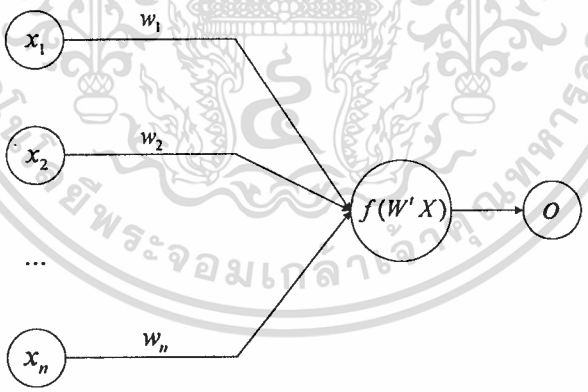
ภาพที่ 3.12




แบบจำลองเซลประสาทของ McCulloch-Pitts

โครงข่ายอีกแบบหนึ่งซึ่งคล้ายกับแบบจำลองของ McCulloch-Pitts แต่แตกต่างกันตรงที่ ค่าของตัวแปรต่างๆ ที่ใช้ในแบบจำลอง ANNS เป็นเลขจำนวนจริง และมีค่าถ่วงน้ำหนักจะได้จากการเรียนรู้ของระบบ ซึ่งแบบจำลองนี้แสดงในภาพที่ 3.13

ภาพที่ 3.13




แบบจำลองโครงข่ายเซลล์ประสาทเทียม

จากภาพที่ 3.13 แสดงโครงข่ายการเชื่อมต่อของบบจำลองเซลล์ประสาทที่สามารถสอนให้โครงข่ายตัดสินใจได้ โดยมี $x_i$ เป็นสัญญาณอินพุต และ $w$ เป็นค่าถ่วงน้ำหนักที่ได้จากการสอนโครงข่าย และแต่ละโหนดในโครงข่ายจะใช้แทนเซลล์ประสาทแต่ละเซลล์ ซึ่งบางครั้งจะเรียกว่าหน่วยประมวลผลพื้นฐาน (Process Element Unit) และมีซิแนปส์ ซึ่งจะเชื่อมต่อโหนดเพื่อใช้ในการส่งสัญญาณการกระตุ้นหรือยับยั้งสัญญาณจะขึ้นกับค่าถ่วงน้ำหนัก $w_i$ และสำหรับสัญญาณเอาต์พุตสามารถจะคำนวณได้ดังนี้

$$o = f(W'x) \tag{3.34}$$

โดยที่ $W$ เป็นเวกเตอร์ของค่าถ่วงน้ำหนักซึ่งสามารถกำหนดได้ดังนี้

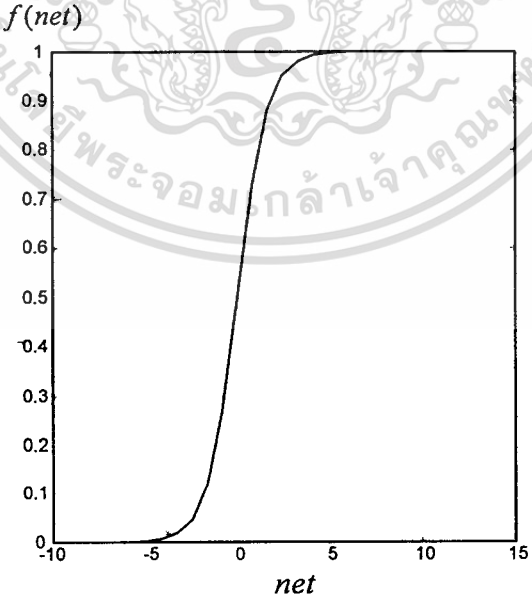
$$W = [w_1, w_2, \ldots, w_n]' \tag{3.35}$$

และ $X$ เป็นเวกเตอร์อินพุต $X = [x_1, x_2, \ldots, x_n]'$ เมื่อ $t$ เป็นตัวดำเนินการทรานสโพสต์ของเมตริก ฟังก์ชันกำหนดสัญญาณเอาต์พุตในสมการที่ 3.34 ถูกเรียกว่าฟังก์ชันการเร่งเร้าหรือแอกติเวชันฟังก์ชัน (Activation Function) และกำหนดให้

$$net = W'X = \sum_{i=1}^{n} w_i x_i \tag{3.36}$$

ฟังก์ชันการเร่งเร้ามีคุณสมบัติคล้ายกับกราฟของศักย์ไฟฟ้าขณะทำงาน มีด้วยกันสองชนิดคือ ชนิดที่เป็นเชิงเส้นและชนิดที่ไม่เป็นเชิงเส้น การที่จะกำหนดว่าฟังก์ชันการเร่งเร้าใดมีคุณสมบัติดีที่สุดคงเป็นไปไม่ได้ เนื่องจาก ANNS ไม่มีโครงข่ายที่แน่นอน นั่นคือโครงข่ายหนึ่งจะใช้กับปัญหาใดปัญหาหนึ่งเท่านั้น ดังนั้นการเลือกฟังก์ชันการเร่งเร้าใดจะต้องพิจารณาให้เหมาะสมกับปัญหานั้นๆ ตัวอย่างแอกติเวชันฟังก์ชันแบบต่อเนื่องที่นิยมใช้กันมากคือฟังก์ชัน Sigmoid ดังแสดงในภาพที่ 3.14

ภาพที่ 3.14




$$f(net) = \frac{1}{1+e^{-net}}$$

แสดง Activation Function แบบ Sigmoid

**3.6.1 การเรียนรู้ของโครงข่ายประสาทเทียม**

การเรียนรู้ของโครงข่าย ANNS จะมีประสิทธิภาพเพียงใดนั้นขึ้นกับค่าถ่วงน้ำหนักของโครงข่าย ซึ่งการฝึกสอน (Training) โครงข่ายก็คือการหาค่าถ่วงน้ำหนักที่เหมาะสมให้แก่โครงข่ายนั้นๆ วิธีการสอน ANNS มีอยู่สองแบบด้วยกันคือ การสอนแบบชี้นำ และการสอนแบบไม่ชี้นำ

**3.6.1.1 การเรียนรู้แบบชี้นำหรือดูแล (Supervised Learning)** การสอนโดยวิธีนี้จะกำหนดเซตของการสอนให้กับโครงข่ายซึ่งเซตนี้ประกอบด้วยอินพุตและเอาต์พุตที่ต้องการ (Output Desired) เมื่อป้อนอินพุตให้กับโครงข่าย โครงข่ายก็จะทำการประมวลผลจนได้คำตอบและค่าถ่วงน้ำหนักออกมาชุดหนึ่ง สำหรับคำตอบที่ได้จากโครงข่ายจะถูกนำมาคำนวณค่าความผิดพลาดโดยวัดเป็นระยะทางว่ามีความห่างจากคำตอบที่ต้องการของอินพุตในชุดเดียวกันมากน้อยเพียงใด ถ้ายังมีความผิดพลาดสูงอยู่ก็จะมีการปรับค่าถ่วงน้ำหนักและทำการสอนต่อไปจนกว่าค่าความผิดพลาดระหว่างคำตอบโครงข่ายกับเอาต์พุตที่ต้องการมีค่าน้อยพอที่จะยอมรับได้จึงจะหยุดการสอน และค่าถ่วงน้ำหนักที่ได้ก็จะเป็นเหมือนฟังก์ชันที่ใช้ในการแปลงข้อมูล

**3.6.1.2 การเรียนรู้แบบไม่มีการชี้นำหรือไม่มีการดูแล (Unsupervised)** การสอนโดยวิธีนี้จะป้อนอินพุตเข้าสู่โครงข่าย และภายในโครงข่ายจะมีเอาต์พุตโหนดอยู่หลายโหนดด้วยกันโดยแต่ละโหนดจะแทนกลุ่มของข้อมูลที่มีคุณสมบัติเหมือนกัน- เมื่อป้อนอินพุตเข้าสู่โครงข่าย โครงข่ายจะคำนวณค่าความสัมพันธ์ที่มีอยู่ภายในเซตของอินพุตโดยอาศัยค่าถ่วงน้ำหนักเป็นตัวแยกความแตกต่างของอินพุตไปเก็บไว้ในโหนดเอาต์พุตของโครงข่าย การสอนโดยวิธีนี้จะไม่สามารถระบุได้ว่าเอาต์พุตโหนดใดเป็นของข้อมูลกลุ่มไหน ผู้ใช้จะต้องกำหนดเองซึ่งแตกต่างจากการสอนแบบชี้นำที่โครงข่ายสามารถระบุกลุ่มเอาต์พุตได้อย่างแน่นอน

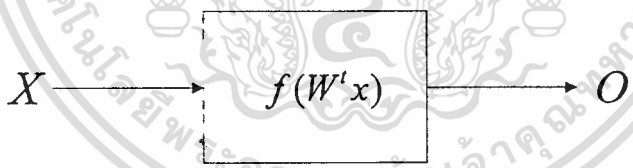
การสอนโครงข่ายเซลล์ประสาทแบบจำลองเป็นการหาฟังก์ชันการแปลง และฟังก์ชันการแปลงที่ได้จะมีคุณสมบัติไม่เป็นเชิงเส้น ซึ่งฟังก์ชันการแปลงของ ANN ในที่นี้คือเซตของค่าถ่วงน้ำหนักของโครงข่าย ดังนั้นฟังก์ชันการแปลงจะมีศักยภาพมากน้อยเพียงใดนั้นจะขึ้นอยู่กับค่าถ่วงน้ำหนักของโครงข่ายนั้นๆ ว่ามีเสถียรภาพมากน้อยเพียงใด และ ค่าถ่วงน้ำหนักคำนวณได้จากการสอนโครงข่าย ซึ่งการสอนโครงข่ายมีหลายแบบด้วยกัน เช่น กฎการสอนของ Hebb, กฎการสอนแบบ Perceptron ของ Rosenblatt, กฎการสอนแบบเดลต้า, กฎของ Widrow-Take-All, และกฎการสอนแบบ Outstar ของ Grossberg

**3.6.2 แบบจำลองของ ANN**

โครงข่ายเซลล์ประสาทที่เป็นแบบจำลองและเป็นโครงข่ายของเซลล์ประสาทจริงของมนุษย์ จะมีการเชื่อมต่อกันของโหนดในลักษณะของโครงข่ายอย่างแน่นหนา เพื่อให้โครงข่ายสามารถเรียนรู้และสามารถจดจำสิ่งที่เรียนรู้มาแล้วได้ ซึ่งการเชื่อมโยงของโครงข่ายจะมีอยู่สองลักษณะด้วยกันคือ

**3.6.2.1 โครงข่ายที่ส่งสัญญาณไปข้างหน้า (Feedforward Networks)** โครงข่ายชนิดนี้จะประกอบด้วยชั้นต่างๆ ของโครงข่ายโดยชั้นแรกจะเป็นอินพุตและชั้นสุดท้ายเป็นชั้นของเอาต์พุต ส่วนระหว่างชั้นอินพุตกับเอาต์พุตอาจจะมีหรือไม่มีชั้นที่แทรกอยู่ภายในก็ได้ ซึ่งจะขึ้นอยู่กับอัลกอริทึมที่ใช้ในการสอนโครงข่าย เช่นถ้าเป็นโครงข่าย Perceptron แบบหลายชั้น (Multilayer Perceptron) ก็จะมีชั้นที่อยู่ระหว่างอินพุตกับเอาต์พุตอีกซึ่งอาจมีมากกว่าหนึ่งชั้นก็ได้ ส่วนโครงข่าย Self-Organizing Map ของ Kohonen จะมีเพียงชั้นของอินพุตกับเอาต์พุตเท่านั้น การเชื่อมต่อระหว่างโครงข่ายแบบ Feedforward จะมีค่าถ่วงน้ำหนักเป็นตัวเชื่อม และสัญญาณ อินพุตที่เข้ามาจะถูกส่งไปตามทิศทางของลูกศรจนถึงชั้นเอาต์พุตโดยไม่มีการป้อนกลับ ดังภาพที่ 3.15

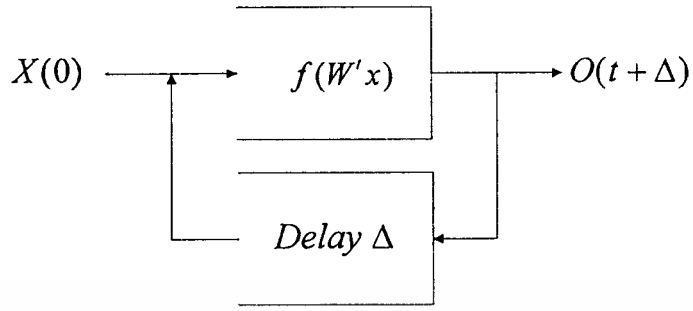
ภาพที่ 3.15




บล็อคไดอะแกรมของโครงข่าย Feedforward

**3.6.2.2 โครงข่ายที่มีการป้อนกลับ (Feedback Networks)** ในส่วนแรกของโครงข่ายนี้จะเป็นโครงข่าย Feedforward เหมือนกับแบบแรก และส่วนที่เพิ่มเข้ามาคือส่วนของการป้อนกลับซึ่งจะมีการหน่วงเวลาไปจากเวลาเดิมเท่ากับ $\Delta$ ดังภาพที่ 3.16

ภาพที่ 3.16

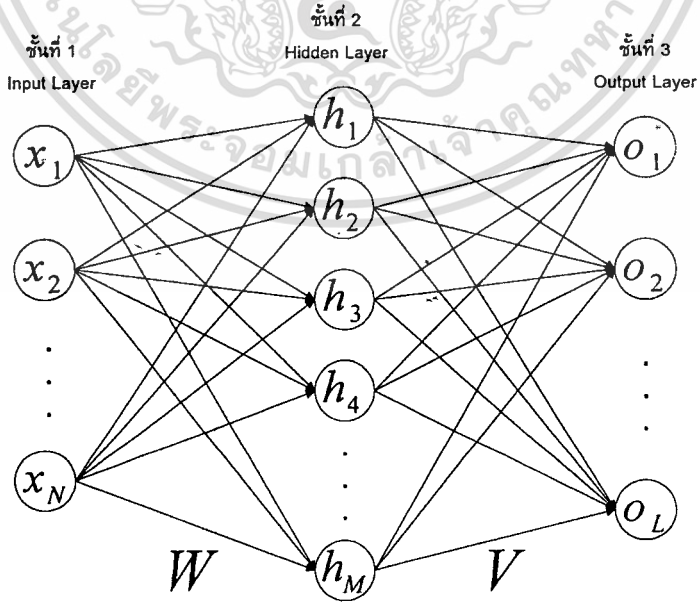



บล็อคไดอะแกรมของโครงข่าย Feedback

**3.6.3 การแพร่กระจายกลับ (Back-propagation)**

การแพร่กระจายกลับหรือแบกโพรพาเกชัน (Back-propagation) เป็นขั้นตอนที่ใช้สอนโครงข่ายแบบ Multilayer Perceptron ซึ่งเป็นแบบจำลองโครงข่ายเซลล์ประสาทที่มีการเชื่อมโยงกันเป็นโครงข่ายแบบเป็นชั้นๆ ดังในภาพที่ 3.17

ภาพที่ 3.17




โครงข่าย Multilayer Perceptron ที่มี 3 ชั้น

โครงข่ายนี้มีการเชื่อมโยงกันสามชั้น ซึ่งประกอบด้วยชั้นของอินพุตซึ่งมีเซลล์ประสาทอยู่ $N$ โหนด ถัดมาเป็นชั้นของฮิดเดนหรือชั้นภายใน (Hidden Layer) ซึ่งประกอบด้วยโหนดต่างๆ จำนวน $M$ โหนด และสุดท้ายคือ ชั้นของเอาต์พุตซึ่งมีโหนดต่างๆ อยู่ $L$ โหนด โครงข่ายแบบ Multilayer Perceptron ในภาพที่ 3.17 แต่ละโหนดในชั้นเดียวกันจะไม่มีการเชื่อมต่อกัน การเชื่อมโยงกันจะมีเฉพาะระหว่างชั้นเท่านั้น และการเชื่อมโยงนี้จะต่อถึงกันทุกโหนด โครงข่ายแบบ Multilayer Perceptron ไม่จำเป็นต้องมีสามชั้นเหมือนในภาพที่ 3.17 อาจจะมีจำนวนชั้นมากกว่านี้ก็ได้ ซึ่งอาจมีสี่ชั้นโดยการเพิ่มชั้นฮิดเดนเข้าไปอีกหนึ่งชั้น หรือถ้าต้องการจำนวนชั้นมากกว่านี้ก็สามารถทำได้

ชั้นฮิดเดนเป็นตัวเพิ่มความสามารถให้แก่โครงข่ายเซลล์ประสาท โครงข่ายเซลล์ประสาทแบบ Multilayer Perceptron ถ้าไม่มีชั้นฮิดเดนก็จะกลายเป็นโครงข่ายแบบ Perceptron

การที่จะกำหนดชั้นของฮิดเดนว่าในโครงข่ายหนึ่งๆ ควรมีฮิดเดนกี่ชั้น และแต่ละชั้นประกอบด้วยกี่โหนดนั้นไม่มีกฎเกณฑ์หรือทฤษฎีที่แน่นอน ดังนั้นการกำหนดจำนวนชั้นและจำนวนโหนดของแต่ละชั้นของฮิดเดนสามารถทำได้โดยการทดลอง

### 3.6.4 วิธีการของกฎเดลต้าเอนกประสงค์

กฎเดลต้า (Delta Rule) ถูกพัฒนาขึ้นมาเพื่อใช้สอนโครงข่ายเซลประสาทจำลอง [25] ซึ่งครั้งแรกใช้ในการสอนโครงข่ายเซลประสาทจำลอง Perceptron ซึ่งเป็นกฎการสอน Perceptron แบบต่อเนื่อง และต่อมาได้พัฒนากฎการสอนเดลต้ารูล์ให้ใช้ได้กับโครงข่ายที่มีการเชื่อมต่อกันหลายชั้นอย่าง Multilayer Perceptron จึงเรียกกฎเดลต้าที่ถูกพัฒนาขึ้นใหม่นี้ว่ากฎเดลต้าเอนกประสงค์ (Generalized Delta Rule:GDR)

ขั้นตอนการสอนโครงข่าย Multilayer Perceptron จะเรียกว่า ขั้นตอนการสอนแบบค่าความผิดพลาดแพร่กรจายกลับ หรือ Error Back-propagation ซึ่งเป็นการแพร่กระจายกลับของค่าความผิดพลาดที่เกิดขึ้นในชั้นเอาต์พุตที่ต้องการกับเอาต์พุตที่คำนวณได้ โดยคำนวณย้อนกลับจากชั้น เอาต์พุตผ่านชั้นฮิดเดนตลอดมาจนถึงชั้นอินพุต เพื่อทำ การปรับค่าวงน้ำหนักจากภาพที่ 3.17 เมื่อทำการสอนโครงข่ายนี้ ซึ่งมีการสอนแบบชี้นำ ดังนั้นในการสอนจะต้องป้อนเซตของข้อมูลที่จะใช้สอนซึ่งประกอบด้วยเซตของอินพุตและเซตของเอาต์พุตที่ต้องการซึ่งจะสอดคล้องกับเซตของอินพุต กำหนดให้ $P$ เป็นจำนวนเซตอินพุตทั้งหมดที่ใช้ในการสอนโครงข่าย ดังนั้นเซตของอินพุตจะมีอยู่ $P$ เซต ถ้าให้ $D$ เป็นเซตของเอาต์พุตที่ต้องการจะได้ว่า $D_p = \{d_1, d_2, ..., d_L\}$ เมื่อ $p = 1,2,...,P$

$W$ เป็น เมตริกซ์ ของค่าถ่วง น้ำหนัก ที่เชื่อมโยงระหว่างชั้นอินพุตกับชั้นฮิดเดนซึ่ง $W$ มีขนาด $M$ แถว และ $N$ คอลัมน์

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{MN} \end{bmatrix} \tag{3.37}$$

$V$ เป็นเมตริกซ์ของค่าถ่วงน้ำหนักที่เชื่อมโยงระหว่างชั้นฮิดเดนไปยังชั้นเอาต์พุต ซึ่งมีขนาด $L$ แถวและ $M$ คอลัมน์

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1M} \\ v_{21} & v_{22} & \cdots & v_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ v_{L1} & v_{L2} & \cdots & v_{LM} \end{bmatrix} \tag{3.38}$$

และ $f(net)$ เป็น Activation Function ดังที่แสดงในภาพที่ 3.14

เมื่อทำการป้อนเซตของข้อมูลที่ใช้สอนเข้าสู่โครงข่ายจะสามารถคำนวณโหนดของฮิดเดนที่ $m$ ได้ดังนี้

$$net_m = \sum_{n=1}^{N} w_{mn} x_n \tag{3.39}$$

$$h_m = f(net_m) \tag{3.40}$$

และสามารถคำนวณเอาต์พุตโหนดที่ $l$ ได้ดังนี้

$$net_l = \sum_{n=1}^{N} v_{lh} x_n \tag{3.41}$$

$$o_l = f(net_l) \tag{3.42}$$

เมื่อคำนวณชั้นของเอาต์พุตครบทุกโหนดแล้ว ขั้นต่อไปเป็นการปรับค่าถ่วงน้ำหนักของโครงข่าย โดยใช้ค่าผลรวมของค่าความคลาดเคลื่อนกำลังสอง ของ ผลลัพธ์ที่ได้จากโครงข่ายกับค่าเอาต์พุตที่ต้องการของแพทเทิร์นที่ $p$ ซึ่ง สามารถคำนวณได้ดังนี้

$$E_p = \frac{1}{2}\sum_{l=1}^{L}(o_{pl} - d_{pl})^2 \tag{3.43}$$

ดังนั้นในการปรับค่าถ่วงน้ำหนักที่เชื่อมต่อระหว่างชั้นเอาต์พุตกับชั้นฮิดเดน สำหรับแพทเทิร์นที่ $p$ ( $H_p$ ) สามารถคำนวณได้ดังนี้

$$\begin{aligned}\Delta V &= -\eta\frac{\partial E_p}{\partial V} \\ &= \alpha V + \eta\delta_p^{V} H_p\end{aligned} \tag{3.44}$$

โดยที่ $\delta_p^V = (D_p - O_p)O_p(1-O_p)$ คือความคลาดเคลื่อนภายในชั้นของเอาต์พุต $\alpha$ เป็นค่าโมเมนตัม (Momentum) ซึ่งเป็นค่าช่วยป้องกันการแกว่ง (Oscillate) ของระบบ $\eta$ เป็นค่าอัตราเร็วในการเรียนรู้ (Learning Rate) ซึ่งเป็นค่าคงที่อยู่ระหว่าง 0.05-0.25 ในการกำหนดค่าโมเมนตัมต้องสัมพันธ์กับค่าอัตราเร็วในการเรียนรู้ คือถ้า อัตราเร็วในการเรียนรู้มีค่ามากแต่ค่าโมเมนต์ตัมมีค่าน้อยจะทำให้โครงข่ายเกิดการแกว่งได้ และการกำหนดค่าทั้งสองนี้จะมีผลต่อเวลาที่ใช้ในการสอน

และการปรับค่าถ่วงน้ำหนักที่เชื่อมต่อระหว่างชั้นฮิดเดนกับชั้นอินพุตสามารถคำนวณได้ดังนี้

$$\Delta W = \alpha W + \eta\delta_p^{W} X_p \tag{3.45}$$

โดยที่ $\delta_p^W = H_p(1-H_p)(\delta_p^V V)$ ซึ่งเป็นค่าความคลาดเคลื่อนภายในชั้นฮิดเดน สำหรับ $a$ และจะมีคุณสมบัติเหมือนกับสมการที่ 3.44

ในกรณีที่โครงข่ายมี $I$ ชั้นสามารถปรับค่าถ่วงน้ำหนักโดยวิธีค่าความผิดพลาดแพร่กระจายกลับได้ดังนี้

$$\Delta W^i = \alpha W^i + \eta\delta_p^i X_p^{i-1} \tag{3.46}$$

เมื่อ $W^i$ เป็นเมตริกซ์ค่าถ่วงน้ำหนักที่อยู่ระหว่างชั้นที่ $i$ กับ $i-1$ $(i=1,2,...,I)$ เป็นเซตของอินพุตแพทเทิร์นที่ $p$ จากชั้นที่ $i-1$ และ $\delta_p^i$ เป็นเวกเตอร์ความคลาดเคลื่อนสำหรับชั้นที่ $i$ สำหรับชั้นเอาต์พุตหรือชั้นที่ $I$ สามารถคำนวณได้ดังนี้

$$\delta_p^I = (D_p - O_p^I)O_p^I(1-O_p^I) \tag{3.47}$$

โดยที่ $O_p^I$ คือชั้นของเอาต์พุต และสำหรับความคลาดเคลื่อนของชั้นที่ $i$ ใดๆ ที่สามารถคำนวณได้ดังนี้

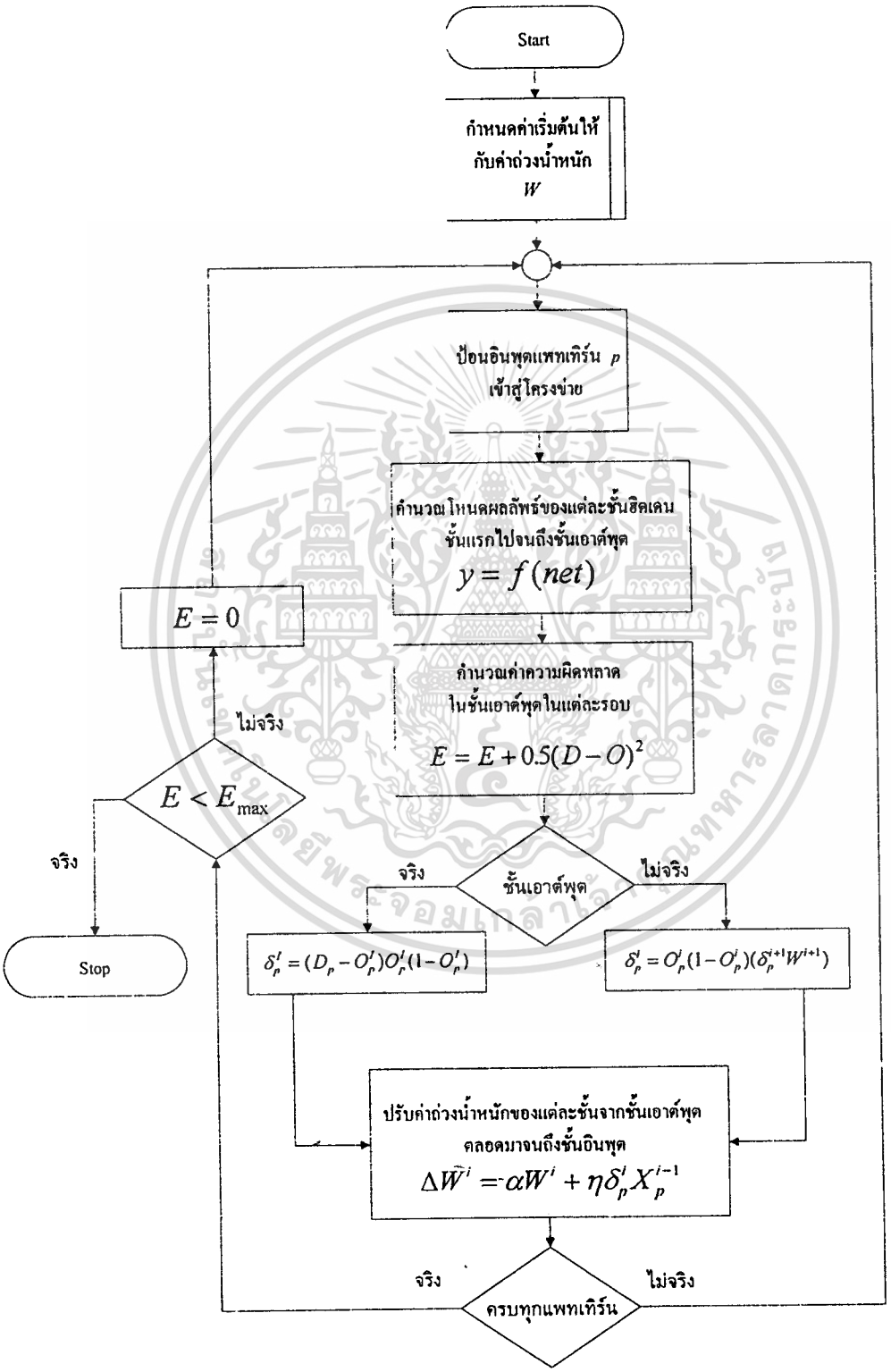
$$\delta_p^i = O_p^i(1-O_p^i)(\delta_p^{i+1}W^{i+1}) \tag{3.48}$$

โดยที่ $O_p^i$ คือเอาต์พุตชั้นที่ $i$ แพทเทิร์นที่ $p$ และสำหรับ $\delta_p^i$ และ $W^{i+1}$ เป็นค่าความคลาดเคลื่อนและเมตริกค่าถ่วงน้ำหนักของชั้นถัดไป

กระบวนการในการเรียนรู้ของโครงข่ายประสาทเทียมแสดงดังภาพที่ 3.18

จากภาพที่ 3.18 ในการฝึกโครงข่ายประสาทเทียม เริ่มต้นจากการให้ค่าเริ่มต้นแบบสุ่มแก่ตัวถ่วงน้ำหนัก แล้วป้อนแพทเทิร์นหนึ่งเป็นอินพุต แล้วคำนวณค่าที่โหนดผลลัพธ์ของแต่ละชั้น จนถึงเอาต์พุตจากนั้นคำนวณค่าความผิดพลาดชั้นเอาต์พุต แล้วคำนวณความคลาดเคลื่อนในชั้นต่างๆ ซึ่งถ้าเป็นชั้นเอาต์พุตจะใช้สูตรดังสมการที่ 3.47 แต่หากเป็นชั้นใดๆ ใช้สูตรดังสมการที่ 3.48 แล้วปรับค่าถ่วงน้ำหนักของแต่ละชั้นจากชั้นเอาต์พุตจนถึงชั้นอินพุต แล้วทำการรับแพทเทิร์นอื่นๆ เข้ามาจนครบทุกแพทเทิร์น แล้วทำการตรวจสอบว่าค่าผิดพลาดเฉลี่ยที่ได้น้อยกว่าที่ยอมรับได้แล้วหรือยัง ถ้าน้อยกว่าก็จะสามารถสิ้นสุดการฝึกได้ทันที แต่ถ้ายังคงมีค่าสูงกว่าอยู่ให้ค่าความคลาดเคลื่อนในชั้นเอาต์พุตเป็นศูนย์ใหม่อีก ครั้งหนึ่งและเริ่มกระบวนการใหม่อีกครั้ง

**แผนภูมิที่ 3.18**



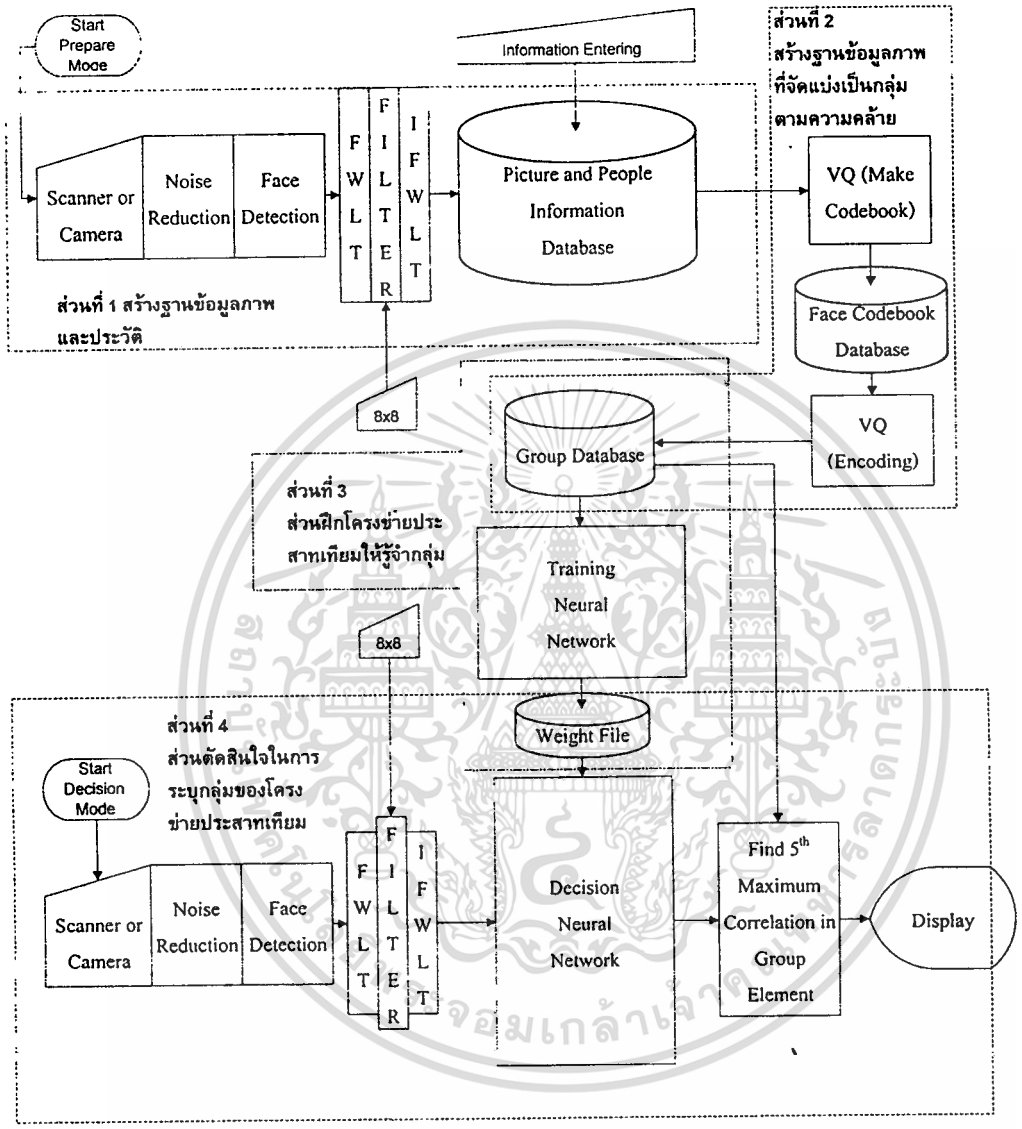

**โฟล์ตชาร์ตของการสอนแบบ Error Back-propgation ที่มี $I$ ชั้น**

# บทที่ 4
# ระบบการรู้จำใบหน้าด้วยโครงข่ายประสาทเทียม

ระบบการรู้จำใบหน้าที่นำเสนอนี้จะเน้นการรู้จำโดยใช้โครงข่ายประสาทเทียมที่รู้จำในลักษณะกลุ่มใบหน้า (คล้ายระบบรู้จำลายนิ้วมือ) เพื่อช่วยแก้ปัญหาการเรียนรู้ใหม่ทุกครั้งที่มีการเปลี่ยนแปลงฐานข้อมูลสำหรับโครงข่ายประสาทเทียม ดังแสดงในภาพที่ 4.1 ผลจากการรู้จำจากโครงข่ายประสาทเทียมจะได้ตัวชี้กลุ่ม (Group ID) ซึ่งก็จะทำให้เราสามารถรู้ว่าภาพที่เข้ามาทดสอบมีความคล้ายกับชุดภาพหรือกลุ่มภาพใดจากระบบที่ใช้โครงข่ายประสาทเทียมนี้ได้ในเวลาที่รวดเร็วไม่เกิน 2 วินาทีแม้ฐานข้อมูลจะมีขนาดใหญ่มากก็ตาม หลังจากการรู้กลุ่มแล้วเราสามารถนำมาหาลำดับความคล้ายเรียงตามลำดับได้อีกโดยการใช้การเทียบภาพด้วยสหสัมพันธ์ที่เลือก 5 อันดับเฉพาะภายในกลุ่มนั้นอีกครั้ง ซึ่งสมาชิกภายในกลุ่มจะมีจำนวนไม่มาก ดังนั้นจึงสามารถค้นหาได้อย่างรวดเร็วกว่าการต้องค้นหาจากฐานข้อมูลทั้งหมดโดยเฉพาะจะมีประโยชน์เมื่อเป็นฐานข้อมูลมีขนาดใหญ่มาก

ระบบทั้งหมดของการรู้จำใบหน้าที่นำเสนอในวิทยานิพนธ์นี้แสดงดังภาพที่ 4.1

แผนภูมิที่ 4.1




ระบบทั้งหมดที่นำเสนอในการรู้จำภาพใบหน้า

**การทำงานของระบบแบ่งเป็น 4 ส่วนใหญ่ ๆ คือ**

1) การเก็บภาพเป็นฐานข้อมูลภาพโดยการใช้เครื่องสแกนเนอร์ และเก็บประวัติบุคคล โดยนำภาพทุกภาพมากำจัดสัญญาณรบกวน ทำการหาส่วนของใบหน้าในภาพ (Face Detection) และนำมาทำการลดรายละเอียดของภาพลง ด้วยการแปลงเวฟเลตเพื่อแปลงภาพไปอยู่ในโดเมนของเวฟเลต แล้วทำการกรองความถี่สูงทิ้งไป และการแปลงกลับของเวฟเลตเพื่อแปลงภาพกลับมาสู่โดเมนภาพปกติ (Spatial Domain) หลังจากนั้นทำการป้อนประวัติของ

บุคคลต่างๆ เป็นส่วนของ Information Database ด้วยถ้าต้องการ โดยข้อมูลทุกตัวจะมี ID เป็นตัวชี้เพื่อป้อนข้อมูลเข้าหรืออ่านข้อมูลตัวนั้นออกไปเสมอ ทำการเก็บภาพและป้อนประวัติจนครบทุกคนจนเป็นระบบฐานข้อมูลของหน่วยงานขึ้นมาตามต้องการ

2) การจัดกลุ่มภาพใบหน้าด้วยเวกเตอร์ควอนไตซ์จะนำภาพมาสร้างตัวเก็บรหัส (Codebook) โดยเราสามารถกำหนดจำนวนกลุ่มได้ตามต้องการ ซึ่งในที่นี้สามารถเลือกกำหนดจำนวนกลุ่มได้เป็น $2^n$ เมื่อ $n = 1,2,...,8$ ซึ่งหมายความว่า เราสามารถเลือกจำนวนกลุ่มได้เป็น 1, 2, 4, 8, 16, 32, 64, 128 และ 256 ไปใช้ได้ตามความเหมาะสม จากนั้นนำภาพทุกภาพในฐานข้อมูลมาทำการหาว่าเป็นสมาชิกในกลุ่มใด โดยการทำกระบวนการเข้ารหัสเพื่อหาความเพี้ยนน้อยที่สุดเมื่อเทียบกับตัวเก็บรหัส ได้ผลลัพธ์เป็นตัวชี้กลุ่มออกมา ทำการบันทึกและจัดกลุ่มข้อมูลใหม่ให้มีการกระจายแบบกลุ่มแทนการเรียงกันตามรหัส

3) การฝึกโครงข่ายประสาทเทียม จะเป็นการนำภาพใบหน้าของบุคคลต่างๆ จากฐานข้อมูลที่มีการจัดกันเป็นกลุ่มเข้ามาเป็นอินพุต และใช้หมายเลขกลุ่มเป็นตัวกำหนดเอาต์พุตของโครงข่ายประสาทเทียม ทำการฝึกให้โครงข่ายประสาทเทียมเรียนรู้ข้อมูลดังกล่าวสำหรับบุคคลต่างๆ จนครบทุกคน การฝึกนี้จะเห็นได้ว่าเป็นการฝึกแบบชี้นำ คือมีคู่ระหว่างอินพุตและเอาต์พุตที่ต้องการกำหนดไว้ให้ในการเรียนรู้ของโครงข่ายประสาทเทียม โดยผลจากการเรียนรู้จะได้เป็นไฟล์ค่าถ่วงน้ำหนัก (Weight File) เพื่อไว้ใช้ต่อไป

4) การตัดสินใจด้วยโครงข่ายประสาทเทียมและการหา 5 อันดับความคล้าย จะเริ่มต้นจากการใช้เครื่องสแกนภาพทำการเก็บภาพเข้ามาตรวจสอบ โดยผ่านการกำจัดสัญญาณรบกวนแล้วผ่านการค้นหาส่วนของใบหน้าในภาพ เมื่อได้ภาพใบหน้าที่ต้องการจะนำมาผ่านการแปลงเวฟเลตแล้วผ่านตัวกรองในเวฟเลตโดเมนเพื่อตัดความถี่สูงทิ้งไปแล้วทำการแปลงกลับเวฟเลตมาอยู่ในโดเมนของภาพปกติจะได้ภาพที่ถูกตัดรายละเอียดของภาพทิ้งไปตามต้องการ จากนั้นจะเป็นการใช้โครงข่ายประสาทเทียมในการตัดสินใจว่าภาพใบหน้าที่เข้ามาเป็นภาพของบุคคลที่เป็นสมาชิกอยู่ในกลุ่มใด โดยการนำไฟล์ค่าถ่วงน้ำหนักมาใช้ เมื่อนำอินพุตคือภาพที่ต้องการตรวจสอบป้อนเข้ามา และจากไฟล์ค่าถ่วงน้ำหนักที่มีอยู่ในโครงข่ายเรียบร้อยแล้ว ก็จะสามารถคำนวณเอาต์พุตออกมาได้ เอาต์พุตที่ได้จะเป็นตัวชี้กลุ่ม ตัวชี้กลุ่มจะเป็นตัวบอกกระบวนการในการเทียบภาพด้วยวิธีการสหสัมพันธ์ในกระบวนการถัดไปว่าจะต้องทำการตรวจสอบภาพที่เข้ามาเทียบกับสมาชิกคนใดบ้างในฐานข้อมูลขนาดใหญ่ การเลือกกระทำเฉพาะสมาชิกในกลุ่มเช่นนี้เพื่อเป็นการเพิ่มความเร็วในการเปรียบเทียบภาพและอีกทั้งหลังจากการตรวจสอบครบทุกสมาชิกในกลุ่มแล้วจะนำมาเรียง 5 อันดับที่มีความคล้ายมากที่สุดตามลำดับได้อีก

ด้วย ผลจากภาพ 5 ภาพที่ค้นพบและผ่านการเรียงลำดับความคล้ายจะนำมาแสดงบนหน้าจอตามลำดับ และโดยถ้าต้องการเลือกดูประวัติของบุคคลใดในนั้นก็สามารถทำได้อย่างง่ายดาย

**4.1 การเก็บภาพเป็นฐานข้อมูลภาพและเก็บประวัติส่วนบุคคล**

โดยการใช้เครื่องสแกนเก็บภาพ (Scanner) หรือกล้องซีซีดี (Camera) ร่วมกับแผงวงจรจับและเก็บภาพ (Image Capturing Card) ในการเก็บภาพ ในที่นี้เนื่องจากไม่สามารถหากล้องซีซีดีที่มีคุณภาพหรือความละเอียดสูงพอมาใช้ได้ จึงเลือกใช้เครื่องสแกนภาพ เพราะคุณภาพของภาพที่ได้มีคุณภาพดีกว่า จะได้ภาพต้นแบบที่จะต้องนำมาผ่านการกำจัดสัญญาณรบกวนที่อาจจะเกิดจากระบบเก็บภาพหรือจากภาพถ่ายต้นแบบเองที่อาจจะรบกวนกระบวนการต่างๆ ในการประมวลผลต่อๆ ไปได้ ในที่นี้ เลือกใช้ตัวกรองเลือกค่ากลาง (Median Filter) มาทำการกำจัดสัญญาณรบกวน จากนั้นจะทำการหาส่วนของใบหน้าในภาพ ซึ่งใช้วิธีการสหสัมพันธ์ร่วมกับการย่อภาพ 3 ระดับเพื่อให้สามารถทำการค้นหาได้แม้ภาพที่เข้ามามีขนาดแตกต่างกัน ซึ่งจากวิธีการดังกล่าวช่วยให้ได้ภาพใบหน้าที่มีขนาดใกล้เคียงกันออกมาด้วยทั้งนี้เพราะมีการตัดภาพที่เป็นใบหน้าในภาพที่ถูกย่อลงด้วยระดับต่างๆ 3 ระดับดังที่กล่าวมาแล้วในบทที่ 3 ซึ่งภาพที่ได้จึงเป็นภาพที่มีขนาดใกล้เคียงกับภาพอ้างอิงที่ใช้ในการค้นหา จากนั้นนำภาพใบหน้าดังกล่าวที่ได้มาทำการแปลงเวฟเลตแบบเร็ว FWLT (Fast Wavelet Transform) เพื่อหาสัมประสิทธิ์ในโดเมนของเวฟเลตแล้วทำการกรองความถี่เพื่อลดรายละเอียดของภาพ (Reduce Information) แล้วผ่านตัวกรองที่สามารถเลือกสัมประสิทธิ์ในส่วนบนซ้ายของโดเมนเวฟเลตที่ต้องการรักษาไว้ในที่นี้คือขนาด 8x8 ในโดเมนของเวฟเลตให้คงอยู่โดยทำการกำจัดสัมประสิทธิ์ที่เหลือนอกเหลือจากพื้นที่ดังกล่าวให้เป็นศูนย์ทั้งหมดแล้วทำการแปลงกลับ IFWLT (Inverse Fast Wavelet Transform) ออกมาเพื่อแปลงภาพกลับสู่โดเมนภาพปกติ (Spatial Domain) ก็จะได้ภาพที่ถูกลดรายละเอียดของภาพตามต้องการ ข้อมูลภาพที่ผ่านการกรองรายละเอียดนี้จะมีขนาด 32x32 จุดภาพเท่าเดิมแต่รายละเอียดความถี่สูงจะหายไป ข้อมูลภาพดังกล่าวจะถูกนำมาเก็บบันทึกลงเป็นฐานข้อมูลส่วนที่เป็นภาพ (Picture Database) โดยมีตัวชี้ (ID) เป็นตัวกำหนดสมาชิกบุคคลต่างๆ ในการจะดึงไปใช้งาน ID ในที่นี้จะมีค่าเรียงกันไปตั้งแต่คนแรกที่เข้ามาเป็น 1 ไปจนถึง คนสุดท้าย

สำหรับส่วนของการเก็บข้อมูลประวัติส่วนบุคคลจะสามารถป้อนเข้าสู่ ID ต่างๆ ได้เมื่อต้องการ โดยมีการบันทึกชื่อและนามสกุลจริง อายุ วันที่เกิด และประวัติต่างๆ

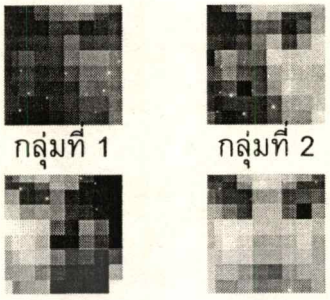
**4.2 การจัดกลุ่มภาพใบหน้าด้วยเวกเตอร์ควอนไตซ์**

**4.2.1 การสร้างตัวเก็บรหัส**

จากฐานข้อมูลที่เป็นภาพใบหน้าที่ถูกลดรายละเอียดของข้อมูลในส่วนความถี่สูงแล้ว จะนำมาทำการจัดกลุ่มของใบหน้าโดยการใช้คุณสมบัติของเวกเตอร์ควอนไตซ์ที่สามารถรวมกลุ่มที่มีความคล้ายกันเข้าด้วยกันโดยมีคุณสมบัติที่เด่นในเรื่องความสามารถในการการปรับเปลี่ยนจากข้อมูลเริ่มต้น (Initial Vector) เพื่อให้ได้ข้อมูลที่มีการแบ่งแยกได้ดีที่สุดในภายหลัง โดยชุดข้อมูลที่ได้มาท้ายสุดนั้นจะเป็นกลุ่มที่ประกอบด้วยสมาชิกที่มีความเพี้ยนจากการจัดกลุ่มน้อยที่สุดเมื่อเทียบกับการจัดกลุ่มสมาชิกชุดอื่นๆ เช่น จากภาพใบหน้าจำนวน 7 ภาพ ที่ประกอบด้วย นาย {ก ,ข, ค, ง, จ, ฉ, ช} เมื่อเราต้องการจัดกลุ่มออกเป็น 2 กลุ่ม การทำงานเริ่มต้นจากการกำหนดเวกเตอร์เริ่มต้นแบบสุ่ม (Random Initialization) สมมติว่าเวกเตอร์ชุดที่ 1 ประกอบด้วย นาย {ก, ข, ค, ง} ชุดที่ 2 ประกอบด้วย นาย {จ, ฉ, ช} แล้วภายหลังจากกระบวนการจัดกลุ่มโดยเวกเตอร์ควอนไตซ์สมมติว่ากลุ่มที่ 1 ได้เป็น นาย {ก, จ, ช} กลุ่มที่ 2 ได้เป็น {ข , ค, ง, ฉ} ผลลัพธ์ที่ได้นี้จะเป็นการเข้าชุดกันที่ดีที่สุดในการแบ่งแยกกลุ่มซึ่งมีความเพี้ยนเฉลี่ยรวมแล้วน้อยกว่าการจัดชุดสลับหรือประกอบด้วยสมาชิกตัวอื่นๆ ซึ่งอันนี้เป็นคุณสมบัติที่ดีอย่างมาก ทำให้เราไม่ต้องพะวงว่าจะให้เวกเตอร์เริ่มต้นหรือใบหน้าเริ่มต้นสำหรับชุดต่างๆ เป็นอย่างไร หมายความว่าทุกอย่างสามารถทำงานได้อย่างอัตโนมัติโดยเวกเตอร์ควอนไตซ์ การกำหนดกลุ่มเวกเตอร์เริ่มต้นนั้น จะมีผลต่อเวลาที่ใช้ในการจัดกลุ่มมากกว่าอย่างอื่น โดยส่งผลกระทบต่อการเลือกสมาชิกกลุ่มที่เป็นผลจากการจัดเพียงเล็กน้อยทั้งนี้เนื่องจากความที่ข้อมูลคล้ายกันมากๆ (จากการทดลองจัดกลุ่มหลายๆ ครั้งโดยเวกเตอร์ควอนไตซ์บางครั้งพบว่าสมาชิกในกลุ่มต่างๆ ก็อาจได้ไม่เหมือนกันได้จากการกำหนดเวกเตอร์ เริ่มต้นต่างกัน เพราะอาจจะมีความคลุมเครือแฝงอยู่ เช่น นาย จ อาจจะคล้ายทั้ง นาย ชในกลุ่ม 1 และอาจคล้ายนาย ข ในกลุ่ม 2 ก็ได้ เป็นต้น) แต่สิ่งนี้ไม่เป็นปัญหาแต่อย่างใดทั้งนี้เพราะเรานำเวกเตอร์ควอนไตซ์มาใช้ในการจัดกลุ่มเบื้องต้นเท่านั้น ข้อมูลที่ได้มาไม่ว่าจะจัดฉลุ่มอย่างไร นั่นคือกลุ่มที่เราถือว่าเป็นเอกฉันท์ที่เราสามารถนำไปใช้โครงข่ายประสาทเทียมในการรู้จำกลุ่มต่อไปได้โดยไม่มีผลแต่อย่างใด นาย จ ไม่ว่าจะอยู่กลุ่ม 1 หรือ 2 ถ้าถูกจัดกลุ่มมาย่อมมีข้อมูลบางอย่างที่ทำให้เกิดเหตุการณ์เช่นนั้น โครงข่ายประสาทเทียมย่อมจะสามารถเรียนรู้ความเป็น นาย จ ในกลุ่มนั้นได้เช่นกัน

ผลจากเวกเตอร์ควอนไตซ์จะได้เป็นตัวเก็บรหัสหรือตัวเก็บภาพใบหน้าที่เป็นผลจากการจัดกลุ่ม เช่น ถ้ามีการจัดเป็น 4 กลุ่มจะได้ตัวเก็บระหัสที่ประกอบด้วยเวกเตอร์ 4 ตัวหรือภาพใบหน้า 4 ภาพ ดังในภาพที่ 4.2

ภาพที่ 4.2




ภาพตัวเก็บรหัสที่ได้จากการจัดกลุ่มเป็น 4 กลุ่ม

**4.2.2 การเข้ารหัส**

เป็นการหาว่าภาพต่างๆ ในฐานข้อมูลเป็นภาพที่ควรจัดเป็นสมาชิกอยู่ในกลุ่มใด โดยจะนำภาพทุกภาพมาทำการหาความเพี้ยนน้อยที่สุด ดังที่ได้กล่าวมาแล้ว ในบทที่ 3 ซึ่งเป็นการหาความใกล้เคียงกับ Codebook ตัวใดมากที่สุดนั่นเอง ซึ่งก็หมายความว่าภาพใบหน้าดังกล่าวมีความคล้ายกับกลุ่มใบหน้ากลุ่มนั้นมากที่สุด ผลลัพธ์จากกระบวนการนี้จะได้เป็นฐานข้อมูลที่จัดเก็บกันเป็นกลุ่มที่มีความคล้ายกัน แทนที่จะจัดเรียงตามลำดับกันแบบปกติแบบเดิม ซึ่งเราจะเรียกว่าฐานข้อมูลกลุ่ม

กลุ่มที่ 1 เก็บภาพทั้งหมด ที่เป็นสมาชิกในกลุ่มที่ 1
กลุ่มที่ 2 เก็บภาพทั้งหมด ที่เป็นสมาชิกในกลุ่มที่ 2

.

.

.

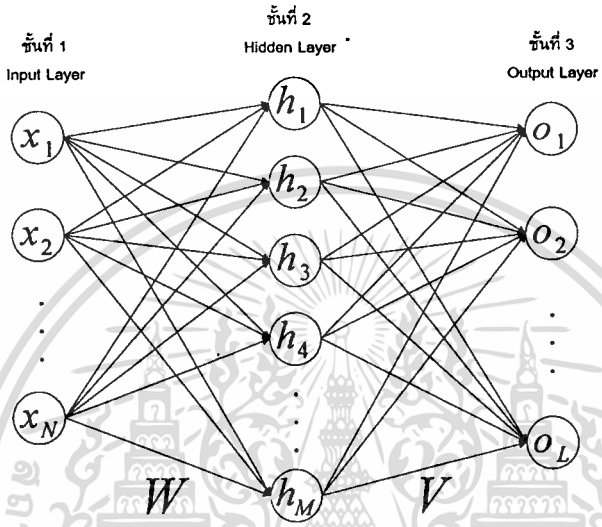
กลุ่มที่ N เก็บภาพทั้งหมด ที่เป็นสมาชิกในกลุ่มที่ N

**4.3 โครงข่ายประสาทเทียมในการรู้จำ**

ในวิทยานิพนธ์นี้ได้ใช้โครงข่ายประสาทเทียมแบบแพร่กลับที่มี 3 ชั้น คือชั้นอินพุต ชั้นฮิดเดน และชั้นเอาต์พุต โดยที่มีหน่วยรับข้อมูลจำนวน 1024 หน่วยเพื่อรับข้อมูลภาพขนาด

32x32 จุดภาพที่นำมาตรวจสอบ ใช้หน่วยที่ซ่อนอยู่จำนวน 30 หน่วย และมีหน่วยผลลัพธ์ 8 หน่วย จึงสามารถจำกลุ่มใบหน้าได้สูงสุด 256 กลุ่ม แสดงรูปร่างดังภาพต่อไปนี้

ภาพที่ 4.3




โครงข่ายประสาทเทียมแบบแพร่กลับ

ในการฝึกให้โครงสร้างของโครงข่ายประสาทเทียมรู้จำกลุ่มใบหน้านั้น เราจะทำการส่งภาพใบหน้าคนๆ จำนวนหลายภาพที่เป็นสมาชิกอยู่ในกลุ่มเดียวกัน ซึ่งกำหนดมาจากเวกเตอร์ควอนไตซ์ ให้โครงข่ายประสาทเทียมรู้จักเป็นกลุ่มหมายเลขเดียวกัน ตัวอย่าง การกำหนดผลลัพธ์ แสดงได้ดังนี้

ภาพบุคคลในกลุ่มที่ 1 กำหนดให้มีหน่วยผลลัพธ์เป็น (0000 0000)
ภาพบุคคลในกลุ่มที่ 2 กำหนดให้มีหน่วยผลลัพธ์เป็น (1000 0000)

.
.
.

ภาพบุคคลในกลุ่มที่ 256 กำหนดให้มีหน่วยผลลัพธ์เป็น (1111 1111)

ในการฝึกให้โครงข่ายประสาทเทียมรู้จำจะเป็นการทำซ้ำ (Iterative) จน error ลดลงถึงจุดที่ได้ตั้งไว้ ก็จะได้ Weight File ออกมา

### 4.4 โครงข่ายประสาทเทียมในการตัดสินใจและการหา 5 อันดับความคล้าย

ในการทดสอบโครงข่ายประสาทเทียมให้รู้จำกลุ่มใบหน้า เรากำหนดให้เวกเตอร์ชื่อ out(1), out(2),….,out(8) เป็นตัวรับค่าจากหน่วยผลลัพธ์ที่ผ่านการตัดเทรสโฮลด์ สำหรับ i=1, 2,…, 8 ดังนี้

$$out[i] = \begin{cases} 1 & if \quad o_i \geq 0.7 \\ 0 & if \quad o_i \leq 0.3 \\ unknow & if \quad 0.3 < o_i < 0.7 \end{cases} \tag{4.1}$$

โดยถ้าได้ผลจากหน่วยผลลัพธ์ที่ผ่านการตัดเทรสโฮลด์เป็น (out(1),out(2),…,out(8)) = (0,0,0,0,0,0,0,0) เราจะได้ผลจากการรู้จำว่าคือบุคคลที่เป็นสมาชิกอยู่ในกลุ่มที่ 1 และถ้าได้ (out(1), out(2), …, out(8)) = (1,0,0,0,0,0,0,0) เราจะได้ผลจากการรู้จำว่าคือบุคคลที่เป็นสมาชิกอยู่ในกลุ่มที่ 2 แต่ถ้าผลลัพธ์จากหน่วยผลลัพธ์อยู่ในช่วง $0.3 < o_i < 0.7$ เพียงตัวใดตัวหนึ่งซึ่งเป็นช่วงที่ไม่รู้จักบุคคลคนนั้น

ตัวชี้กลุ่มจะเป็นตัวบอกกระบวนการในการเทียบภาพด้วยวิธีการสหสัมพันธ์ในกระบวนการถัดไปว่าจะต้องทำการตรวจสอบภาพที่เข้ามาเทียบกับสมาชิกคนใดบ้าง หลังการทำสหสัมพันธ์แล้วจะนำมาเรียง 5 อันดับที่มีความคล้ายมากที่สุดตามลำดับ

# บทที่ 5

## การทดลองและผลการทดลอง

ในการทดลองจะทำการทดสอบการค้นหาภาพใบหน้า (Face detection) ทดสอบฝึกสอนโครงข่ายประสาทเทียมในสามกรณีที่ต่างกันคือ

1) การระบุบุคคลจากข้อมูลภาพระดับสีเทาโดยตรง

2) การระบุบุคคลจากข้อมูลที่ถูกลดรายละเอียดด้วยการแปลงเวฟเลต

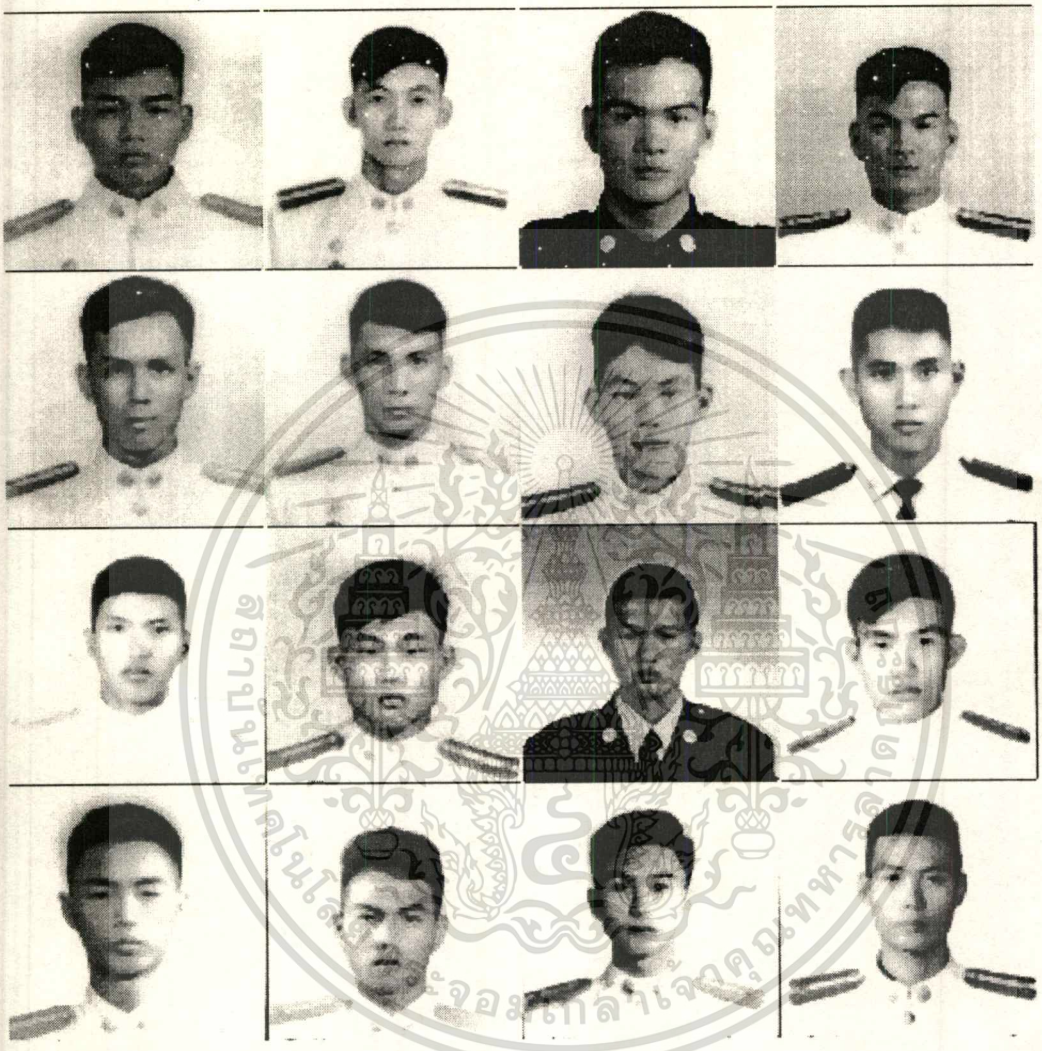
3) การรู้จำกลุ่มจากการรวมกลุ่มข้อมูลซ้ำซ้อนที่ถูกลดรายละเอียดด้วยการแปลงเวฟเลตให้เป็นกลุ่มเดียวกันด้วยเวกเตอร์ควอนไตซ์

และการทดสอบผลการหาลำดับความคล้ายเพื่อระบุบุคคลด้วยวิธีการสหสัมพันธ์จากผลการระบุกลุ่มของโครงข่ายประสาทเทียม

### 5.1 ผลการค้นหาภาพใบหน้า

จากการเก็บภาพด้วยเครื่องสแกนภาพ เก็บในรูปแบบภาพระดับสีเทา (Gray Scale) ขนาด 8 บิต หรือมีจำนวนสี 256 ระดับ โดยใช้ความละเอียด 100 จุดภาพต่อนิ้ว (dpi) ภาพถ่ายต้นแบบอาจเป็นภาพขนาด 1 นิ้วหรือ 2 นิ้ว ดังนั้นขนาดภาพที่ได้จากการสแกนอาจมีจำนวนจุดภาพไม่เท่ากัน จึงนำภาพดังกล่าวมาลดขนาดลงทุกภาพให้เหลือเป็นภาพขนาด 128x128 จุดภาพ และทำการกำจัดสัญญาณรบกวน โดยใช้ตัวกรองแบบเลือกค่ากลาง แสดงผลลัพธ์ได้ดังภาพที่ 5.1

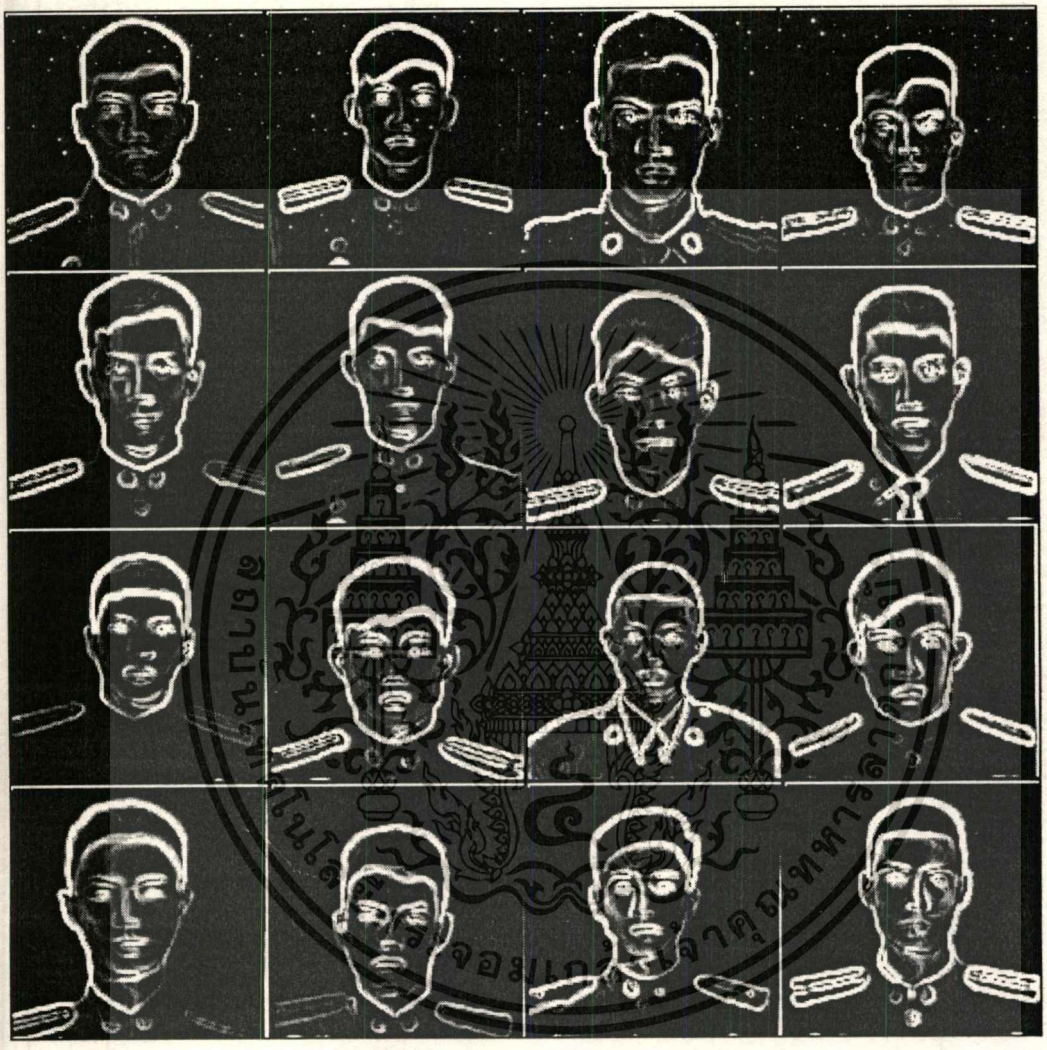
ภาพที่ 5.1



ภาพที่ย่อขนาดให้เล็กลงเหลือ 128x128 จุดภาพ

นำภาพมาทำการหาขอบภาพ แสดงผลลัพธ์ได้ดังภาพที่ 5.2

ภาพที่ 5.2



ภาพขอบจากการใช้วิธีโซเบล

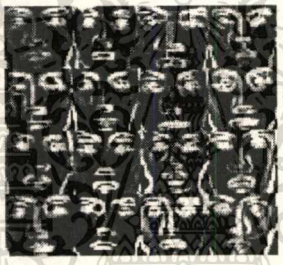
เมื่อนำภาพมาย่อเป็นสามระดับดังที่กล่าวมาแล้วดังในบทที่ 3 และทำการเทียบภาพกับภาพอ้างอิงดังในภาพที่ 5.3 ในการค้นหาบริเวณที่สนใจทั่วทุกบริเวณทั้งภาพด้วยวิธีการสหสัมพันธ์นั้นต้องใช้เวลาในมาก ดังนั้นจึงต้องเลือกกำหนดให้ค้นหาเฉพาะในบริเวณที่คาดว่าจะมีภาพดังกล่าวอยู่เท่านั้น ดังที่ได้กล่าวมาแล้วในบทที่ 3 ซึ่งยังผลให้สามารถลดระยะเวลาในการค้นหาได้อย่างมาก จากภาพทั้งหมดที่นำมาใช้นั้นเป็นภาพจากบัตรซึ่งพบภาพใบหน้าในบริเวณใกล้ตรงกลางภาพถ่ายเสมอ ดังนั้นในที่นี้จึงเริ่มทำการค้นหาในส่วนบนด้านซ้ายของ

ภาพถ่ายลงมาจนถึงตรงกลางภาพถ่ายเท่านั้น และจากการทดลองก็ประสบความสำเร็จในการค้นหาด้วยดี ดังตัวอย่างภาพผลลัพธ์ดังในภาพที่ 5.4 และ 5.5 สำหรับ ภาพระดับสีเทาที่ได้และเมื่อแสดงให้เห็นถึงส่วนของใบหน้าที่โดนตัดออกไปจากภาพต้นแบบ แสดงได้ดังภาพที่ 5.6

ภาพที่ 5.3

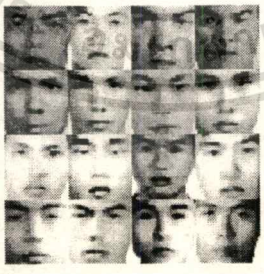
ภาพขอบของใบหน้าอ้างอิง

ภาพที่ 5.4



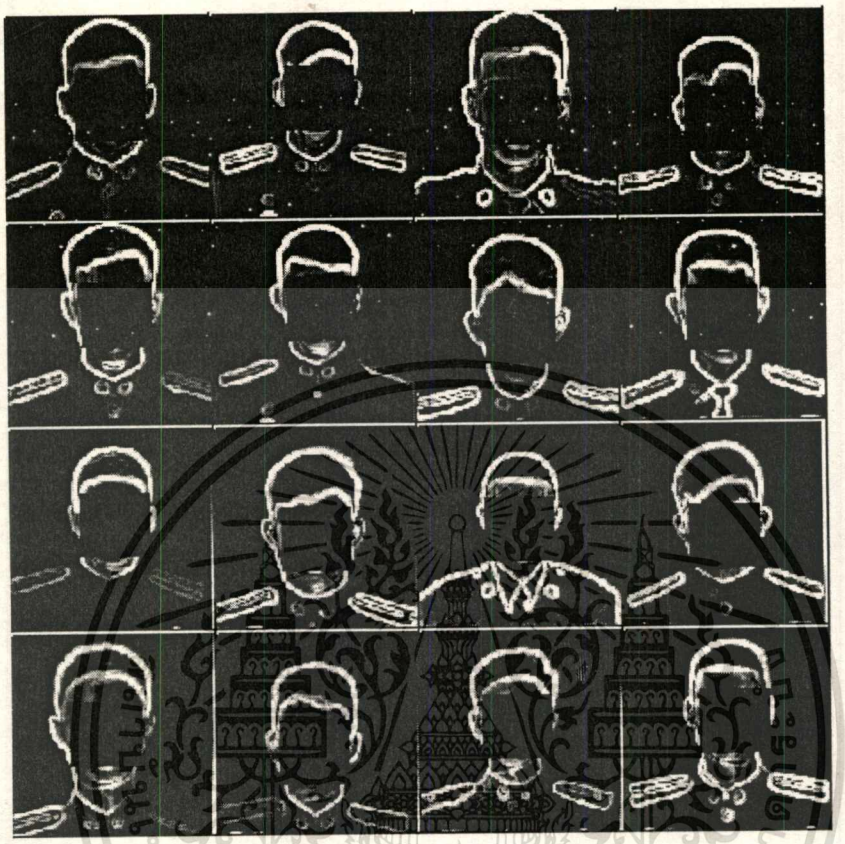
ภาพขอบของใบหน้าที่ค้นพบ

ภาพที่ 5.5



ภาพของใบหน้าที่ค้นพบ

ภาพที่ 5.6



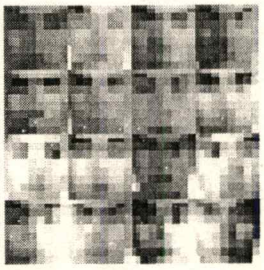
แสดงส่วนที่ภาพใบหน้าถูกตัดออกไป

ซึ่งจากการทดลองจับความเร็วในการประมวลผลเฉพาะส่วนของการค้นหาใบหน้าในภาพ (ไม่รวมการตรวจสอบว่าเป็นบุคคลใด) โดยเฉลี่ยจะมีความเร็วประมาณ 0.9 วินาทีต่อภาพ

**5.2 ผลการลดรายละเอียดของข้อมูลภาพด้วยการแปลงเวฟเลต**

จากข้อมูลภาพจากภาพ 5.5 ซึ่งได้จากการตรวจจับใบหน้า เมื่อนำมาผ่านการแปลงเวฟเลต และทำการกรองความถี่โดยเลือกให้เหลือเพียง 8x8 สัมประสิทธ์ทางด้านบนซ้ายเท่านั้น แล้วแปลงกลับเวฟเลตออกมา จะได้ภาพที่ถูกลดรายละเอียดความถี่สูงได้ดังภาพที่ 5.7 ซึ่งข้อมูลภาพบุคคลต่างๆ จะมีความซ้ำซ้อนหรือคล้ายกันขึ้นมา ซึ่งต้องนำไปผ่านการจัดกลุ่มด้วยเวกเตอร์ควอนไตซ์ต่อไป

ภาพที่ 5.7



การลดข้อมูลภาพโดยเลือกสัมประสิทธ์ในโดเมนของเวฟเลต ขนาด 8x8

**5.3 ผลการจัดกลุ่มข้อมูลด้วยเวกเตอร์ควอนไตซ์**

โดยการใช้เวกเตอร์ควอนไตซ์ จากการทดลองจะนำข้อมูลภาพจากภาพที่ 5.7 จำนวน 16 คนมาทำการจัดเป็น 4 กลุ่ม ในการสร้างตัวเก็บรหัส สามารถแสดงผลลัพธ์ (Codebook) ได้ดังภาพที่ 5.8 และสมาชิกในกลุ่มเวกเตอร์ต่างๆ ที่ได้จากการถอดรหัส แสดงได้ดังภาพที่ 5.9

ภาพที่ 5.8


แสดงการนำข้อมูล 16 คนที่ผ่านการแปลงเวฟเลตมาจัดกลุ่ม
ด้วย เวกเตอร์ควอนไตซ์เป็น 4 กลุ่ม

ภาพที่ 5.9

(a) สมาชิกในกลุ่มที่ 1

(b) สมาชิกในกลุ่มที่ 2

(c) สมาชิกในกลุ่มที่ 3

(d) สมาชิกในกลุ่มที่ 4

แสดงภาพสมาชิกในกลุ่มต่างๆ ทั้ง 4 กลุ่ม

**5.4 ผลการรู้จำกลุ่มภาพใบหน้าด้วยโครงข่ายประสาทเทียม**

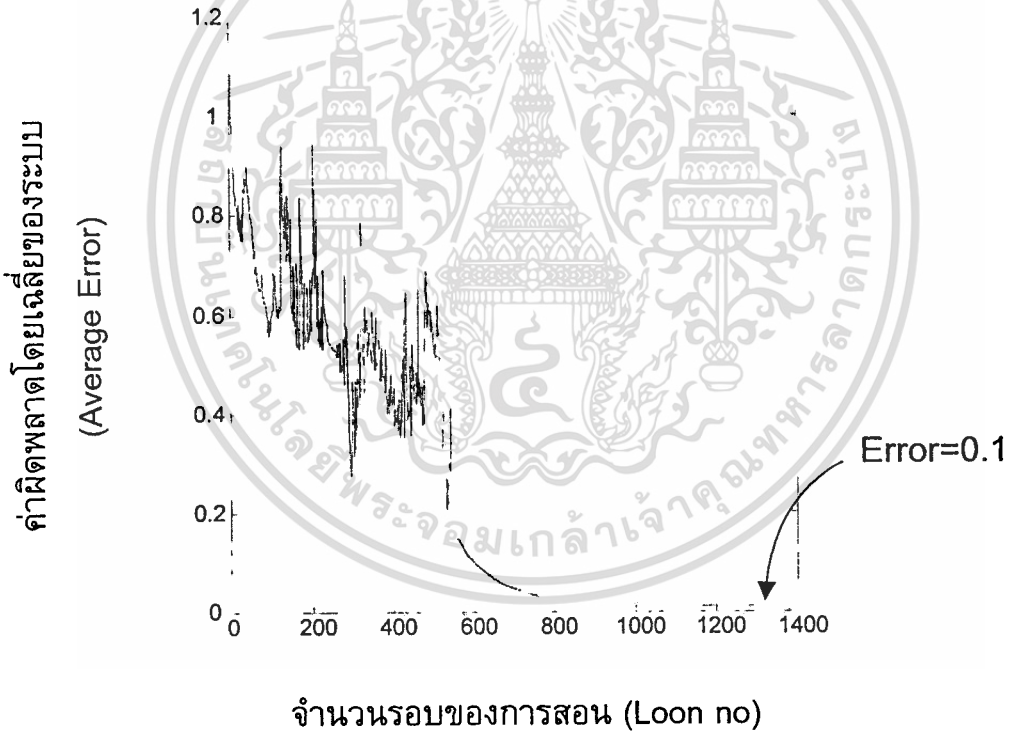
ในวิทยานิพนธ์นี้เพื่อพิสูจน์หลักการ คุณสมบัติในการเรียนรู้ข้อมูลแบบกลุ่มจึงได้ใช้โครงข่ายประสาทเทียมแบบแพร่กลับที่มีจำนวนชั้นและจำนวนโหนดต่างๆ คงที่ โดยใช้โครงข่าย 3 ชั้น คือชั้นอินพุต ชั้นฮิดเดน และชั้นเอาต์พุต โดยที่มีหน่วยรับข้อมูลจำนวน 1024 หน่วย เพื่อรับข้อมูลภาพขนาด 32x32 จุดภาพที่นำมาตรวจสอบ ใช้หน่วยที่ซ่อนอยู่จำนวน 30 หน่วย และมีหน่วยผลลัพธ์ 8 หน่วย ดังที่ได้กล่าวมาแล้วในบทที่ 4

### 5.4.1 การฝึกโครงข่ายประสาทเทียมในการระบุบุคคลจากข้อมูลภาพระดับสีเทาโดยตรง

จากการเรียนรู้จากภาพระดับสีเทาจากภาพที่ 5.5 ที่เป็นภาพต้นแบบโดยตรงโดยไม่ได้ผ่านการลดข้อมูลด้วยเวฟเลตและการจัดกลุ่ม นั้นสามารถแสดงกราฟการเรียนรู้ได้ดังภาพที่ 5.10

ในการฝึกให้เกิดการเรียนรู้นั้นจะทำจนกระทั่งกราฟตกลงจนถึงค่า Average Error ที่ยอมรับได้ ในที่นี้คือ 0.1 ก็จะสิ้นสุดกระบวนการฝึก

ภาพที่ 5.10



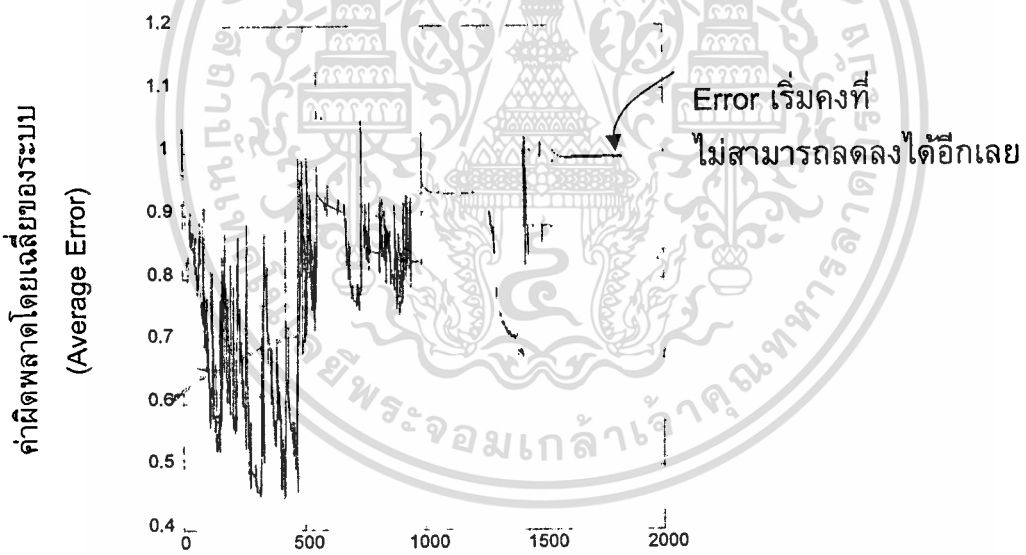

กราฟการเรียนรู้ที่ใช้ภาพระดับสีเทาจากภาพใบหน้าจากภาพที่ 5.5 โดยตรง

**5.4.2 การฝึกโครงข่ายประสาทเทียมในการระบุบุคคลจากข้อมูลที่ถูกลดรายละเอียดด้วยการแปลงเวฟเลต**

จากการเรียนรู้จากภาพระดับสีเทาที่ผ่านการลดข้อมูลด้วยเวฟเลตจากภาพที่ 5.7 โดยยังไม่ได้รับการลดข้อมูลที่ซ้ำซ้อนหรือยังไม่ผ่านการจัดกลุ่ม แสดงกราฟการเรียนรู้ได้ภาพที่ 5.11

ซึ่งจากกราฟจะแสดงให้เห็นอย่างชัดเจนว่าไม่สามารถเรียนรู้ได้ทั้งนี้เนื่องจากการลดข้อมูลทำให้ภาพมีการซ้ำของข้อมูลในแต่ละคน จนระบบโครงข่ายประสาทเทียมไม่สามารถแยกแยะระบุบุคคลได้ ในกราฟมีการขึ้นๆ ลงๆ อย่างมากและไม่มีการเปลี่ยนแปลงอีกเลยที่จุดซึ่งมีค่า Average Error ที่สูงมากด้วย

ภาพที่ 5.11

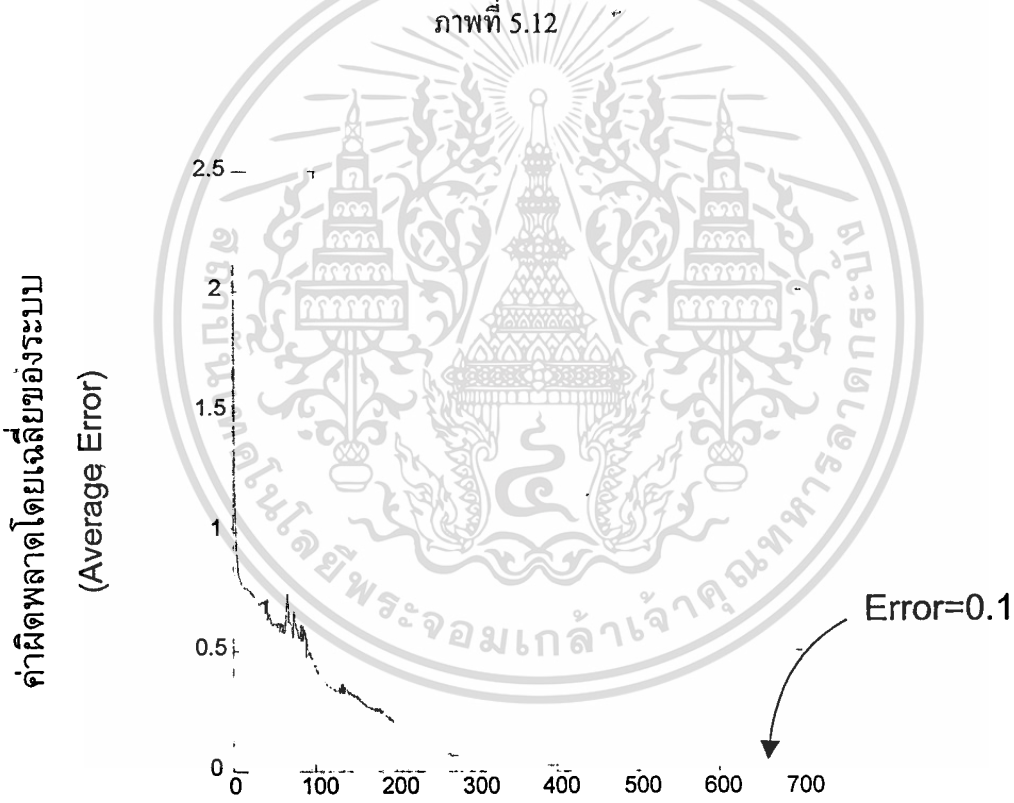



กราฟการเรียนรู้ที่ใช้ ข้อมูลภาพที่ผ่านการลดข้อมูล ด้วยการแปลงเวฟเลต

**5.4.3 การฝึกโครงข่ายประสาทเทียมในการรู้จำกลุ่มจากการรวมกลุ่มข้อมูลซ้ำซ้อนที่ถูกลดรายละเอียดด้วยการแปลงเวฟเลตให้เป็นกลุ่มเดียวกันด้วยเวกเตอร์ควอนไตซ์**

การนำภาพที่ผ่านการแปลงเวฟเลตมาผ่านการจัดกลุ่มด้วยเวกเตอร์ควอนไตซ์ เพื่อหาภาพที่มีความคล้ายคลึงกันให้มาอยู่ในประเภทเดียวกัน ดังภาพที่ 5.12 ซึ่งเป็นกราฟการเรียนรู้ที่แสดงให้เห็นว่า หลังจากการจัดกลุ่มแล้ว โครงข่ายประสาทเทียมสามารถเรียนรู้กลุ่ม ได้ เป็นอย่างดี โดยมีการลู่ค่าความผิดพลาดเข้าจุดต่ำสุดได้ดีและเร็วที่สุด ซึ่งนี่คือผลจากวิธีการที่นำเสนอในวิทยานิพนธ์นี้

ภาพที่ 5.12




**กราฟการเรียนรู้ที่ใช้ข้อมูลภาพที่ผ่านการลดข้อมูลด้วยการแปลงเวฟเลต และถูกจัดกลุ่มด้วยเวกเตอร์ควอนไตซ์**

**5.5 ผลการหาลำดับความคล้ายเพื่อระบุบุคคลจากผลการระบุกลุ่มของโครงข่ายประสาทเทียม**

หลังจากกระบวนการตัดสินใจของโครงข่ายประสาทเทียมที่สามารถบอกได้ว่าภาพใบหน้าดังกล่าวนั้นอยู่ในกลุ่มใดแล้ว จะใช้วิธีการสหสัมพันธ์เปรียบเทียบภาพเฉพาะในกลุ่มนั้นเป็นกระบวนการถัดไปเพื่อหาลำดับความคล้ายที่จะบอกถึงภาพใบหน้าที่คล้ายที่สุดตามลำดับ โดยสามารถแสดงตัวอย่างหนึ่งของการทดสอบตามลำดับได้ดังนี้

มีภาพใบหน้าอยู่ 2 ภาพคือภาพที่ 5.13(a) และ (b) เมื่อใช้โครงข่ายประสาทเทียมตัดสินใจเพื่อหากลุ่มพบว่าภาพที่ 5.13(a) อยู่ในกลุ่มที่ 1 และภาพที่ 5.13(b) อยู่ในกลุ่มที่ 4 ดังแสดงภาพทั้งหมดในกลุ่มทั้งสองดังภาพที่ 5.14

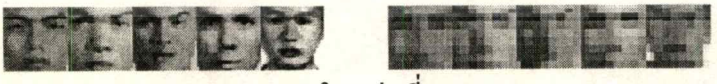
ภาพที่ 5.13

(a) ภาพบุคคลที่ 1

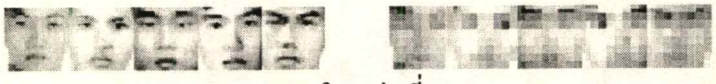
(b) ภาพบุคคลที่ 2

ภาพที่นำมาทดสอบ

ภาพที่ 5.14



(a) ภาพในกลุ่มที่ 1



(b) ภาพในกลุ่มที่ 4

แสดงภาพต้นแบบและภาพการลดรายละเอียดข้อมูลด้วยการแปลงเวฟเลต
ของกลุ่มที่ตรวจพบภาพบุคคลที่ 1 และ 2 ของตัวอย่าง

จากนั้นนำมาหาลำดับความคล้ายด้วยวิธีการสหสัมพันธ์ โดยทดสอบภาพบุคคลที่ 1 จากภาพ 5.13(a) ที่อยู่ในกลุ่มที่ 1 ดังภาพ 5.14(a) ได้ผลดังภาพที่ 5.15

ภาพที่ 5.15

0.99 0.869 0.860 0.754 <0.1

แสดงค่าลำดับความคล้ายภาพบุคคลที่ 1 เมื่อเทียบกับภาพในกลุ่มที่ 1

และทดสอบภาพบุคคลที่ 2 จากภาพที่ 5.13(b)ที่อยู่ในกลุ่มที่ 4 จากภาพที่ 5.14(b) ได้ผลดังภาพที่ 5.16

ภาพที่ 5.16

0.98 0.602 0.366 0.165 <0.1

แสดงค่าลำดับความคล้ายภาพบุคคลที่ 2 เมื่อเทียบกับภาพในกลุ่มที่ 4

จากการทดลองนี้แสดงให้เห็นว่าภาพที่คล้ายที่สุดที่ได้คือภาพบุคคลคนเดียวกับภาพที่เข้ามาทดสอบ ซึ่งถือว่านี่คือกระบวนการสุดท้ายที่หลังจากเราสามารถรู้กลุ่มด้วยโครงข่ายประสาทเทียมได้แล้ว เมื่อทำการหาลำดับความคล้ายแล้วเราก็สามารถรู้ตัวบุคคลได้ในที่สุด ซึ่งวิธีการนี้จะมีความเร็วกว่าวิธีการที่ต้องทำการเทียบภาพเพื่อหาความคล้ายโดยตรงทุกๆ ภาพในฐานข้อมูลภาพขนาดใหญ่

**5.6 สรุปผลการทดลอง**

ในการทดลองได้ป้อนภาพที่ได้เคยผ่านการเรียนรู้มาแล้วพบว่าสามารถระบุกลุ่มได้อย่างถูกต้อง และเมื่อมีการเปลี่ยนแปลงเกิดขึ้น เช่น ทดลองหมุนภาพ 1 ,2 และ 4 องศาพบว่าก็สามารถบอกกลุ่มได้อย่างถูกต้อง 100 % เช่นกัน อีกทั้งนอกจากการทดลองที่กล่าวมาแล้วนั้น เมื่อป้อนภาพที่ไม่เคยได้รับการเรียนรู้มาก่อนก็สามารถระบุกลุ่มภาพใบหน้าที่คล้ายที่สุดได้เป็นอย่างดี ซึ่งแสดงให้เห็นถึงความเข้ากันได้ของการลดข้อมูลด้วยการแปลงเวฟเลต การจัดกลุ่มด้วยเวกเตอร์ควอนไตซ์ และการรู้จำกลุ่มใบหน้า ด้วยโครงข่ายประสาทเทียม

# บทที่ 6

# สรุปผลการวิจัยและแนวทางในการพัฒนา

## 6.1 สรุปผลการวิจัย

ในการใช้โครงข่ายประสาทเทียมในการรู้จำกลุ่ม ที่มีการจัดกลุ่มข้อมูลโดยใช้เวกเตอร์ควอนไตซ์ก่อนการเรียนรู้เพื่อลดความซ้ำซ้อนของข้อมูลที่ผ่านการแปลงเวฟเลตมานั้น ทำให้**สามารถรู้จำกลุ่มได้เท่ากับ 100%** และอีกทั้งความไวต่อการเปลี่ยนแปลงของข้อมูลจากการหมุนก็ลดลง ทำให้สามารถรู้จำข้อมูลภาพบุคคลแบบจำเป็นกลุ่มได้อย่างมีประสิทธิภาพ และมีผลพลอยได้คือถ้าภาพที่เข้ามาเป็นภาพที่โครงข่ายประสาทเทียมยังไม่เคยเรียนรู้มาก่อน ก็สามารถระบุกลุ่มที่ภาพนั้นมีความคล้ายได้ด้วย ซึ่งอันนี้เป็นข้อดีอย่างมาก

ปกติในการเรียนรู้ของ**โครงข่ายประสาทเทียมในการระบุบุคคล**นั้นต้องใช้ภาพหลายภาพที่เป็นภาพบุคคลเดียวกันแต่มีความแตกต่างกัน ทางด้าน การเลื่อน การหมุน เป็นต้น ป้อนให้รู้จำ ซึ่งถ้าภาพที่ป้อนนั้น มีน้อยและไม่ครอบคลุมกรณีของการแปรเปลี่ยนจะทำให้การตัดสินใจของโครงข่ายประสาทเทียมบอกออกมาว่า ไม่รู้จักบุคคลนั้นได้บ่อยๆ แต่ถ้าใช้**ระบบโครงข่ายประสาทเทียมแบบการรู้จำกลุ่ม**แล้วสามารถแก้ปัญหาดังกล่าวได้ ทั้งนี้เนื่องจากมีการป้อนภาพบุคคลต่างๆ ที่อยู่ในกลุ่มเดียวกันเข้าไปให้รู้จำเป็นเอาต์พุตเดียวกันนั้น ทำให้เหมือนได้ข้อมูลอินพุตที่มีการแปรเปลี่ยนซึ่งช่วยครอบคลุมกรณีการแปรเปลี่ยนได้โดยที่เราไม่ได้ตั้งใจ ดังนั้นนี่คือเหตุผลว่าทำไมระบบที่นำเสนอสามารถ รับการแปรเปลี่ยนทางการเลื่อน และการหมุนภาพได้ และสามารถรู้จำกลุ่มความคล้ายของภาพที่ยังไม่เคยได้รับการฝึกมาเลยได้อีกด้วย

## 6.2 ปัญหาที่เกิดขึ้น

6.2.1 ในการค้นหาส่วนของใบหน้าในภาพนั้นพบความผิดพลาดในภาพบางภาพอยู่ ตัวอย่างดังภาพที่ 6.1 ทั้งนี้สาเหตุหนึ่งเนื่องมาจากขนาดของภาพอ้างอิง (Reference Image) ที่ใช้ในการเทียบภาพเพื่อค้นหาส่วนของใบหน้านั้นแตกต่างกับขนาดของภาพที่เข้ามาทดสอบมากเกินไป ทำให้ไปเทียบกับส่วนอื่นของใบหน้าแล้วมีความคล้ายมากกว่านั่นเอง ดังนั้นจึงยังมีความยากลำบากที่จะเลือกภาพใบหน้าอ้างอิงและเปอร์เซนต์การย่อขยายที่เหมาะสมในการ

ค้นหาเพื่อให้ได้ครอบคลุมทุกภาพ ดังนั้นในการค้นหาส่วนของใบหน้านี้ก็ยังคงต้องพัฒนาให้ดีต่อไปอีกเพื่อให้ค้นหาได้กับภาพทุกภาพทุกขนาดอย่างแท้จริง

ภาพที่ 6.1

ตัวอย่างภาพที่ค้นหาส่วนของใบหน้าผิดพลาด

6.2.2 ในการหาภาพมาทดสอบยังคงใช้ภาพจำนวนน้อยอยู่ ซึ่งเพียงเพื่อทดสอบแนวความคิดเบื้องต้นของตนเองเท่านั้น และถึงแม้ภาพบุคคลคนเดียวกันจะใช้ภาพเดียวกันในการตรวจสอบระบุบุคคลก็ตาม แต่ภาพเมื่อเก็บเข้ามาคนละครั้งกันหรือจากเครื่องสแกนเก็บภาพคนละเครื่องกันก็อาจจะมีการเปลี่ยนแปลงไปมากก็เป็นไปได้ ซึ่งจะต้องปรับปรุงโปรแกรมให้สามารถทดสอบหลายๆ ลักษณะของภาพที่เปลี่ยนแปลงไป และมีทดสอบกับภาพจำนวนมากได้ด้วย

6.2.3 หากภาพที่เข้ามามีการเปลี่ยนแปลงมากๆ ต้องมีกระบวนการในการปรับภาพก่อนการประมวลผล (Preprocessing) ที่ดีด้วย เช่น ปรับแสง การเอียงหรือวางตัวของใบหน้าและขนาดของใบหน้าเป็นต้น

6.2.4 การเรียนรู้ของโครงข่ายประสาทเทียมในการรู้จำกลุ่มนั้น ภาพบางภาพที่ไม่เคยรู้จักหรือมีความแตกต่างจากกลุ่มภาพที่ได้เคยเรียนรู้มาแล้วอย่างมาก ยังคงเป็นภาพที่โครงข่ายประสาทเทียมไม่รู้จักอยู่ดี ดังนั้นภาพที่เลือกเข้ามาจัดกลุ่ม และให้โครงข่ายประสาทเทียมเรียนรู้ก็ยังคงต้องใช้ภาพที่จำนวนมากและครอบคลุมพอสมควรมาฝึกสอนให้รู้จำ

6.2.5 เพื่อทดสอบแนวความคิด โปรแกรมภาษาซีที่ใช้เขียนขึ้นมานี้ทำงานแต่ละส่วนแยกกัน และนำมาทำงานเรียงลำดับกันไปเป็นลักษณะของแบตช์โพรเซส (Batch Process) เช่น การกำจัดสัญญาณรบกวน การหาขอบภาพ การหาส่วนของใบหน้า การแปลงเวฟเลต เวกเตอร์ควอนไตซ์ โครงข่ายประสาทเทียม และ การทำสหสัมพันธ์ 5 อันดับความคล้าย ซึ่งแต่ละโปรแกรมภาษาซีแต่ละตัวนั้นจะแยกอิสระจากกัน ดังนั้นในการผ่านจากโปรแกรมหนึ่งไปอีกโปรแกรมหนึ่งจะต้องทำการบันทึกภาพผลลัพธ์แต่ละกระบวนการทุกขั้นตอน ทำให้ค่อนข้างสิ้นเพียงเนื้อที่ในการเก็บภาพผลลัพธ์ต่างๆ และใช้เวลามากกว่าความเป็นจริงในการประมวลผล แต่ถึงแม้วิธีการทดลองแบบนี้ทำให้เกิดปัญหาหลายอย่างขึ้นก็ตาม แต่มันก็ยังมีข้อ

ดี คือ มี ความยืดหยุ่นสูง สามารถเปลี่ยนแปลงลำดับการประมวลผลได้ง่าย และปรับปรุงกระบวนการในแต่ละขั้นให้ดีขึ้นได้อย่างอิสระมากทีเดียว

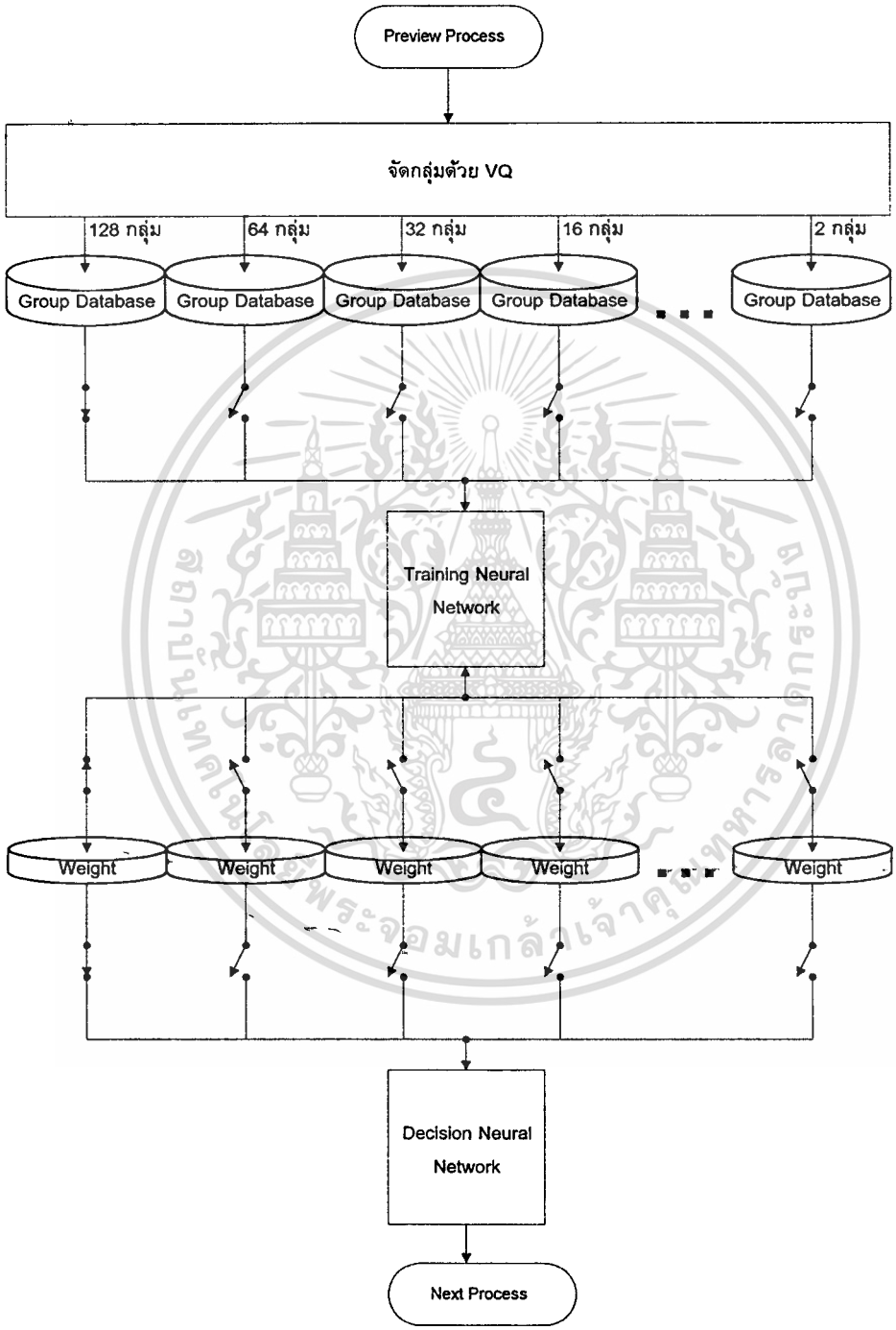
6.2.6 เนื่องจากภาพที่นำมาใช้ตรวจสอบเพื่อระบุกลุ่มหรือบุคคลในที่นี้เป็นภาพใบเดียวกันที่ผ่านการสแกนเก็บภาพคนละครั้ง ดังนั้นในการนำไปประยุกต์ใช้งานจึงมีอยู่ในวงจำกัดเท่านั้น เพื่อให้มีการใช้งานหรือไปประยุกต์ได้มากกว่านี้ จะต้องปรับปรุงทำให้สามารถใช้ได้กับภาพจากกล้องวิดีโอหรือภาพบุคคลคนเดียวกันที่เป็นคนละภาพได้ด้วย

**6.3 แนวทางในการพัฒนาต่อไป**

เนื่องจากระบบที่นำเสนอนี้สามารถระบุกลุ่มซึ่งจำนวนของกลุ่มนั้นขึ้นกับการออกแบบว่าจะแบ่งออกเป็นจำนวนเท่าใด ในที่นี้ต้องเลือกให้มีจำนวนกลุ่มมากที่สุด แต่จะต้องน้อยกว่าจำนวนคนสมาชิกทั้งหมดของระบบ เช่น ถ้า ระบบของเราประกอบด้วยคน 1000 คน เราจะแบ่งกลุ่มออกเป็น 128 กลุ่มเพื่อให้แต่ละกลุ่มมีสมาชิกประมาณ 7 คน และในการค้นหาก็จะได้รายชื่อคนที่คล้าย 5 อันดับออกมา ซึ่งนี่คือผลจากการจัดกลุ่มออกเป็น 128 กลุ่ม แต่ถ้าเกิดความผิดพลาดจากการค้นหา โครงข่ายประสาทเทียมตัดสินใจผิด และต้องการค้นหาชั้นที่มีการแบ่งกลุ่มออกเป็น 64 กลุ่ม เราต้องทำกระบวนการเวกเตอร์ควอนไตซ์ใหม่อีกครั้งหนึ่ง และทำการฝึกให้โครงข่ายประสาทเทียมอีกครั้ง ซึ่งจะเห็นว่าใช้เวลามากทีเดียว

ดังนั้น ระบบควรได้รับการออกแบบเป็นชั้นชั้น ซึ่งควรประกอบด้วย เวกเตอร์ควอนไตซ์ที่มีการแบ่งกลุ่มไว้หลายระดับ เช่นมีการแบ่ง 2 กลุ่ม, 4 กลุ่ม, 8 กลุ่ม, 16 กลุ่ม, 32 กลุ่ม, 64 กลุ่ม, 128 กลุ่ม, 256 กลุ่ม,...เป็นต้น และโครงข่ายประสาทเทียมที่มีการเรียนรู้การแบ่งกลุ่มดังกล่าวทุกๆ ระดับโดยเก็บไว้เป็นไฟล์น้ำหนักหลายตัว ซึ่งจะมีสวิทช์คอยเลือกไฟล์น้ำหนักมา ใช้ งานได้ตามต้องการ ถ้าการค้นหาในระดับสูงไม่พบ ก็ยังมีความหวังว่าจะพบใบหน้าที่คล้ายในระดับที่ต่ำๆ ลงไป ได้ตามลำดับ ซึ่งระบบแบบนี้จะช่วยลดเวลาในการค้นหาในฐานข้อมูลภาพขนาดใหญ่ได้เป็นอย่างมาก ระบบแสดงดังในภาพที่ 6.1

แผนภูมิที่ 6.1




**ระบบรู้จำกลุ่มด้วยโครงข่ายประสาทเทียมแบบจัดกลุ่มหลายชั้นความคล้าย**

# บรรณานุกรม

[1] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," IEEE Trans. Pattern Anal. Machine Intell., vol. 15, pp 1042-1052, Oct.1993.

[2] T. Kanade, "Picture processing by computer complex and recognition of human faces,"Ph.D. dissertation, Kyoto Univ., Japan,1973.

[3] I.J. Cox, J. Ghosn, and P.N. Yianilos, "Feature-based face recognition using mixture-distance," in Vision and Pattern Recognition. Piscataway, NJ:IEEE Press, 1996.

[4] H.U. Bauer and K.R. Pawelzik, "Quantifying the neighborhood preservation of self-organizing feature maps, "IEEE Trans. Neural Networks, vol. 3, pp. 570-579, 1992.

[5] M. Turk and A. Pentland, "Eigenfaces for recognition," J. Cognitive Neurosci., vol. 3, pp. 71-86, 1991.

[6] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," in Proc. IEEE Conf. Comput. Vision Pattern Recognition 1994.

[7] A. Pemtland, T. Starner, N. Etcoff, A. Masoiu, O. Oliyide, and M. Turk, "Experiments with eigenfaces, "in Proc. Looking at people Wkshp, Int. Joint Conf. Artificial Intell., Chamberry, France, 1993.

[8] B. Moghaddam and A. Pentland, "Face recognition using view-based and modular eigenspaces," in Automat. Syst. Identification Inspection of Humans, vol. 2257, 1994.

[9] D.L. Swets and J.J. Weng, "Using discriminant eigenfeatures for image retrieval," IEEE Trans. Pattern Anal. Machine Intell., to appear, 1996.

[10]D. DeMers and G. W. Cottrell, "Nonlinear dimensionality reduction, " in Advances in Neural Information Processing System 5, S.J. Hanson, J.D. Cowan, and C. Lee Giles, Eds. San Mateo, CA:Morgan Kaufmann, 1993,pp. 580-587.

[11]S. Lawrence, C.L. Giles, A.C. Tsoi,and A.D. Back, "Face recognition: A convolutional neural-network approach, " IEEE Trans. Neural Networks, vol. 8, no.1, January 1997.

[12]J.J. Atick, P.A. Griffin and A.N. Redlich,"Statistical Approach to shape from shading reconstruction of 3d face surfaces from single 2d images," Computational Neuroscience Laboratory, The Rockefeller University 1230 York Avenue New York,NY 10021-6399.

[13]P.S. Penev and J.J. Atick,"Local Feature Analysis: A general statistical theory for object representation," Computational Neuroscience Laboratory, The Rockefeller University 1230 York Avenue New York,NY 10021-6399.

[14]R.C. Gonzalez and R.E. Woods, Digital Image Processing, Addison-Wesley Publish Company, Inc.,1992.

[15]"Manual of Remote Sensing", Vol. 1, The American Society of Photogrammetry, Falls Church, va., 1975.

[16]P.M. Bently and J.T.E. McDonnell,"Wavelet Transforms: and Introduction," Electronics & Communication Engineering Journal, August 1994.

[17]R.K. Young, Wavelet Thory and Its Application, Kluwer Academic Publishers, 1993.

[18]C.K. Chui, Wavelet: A tutorial in Theory and Applications, Academic Press,1992.

[19]N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A Review,"IEEE Trans. Commun., vol. COM-36, pp. 957-971, Aug 1980.

[20]Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," IEEE Trans. Commun., vol COM-28, pp. 84-95, Jan 1980.

[21]A.Gersho and R. M. Gray, "Vector Quantization and Signal Compression," Kluwer Academic Publishers, Boston, London, 1991.

[22]L.Fausett , Fundamentals of Neural Networks, Prentice-Hall International, Inc.,1994.

[23]R.J. Schalkoff, Pattern Recognition: Statistical, Structural and Neural Approaches, John Wiley & Sons, Inc., 1992.

[24]สาธิต อินทจักร์ และ รศ.ดร. กิตติ ไพฑูรย์วัฒนกิจ Soft Computong : นิวรอลเน็ตเวิร์ก สาร NECTEC ศูนย์เทคโนโลยีอิเล็กทรอนิกส์ และคอมพิวเตอร์แห่งชาติ ปีที่ 4 ฉบับที่ 16 พฤษภาคม-มิถุนายน 2540 หน้า 26-42

[25]D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, Parallel Distributed Processing: Exploration in the Microstrucure of Cognition, Vol. 1: Foundation, MIT Press, Massachusette, USA., 1986.

# ภาคผนวก

# ภาคผนวก ก

## ผลงานวิจัยที่ได้รับการตีพิมพ์

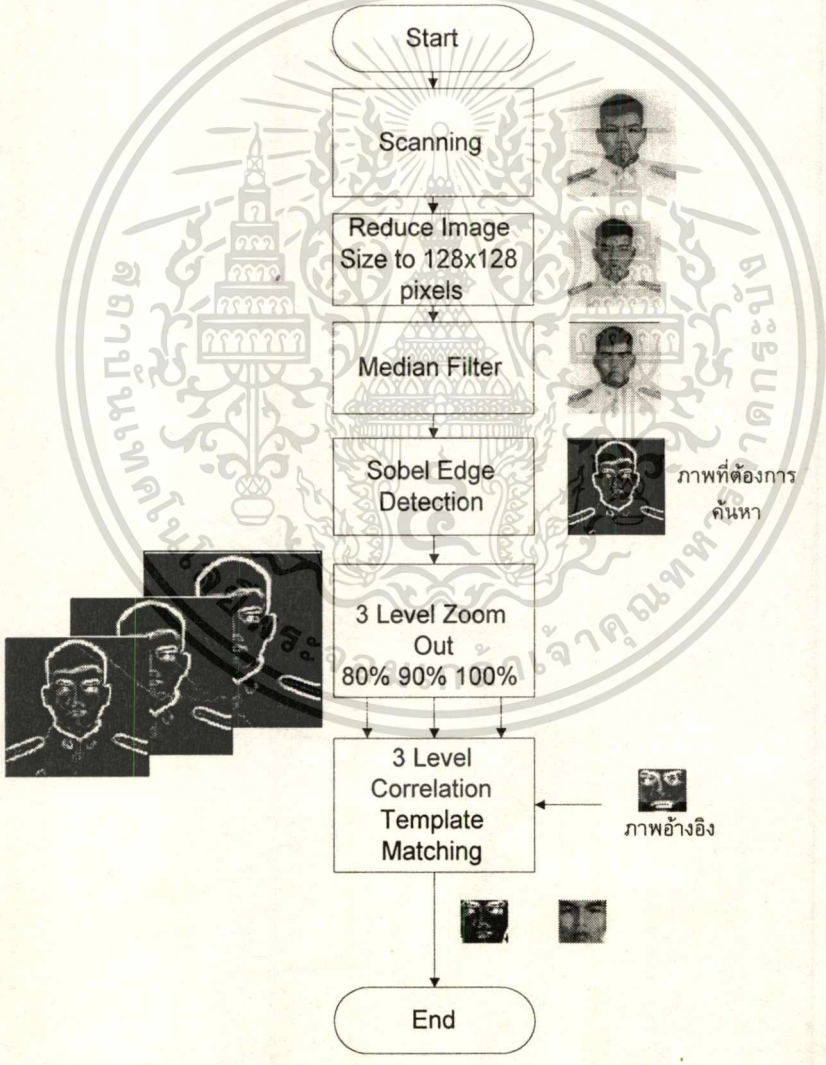
[1] วาภิรมย์ มนัสรังษี ครรชิต ไมตรี และ วิเชียร ศรีเสือขาม, "AN ARCHITECTURE FOR FACE RECOGNITION SYSTEM," 19th Congress on Science and Technology of Thailand, Hat Yai, Songkhla, 27-29 October 1993.

# ภาคผนวก ข

## การค้นหาภาพใบหน้า

## (Face Detection)

กระบวนทั้งหมดในการค้นหาใบหน้าซึ่งวิทยานิพนธ์นี้นำเสนอนั้น สามารถแสดงดังรูปต่อไปนี้




**การค้นหาภาพใบหน้าที่ใช้การหาสหสัมพันธ์ใบหน้าอ้างอิงกับภาพย่อ 3 ระดับเพื่อให้ทำงานได้กรณีภาพที่เข้ามาตรวจสอบมีการเปลี่ยนแปลงขนาด**

ซึ่งในการค้นหาภาพใบหน้าในภาพจะเป็นไปโดยอัตโนมัติ ที่ประกอบด้วย 3 ขั้นตอน ดังนี้

**1. การเตรียมภาพใบหน้าอ้างอิง (Preparing the reference face)**

การเตรียมใบหน้าอ้างอิงนั้น สามารถทำได้โดยการนำภาพมาหาขอบด้วยวิธีการของโซเบลที่คิดว่ามีตา จมูก และปาก ที่สมบูรณ์แบบและภาพใบหน้ามีขนาดเล็กที่สุดในบรรดาภาพทั้งหลาย ทั้งนี้เนื่องจากวิธีการที่นำเสนอภาพทุกภาพที่เข้ามาตรวจสอบจะถูกย่อลงเข้าหาภาพอ้างอิงนั่นเอง

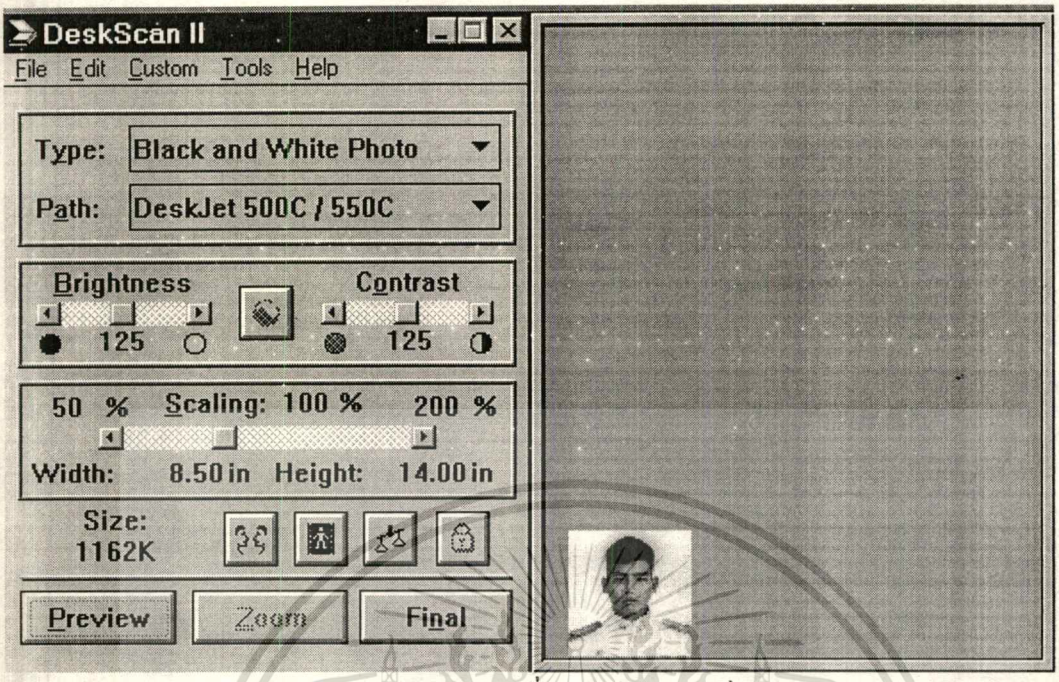
(a) ภาพต้นแบบ

(b) ภาพส่วนใบหน้าที่เลือกตัดมาจากภาพ (a) แบบแมนนวล

ภาพต้นแบบที่เลือกมาและส่วนที่ตัดมาใช้อ้างอิงเพื่อการค้นหา (ผ่านการลบขอบแก้มทิ้งไป)

**2. การเก็บภาพที่เข้ามาตรวจสอบ ด้วยโปรแกรม DeskScan II**

เนื่องจากในการควบคุมเครื่องสแกนภาพบนระบบปฏิบัติการดอสเป็นไปด้วยความยากลำบากกว่าบนวินโดว์ และโปรแกรมควบคุมบนวินโดว์ที่ให้มากับเครื่องสแกนภาพของ HP scanner นั้นก็ทำงานได้ดีเยี่ยมอยู่แล้ว ดังนั้นจึงได้ใช้โปรแกรมสำเร็จคือ DeskScan II มาเป็นตัวช่วยในการเก็บภาพที่จะทำการบันทึกภาพที่ได้นั้นลงไปบนไฟล์ชั่วคราวในชื่อที่ได้นัดหมายไว้กับโปรแกรมของเรา เช่น (t.pcx) แล้วทำการเปิดภาพดังกล่าวไปใช้งานต่อไป โปรแกรม DeskScan II มีหน้าตา ดังรูปต่อไปนี้




โปรแกรม DeskScan II ที่นำมาใช้ในการเก็บภาพ

ภาพที่ได้จากการสแกนเก็บภาพในที่นี้ใช้ความละเอียด 100 dpi ที่ให้ความละเอียดระดับปานกลาง จะถูกกำหนดพื้นที่ภาพด้วยผู้ใช้เอง โดยสามารถเลือกให้กว้างเพียงพอที่ครอบคลุมทุกส่วนที่สำคัญของบุคคลนั้นได้ตามต้องการ ดังแสดงรูปตัวอย่างหนึ่งดังรูปข้างล่างนี้ ซึ่งเราจะบันทึกลงไปบนชื่อที่ได้นัดหมายกันไว้แล้ว เพื่อจะได้สามารถนำไปใช้งานได้ต่อไป บนระบบปฏิบัติการดอส

ภาพใบหน้าที่ได้จากเครื่องสแกนเก็บภาพ ที่บันทึกลงไปในดิสก์

**3. นำภาพดังกล่าวไปผ่านโปรแกรม Face Detection** เพื่อหาส่วนที่เป็นภาพใบหน้าเท่านั้น ดังวิธีการที่ได้นำเสนอ โดยผ่านกระบวนการตามลำดับ คือ

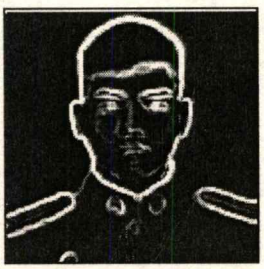
- ลดขนาดให้เหลือ 128x128 เพื่อปรับภาพทุกภาพให้มีขนาดเท่ากันทั้งหมด

ภาพที่ผ่านการปรับขนาดเป็น 128x128

- ผ่านการกรองสัญญาณรบกวนด้วย Median Filter

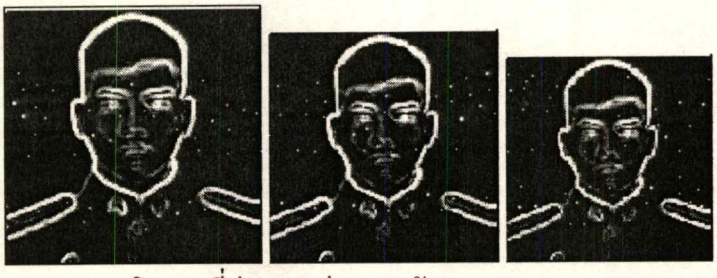
ภาพที่ผ่านการลดสัญญาณรบกวนแล้ว

- ผ่านการหาขอบภาพ ด้วย Sobel Edge Detection ซึ่งเหตุผลเดียวที่ใช้ภาพขอบทั้งนี้ก็เพราะว่าขอบของภาพแสดงลักษณะเด่นที่ต้องการออกมาอย่างชัดเจน คือ มีส่วนของ ตา จมูก และ ปาก ซึ่งจะง่ายและเที่ยงตรงกว่าการค้นหาที่ใช้ภาพระดับสีเทาบริเวณใบหน้าธรรมดา และปกติการทำสหสัมพันธ์นั้นมักนิยมใช้ในการหาบริเวณที่เด่นชัดมากๆ เช่น การตัดกันของถนนในภาพถ่ายดาวเทียม สำหรับในการประมวลผลภาพระยะไกล (Remote Sensing) นั่นเอง และจากการทดลองจากข้อมูลจำนวนมาก ก็พิสูจน์ได้อย่างเด่นชัดว่า การนำขอบของภาพมาใช้ในการค้นหาภาพใบหน้านั้น สามารถได้ภาพออกมาถูกต้องกว่ามากจริงๆ



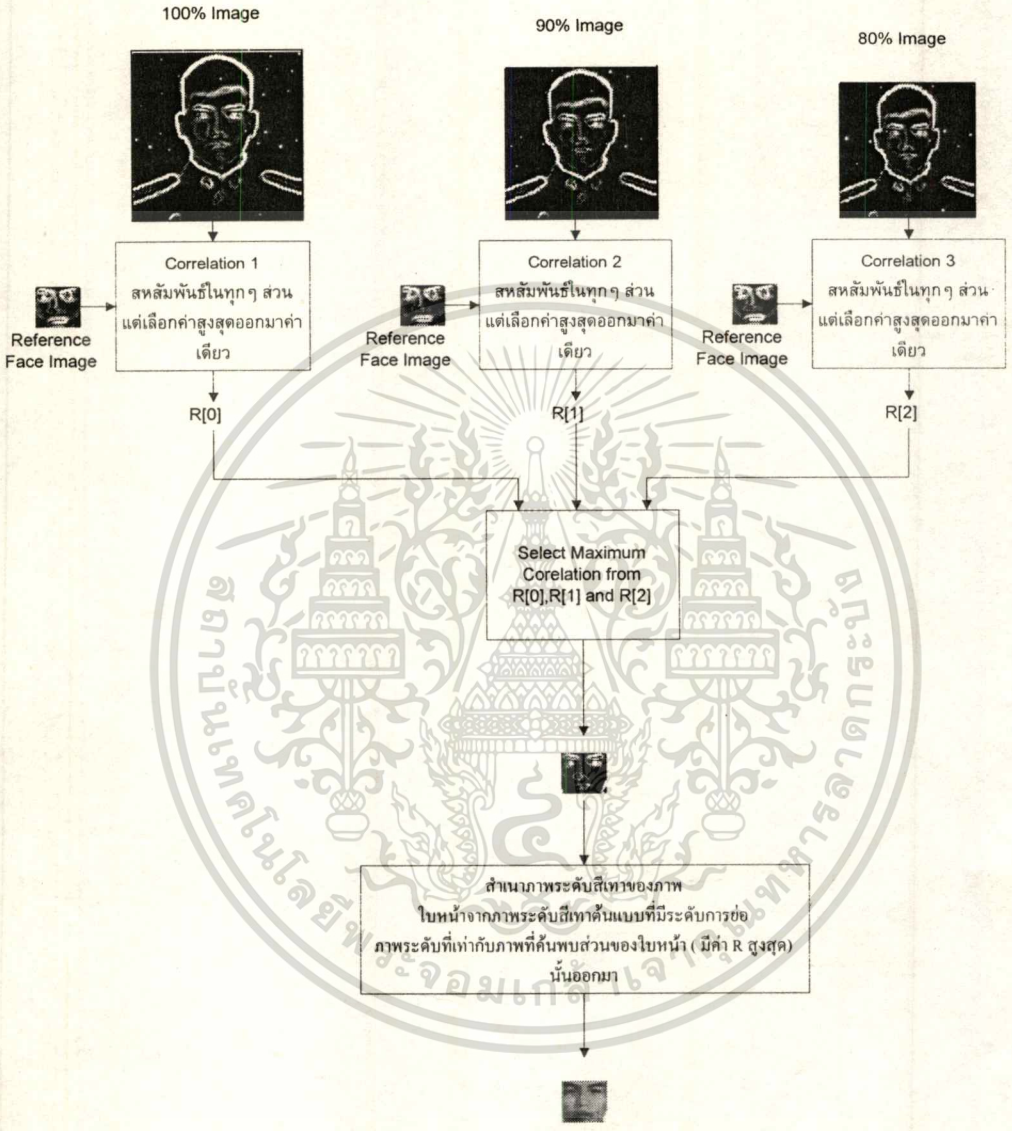
ภาพขอบโซเบล

- นำภาพที่ต้องการค้นหาส่วนของใบหน้ามาทำการย่อเป็น 3 ระดับ คือ ปกติที่เป็น 100% ลดลงเหลือ 90% และ 80% ตามลำดับ ดังรูปต่อไปนี้



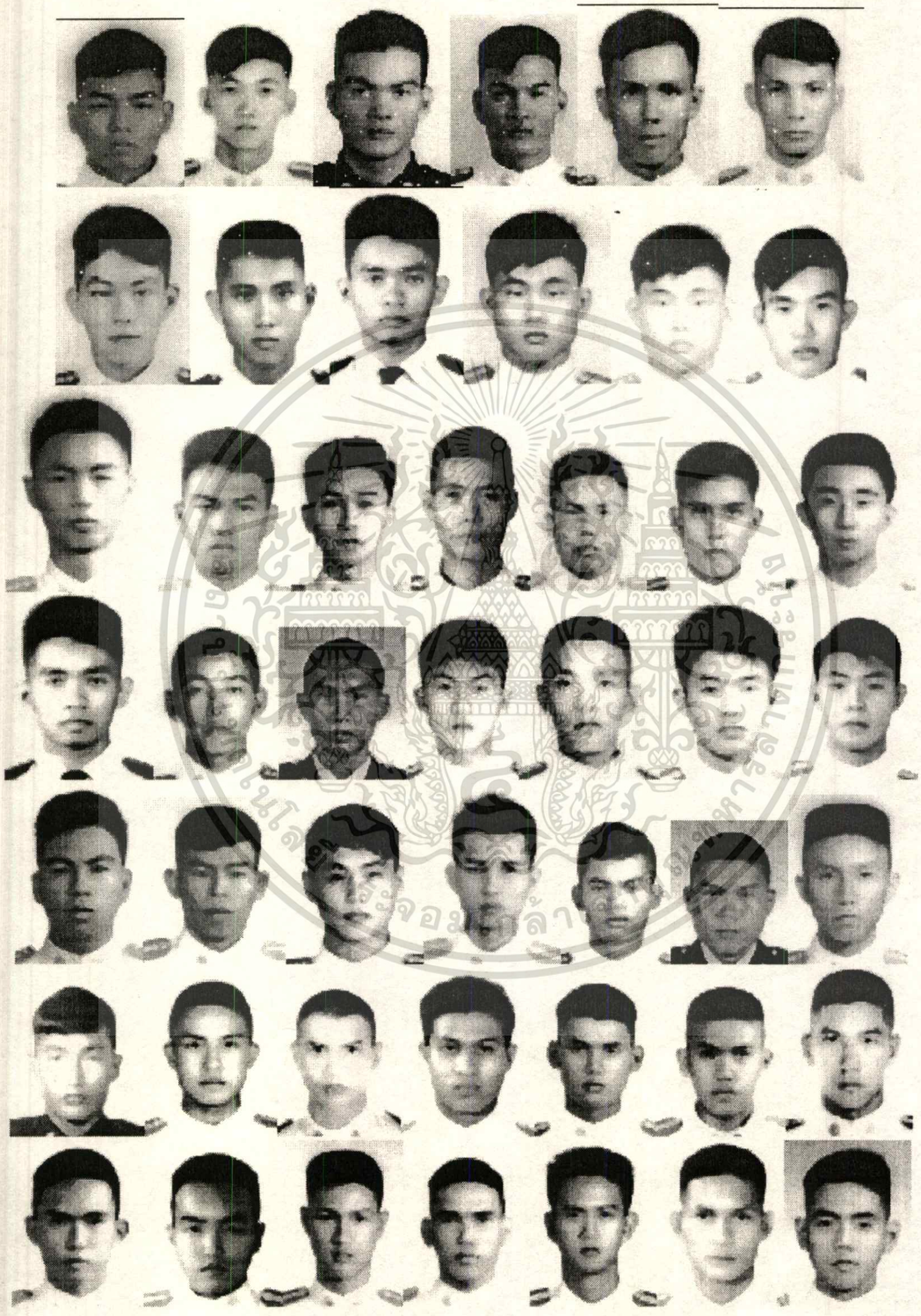
ภาพขอบโซเบลที่ผ่านการย่อ 3 ระดับ 100% 90% และ 80%

- นำภาพ 3 ระดับดังกล่าวจำนวน 3 ภาพนั้น มาเทียบกับภาพใบหน้าอ้างอิงที่ได้นำเตรียมไว้แล้วในหัวข้อที่ 1 (ในขั้นตอนการเตรียมภาพใบหน้าอ้างอิง ด้วยวิธีการสหสัมพันธ์) ดังแสดงขั้นตอนอย่างละเอียด ดังรูปต่อไปนี้ ภาพใบหน้าที่แท้จริงจะเป็นภาพที่มีค่าสหสัมพันธ์สูงที่สุด

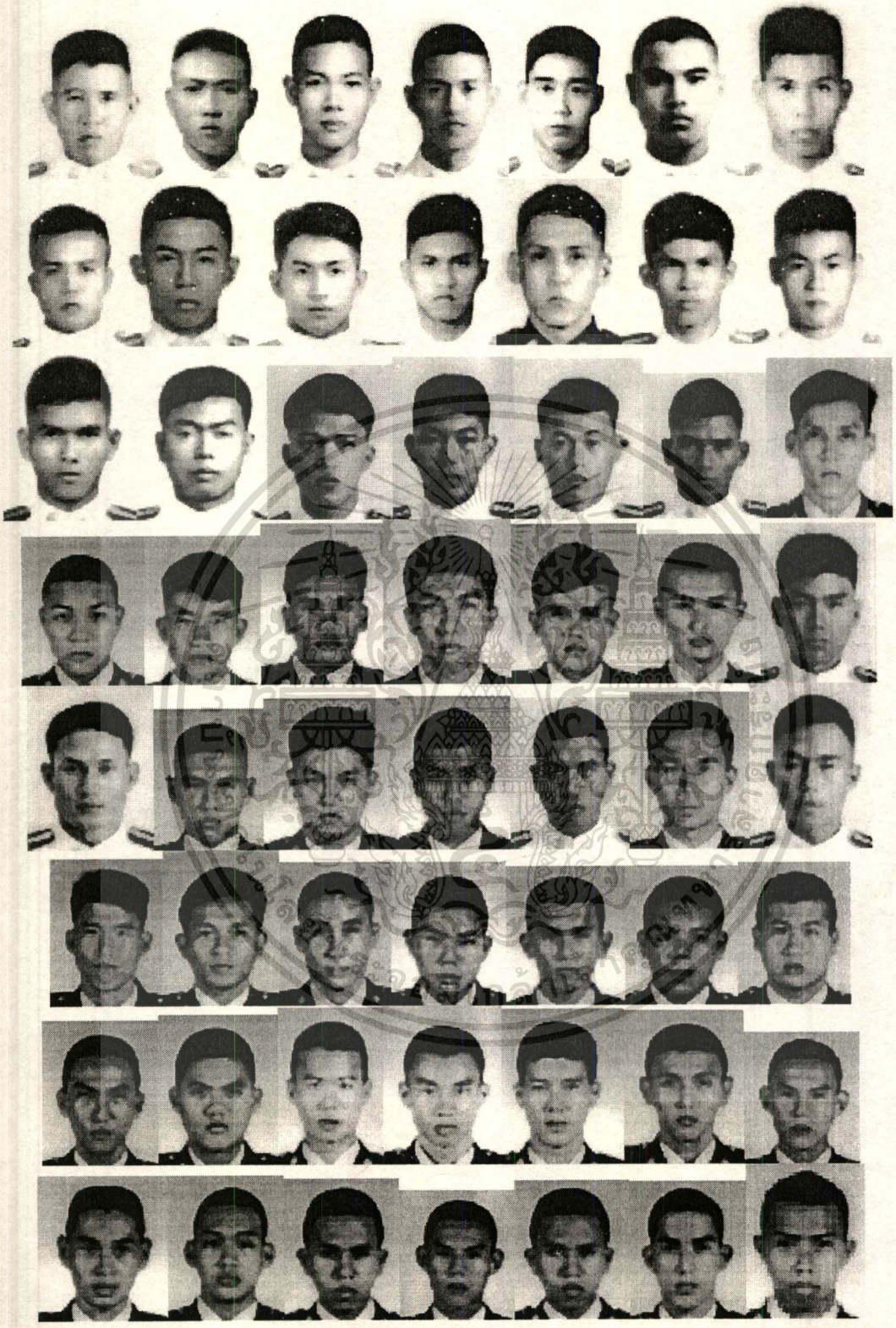



ขั้นตอนการหาส่วนของใบหน้าจากภาพ 3 ระดับ

ตัวอย่างต้นแบบจำนวน 150 ภาพที่ได้มาจากการสแกนเก็บภาพ แสดงได้ดังนี้



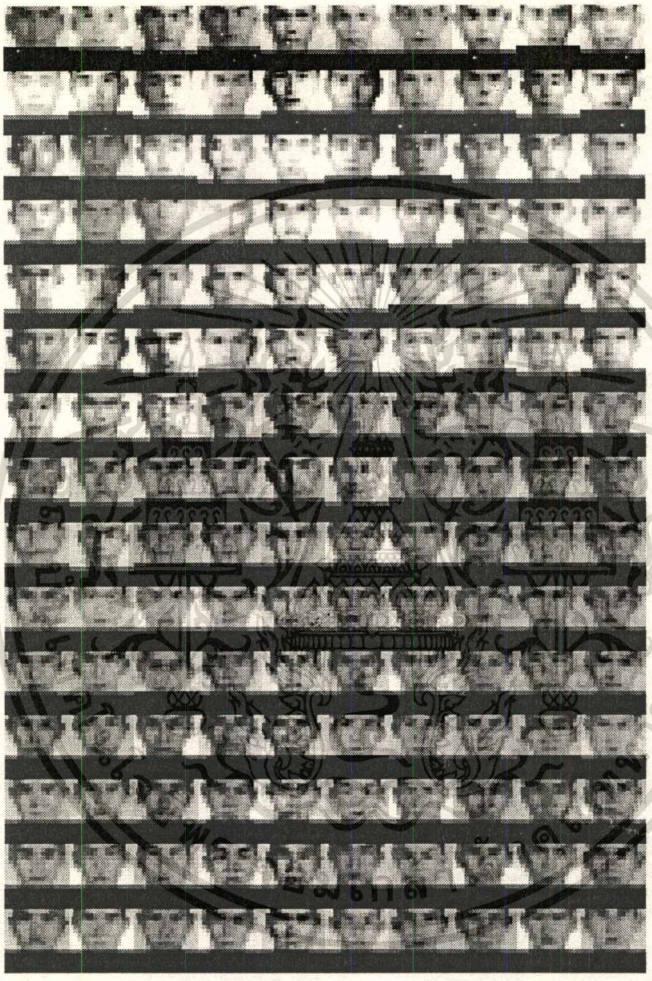
ภาพต้นแบบจากการสแกนด้วยเครื่องสแกนภาพ ความละเอียด 100 dpi



ภาพต้นแบบจากการสแกนด้วยเครื่องสแกนภาพ ความละเอียด 100 dpi (ต่อ)

ภาพต้นแบบจากการสแกนด้วยเครื่องสแกนภาพ ความละเอียด 100 dpi (ต่อ)

ตัวอย่าง ภาพผลจาการค้นหาส่วนของใบหน้า ได้มาจากการค้นหาอย่างอัตโนมัติ จำนวน 150 ภาพแสดงได้ดังนี้



ภาพใบหน้าผลลัพธ์ 150 ภาพ ที่สามารถค้นหาออกมาได้อย่างอัตโนมัติทั้งหมด

# ภาคผนวก ค

## โปรแกรม

โปรแกรมที่ใช้ในการทดลองได้เขียนด้วยภาษาซี โดยใช้ตัวแปรภาษาของบอร์แลนด์ รุ่น 3.1 ผสมกับ WatcomC รุ่น 10.0 แต่ละส่วนสามารถทำงานแบบเป็นอิสระจากกันได้ ทำให้สามารถทำงานต่อเนื่องกันต่อๆ กันไปได้ในลักษณะแบตโพรเซส (Batch Process) ที่ซึ่งประกอบด้วยโปรแกรม ดังนี้

**1) Median Filter**

**2) Sobel Edge Detection**

**3) Face Detection by Template Matching and 3 Level Zooming**

**4) Wavelet Transform**

**5) Vector Quantization**

**-VQ Making Codebook**

**-VQ Encoding**

**6) Neural Network**

**7) Template Matching with 5$^{th}$ Best Match**

## /* Median Filter */

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <stdlib.h>
#include <mem.h>
#include <string.h>
#include <math.h>
#include <complex.h>

#define  PI 3.1415926
#define  LIMIT 4096
#define  THDLOW 50

#define  uc unsigned char
short int THD=128,number;
uc *data,*datb,*datc,*datd,*date,*datf;
short int  *xxx;
short int  *yyy;

long xsize,ysize;

typedef struct {
      char manufacturer;//1
      char version;    //1
      char encoding;   //1
      char bits_per_pixel;//1
      short int  xmin,ymin;  //4
      short int  xmax,ymax; //4
      short int  hres; //2
      short int  vres;//2
      char __palette[48];//48
      char reserved;     //1
      char color_planes;  //1
      short int  bytes_per_line;//2
      short int  palette_type;//2
      char filler[58];//58
      } PCXHEAD;
typedef struct{
  unsigned char RED;
  unsigned char GREEN;
  unsigned char BLUE;
}PCXPALETTE;

short int readpcxline(unsigned char huge *p,FILE
*fp,short int bytes);
void writepcxline(unsigned char *p,FILE
*fp,short int n);
short int  iminimum(short int i,short int a[]);
void swapvalue(short int *a,short int *b);

void read_pcx_size(long *xsize,long *ysize,char
*filename);
void allocate_buff(uc *buff,long xsize,long ysize);
void read_pcx_file(uc *buff,char *filename,long
xsize,long ysize);
void median(uc*,uc*);

void writepcxline(unsigned char *p,FILE
*fp,short int n)
{
 unsigned short int i=0,j=0,t=0;
 do
 {
  i=0;
  while((p[t+i]==p[t+i+1]) && ((t+i) < n) && (i
< 63))
      ++i;

  if(i>0)
   {
   fputc(i | 0xc0,fp);
   fputc(p[t],fp);
   t+=i;
   j+=2;
   }
  else
   {
   if(((p[t]) & 0xc0) == 0xc0)
    {
    fputc(0xc1,fp);
    ++j;
    }
   fputc(p[t++],fp);
   ++j;
   }
}
```

```
 while(t<n);
}

short int writepcx(uc *data_out,char
*filename,long xsize,long ysize)
{
FILE *fp;
short int x,y;
unsigned char _outpalette[768],buf[1024];
PCXHEAD   pcx;
 if((fp = fopen(filename,"wb"))==NULL)
  {
  printf("Error saving %s\n",filename);
  exit(1);
  }

 for(x=0;x<256;x++)
  {
  _outpalette[x*3]  =x;
  _outpalette[x*3+1]=x;
  _outpalette[x*3+2]=x;
  }

 memset((char*)&pcx,0,128);
 pcx.manufacturer=10;
 pcx.encoding=1;
 pcx.xmin=0;
 pcx.ymin=0;
 pcx.xmax=xsize-1;
 pcx.ymax=ysize-1;
 pcx.color_planes=1;
 pcx.bytes_per_line=xsize;
 pcx.bits_per_pixel=8;
 pcx.version=5;
 fwrite((char*)&pcx,1,128,fp);

 for(y=0;y<ysize;y++)
   {
   for(x=0;x<xsize;x++)
     buf[x]=*(data_out+y*xsize+x+4);
   writepcxline(buf,fp,xsize);
   }
 fputc(0x0c,fp);
 fwrite(_outpalette,1,768,fp);
 fclose(fp);
 return(1);
}

short int readpcxline(unsigned char huge *p,FILE
*fp,short int bytes)
{
 unsigned int n=0,i,c;
 do
 {
  c=fgetc(fp)&0xff;
  if((c&0xc0)==0xc0)
    {
    i=c&0x3f;
    c=fgetc(fp);
    while(i--) p[n++]=c;
    }
  else p[n++]=c;
 }
 while(n<bytes);
 return(n);
}

void read_pcx_file(uc *buff,char *filename,long
xsize,long ysize)
{
FILE          *fh;
PCXHEAD       pcx;
PCXPALETTE    pal[256];
PCXPALETTE    bufpal[1];
long          imgsize;
long          bytes;
short int     ch,i;

 if((fh=fopen(filename,"rb"))==NULL)
   {
   printf("\n File [%s] not found",filename);
   exit(1);
   }

 if(fread((char *)&pcx,1,128,fh) != 128)
   {
```

```
    printf("\nPCXHEADER of [%s] file
error",filename);
    exit(1);
   }

 bytes = pcx.bytes_per_line;
 xsize = pcx.xmax-pcx.xmin+1;
 ysize = pcx.ymax-pcx.ymin+1;

 if(pcx.bits_per_pixel==8)
    bytes=xsize;
 else
   {
    printf("\nUse PCX 256 color image only");
    exit(1);
   }

 if(fseek(fh,-769L,SEEK_END))
 {
   printf("PCXPALETTE seek error\n");
   exit(1);
 }

 ch=fgetc(fh);
 if(ch!=0xC)
    printf("PCXPALETTE error");

 for(i=0;i<256;i++)
    {
     fread(&pal[i],768,1,fh);
    }


 fseek(fh,128L,SEEK_SET);
 for(i=0;i<ysize;++i)
 {
  readpcxline(&buff[i*xsize+4],fh,bytes);
 }

 fclose(fh);
}
```

```
void read_pcx_size(long *xsize,long *ysize,char
*filename)
{
FILE          *fh;
PCXHEAD       pcx;
PCXPALETTE    pal[256];
PCXPALETTE    bufpal[1];
long          imgsize;
long          bytes;
short int     ch,i;

 if((fh=fopen(filename,"rb"))==NULL)
   {
   printf("\n File [%s] not found",filename);
   exit(1);
   }

 if(fread((char *)&pcx,1,128,fh) != 128)
   {
    printf("\nPCXHEADER of [%s] file
error",filename);
    exit(1);
   }

 *xsize = (pcx.xmax-pcx.xmin)+1;
 *ysize = (pcx.ymax-pcx.ymin)+1;

 fclose(fh);
}

short int iminimum(short int i,short int a[])
{
 short int j,imin;              //local variable
in this function
 imin=i;
 for(j=i+1;j<9;j++)          //loop i+1 to 4 for
find array index-
  {                          //of minimum value
  if(a[j]<a[imin])
     imin=j;
  }
 return imin; //return array index of minimum
}
```

```
void swapvalue(short int *a,short int *b)
{
 short int c;                    //local variable
in this function
 c=*a;
 *a=*b;
 *b=c;
}

void median(uc *datain,uc *dataout)//median
3x3
{
 short int x,y,i,imin,a[9];

 //top+bottom
 for(x=0;x<xsize;x++)
   {
    *(dataout+xsize+x+4)=*(datain+xsize+x+4);
    *(dataout+(ysize-
1)*xsize+x+4)=*(datain+(ysize-1)*xsize+x+4);
   }

 //left+right
 for(y=0;y<ysize;y++)
   {
    *(dataout+y*xsize+4)=*(datain+y*xsize+4);
    *(dataout+y*xsize+xsize-
1+4)=*(datain+y*xsize+xsize-1+4);
   }

 for(y=1;y<ysize-1;y++)
  for(x=1;x<xsize-1;x++)
   {
    //0 1 2
    //3 4 5
    //6 7 8
    a[0]=*(datain+(y-1)*xsize+x+4-1);
    a[1]=*(datain+(y-1)*xsize+x+4);
    a[2]=*(datain+(y-1)*xsize+x+4+1);
    a[3]=*(datain+y*xsize+x+4-1);

    a[4]=*(datain+y*xsize+x+4);
    a[5]=*(datain+y*xsize+x+4+1);
    a[6]=*(datain+(y+1)*xsize+x+4-1);
    a[7]=*(datain+(y+1)*xsize+x+4);
    a[8]=*(datain+(y+1)*xsize+x+4+1);

    for(i=0;i<9;i++) //loop 0-8 for Selection
sort algorithm
     {
     imin=iminimum(i,&a[0]);
     swapvalue(&a[i],&a[imin]);
     }
    *(dataout+y*xsize+x+4)=a[4];
   }
 }

void main(short int argc,char *argv[])
{
short int  i,ch,x,y,loop,DIM=64;
char name1[40];
char name2[40];
long newxsize,newysize;

 if(argc<3)
   {
   printf("\nCorrect the parameter please");
   printf("\n%s  input.pcx
output.pcx\n",argv[0]);
   exit(1);
   }

 sprintf(name1,"%s",argv[1]);
 sprintf(name2,"%s",argv[2]);

printf("\nfrom %s to %s",name1,name2);
 read_pcx_size(&xsize,&ysize,name1);
 printf("\nXsize=%ld Ysize=%ld",xsize,ysize);

//original image
 data=(uc*)calloc((xsize*ysize)+4,1);
 if(!data)
   {
   printf("\nAlloc buffer error");
   exit(1);
   }
```

```
data[0]=(xsize-1)&0x00ff;
data[1]=((xsize-1)&0xff00)>>8;
data[2]=(ysize-1)&0x00ff;
data[3]=((ysize-1)&0xff00)>>8;
printf("\n&alloc data=%p",data);
read_pcx_file(data,name1,xsize,ysize);


datb=(uc*)calloc((xsize*ysize)+4,1);
if(!datb)
  {
  printf("\nAlloc buffer error");
  exit(1);
  }
datb[0]=(xsize-1)&0x00ff;
datb[1]=((xsize-1)&0xff00)>>8;
datb[2]=(ysize-1)&0x00ff;
datb[3]=((ysize-1)&0xff00)>>8;
printf("\n&alloc datb=%p",datb);


median(data,datb);//from data to datb

writepcx(datb,name2,xsize,ysize);

free(data);
free(datb);
}
```

## /*Sobel Edge Detection*/

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <stdlib.h>
#include <mem.h>
#include <string.h>
#include <math.h>
#include <complex.h>

#define  PI 3.1415926
#define  LIMIT 4096
#define  THDLOW 50

#define  uc unsigned char
short int THD=128,number;
uc *data,*datb,*datc,*datd,*date,*datf;
short int  *xxx;
short int  *yyy;

long xsize,ysize;

typedef struct {
      char manufacturer;//1
      char version;    //1
      char encoding;   //1
      char bits_per_pixel;//1
      short int  xmin,ymin;  //4
      short int  xmax,ymax; //4
      short int  hres; //2
      short int  vres;//2
      char __palette[48];//48
      char reserved;     //1
      char color_planes;  //1
      short int  bytes_per_line;//2
      short int  palette_type;//2
      char filler[58];//58
      } PCXHEAD;
typedef struct{
  unsigned char RED;
  unsigned char GREEN;
  unsigned char BLUE;
}PCXPALETTE;

short int readpcxline(unsigned char huge *p,FILE
*fp,short int bytes);
void writepcxline(unsigned char *p,FILE
*fp,short int n);

void read_pcx_size(long *xsize,long *ysize,char
*filename);
void allocate_buff(uc *buff,long xsize,long ysize);
void read_pcx_file(uc *buff,char *filename,long
xsize,long ysize);

void writepcxline(unsigned char *p,FILE
*fp,short int n)
{
 unsigned short int i=0,j=0,t=0;
 do
 {
 i=0;
  while((p[t+i]==p[t+i+1]) && ((t+i) < n) && (i
 < 63))
        ++i;
  if(i>0)
   {
   fputc(i | 0xc0,fp);
   fputc(p[t],fp);
   t+=i;
   j+=2;
   }
  else
   {
   if(((p[t]) & 0xc0) == 0xc0)
    {
    fputc(0xc1,fp);
    ++j;
    }
   fputc(p[t++],fp);
   ++j;
   }
 }
 while(t<n);
}
```

```
short int writepcx(uc *data_out,char
*filename,long xsize,long ysize)
{
FILE *fp;
short int x,y;
unsigned char _outpalette[768],buf[1024];
PCXHEAD   pcx;
 if((fp = fopen(filename,"wb"))==NULL)
  {
  printf("Error saving %s\n",filename);
  exit(1);
  }

 for(x=0;x<256;x++)
  {
  _outpalette[x*3]  =x;
  _outpalette[x*3+1]=x;
  _outpalette[x*3+2]=x;
  }

 memset((char*)&pcx,0,128);
 pcx.manufacturer=10;
 pcx.encoding=1;
 pcx.xmin=0;
 pcx.ymin=0;
 pcx.xmax=xsize-1;
 pcx.ymax=ysize-1;
 pcx.color_planes=1;
 pcx.bytes_per_line=xsize;
 pcx.bits_per_pixel=8;
 pcx.version=5;
 fwrite((char*)&pcx,1,128,fp);

 for(y=0;y<ysize;y++)
   {
   for(x=0;x<xsize;x++)
     buf[x]=*(data_out+y*xsize+x+4);
   writepcxline(buf,fp,xsize);
   }
 fputc(0x0c,fp);
 fwrite(_outpalette,1,768,fp);
 fclose(fp);
 return(1);
}
```

```
short int readpcxline(unsigned char huge *p,FILE
*fp,short int bytes)
{
 unsigned int n=0,i,c;
 do
 {
  c=fgetc(fp)&0xff;
  if((c&0xc0)==0xc0)
    {
    i=c&0x3f;
    c=fgetc(fp);
    while(i--) p[n++]=c;
    }
  else p[n++]=c;
 }
 while(n<bytes);
 return(n);
}

void read_pcx_file(uc *buff,char *filename,long
xsize,long ysize)
{
FILE            *fh;
PCXHEAD         pcx;
PCXPALETTE      pal[256];
PCXPALETTE      bufpal[1];
long            imgsize;
long            bytes;
short int       ch,i;

 if((fh=fopen(filename,"rb"))==NULL)
   {
   printf("\n File [%s] not found",filename);
   exit(1);
   }

 if(fread((char *)&pcx,1,128,fh) != 128)
   {
    printf("\nPCXHEADER of [%s] file
error",filename);
    exit(1);
   }
```

```
bytes = pcx.bytes_per_line;
xsize = pcx.xmax-pcx.xmin+1;
ysize = pcx.ymax-pcx.ymin+1;

if(pcx.bits_per_pixel==8)
   bytes=xsize;
else
  {
   printf("\nUse PCX 256 color image only");
   exit(1);
  }

if(fseek(fh,-769L,SEEK_END))
{
  printf("PCXPALETTE seek error\n");
  exit(1);
}

ch=fgetc(fh);
if(ch!=0xC)
   printf("PCXPALETTE error");

for(i=0;i<256;i++)
   {
    fread(&pal[i],768,1,fh);
   }

fseek(fh,128L,SEEK_SET);
for(i=0;i<ysize;++i)
{
 readpcxline(&buff[i*xsize+4],fh,bytes);
}

fclose(fh);
}

void read_pcx_size(long *xsize,long *ysize,char
*filename)
{
FILE          *fh;
PCXHEAD       pcx;
PCXPALETTE    pal[256];
PCXPALETTE    bufpal[1];
long          imgsize;
long          bytes;
short int     ch,i;

if((fh=fopen(filename,"rb"))==NULL)
  {
  printf("\n File [%s] not found",filename);
  exit(1);
  }

if(fread((char *)&pcx,1,128,fh) != 128)
  {
   printf("\nPCXHEADER of [%s] file
error",filename);
   exit(1);
  }

 *xsize = (pcx.xmax-pcx.xmin)+1;
 *ysize = (pcx.ymax-pcx.ymin)+1;

 fclose(fh);
 }

 void edge(uc *datain,uc *dataout)
 {
 short int
 z2,z4,z6,z8,z1,z3,z7,z9,result,resultx,resulty,x,y;
  //sobel
  for(y=1;y<ysize-1;y++)
   for(x=1;x<xsize-1;x++)
    {
     z2=*(datain+(y-1)*xsize+x+4);
     z4=*(datain+y*xsize+x+4-1);
     z6=*(datain+y*xsize+x+4+1);
     z8=*(datain+(y+1)*xsize+x+4);
     resultx=(z7+2*z8+z9)-(z1+2*z2+z3);

     z1=*(datain+(y-1)*xsize+x+4-1);
     z3=*(datain+(y-1)*xsize+x+4+1);
     z7=*(datain+(y+1)*xsize+x+4-1);
     z9=*(datain+(y+1)*xsize+x+4+1);
     resulty=(z3+2*z6+z9)-(z1+2*z4+z7);
```

```
      result=(short
int)sqrt((double)(resultx*(long)resultx +
resulty*(long)resulty));
      if(result>255)
            result=255;
      else if(result<0)
            result=0;

      *(dataout+y*xsize+x+4)=result;
     }
}

void main(short int argc,char *argv[])
{
short int  i,ch,x,y,loop,DIM=64;
char name1[40];
char name2[40];
long newxsize,newysize;

 if(argc<3) printf("\nface <input.pcx>
<output.pcx> \n"),exit(1);
 sprintf(name1,"%s",argv[1]);
 sprintf(name2,"%s",argv[2]);

 printf("\nfrom %s to %s",name1,name2);

 read_pcx_size(&xsize,&ysize,name1);
 printf("\nXsize=%ld Ysize=%ld",xsize,ysize);

//original image
 data=(uc*)calloc((xsize*ysize)+4,1);
 if(!data)
   {
   printf("\nAlloc buffer error");
   exit(1);
   }
 data[0]=(xsize-1)&0x00ff;
 data[1]=((xsize-1)&0xff00)>>8;
 data[2]=(ysize-1)&0x00ff;
 data[3]=((ysize-1)&0xff00)>>8;
 printf("\n&alloc data=%p",data);
 read_pcx_file(data,name1,xsize,ysize);

 datb=(uc*)calloc((xsize*ysize)+4,1);
 if(!datb)
   {
   printf("\nAlloc buffer error");
   exit(1);
   }
 datb[0]=(xsize-1)&0x00ff;
 datb[1]=((xsize-1)&0xff00)>>8;
 datb[2]=(ysize-1)&0x00ff;
 datb[3]=((ysize-1)&0xff00)>>8;
 printf("\n&alloc datb=%p",datb);

 edge(data,datb);

 writepcx(datb,name2,xsize,ysize);

 free(data);
 free(datb);
}
```

```
/*Face Detection using
Correlation
and 3 level Zooming*/
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <stdlib.h>
#include <mem.h>
#include <string.h>
#include <time.h>
#include <math.h>
#include <dos.h>
short int xst,yst,xnd,ynd;
short int DIM;
#define THD 128
#define  uc unsigned char
long      xsize,ysize,imgsize,xbox,
ybox,xref,yref,xr,yr,xnew,ynew;
long
newxsize2,newysize2,newxsize3,newysize3;
uc *datainedge, *dataref, *dataingray, *dataout,
*datazoom2, *datazoom3;
uc *datazoom2g,*datazoom3g;
short int xcen,ycen,maxx,maxy,maxt=10,mode;
char str1[40];
char str2[40];
char str[40];
char nameing[40];
char nameine[40];
char nameref[40];
char namepos[40];
char
nameout[40],nameout2[40],nameout3[40];
short int STY=10,YBOX=10;

void main(short int argc,char *argv[])
{
short int  x,y,xmin,xmax,firsty;
FILE       *fh;
PCXHEAD    pcx;
PCXPALETTE pal[256];
short int
z2,z4,z6,z8,z5,z1,z3,z7,z9,result,resultx,resulty;
double fresult;
float like[3];
long xlike[3],ylike[3];
long likemax,likeitem;
 if(argc!=8)
   {
   printf("\nCorrect the parameter please");
   printf("\n%s  ref.pcx  ref.txt  ingray.pcx
inedge.pcx  out.pcx out2.pcx out3.pcx",argv[0]);
   printf("\nref.txt is position file for assign box
template");
   exit(1);
   }
 sprintf(nameref ,"%s",argv[1]);
 sprintf(namepos ,"%s",argv[2]);
 sprintf(nameing ,"%s",argv[3]);
 sprintf(nameine ,"%s",argv[4]);
 sprintf(nameout ,"%s",argv[5]);
 sprintf(nameout2 ,"%s",argv[6]);
 sprintf(nameout3 ,"%s",argv[7]);
 if((fh=fopen(nameing,"rb"))==NULL)
   {
   printf("\nFile not found");
   exit(1);
   }
 fread((char *)&pcx,1,sizeof(PCXHEAD),fh);
 fseek(fh,-769L,SEEK_END); fgetc(fh);
 for(x=0;x<256;x++)

if(fread(&pal[x],1,(size_t)sizeof(PCXPALETTE),f
h) != sizeof(PCXPALETTE))
     printf("fread PALETTE error"),exit(1);
 xsize = pcx.xmax-pcx.xmin+1; ysize =
pcx.ymax-pcx.ymin+1;
 imgsize=xsize*ysize;

if((dataingray=(uc*)calloc(imgsize+4,1))==NUL
L) exit(1);
 fseek(fh,128L,SEEK_SET);
 for(y=0;y<ysize;y++)
readpcxline(&dataingray[y*xsize+4],fh,xsize);
```

```
dataingray[0]=(xsize-1)&0x00ff;
dataingray[1]=((xsize-1)&0xff00)>>8;
dataingray[2]=(ysize-1)&0x00ff;
dataingray[3]=((ysize-1)&0xff00)>>8;
fclose(fh);
if((fh=fopen(nameine,"rb"))==NULL)
  {
  printf("\nFile not found");
  exit(1);
  }
fread((char *)&pcx,1,sizeof(PCXHEAD),fh);
fseek(fh,-769L,SEEK_END); fgetc(fh);
for(x=0;x<256;x++)

if(fread(&pal[x],1,(size_t)sizeof(PCXPALETTE),f
h) != sizeof(PCXPALETTE))
    printf("fread PALETTE error"),exit(1);
xsize = pcx.xmax-pcx.xmin+1; ysize =
pcx.ymax-pcx.ymin+1;
imgsize=xsize*ysize;

if((datainedge=(uc*)calloc(imgsize+4,1))==NUL
L) exit(1);
fseek(fh,128L,SEEK_SET);
for(y=0;y<ysize;y++)
readpcxline(&datainedge[y*xsize+4],fh,xsize);
datainedge[0]=(xsize-1)&0x00ff;
datainedge[1]=((xsize-1)&0xff00)>>8;
datainedge[2]=(ysize-1)&0x00ff;
datainedge[3]=((ysize-1)&0xff00)>>8;
fclose(fh);
if((fh=fopen(nameref,"rb"))==NULL)
printf("\nFile #1 not found"),exit(1);
fread((char *)&pcx,1,sizeof(PCXHEAD),fh);
fseek(fh,-769L,SEEK_END); fgetc(fh);
for(x=0;x<256;x++)

if(fread(&pal[x],1,(size_t)sizeof(PCXPALETTE),f
h) != sizeof(PCXPALETTE))
    printf("fread PALETTE error"),exit(1);
xr = pcx.xmax-pcx.xmin+1; yr = pcx.ymax-
pcx.ymin+1;
imgsize=xr*yr;

if((dataref=(uc*)calloc(imgsize+4,1))==NULL)
exit(1);
fseek(fh,128L,SEEK_SET);
for(y=0;y<yr;y++)
readpcxline(&dataref[y*xr+4],fh,xr);
dataref[0]=(xr-1)&0x00ff; dataref[1]=((xr-
1)&0xff00)>>8;
dataref[2]=(yr-1)&0x00ff; dataref[3]=((yr-
1)&0xff00)>>8;
fclose(fh);
//read start of Y coordinate to search
if((fh=fopen(namepos,"rt"))==NULL)
    printf("\nResult Position File can not
write"),exit(1);
fscanf(fh,"%d %d",&STY,&YBOX);
STY=0;
fclose(fh);
newxsize2=xsize*0.9;
newysize2=xsize*0.9;
printf("\nnewXsize=%ld
newYsize=%ld",newxsize2,newysize2);
if((datazoom2=(uc*)calloc((newxsize2*newysize
2)+4,1))==NULL)
  {
  printf("\nAlloc buffer error");
  exit(1);
  }
if((datazoom2g=(uc*)calloc((newxsize2*newysiz
e2)+4,1))==NULL)
  {
  printf("\nAlloc buffer error");
  exit(1);
  }
datazoom2[0]=(newxsize2-1)&0x00ff;
datazoom2[1]=((newxsize2-1)&0xff00)>>8;
datazoom2[2]=(newysize2-1)&0x00ff;
datazoom2[3]=((newysize2-1)&0xff00)>>8;
datazoom2g[0]=(newxsize2-1)&0x00ff;
datazoom2g[1]=((newxsize2-1)&0xff00)>>8;
datazoom2g[2]=(newysize2-1)&0x00ff;
datazoom2g[3]=((newysize2-1)&0xff00)>>8;

resize(datainedge,datazoom2,newxsize2,newysi
ze2);
```

```
resize(dataingray,datazoom2g,newxsize2,newysize2);
 newxsize3=xsize*0.8;
 newysize3=xsize*0.8;
 printf("\nnewXsize=%ld newYsize=%ld",newxsize3,newysize3);

if((datazoom3=(uc*)calloc((newxsize3*newysize3)+4,1))==NULL)
  {
  printf("\nAlloc buffer error");
  exit(1);
  }
if((datazoom3g=(uc*)calloc((newxsize3*newysize3)+4,1))==NULL)
  {
  printf("\nAlloc buffer error");
  exit(1);
  }
 datazoom3[0]=(newxsize3-1)&0x00ff;
 datazoom3[1]=((newxsize3-1)&0xff00)>>8;
 datazoom3[2]=(newysize3-1)&0x00ff;
 datazoom3[3]=((newysize3-1)&0xff00)>>8;
 datazoom3g[0]=(newxsize3-1)&0x00ff;
 datazoom3g[1]=((newxsize3-1)&0xff00)>>8;
 datazoom3g[2]=(newysize3-1)&0x00ff;
 datazoom3g[3]=((newysize3-1)&0xff00)>>8;

resize(datainedge,datazoom3,newxsize3,newysize3);

resize(dataingray,datazoom3g,newxsize3,newysize3);
 like[0]= Correlation (datainedge ,xsize,ysize ,dataref,xr,yr,&xlike[0],&ylike[0]);
 like[1]= Correlation(datazoom2,newxsize2,newysize2, dataref,xr,yr,&xlike[1],&ylike[1]);
 like[2]= Correlation(datazoom3,newxsize3,newysize3, dataref,xr,yr,&xlike[2],&ylike[2]);
 likemax=0;
 for(x=0;x<3;x++)
   {
   printf("\nlike[%d]=[%f%%] found at %ld %ld",x,like[x],xlike[x],ylike[x]);
   if(like[x]>likemax)
     {
      likemax=like[x];
      likeitem=x;
     }
   }
 if(likeitem==0)
   {
   xst=xlike[0];
   yst=ylike[0];
   xnd=xlike[0]+xr;
   ynd=ylike[0]+yr;
   for(y=0;y<ysize;y++)
     for(x=0;x<xsize;x++)

 if(!(x>=xst&&x<=xnd&&y>=yst&&y<=ynd))
            *(datainedge+y*xsize+x+4)=0;
 for(y=ylike[0];y<ylike[0]+yr;y++)
     for(x=xlike[0];x<xlike[0]+xr;x++)
          *(dataref+(y-ylike[0])*xr+(x-xlike[0])+4)=*(dataingray+y*xsize+x+4);
 writepcx(dataref,nameout,xr,yr);
 }
  else if(likeitem==1)
   {
   xst=xlike[1];
   yst=ylike[1];
   xnd=xlike[1]+xr;
   ynd=ylike[1]+yr;
   for(y=0;y<newysize2;y++)
     for(x=0;x<newxsize2;x++)

if(!(x>=xst&&x<=xnd&&y>=yst&&y<=ynd))

*(datazoom2+y*newxsize2+x+4)=0;
for(y=ylike[1];y<ylike[1]+yr;y++)
     for(x=xlike[1];x<xlike[1]+xr;x++)
```

```
        *(dataref+(y-ylike[1])*xr+(x-
xlike[1])+4)=*(datazoom2g+y*newxsize2+x+4);
writepcx(dataref,nameout,xr,yr);
}
 else if(likeitem==2)
  {
   xst=xlike[2];
   yst=ylike[2];
   xnd=xlike[2]+xr;
   ynd=ylike[2]+yr;
   for(y=0;y<newysize3;y++)
    for(x=0;x<newxsize3;x++)

if(!(x>=xst&&x<=xnd&&y>=yst&&y<=ynd))

*(datazoom3+y*newxsize3+x+4)=0;
   for(y=ylike[2];y<ylike[2]+yr;y++)
    for(x=xlike[2];x<xlike[2]+xr;x++)
        *(dataref+(y-ylike[2])*xr+(x-
xlike[2])+4)=*(datazoom3g+y*newxsize3+x+4);
writepcx(dataref,nameout,xr,yr);
}
 free(dataref);
 free(dataingray);
 free(datainedge);
 free(datazoom2);
 free(datazoom3);
 free(dataout);
}
```

**float Correlation(unsigned char *data,long xsize,long ysize,unsigned char *dataref,long xr,long yr,long *xlike_,long *ylike_)**

```
{
int x,y;
float          R,ratio;
long
        A,B,p_,s_,C,boxsize,P,S,xlike,ylike;
short int xpos,ypos,xsthead,
xndhead,ysthead,yndhead,xstsearch,ystsearch;
short int xndsearch,yndsearch;
double         cat,Bf,Cf,like;
boxsize=xr*yr;
p_=0;
for(y=0;y<yr;y++)
 for(x=0;x<xr;x++)
        p_+=*(dataref+y*xr+x+4);
p_=p_/boxsize;

B=0;
for(y=0;y<yr;y++)
 for(x=0;x<xr;x++)
      {
       cat=(double)*(dataref+y*xr+x+4)-p_;
       B+=pow(cat,2);
      }
 Bf=sqrt((long double)B);
 xstsearch=xsize>>2;
 xndsearch=xsize>>1;

 if(xndsearch>xsize-xr)
      xndsearch=xsize-xr;
 ystsearch=ysize>>2;//STY-(YBOX>>1);
 yndsearch=ysize>>1;//STY+(YBOX>>1);
 if(yndsearch>ysize-yr)
      yndsearch=ysize-yr;
 xlike=ylike= like=0.0;

for(ypos=ystsearch;ypos<yndsearch;ypos+=2)

for(xpos=xstsearch;xpos<xndsearch;xpos+=2)
 {
      s_=0;
      for(y=0;y<yr;y++)
      for(x=0;x<xr;x++)

s_+=*(data+(y+ypos)*xsize+(x+xpos)+4);
      s_=s_/boxsize;
      A=C=0;
      for(y=0;y<yr;y++)
      for(x=0;x<xr;x++)
        {
          P=*(dataref+y*xr+x+4)-p_;

S=*(data+(y+ypos)*xsize+(x+xpos)+4)-s_;
          A+=P*S;
```

```
            C+=S*S;
          }
        Cf=sqrt((long double)C);
        R=(float)A/(Bf*Cf);
        if(R>like)
          {
           like=R;    xlike=xpos;
           ylike=ypos;
          }
    }
 *xlike_=xlike;
 *ylike_=ylike;
 return(like);
}
```

**void resize(uc *datain,uc *dataout,long XDIM,long YDIM)**

```
{
short int x,y;
float ratio;
 ratio=xsize/(float)XDIM;
 //making reduce of testing image
 for(y=0;y<ysize;y++)
  for(x=0;x<xsize;x++)
    {
     short int i,j;
     i=x/ratio;
     j=y/ratio;
     if(i>(XDIM-1))        i=XDIM-1;
     else if(i<0)          i=0;
     if(j>(YDIM-1))        j=YDIM-1;
     else if(j<0)          j=0;

*(dataout+j*XDIM+i+4)=*(datain+y*xsize+x+4);
    }
}
```

**/* Two dimensional wavelet transform */**

```
#define MAX
#include "lib.h"
#include <math.h>
#include <malloc.h>
#include <stdio.h>

#define FORWARD 1
#define INVERSE -1

float rec[16] =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

void fwt_1d (float *data,  int N, int direction);
void wavelet (float *data, int N, int direction);
void Haar_2 (float *data,  int N, int direction);
void fwt_2d (IMAGE im, float **result);
void fwti_2d (float **result, IMAGE im);
float ** f2d (int i, int j);
void NEXT();

void NEXT ()
{
        static int i = 0;
        static char xc[5] = {'|', '/', '-', '\\', '
'};

        fprintf (stdout, "%c%c", 010,
xc[i++]);
        fflush (stdout);
        if (i>3) i = 0;
}

int main(int argc, char *argv[])
{
        IMAGE im, tmp;
        float **result, xmax, xmin, xd;
        int maxr, i, j, k;
        FILE *f;

        if (argc<3)
        {
          printf ("2d wavelet transform -
<input> <output>\n");
          printf ("File of raw wavelet
coefficients saved in 'wtout'\n");
          exit (1);
        }

        printf ("Computing a 2d wavelet
transform on a Haar basis:\n");
        im = Input_PBM (argv[1]);
        if (im == 0)
        {
                printf ("Can't open file
'%s'.\n", argv[1]);
                return 0;
        }
        result = f2d(im->info->nr, im->info-
>nc);
        printf ("Forward: ");

        fwt_2d (im, result);

        tmp = newimage (im->info->nr, im-
>info->nc);
        xmax = -1.0e12; xmin = -xmax;
        for (i=0; i<im->info->nr; i++)
          for (j=0; j<im->info->nc; j++)
          {
            if (xmax < result[i][j]) xmax =
result[i][j];
            if (xmin > result[i][j]) xmin =
result[i][j];
          }
        xd = xmax-xmin;
        for (i=0; i<im->info->nr; i++)
          for (j=0; j<im->info->nc; j++)
            tmp->data[i][j] = (unsigned char)
((result[i][j]-xmin)/xd * 255);
        Output_PBM (tmp, "wtout");
        printf ("\nWavelet transform saved in
file 'wtout'.\n");
        freeimage (tmp);
```

```
        printf ("Reconstructing the data from
the wavelets:\n");
        printf ("Enter max row/column for
reconstruction:\n");
        scanf ("%d", &maxr);

/* Remove all coefficients except those in
rows/cols 1-maxr */
        if (maxr > 0)
        for (i=0; i<im->info->nr; i++)
          for (j=0; j<im->info->nc; j++)
            if (i>maxr || j > maxr) result[i][j] =
0.0;

/* back-transform */
        printf ("Inverse transform: ");
        fwti_2d (result, im);
        Output_PBM (im, argv[2]);
        printf ("Restored image saved in
'%s'.\n", argv[2]);
        return 0;
}

void fwt_1d (float *data, int N, int direction)
{
        int thisn;

        if (N > 2)
        {
          if (direction >= 0)
          {
            thisn = N;
            while (thisn >= 2)
            {
              wavelet (data, thisn, direction);
              thisn /= 2;
            }
          } else if (direction == INVERSE)
          {
            thisn = 2;
            while (thisn <= N)
            {
              wavelet (data, thisn, direction);
              thisn *= 2;
            }
          }
        }
}

/* Image is preferably square; size of each
dimension is a power of two */
void fwt_2d (IMAGE im, float **result)
{
        int i,j,k;
        float *tmp;

/* Copy to the floating point array */
        for (i=0; i<im->info->nr; i++)
          for (j=0; j<im->info->nc; j++)
            result[i][j] = (float)(im->data[i][j]);

/* Transform the rows */
        for (i=0; i<im->info->nr; i++)
        {
          fwt_1d (result[i], im->info->nc,
FORWARD);
          NEXT();
        }

/* Transform the columns */
        tmp = (float *)malloc (sizeof(float)*im-
>info->nr);
        for (j=0; j<im->info->nc; j++)
        {
          for (i=0; i<im->info->nr; i++)
            tmp[i] = result[i][j];
          fwt_1d (tmp, im->info->nr,
FORWARD);
          for (i=0; i<im->info->nr; i++)
            result[i][j] = tmp[i];
          NEXT();
        }
        free(tmp);
}
```

```
void fwti_2d (float **result, IMAGE im)
{
        int i,j,k;
        float *tmp;

/* Transform the rows */
        printf ("\nRows: ");
        for (i=0; i<im->info->nr; i++)
        {
          fwt_1d (result[i], im->info->nc,
INVERSE);
          NEXT();
        }

        printf ("\n Columns: ");
/* Transform the columns */
        tmp = (float *)malloc (sizeof(float)*im-
>info->nr);
        for (j=0; j<im->info->nc; j++)
        {
          for (i=0; i<im->info->nr; i++)
            tmp[i] = result[i][j];
          fwt_1d (tmp, im->info->nr,
INVERSE);
          for (i=0; i<im->info->nr; i++)
            result[i][j] = tmp[i];
          NEXT();
        }
        printf ("\n");
        free (tmp);

/* Copy into the image */
        for (i=0; i<im->info->nr; i++)
          for (j=0; j<im->info->nc; j++)
            if (result[i][j] > 255.0) im->data[i][j]
= 255;
            else if (result[i][j] < 0) im->data[i][j]
= 0;
            else im->data[i][j] = (unsigned
char)result[i][j];
}
```

```
void wavelet (float *data, int N, int direction)
{
        Haar_2 (data, N, direction);
}

void Haar_2 (float *data, int N, int direction)
{
        int i,j,nover2;
        float h0, h1, *tmp;
        h0 = h1 = 0.5;
        tmp = (float *)malloc (sizeof(float)*N);

        nover2 = N/2;
        if (direction == FORWARD)
        {
          i=0;
          for (j=0; j<N-1; j+=2)
          {
            tmp[i]       = h0*data[j] +
h1*data[j+1];
            tmp[i+nover2] = h0*data[j] -
h1*data[j+1];
            i++;
          }
        } else if (direction == INVERSE)
        {
/* Comment the next line out for symmetric
scaling */
          h1 = h0 = 1.0;
          i = j = 0;
          do
          {
            tmp[j]   = h0*data[i] +
h1*data[i+nover2];
            tmp[j+1] = h0*data[i] -
h1*data[i+nover2];
            j += 2; i++;
          } while (i <= nover2);
        }
        for (i=0; i<N; i++)
          data[i] = tmp[i];
        free(tmp);
}
```

## /* LBG Vector Quantization */

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>

#define uc      unsigned char
#define N       256   /* number of levels
*/
#define k       16    /* block length
*/
#define e       0.001 /* distortion threshold
*/
#define n       4096 /* length of training
sequence   */
#define SizeImg  256 /* Image size =
256*256     */
#define SizeDim  4    /* Image block
=4*4(16 pixel) */
#define Infinite 9.99E+62

void  Initial(void);
double Find_D1(void);
double Distor(int,int);
void  Partition(void);
void  Update_codebook(void);
void  Display(void);
void  SaveCodebook(void);

uc huge s[SizeImg][SizeImg];
unsigned int  m,num[N];
static float huge x[n][k],y[N][k],updatey[N][k];
double D0,D1,errortest;
FILE *file1,*file2,*file3;

void main(void)
{
 clrscr();
 Initial();
 do{
  Partition();
  D1 = Find_D1();
  errortest = (D0-D1)/D1;
  printf("D0 = %f\n",D0);
  printf("D1 = %f\n",D1);
  printf("(D0-D1)/D1 = %f\n",errortest);
  if(errortest > e){
   Update_codebook();
  D0 =D1;
  }
  if(errortest <= e){
   printf("\n!!! TRAINING END !!!");
  }
  SaveCodebook();
 }while(errortest > e);
 SaveCodebook();
}

void Initial(void)
{
 int c,d,i,j,col,row;
 printf("\nOPEN FILE ...");
 file1 =
fopen("c:\\vq\\image\\tm2.img","rb");
 if(file1 == NULL){
  printf("\nOPEN FILE FAILURE ...!!!");
  exit(1);
 }
 printf("\nOK.");

 for(row=0;row<SizeImg;row++){
  fread(s[row],1,SizeImg,file1);
 }
 fclose(file1);
 printf("\nLOAD DATA OK.");

 c=0;
 for(row=0;row<SizeImg;row=row+SizeDim){
  for(col=0;col<SizeImg;col=col+SizeDim){
   d=0;
   for(i=0;i<SizeDim;i++){
        for(j=0;j<SizeDim;j++){
          x[c][d]=s[row+i][col+j];
          if(c<N){
            y[c][d]=x[c][d];
          }
```

```
            d++;
          }
      }
      c++;
    }
  }

  D0 = Infinite;
  m = 0;
  printf("\nm = 0\n");
  printf("Waiting...!!!");
}

double Find_D1(void)
{
  int i,j;
  double minD,minD_temp,sum_minD,result;

  sum_minD = 0.0;
  for(i=0;i<n;i++){          /* number of
sequence */
    minD = Infinite;
    for(j=0;j<N;j++){        /* number of
book code */
      minD_temp = Distor(i,j);
      if(minD_temp < minD )
          minD = minD_temp;
    }
    sum_minD = sum_minD + minD;
  }
  result = sum_minD/n;       /* number of
sequence */
  return(result);
}


double Distor(int indexx,int indexy)
{
  int i;
  double sum,ab;
  sum = 0.0;
  for(i=0;i<k;i++){
    ab = fabs((double)(x[indexx][i]-y[indexy][i]));
    sum = sum + (ab*ab)/k;
  }
  return(sum);
}


void Partition(void)
{
  int i,j,p,q,e2;
  double e0,e1;
  for(j=0;j<N;j++)
  {          /* number of codebook */
  num[j] = 0;
  for(q=0;q<k;q++)
    {  /*numberofblock length */
        updatey[j][q] = 0;
    }
    for(i=0;i<n;i++)
    {/*number of trainingsequence */
     e2 = 1;
     e0 = Distor(i,j);
     p  = 0;
     do{
      if(j != p){
       e1 = Distor(i,p);
       if(!(e0 <= e1))
         e2 = 0;
       if(e2 == 0)
         break;
      }
      p++;
    }while(p<N);   /* number of codebook */
    if(e2){
        for(q=0;q<k;q++)
        {   /* number of block length */
          if(q==0)
          {
            num[j]=num[j]+1;
          }
          updatey[j][q] = updatey[j][q] + x[i][q];
        }
    }
  }
 }
}
```

```
void Update_codebook(void)
{
  int i,j;
  for(i=0;i<N;i++){
    for(j=0;j<k;j++){
      if(num[i] != 0)
          y[i][j] = updatey[i][j]/num[i];
      if((num[i] == 0)&&(j== 0))
          printf("\nDevide By 0 at N =
%d\n",i);
    }
  }
  m++;
  printf("m = %d\n",m);
  printf("\nWaiting...!!!\n");
}

void Display(void)
{
  int i,j;
  for(i=0;i<k;i++){
    for(j=0;j<N;j++){
      printf("y[%d][%d] = %9f ",j,i,y[j][i]);
    }
  }
}

void SaveCodebook(void)
{
  int row;
  printf("\nOPEN FILE TO SAVE ...");
  file2 =
fopen("c:\\vq\\codebook\\cb_tm2.dat","wb");
  if(file2 == NULL){
    printf("\nOPEN FILE TO SAVE FAILURE
...!!!");
    exit(1);
  }
  printf("OK1. ");

  for(row=0;row<N;row++){
    fwrite(y[row],sizeof(float),k,file2);
  }
  fclose(file2);
  file3 =
fopen("c:\\vq\\codebook\\d1_tm2.dat","wb");
  if(file3 == NULL){
    printf("\nOPEN FILE TO SAVE FAILURE
...!!!");
    exit(1);
  }
  printf("OK2.\n");
  fwrite(&D1,sizeof(double),1,file3);
  fclose(file3);
}
```

```
/* VQ Encoder */

#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <malloc.h>
#include <dos.h>
#define  uc        unsigned char
#define  e  0.001/*distortion threshold*/

#define  N  256 /*number of levels    */

int vectorlength=256; /* block length  */
int numvector=1;
int SizeImg=16; /* Image size = 512*512 */
int SizeDim=16;/*Imageblock=4*4(16pixel)*/

#define  Infinite  3.4E+38

void   MemAlloc(void);
void   Load_Codebook(void);
void   Find_Ri(void);
void   Sort_Ri(void);
void   Find_D(void);
void   Load_Image(void);
void   Find_Index_Image(void);
void   Load_Vector(void);
void   Find_Index_Vector(void);

char
namecodebk[40],nameimgin[40],name_idx_out[4
0],name4[40],name_org_out[40];
char name_con_out[40],name_gidx_out[40];
uc *vector;
uc  **x;
float  **dij,**ri,**y;
FILE *file1,*file2,*file3,*file4;

void main(int argc,char *argv[])
{
int assmode;

 if(argc!=10)
    {
    printf("mode 0) image data that have sub
image");
    printf("\nEx.");
    printf("\n%s  <in_codebook> <in_image>
<out_index> <lead_out_grp_idx> <out_org>
<out_con> <mode> <SizeDIM>
<SizeImg>\n",argv[0]);
    printf("\nif you have single sub image in
your image SizeDIM==SizeImg\n");

    printf("mode 1) vector text data");
    printf("\nEx.");
    printf("\n%s  <in_codebook> <in_vector>
<out_index> <lead_out_grp_idx> <out_org>
<out_con> <mode> <vectorlength>
<numvector>\n",argv[0]);
    printf("\n now numvector not used in
program");

    exit(1);
    }

  sprintf(namecodebk,"%s",argv[1]);
  sprintf(nameimgin,"%s",argv[2]);
  sprintf(name_idx_out,"%s",argv[3]);
  sprintf(name_gidx_out,"%s",argv[4]);

sprintf(name_org_out,"%s",argv[5]);//out_index
sprintf(name_con_out,"%s",argv[6]);//out_index
  assmode=atoi(argv[7]);
  if(assmode<0 || assmode>1)
    {
    printf("Please select the assign mode0or1");
    exit(1);
    }

  if(assmode==0)//image
  {
    SizeDim=atoi(argv[8]);//sub image size
    vectorlength=SizeDim*SizeDim;

    SizeImg=atoi(argv[9]);
```

```
    numvector=SizeImg*SizeImg/vectorlength;
  }
  else if(assmode==1)//vector
  {
    vectorlength=atoi(argv[8]);
    SizeDim=1;//(int)sqrt(vectorlength);
    numvector=atoi(argv[9]);
    SizeImg=1;//16;//not used
  }

  vector=(uc*)calloc(vectorlength,sizeof(uc));
  if(!vector)
    {
    printf("\nAlloc buffer name [(uc*)vector]
error");
    exit(1);
    }

  MemAlloc();
  Load_Codebook();
  if(assmode==0)
    {
    Load_Image();
    Find_Index_Image();
    }
  else if(assmode==1)
    {
    Load_Vector();
    Find_Index_Vector();
    }
  free(vector);
  free((void **)x);
  free((void **)dij);
  free((void **)ri);
  free((void **)y);
  printf("\nEND ....!!!");
}

void MemAlloc()
{
  unsigned int i;
  if(((y = (float **) malloc(sizeof(float
*)*N))==NULL)){
    printf("memory allocation error...!!!");
  }
  for(i=0;i<N;i++)
    if(((y[i] = (float *)
calloc(vectorlength,sizeof(y)))==NULL)){
      printf("memory allocation error...y[%d]",i);
      exit(1);
    }

  if(((ri = (float **) malloc(sizeof(float
*)*N))==NULL)){
    printf("memory allocation error...!!!");
  }
  for(i=0;i<N;i++)
    if(((ri[i] = (float *)
calloc(2,sizeof(ri)))==NULL)){
      printf("memory allocation error...ri[%d]",i);
      exit(1);
    }
  if(((x = (uc **) malloc(sizeof(uc
*)*SizeImg))==NULL)){
    printf("memory allocation error...!!!");
  }
  for(i=0;i<SizeImg;i++)
    if(((x[i] = (uc *)
calloc(SizeImg,sizeof(x)))==NULL)){
      printf("memory allocation error...dij[%d]",i);
      exit(1);
    }
}
void Load_Codebook(void)
{
  int row;
  if((file1 = fopen(namecodebk,"rb")) == NULL)
  {
    printf("\nOPEN FILE FAILURE ...!!!");
    exit(1);
  }
  for(row=0;row<N;row++)
    fread(y[row],sizeof(float),vectorlength,file1);
  fclose(file1);
  Find_Ri();
  Sort_Ri();
}
```

```
void Find_Ri(void)
{
  unsigned int  i,j,index;

  for(index=0;index<N;index++){
    ri[index][0]=index;
    ri[index][1]=0;
    for(i=0;i<vectorlength;i++){
      ri[index][1] = ri[index][1] +
(y[index][i]*y[index][i]);
    }
    ri[index][1]= sqrt(ri[index][1]);
   }
}

void Sort_Ri(void)
{
  float temp0,temp1;
  unsigned int i,j;

  for(i=0;i<(N-1);i++){
    for(j=0;j<(N-1-i);j++){
      if(ri[j][1]>ri[j+1][1]){
            temp0 = ri[j][0];
            temp1 = ri[j][1];
            ri[j][0] = ri[j+1][0];
            ri[j][1] = ri[j+1][1];
            ri[j+1][0] = temp0;
            ri[j+1][1] = temp1;
      }
    }
  }
}

void Find_D()
{
  unsigned int  i,j,index;
  for(i=0;i<N;i++)
    for(j=0;j<N;j++){
      dij[i][j]=0;
      if(i != j){
          for(index=0;index<vectorlength;index+
+){
            dij[i][j] = dij[i][j] + pow((y[i][index]-
y[j][index]),2);
          }
          dij[i][j]= sqrt(dij[i][j]);
      }
    }
}

void Load_Vector(void)//Text file
{
FILE *file1;
unsigned int d;
short int temp;
  if((file1 = fopen(nameimgin,"rt")) == NULL)
  {
    printf("\nOpen VECTOR file name [%s]
failure ...!!!",nameimgin);
    exit(1);
  }
  for(d=0;d<vectorlength;d++)
    {
    fscanf(file1,"%d ",&temp);
    if(temp>255) temp=255;
    if(temp<0)   temp=0;
    vector[d]=temp;
    printf("%d ",vector[d]);
    }
  fclose(file1);
}

void Load_Image(void)
{
  unsigned int row;
  if((file2 = fopen(nameimgin,"rb")) == NULL)
  {
    printf("\nOPEN FILE FAILURE ...!!!");
    exit(1);
  }
  for(row=0;row<SizeImg;row++)
    fread(x[row],sizeof(uc),SizeImg,file2);
  fclose(file2);
}
```

```
void Find_Index_Image(void)
{
  uc out_index;
  unsigned int row,col,i,j,d,c,index;
  float Rx,hi,distor;
  float min_distor;

 /* OPEN INDEX TEXT FILE */
  printf("\nOPEN INDEX TEXT FILE .....");
  file3 = fopen(name_idx_out,"wt");
  if(file3 == NULL){
  printf("\nOPEN FILE FAILURE ...!!!");
    exit(1);
  }
  printf("OK.");
  printf("\nCOMPRESSING.....!!!\n");

  c = 1;
  for(row=0;row<SizeImg;row=row+SizeDim)
  {
    for(col=0;col<SizeImg;col=col+SizeDim)
      {
      /* RESHAPE BLOCKS IMAGE(4*4) TO
VECTORS IMAGE(1*16) */
      d=0;
      for(i=0;i<SizeDim;i++)
      {
          for(j=0;j<SizeDim;j++)
          {
            vector[d]=x[row+i][col+j];
            d++;
          }
      }

      /* Compute Rx */
      Rx = 0;
      for(i=0;i<vectorlength;i++){
          Rx = Rx + ((float)vector[i]*vector[i]);
      }
      Rx = sqrt(Rx);

      /* Identify Ci*/
      for(index = 0; index < N; index++){
          if(Rx<ri[index][1]){
              break;
          }
      }
      if(index != 0)
          if(abs(Rx-ri[index-1][1])<abs(Rx-
ri[index][1])){
              index = index-1;
          }

      /* Compute hi */
      index = ri[index][0];
      hi = 0;
      for(i=0;i<vectorlength;i++){
          hi = hi + (y[index][i]-
vector[i])*(y[index][i]-vector[i]);
      }
      hi = sqrt(hi);

      /* Find Output Index */
      min_distor = Infinite;
      for(i=0;i<N;i++){
          if((ri[i][1]>=(Rx-hi)) &&
(ri[i][1]<=(Rx+hi))){
              index  = ri[i][0];
              distor = 0;
              for(j=0;j<vectorlength;j++){
                  distor = distor + (y[index][j]-
vector[j])*(y[index][j]-vector[j]);
              }
              distor = sqrt(distor);
              if(distor<min_distor){
                min_distor = distor;
                out_index = index;
              }
          }
      }
      printf("\r%5d",c);
      c++;
      printf("\nCode=%d",out_index);
      fprintf(file3,"%d",out_index);
    }
  }
```

```
  /* OPEN GROUP INDEX FILE */
  sprintf(name4,"grp%d.idx",out_index);
  printf("\nOPEN GROUP INDEX FILE .....");
  file4 = fopen(name4,"at");
  if(file4 == NULL)
  {
    printf("\nOPEN FILE FAILURE ...!!!");
    exit(1);
  }
  fprintf(file4,"\n%s",nameimgin);
  fclose(file4);

  /* OPEN GROUP INDEX FILE */
  sprintf(name4,"grpo%d.idx",out_index);
  printf("\nOPEN GROUP INDEX FILE .....");
  file4 = fopen(name4,"at");
  if(file4 == NULL)
  {
    printf("\nOPEN FILE FAILURE ...!!!");
    exit(1);
  }
  fprintf(file4,"\n%s",name_org_out);
  fclose(file4);

  /* OPEN GROUP INDEX FILE */
  sprintf(name4,"grpc%d.idx",out_index);
  printf("\nOPEN GROUP INDEX FILE .....");
  file4 = fopen(name4,"at");
  if(file4 == NULL)
  {
    printf("\nOPEN FILE FAILURE ...!!!");
    exit(1);
  }
  fprintf(file4,"\n%s",name_con_out);
  fclose(file4);

  fclose(file3);
}

void Find_Index_Vector(void)
{
  uc out_index;
  unsigned int row,col,i,j,d,index;
  float Rx,hi,distor;
  float min_distor;

  printf("\nOPEN OUTPUT INDEX TEXT
FILE.....");
  if((file3 = fopen(name_idx_out,"wt")) ==
NULL)
    {
    printf("\nOPEN FILE FAILURE ...!!!");
    exit(1);
    }
    /* RESHAPE BLOCKS IMAGE(4*4) TO
VECTORS IMAGE(1*16) */
    /* Compute Rx */
    Rx = 0;
    for(i=0;i<vectorlength;i++){
        Rx = Rx + ((float)vector[i]*vector[i]);
    }
    Rx = sqrt(Rx);
    /* Identify Ci*/
    for(index = 0; index < N; index++){
      if(Rx<ri[index][1]){
        break;
      }
    }
    if(index != 0)
      if(abs(Rx-ri[index-1][1])<abs(Rx-
ri[index][1])){
        index = index-1;
      }

    /* Compute hi */
    index = ri[index][0];
    hi = 0;
    for(i=0;i<vectorlength;i++){
        //hi = hi + pow((y[index][i]-
vector[i]),2);
        hi = hi + (y[index][i]-
vector[i])*(y[index][i]-vector[i]);
    }
    hi = sqrt(hi);

    /* Find Output Index */
    min_distor = Infinite;
    for(i=0;i<N;i++){
```

```
        if((ri[i][1]>=(Rx-hi)) &&
(ri[i][1]<=(Rx+hi))){
          index  = ri[i][0];
          distor = 0;
          for(j=0;j<vectorlength;j++){
            distor = distor + (y[index][j]-
vector[j])*(y[index][j]-vector[j]);

          }
          distor = sqrt(distor);
          if(distor<min_distor){
            min_distor = distor;
            out_index = index;
          }
        }
    }
  printf("\nCode=%d",out_index);
  fprintf(file3,"%d",out_index);
sprintf(name4,"%s%d.idx",name_gidx_out,out_i
ndex);
  printf("\nOPEN GROUP INDEX FILE .....");
  if((file4 = fopen(name4,"at")) == NULL)
  {
    printf("\nOPEN FILE FAILURE ...!!!");
    exit(1);
  }
  fprintf(file4,"\n%s",nameimgin);
fclose(file4);

  /* OPEN GROUP INDEX FILE */

sprintf(name4,"%so%d.idx",name_gidx_out,out_
index);
  printf("\nOPEN GROUP INDEX FILE .....");
  file4 = fopen(name4,"at");
  if(file4 == NULL)
  {
    printf("\nOPEN FILE FAILURE ...!!!");
    exit(1);
  }
  fprintf(file4,"\n%s",name_org_out);
  fclose(file4);

  /* OPEN GROUP INDEX FILE */
sprintf(name4,"%sc%d.idx",name_gidx_out,out_i
ndex);
  printf("\nOPEN GROUP INDEX FILE .....");
  file4 = fopen(name4,"at");
  if(file4 == NULL)
  {
    printf("\nOPEN FILE FAILURE ...!!!");
    exit(1);
  }
  fprintf(file4,"\n%s",name_con_out);
  fclose(file4);

  fclose(file3);
}
```

```
/*Neural Network*/
#include "layer.cpp"
unsigned char *addr = (unsigned char
*)0xa0000;
#define MAXNAME   40
FILE *mtfile;
float mtbuf;

void main(int argc,char *argv[])
{

    union REGS r;//medit
    short int ppcolor;//mmedit
    char TRAINING_FILE[MAXNAME];
    char WEIGHTS_FILE[MAXNAME];
    char OUTPUT_FILE[MAXNAME];
    int  ind;
    double error_tolerance=0.1;
    double total_error=0.0;
    double avg_error_per_cycle=0.0;
    double error_last_cycle=0.0;
    double avgerr_per_pattern=0.0; // for the
latest cycle
    double error_last_pattern=0.0;
    double learning_parameter=0.02;
    double alpha; // momentum parameter
    double NF; // noise factor
    double new_NF;

    unsigned temp, startup, start_weights;
    long int vectors_in_buffer;
    long int max_cycles;
    long int patterns_per_cycle=0;
    long int total_cycles, total_patterns;
    int  i;

// create a network object
    network backp;

    FILE * training_file_ptr, * weights_file_ptr, *
output_file_ptr;
    FILE * test_file_ptr, * data_file_ptr;
// open output file for writing
    strcpy(TRAINING_FILE,"training.dat");
    strcpy(WEIGHTS_FILE,"weights.dat");
    strcpy(OUTPUT_FILE, "output.dat");
    strcpy(TEST_FILE,   "test.dat");
// all switch
// -l learning phase
// -t testing  phase
// -w weight file's name                                /
// -o output file's name
// -i input file's name
// -L total layer
// -0 number of processing unit in input layer
// -1 number of processing unit in 1'st hidden
layer
// -2 number of processing unit in 2'nd hidden
layer
// -3 number of processing unit in 3'rd hidden
layer
// -4 number of processing unit in 4'th hidden
layer
// -e total epoach
// -a learning rate
// -b momentum
// -E Minimum Error
// -f noise factor
    if(argc>1)
     {
        /*** Scan command line ***/
       for (ind = 1; ind < argc; ind++) {
          /*** Parse switches ***/
          if (argv[ind][0] == '-') {
                switch (argv[ind][1]) {
                case 'l':
strcpy(TRAINING_FILE, argv[++ind]);
                        temp = 1;
                        //start_weights =
atoi(argv[++ind]);
                start_weights = 0;
                        if(
!(start_weights==0 | | start_weights==1))
                            exit(0);
                        if
((training_file_ptr=fopen(TRAINING_FILE,"r"))=
=NULL)
                        {
```

```
                    cout <<
"problem opening training file\n";
                    exit(1);
                }

data_file_ptr=training_file_ptr; // training on
                break;
            case 't': strcpy(TEST_FILE,
argv[++ind]);
                temp = 0;
                start_weights = 1;
                if
((test_file_ptr=fopen(TEST_FILE,"r"))==NULL)
                {
                    cout <<
"problem opening test file\n";
                    exit(1);
                }

data_file_ptr=test_file_ptr; // training off
                break;
            case 'w':
strcpy(WEIGHTS_FILE, argv[++ind]);
                break;
            case 'o':
strcpy(OUTPUT_FILE, argv[++ind]);
                break;
            /*
            case 'e':
max_cycles=atoi(argv[++ind]);
                break;
            case 'L': ++ind;

backp.set_num_of_layer(atoi(argv[ind]));
                break;
            */
            case '0': ++ind;

backp.set_layer_size(0,atoi(argv[ind]));
                break;
            case '1': ++ind;

backp.set_layer_size(1,atoi(argv[ind]));
                break;
            case '2': ++ind;

backp.set_layer_size(2,atoi(argv[ind]));
                break;
            case '3': ++ind;

backp.set_layer_size(3,atoi(argv[ind]));
                break;
            case '4': ++ind;

backp.set_layer_size(4,atoi(argv[ind]));
                break;
            case 'a': learning_parameter
= atof(argv[++ind]);
                break;
            /*
            case 'b': alpha =
atof(argv[++ind]);
                break;
            case 'E': error_tolerance =
atof(argv[++ind]);
                break;
            case 'f': NF =
atof(argv[++ind]);
                break;
            */
            default : printf("Unknown
switch '%c'\n", argv[ind][1]);
                break;
            }
        }
    }
    max_cycles=20000;
    backp.set_num_of_layer(3);
    alpha = 0;
    error_tolerance = 0.01;
    NF = 0.0;

    if
((output_file_ptr=fopen(OUTPUT_FILE,"w"))==
NULL)
    {
        cout << "problem opening output
file\n";
```

```
            exit(1);
        }
        backp.set_training(temp);
    }   /* end of setting by parameter passing */
    else {
      if
((output_file_ptr=fopen(OUTPUT_FILE,"w"))==
NULL)
      {
            cout << "problem opening output
file\n";
            exit(1);
      }

// enter the training mode : 1=training on
0=training off
      cout << "-------------------------\n";
      cout << " Backpropagation simulator \n";
      cout << "-------------------------\n";
      cout << "Please enter 1 for TRAINING on,
or 0 for off: \n\n";
      cout << "Use training to change weights
according to your\n";
      cout << "expected outputs. Your training.dat
file should contain\n";
      cout << "a set of inputs and expected
outputs. The number of\n";
      cout << "inputs determines the size of the
first (input) layer\n";
      cout << "while the number of outputs
determines the size of the\n";
      cout << "last (output) layer :\n\n";

      cin >> temp;
      backp.set_training(temp);

      if (backp.get_training_value() == 1)
      {
            cout << "--> Training mode is *ON*.
weights will be saved\n";
            cout << "in the file weights.dat at the
end of the\n";
            cout << "current set of input (training)
data\n";
      }
      else
      {
            cout << "--> Training mode is *OFF*.
weights will be loaded\n";
            cout << "from the file weights.dat and
the current\n";
            cout << "(test) data set will be used.
For the test\n";
            cout << "data set, the test.dat file
should contain\n";
            cout << "only inputs, and no expected
outputs.\n";
      }
      if (backp.get_training_value()==1)
      {
            // ---------------------------
            //Read in values for the error_tolerance,
            //and the learning_parameter
            // ---------------------------
            cout << " Please enter in the
error_tolerance\n";
            cout << " --- between 0.001 to
100.0, try 0.1 to start --\n";
            cout << "\n";
            cout << "and the learning_parameter,
beta\n";
            cout << " --- between 0.01 to 1.0,
try 0.5 to start -- \n\n";
            cout << " separate entries by a
space\n";
            cout << " example: 0.1 0.5 sets
defaults mentioned :\n\n";

            cin >> error_tolerance >>
learning_parameter;
      // ---------------------------
      //      Read in values for the momentum
      //      parameter, alpha (0-1.0)
      //      and the noise factor, NF (0-1.0)
      // ---------------------------
            cout << "Enter values now for the
momentum \n";
            cout << "parameter, alpha(0-1.0)\n";
```

```
        cout << " and the noise factor, NF (0-
1.0)\n";
        cout << "You may enter zero for either
of these\n";
        cout << "parameters, to turn off the
momentum or\n";
        cout << "noise features.\n";
        cout << "If the noise feature is used, a
random\n";
        cout << "component of noise is added
to the inputs\n";
        cout << "This is decreased to 0 over
the maximum\n";
        cout << "number of cycles
specified.\n";
        cout << "enter alpha followed by NF,
e.g., 0.3 0.5\n";

        cin >> alpha >> NF;
//-----------------------------------
// open training file for reading
//-----------------------------------
        if
((training_file_ptr=fopen(TRAINING_FILE,"r"))=
=NULL)
        {
           cout << "problem opening training
file\n";
           exit(1);
        }
        data_file_ptr=training_file_ptr; //
training on

        // Read in the maximum number of
cycles
        // each pass through the input data
file is a cycle
        cout << "Please enter the maximum
cycles for the simulation\n";
        cout << "A cycle is one pass through
the data set.\n";
        cout << "Try a value of 10 to start
with\n";
        cin >> max_cycles;
        cout << "Do you want to read weights
from weights.dat to start?\n";
        cout << "Type 1 to read from file, 0 to
randomize starting weights\n";
        cin >> start_weights;
   } else {
        if
((test_file_ptr=fopen(TEST_FILE,"r"))==NULL)
        {
           cout << "problem opening
test file\n";
           exit(1);
        }
        data_file_ptr=test_file_ptr; // training
off
   }
    // get layer information
    backp.get_layer_info();
} /* end of setting from keyboard */

// training: continue looping until the total error
is less than
//          the tolerance specified, or the
maximum number of
//          cycles is exceeded; use both
the forward signal propagation
//          and the backward error
propagation phases. If the error
//          tolerance criteria is satisfied,
save the weights in a file.
// no training: just proceed through the input
data set once in the
//          forward signal propagation
phase only. Read the starting
//          weights from a file.
// in both cases report the outputs on the
screen

// intialize counters
  //medit
  ppcolor=2;
  r.w.ax = 0x0013;
  int386(0x10,&r,&r);
```

```
  mtfile = fopen("error.out","wb");
  if(mtfile == NULL)
  {
    printf("\nOPEN error.txt FILE TO SAVE
FAILURE ...!!!");
    exit(1);
  }
total_cycles=0; // a cycle is once through all
the input data
    total_patterns=0; // a pattern is one entry in
the input data
    new_NF=NF;
// set up the network connections
    backp.set_up_network();
// initialize the weights
    if
((backp.get_training_value()==1)&&(start_weigh
ts!=1))
    {
          // randomize weights for all layers;
there is no
          // weight matrix associated with the
input layer
          // weight file will be written after
processing

          backp.randomize_weights();
          // set up the noise factor value
          backp.set_NF(new_NF);
    } else {
          // read in the weight matrix defined by
a
          // prior run of the backpropagation
simulator
          // with training on
          if
((weights_file_ptr=fopen(WEIGHTS_FILE,"r"))=
=NULL)
          {
                cout << "problem opening
weights file\n";
                exit(1);
          }
          backp.read_weights(weights_file_ptr);
          fclose(weights_file_ptr);
    }
// main loop
// if training is on, keep going through the
input data
//                until the error is acceptable or
the maximum number of cycles
//                is exceeded.
// if training is off, go through the input data
once. report outputs
// with inputs to file output.dat

    startup=1;
    vectors_in_buffer = MAX_VECTORS; //
startup condition
    total_error = 0;

    //int totalVector =
backp.fill_IObuffer(data_file_ptr);
    while (  ((backp.get_training_value()==1)
          && (avgerr_per_pattern >
error_tolerance)
          && (total_cycles < max_cycles)
          && (vectors_in_buffer !=0))
          ||
((backp.get_training_value()==0)
          && (total_cycles < 1))
          ||
((backp.get_training_value()==1)
          && (startup==1)) )
    {
          startup=0;
          error_last_cycle=0; // reset for each
cycle
          patterns_per_cycle=0;
          backp.update_momentum(); //added
to reset momentum matrices each cycle
// process all the vectors in the datafile
// going through one buffer at a time
// pattern by pattern
        //int totalVector =
backp.fill_IObuffer(data_file_ptr);
          while
((vectors_in_buffer==MAX_VECTORS))
```

```
        {
vectors_in_buffer=backp.fill_IObuffer(data_file_ptr
); // fill buffer
            //vectors_in_buffer = totalVector;
        if (vectors_in_buffer < 0)
          {
                cout << "error in reading in
vectors, aborting\n";
                cout << "check that there are
on extra linefeeds\n";
                cout << "in your data file, and
that the number\n";
                cout << "of layers and size of
layers match the\n";
                cout << "the parameters
provided.\n";
                exit(1);
          }
        //cout << "\nvectors in buffer" <<
vectors_in_buffer;
// process vectors
          for (i=0; i<vectors_in_buffer; i++)
          {
            // get next pattern
                backp.set_up_pattern(i);
                total_patterns++;
                patterns_per_cycle++;
                // forward propagate
                backp.forward_prop();
                if
(backp.get_training_value()==0)

backp.write_outputs(output_file_ptr);

// back_propagate, if appropriate
                if
(backp.get_training_value()==1)
                {

backp.backward_prop(error_last_pattern);
                    error_last_cycle
+=error_last_pattern*error_last_pattern;
                    avgerr_per_pattern=
((double)sqrt((double)error_last_cycle/patterns_p
er_cycle));
                // if it's not the last cycle,
update weights
                    if ((avgerr_per_pattern >
error_tolerance)
                        && (total_cycles+1
< max_cycles))
                    backp.update_weights(learning_paramet
er,alpha);
                    // backp.list_weights(); // can
                    // see change in weights by
                    // using list_weights before and
                    // after back_propagation
                        }
                    }
                    error_last_pattern = 0;
                }
                total_error += error_last_cycle;
                total_cycles++;

// update NF
// gradually reduce noise to zero
                if (total_cycles>0.7*max_cycles)
                    new_NF = 0;
                else if (total_cycles>0.5*max_cycles)
                    new_NF = 0.25*NF;
                else if (total_cycles>0.3*max_cycles)
                    new_NF = 0.50*NF;
                else if (total_cycles>0.1*max_cycles)
                    new_NF = 0.75*NF;

                backp.set_NF(new_NF);

// most character displays are 25 lines
// user will see a corner display of the cycle
count
// as it changes
```

```
      //cout <<
"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n
\n\n\n\n\n\n";
         //cout << "\nArmote" << total_cycles
<< "\t" << avgerr_per_pattern ;
         {//medit
         short int ypos;
         ypos=(short int)total_cycles;
         if(ypos%200==0)
           ppcolor++;
         if(ypos>=200)
           ypos=ypos%200;
         if(ppcolor>15)
           ppcolor=2;
         if(kbhit())
          {
           r.w.ax = 0x0003;
           int386(0x10,&r,&r);
           goto pass;
          }
         memset(addr+(long)ypos*320,ppcolor,
(long)(avgerr_per_pattern*320));
         }

         mtbuf=(float)avgerr_per_pattern;
         fwrite(&mtbuf,sizeof(float),1,mtfile);
         fseek(data_file_ptr, 0L, SEEK_SET);
// reset the file pointer
// to the beginning of
// the file
vectors_in_buffer = MAX_VECTORS; // reset
   } // end main loop

pass:
  //medit
  getch();
  r.w.ax = 0x0003;
  int386(0x10,&r,&r);
  fclose(mtfile);
   if (backp.get_training_value()==1)
   {
         if
((weights_file_ptr=fopen(WEIGHTS_FILE,"w"))=
=NULL)
         {
   cout << "problem opening weights file\n";
          exit(1);
         }
   }
   cout << "\n\n";
   cout << "-----------------------\n";
   cout << "       done:    results in file
"<<OUTPUT_FILE<<"\n";
   cout << "                training: last vector
only\n";
   cout << "                not training: full
cycle\n\n";
   if (backp.get_training_value()==1)
   {
         backp.write_weights(weights_file_ptr);
         backp.write_outputs(output_file_ptr);
         avg_error_per_cycle=(double)sqrt((doub
le)total_error/total_cycles);
         error_last_cycle=(double)sqrt((double)er
ror_last_cycle);
         fclose(weights_file_ptr);
         cout << "weights saved in file
"<<WEIGHTS_FILE<<"\n";
         cout << "\n";
         cout << "---->average error per cycle
= " << avg_error_per_cycle << " <---\n";
         cout << "---->error last cycle = " <<
error_last_cycle << " <---\n";
         cout << "->error last cycle per
pattern= " << avgerr_per_pattern << " <---\n";
   }
   cout << "------------>total cycles = " <<
total_cycles << " <---\n";
   cout << "------------>total patterns = "
<< total_patterns << " <---\n";
   cout << "-----------------------\n";
// close all files
   fclose(data_file_ptr);
   fclose(output_file_ptr);
}
```

## /*Neural Network Function*/

```
// layer.cpp
// compile for floating point hardware if
available
#include <stdio.h>
#include <iostream.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>

#include <conio.h>
#include <malloc.h>
#include <mem.h>
#include <dos.h>

#include "layer.h"

char TEST_FILE[40];

// squashing function
// use sigmoid -- can customize to something
// else if desired; can add a bias term too
//
inline double squash(double input)
{
        return (double)(1.0/(1.0+exp(-
(double)input)));
}

// random number generator
// will return a doubleing point
// value between -1 and 1

inline double randomweight(unsigned init)
{
   int num;

   if (init==1)     // seed the generator
        srand ((unsigned)time(NULL));
   num=rand() % 100;
   return 2.0*(double(num/100.00))-1;
}

// the next function is needed for Turbo C++
// and Borland C++ to link in the appropriate
// functions for fscanf doubleing point formats:
static void force_fpf(void)
{
        double x, *y;
        y=&x;
        x=*y;
}

// ---------------------------------
//        input layer
//---------------------------------
input_layer::input_layer(int i, int o)
{
   num_inputs=i;
   num_outputs=o;

   outputs = new double[num_outputs];
   orig_outputs = new double[num_outputs];
   if ((outputs==0)||(orig_outputs==0))
   {
        cout << "not enough memory\n";
        cout << "choose a smaller
architecture\n";
        exit(1);
   }
   noise_factor=0;
}

input_layer::~input_layer()
{
 //delete [num_outputs] outputs;
 //delete [num_outputs] orig_outputs;
   delete [] outputs;
   delete [] orig_outputs;
}


void input_layer::calc_out()
{
 //add noise to inputs
 // randomweight returns a random number
```

```
// between -1 and 1
    int i;

    for (i=0; i<num_outputs; i++)
        outputs[i] =
orig_outputs[i]*(1+noise_factor*randomweight(0
));

}

void input_layer::set_NF(double noise_fact)
{
noise_factor=noise_fact;
}

// ---------------------------------
//          output layer
//----------------------------------
output_layer::output_layer(int ins, int outs)
{
    int i, j, k;

    num_inputs=ins;
    num_outputs=outs;
    weights = new
double[num_inputs*num_outputs];
    output_errors = new double[num_outputs];
    back_errors = new double[num_inputs];
    outputs = new double[num_outputs];
    expected_values = new double[num_outputs];
    cum_deltas = new
double[num_inputs*num_outputs];
    past_deltas = new
double[num_inputs*num_outputs];

    if
((weights==0)||(output_errors==0)||(back_er
rors==0)
        ||(outputs==0)||(expected_values==
0)
        ||(past_deltas==0)||(cum_deltas==0
))
    {
        cout <<"not enough memory\n";
        cout << "choose a smaller
architecture\n";
        exit(1);
    }
// zero cum_deltas and past_deltas matrix
    for (i=0; i< num_inputs; i++)
    {
        k=i*num_outputs;
        for (j=0; j< num_outputs; j++)
        {
            cum_deltas[k+j]=0;
            past_deltas[k+j]=0;
        }
    }
}

output_layer::~output_layer()
{
// some compilers may require the array
// size in the delete statement; those
// conforming to Ansi C++ will not
delete [] weights;
delete [] output_errors;
delete [] back_errors;
delete [] outputs;
delete [] past_deltas;
delete [] cum_deltas;

//delete [num_outputs*num_inputs] weights;
//delete [num_outputs] output_errors;
//delete [num_inputs] back_errors;
//delete [num_outputs] outputs;
//delete [num_outputs*num_inputs] past_deltas;
//delete [num_outputs*num_inputs] cum_deltas;

}

void output_layer::calc_out()
{
    int i,j,k;
    double accumulator=0.0;

    for (j=0; j<num_outputs; j++)
    {
```

```
        for (i=0; i<num_inputs; i++)
        {
                k=i*num_outputs;
        outputs[j]=weights[k+j]*inputs[i];
                accumulator+=outputs[j];
        }
        // use the sigmoid squash function
        outputs[j]=squash(accumulator);
        accumulator=0;
        }
}

void output_layer::calc_error(double & error)
{
    int i, j, k;
    double accumulator=0;
    double total_error=0;

    for (j=0; j<num_outputs; j++)
    {
        output_errors[j] = expected_values[j]-
outputs[j];
        total_error+=output_errors[j];
    }
    error=total_error;

    for (i=0; i<num_inputs; i++)
    {
        k=i*num_outputs;
        for (j=0; j<num_outputs; j++)
        {

        back_errors[i]=weights[k+j]*output_error
s[j];
                accumulator+=back_errors[i];
        }
        back_errors[i]=accumulator;
        accumulator=0;
        // now multiply by derivative of
        // sigmoid squashing function, which is
        // just the input*(1-input)
        back_errors[i]*=inputs[i]*(1.0-inputs[i]);
    }
}

void output_layer::randomize_weights()
{
    int i, j, k;
    const unsigned first_time=1;
    const unsigned not_first_time=0;
    double discard;

    discard=randomweight(first_time);

    for (i=0; i< num_inputs; i++)
    {
        k=i*num_outputs;
        for (j=0; j< num_outputs; j++)

        weights[k+j]=randomweight(not_first_ti
me);
    }
}
void output_layer::update_weights(const double
beta,const double alpha)
{
    int i, j, k;
    double delta;
// learning law: weight_change =
//              beta*output_error*input +
alpha*past_delta
    for (i=0; i< num_inputs; i++)
    {
        k=i*num_outputs;
        for (j=0; j< num_outputs; j++)
        {
        delta=beta*output_errors[j]*inputs[i]
                +alpha*past_deltas[k+j];
        weights[k+j] += delta;
        cum_deltas[k+j]+=delta; //current cycle
        }
    }
}

void output_layer::update_momentum()
{
// This function is called when a
```

```
// new cycle begins; the past_deltas
// pointer is swapped with the
// cum_deltas pointer. Then the contents
// pointed to by the cum_deltas pointer
// is zeroed out.
   int i, j, k;
   double * temp;

// swap
   temp = past_deltas;
   past_deltas=cum_deltas;
   cum_deltas=temp;

// zero cum_deltas matrix
// for new cycle
   for (i=0; i< num_inputs; i++)
   {
         k=i*num_outputs;
         for (j=0; j< num_outputs; j++)
            cum_deltas[k+j]=0;
   }
}

void output_layer::list_weights()
{
   int i, j, k;

   for (i=0; i< num_inputs; i++)
   {
         k=i*num_outputs;
         for (j=0; j< num_outputs; j++)
            cout << "weight["<<i<<","<<j<<"] is:
"<<weights[k+j];
   }
}

void output_layer::list_errors()
{
   int i, j;

   for (i=0; i< num_inputs; i++)
         cout << "backerror["<<i<<"] is :
"<<back_errors[i]<<"\n";
   for (j=0; j< num_outputs; j++)
         cout << "outputerrors["<<j<<"] is:
"<<output_errors[j]<<"\n";

}

void output_layer::write_weights(int
layer_no,FILE * weights_file_ptr)
{
   int i, j, k;

// assume file is already open and ready for
// writing

// prepend the layer_no to all lines of data
// format:
//       layer_no weight[0,0] weight[0,1] ...
//       layer_no weight[1,0] weight[1,1] ...
//          ...

   for (i=0; i< num_inputs; i++)
   {
         fprintf(weights_file_ptr,"%i ",layer_no);
         k=i*num_outputs;
         for (j=0; j< num_outputs; j++)
         {
            fprintf(weights_file_ptr,"%lf
",weights[k+j]);
         }
         fprintf(weights_file_ptr,"\n");
   }
}

void output_layer::read_weights(int layer_no,FILE
* weights_file_ptr)
{
   int i, j, k;
// assume file is already open and ready for
// reading

// look for the prepended layer_no
// format:
//       layer_no weight[0,0] weight[0,1] ...
//       layer_no weight[1,0] weight[1,1] ...
```

```
//                    ...
   while (1)
   {
         fscanf(weights_file_ptr,"%i",&j);
         if ((j==layer_no)||
(feof(weights_file_ptr)))
            break;
         else
         {
         while (fgetc(weights_file_ptr) != '\n')
                  {;}// get rest of line
         }
   }
   if (!(feof(weights_file_ptr)))
   {
         // continue getting first line
         i=0;
         for (j=0; j< num_outputs; j++)
         {

fscanf(weights_file_ptr,"%lf",&weights[j]); //
i*num_outputs = 0
         }
         fscanf(weights_file_ptr,"\n");
         // now get the other lines
         for (i=1; i< num_inputs; i++)
         {

fscanf(weights_file_ptr,"%i",&layer_no);
            k=i*num_outputs;
            for (j=0; j< num_outputs; j++)
            {

         fscanf(weights_file_ptr,"%lf",&weights[k
+j]);
            }
         }
         fscanf(weights_file_ptr,"\n");
   }
   else cout << "end of file reached\n";
}
```

```
void output_layer::list_outputs()
{
   int j;

   for (j=0; j< num_outputs;j++)
   {
         cout << "outputs[" <<j<<"] is:
"<<outputs[j]<<"\n";
   }
}

// ---------------------------------
//        middle layer
//---------------------------------

middle_layer::middle_layer(int i, int
o):output_layer(i,o)
{
}

middle_layer::~middle_layer()
{
delete [] weights;
delete [] output_errors;
delete [] back_errors;
delete [] outputs;

//delete [num_outputs*num_inputs] weights;
//delete [num_outputs] output_errors;
//delete [num_inputs] back_errors;
//delete [num_outputs] outputs;
}

void middle_layer::calc_error()
{
   int i, j, k;
   double accumulator=0;

   for (i=0; i<num_inputs; i++)
   {
         k=i*num_outputs;
         for (j=0; j<num_outputs; j++)
         {
```

```
back_errors[i]=weights[k+j]*(*(output_errors+j));
                accumulator+=back_errors[i];
            }
            back_errors[i]=accumulator;
            accumulator=0;
            // now multiply by derivative of
            // sigmoid squashing function, which is
            // just the input*(1-input)
            back_errors[i]*=(inputs[i]*(1.0-
inputs[i]));
        }
}

network::network()
{
    position=0L;
}

network::~network()
{
 delete []buffer;
}

void network::set_training(const unsigned &
value)
{
    training=value;
}

unsigned network::get_training_value()
{
    return training;
}

void network::get_layer_info()
{
    int i;
//-----------------------------------
//        Get layer sizes for the network
// ----------------------------------
cout << " Please enter in the number of layers
for your network.\n";
cout << " You can have a minimum of 3 to a
maximum of 5. \n";
cout << " 3 implies 1 hidden layer; 5 implies 3
hidden layers : \n\n";

cin >> number_of_layers;

cout << " Enter in the layer sizes separated by
spaces.\n";
cout << " For a network with 3 neurons in the
input layer,\n";
cout << " 2 neurons in a hidden layer, and 4
neurons in the\n";
cout << " output layer, you would enter: 3 2 4
.\n";
cout << " You can have up to 3 hidden
layers,for five maximum entries :\n\n";

    for (i=0; i<number_of_layers; i++)
    {
        cin >> layer_size[i];
    }

// ----------------------------------
// size of layers:
//input_layer     layer_size[0]
//output_layer    layer_size[number_of_layers-1]
//middle_layers   layer_size[1]
//optional: layer_size[number_of_layers-3]
//optional: layer_size[number_of_layers-2]
//----------------------------------
}

void network::set_up_network()
{
    int i,j,k;
//-----------------------------------
// Construct the layers
//-----------------------------------
    layer_ptr[0] = new
input_layer(0,layer_size[0]);

    for (i=0;i<(number_of_layers-1);i++)
    {
```

```
        layer_ptr[i+1] = new
middle_layer(layer_size[i],layer_size[i+1]);
    }
    layer_ptr[number_of_layers-1] = new
 output_layer(layer_size[number_of_layers-
2],layer_size[number_of_layers-1]);

    for (i=0;i<(number_of_layers-1);i++)
    {
        if (layer_ptr[i] == 0)
        {
            cout << "insufficient memory\n";
            cout << "use a smaller
architecture\n";
            exit(1);
        }
    }
//-------------------------------------
// Connect the layers
//-------------------------------------
// set inputs to previous layer outputs for all
//layers,
//except the input layer
  for (i=1; i< number_of_layers; i++)
        layer_ptr[i]->inputs = layer_ptr[i-1]-
>outputs;
// for back_propagation, set output_errors to
//next layer
//back_errors for all layers except the output
//layer and input layer
    for (i=1; i< number_of_layers -1; i++)
        ((output_layer *)layer_ptr[i])-
>output_errors =
                    ((output_layer
*)layer_ptr[i+1])->back_errors;

// define the IObuffer that caches data from
// the datafile
    i=layer_ptr[0]->num_outputs;// inputs
    j=layer_ptr[number_of_layers-1]-
>num_outputs; //outputs
    k=MAX_VECTORS;
    buffer=new double[(i+j)*k];
    if (buffer==0)
    {
        cout << "insufficient memory for
buffer\n";
        exit(1);
    }
}

void network::randomize_weights()
{
    int i;

    for (i=1; i<number_of_layers; i++)
        ((output_layer *)layer_ptr[i])-
>randomize_weights();
}

void network::update_weights(const double beta,
const double alpha)
{
    int i;

    for (i=1; i<number_of_layers; i++)
        ((output_layer *)layer_ptr[i])-
>update_weights(beta,alpha);
}

void network::update_momentum()
{
    int i;

    for (i=1; i<number_of_layers; i++)
        ((output_layer *)layer_ptr[i])-
>update_momentum();
}

void network::write_weights(FILE *
weights_file_ptr)
{
    int i;
    for (i=1; i<number_of_layers; i++)
        ((output_layer *)layer_ptr[i])-
>write_weights(i,weights_file_ptr);
}
```

```
void network::read_weights(FILE *
weights_file_ptr)
{
    int i;

    for (i=1; i<number_of_layers; i++)
        ((output_layer *)layer_ptr[i])-
>read_weights(i,weights_file_ptr);
}

void network::list_weights()
{
    int i;

    for (i=1; i<number_of_layers; i++)
    {
        cout << "layer number : " <<i<< "\n";
        ((output_layer *)layer_ptr[i])-
>list_weights();
    }
}

void network::list_outputs()
{
    int i;

    for (i=1; i<number_of_layers; i++)
    {
        cout << "layer number : " <<i<< "\n";
        ((output_layer *)layer_ptr[i])-
>list_outputs();
    }
}

void network::write_outputs(FILE *outfile)
{
    int i, ins, outs, outbit[8],reject;

    ins=layer_ptr[0]->num_outputs;
    outs=layer_ptr[number_of_layers-1]-
>num_outputs;
    double temp;
    fprintf(outfile,"for input vector:\n");
    for (i=0; i<ins; i++)
    {
        temp=layer_ptr[0]->outputs[i];
        fprintf(outfile,"%lf  ",temp);
    }
    fprintf(outfile,"\noutput vector is:\n");

    for (i=0; i<outs; i++)
    {
        temp=layer_ptr[number_of_layers-1]-
>outputs[i];
        fprintf(outfile,"%lf  ",temp);
    }
    reject=0;
    for (i=0; i<outs; i++)
    {
        temp=layer_ptr[number_of_layers-1]-
>outputs[i];
        if(temp>=0.9)//medit
        {
        temp=1.0;
        }
        else if (temp<=0.3)
        {
        temp=0.0;
        }
        else
        {
        temp=0.0;
        reject=1;
        }
        if(i<8)
          outbit[i]=(int)temp;
        fprintf(outfile,"%lf  ",temp);
    }
    if(reject==1)
     {
      for (i=0; i<outs; i++)
          {
          outbit[i]=1;
          }
     }
    //SINGLE RESULT FILE
    FILE *fh;
    int temp1=0;
```

```
    char *n;
    char str[40],str1[40];
    sprintf(str,"%s",TEST_FILE);
    n=strchr(str,'.');
    strcpy(n,".res");
    strcpy(str1,str);
    if((fh=fopen(str1,"wt"))==NULL)
{printf("can't write"); exit(1);}
    for(i=0; i<8; i++)
        temp1+=((outbit[i])<<(7-i));
    fprintf(fh,"%d\n",temp1);
    fclose(fh);
    }
    {//GROUP RESULT FILE
    FILE *fh;
    int temp1=0;
    if((fh=fopen("outcode.res","at"))==NULL)
{printf("can't write"); exit(1);}
    for(i=0; i<8; i++)
        temp1+=((outbit[i])<<(7-i));
    fprintf(fh,"%d\n",temp1);
    fclose(fh);
    }

    if (training==1)
    {
        fprintf(outfile,"\nexpected output
vector is:\n");
        for (i=0; i<outs; i++)
        {
            temp=((output_layer
*)(layer_ptr[number_of_layers-1]))->
            expected_values[i];
            fprintf(outfile,"%lf  ",temp);
        }
    }
    fprintf(outfile,"\n-----------------\n");
}

void network::list_errors()
{
    int i;

    for (i=1; i<number_of_layers; i++)
    {
        cout << "layer number : " <<i<< "\n";
        ((output_layer *)layer_ptr[i])-
>list_errors();
    }
}

int network::fill_IObuffer(FILE * inputfile)
{
// this routine fills memory with
// an array of input, output vectors
// up to a maximum capacity of
// MAX_INPUT_VECTORS_IN_ARRAY
// the return value is the number of read
// vectors
    int i, k, count, veclength;
    int ins, outs;
    ins=layer_ptr[0]->num_outputs;
    outs=layer_ptr[number_of_layers-1]-
>num_outputs;
    if (training==1)
        veclength=ins+outs;
    else
        veclength=ins;
    count=0;
while((count<MAX_VECTORS)&&(!feof(inputfil
e)))
    {
        k=count*(veclength);
        for (i=0; i<veclength; i++)
        {
        fscanf(inputfile,"%lf",&buffer[k+i]);
        }
        fscanf(inputfile,"\n");
        count++;
    }
    if (!(ferror(inputfile))) return count;
    else return -1; // error condition
}

void network::set_up_pattern(int buffer_index)
{
// read one vector into the network
    int i, k;
```

```
    int ins, outs;

    ins=layer_ptr[0]->num_outputs;
    outs=layer_ptr[number_of_layers-1]-
>num_outputs;
    if (training==1)
        k=buffer_index*(ins+outs);
    else
        k=buffer_index*ins;
    for (i=0; i<ins; i++)
    {
        ((input_layer*)layer_ptr[0])-
>orig_outputs[i]=buffer[k+i]/255.0;
    }
    if (training==1)
    {
      for (i=0; i<outs; i++)
          ((output_layer
*)layer_ptr[number_of_layers-1])-
>expected_values[i]
          = buffer[k+i+ins];
    }
}

void network::forward_prop()
{
    int i;

    for (i=0; i<number_of_layers; i++)
    {
        layer_ptr[i]->calc_out(); //polymorphic
function
    }
}

void network::backward_prop(double & toterror)
{
    int i;

// error for the output layer
    ((output_layer*)layer_ptr[number_of_layers-
1])->calc_error(toterror);
// error for the middle layer(s)
    for (i=number_of_layers-2; i>0; i--)
        ((middle_layer*)layer_ptr[i])-
>calc_error();
}

void network::set_NF(double noise_fact)
{
    ((input_layer*)layer_ptr[0])-
>set_NF(noise_fact);
}
void network::set_layer_size(int ind,int size)
{
    if( size > 0)
        layer_size[ind] = size;
    else
        exit(0);
}
void network::set_num_of_layer(int n)
{
    if( 3<=n && n<=5 )
        number_of_layers = n;
    else
        exit(0);
}
```

**/*Template Matching**

**The First 5 Best Match*/**

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <stdlib.h>
#include <mem.h>
#include <string.h>
#include <time.h>
#include <math.h>
#include <dos.h>

short int xst,yst,xnd,ynd;
short int DIM;

#define THD 128
#define LIKETHD 0.75

unsigned char *addr = (unsigned char
*)0xa0000000;//medit

#define  uc unsigned char
long
        xsize,ysize,imgsize,xbox,ybox,xref,yref,xr,
yr,xnew,ynew;
long
newxsize2,newysize2,newxsize3,newysize3;

uc *dataref, *dataingray;

short int xcen,ycen,maxx,maxy,maxt=10,mode;
char str1[40];
char str2[40];
char str[40];

char *nameing;

char nameref[40];

short int STY=10,YBOX=10;
short int readpcxline(unsigned char *p,FILE
*fp,long bytes);
void writepcxline(unsigned char *p,FILE
*fp,short int n);
long lread(FILE *fh, uc *data, long imgsize);

typedef struct
{
  char manufacturer;
  char version;
  char encoding;
  char bits_per_pixel;
  short int  xmin,ymin;
  short int  xmax,ymax;
  short int  hres;
  short int  vres;
  char __palette[48];
  char reserved;
  char color_planes;
  short int  bytes_per_line;
  short int  palette_type;
  char filler[58];
}PCXHEAD;

typedef struct
{
  char RED;
  char GREEN;
  char BLUE;
}PCXPALETTE;

PCXHEAD   pcx;
PCXPALETTE pal[256];

void writepcxline(unsigned char *p,FILE
*fp,short int n)
{
 unsigned short int i=0,j=0,t=0;
 do
 {
  i=0;
  while((p[t+i]==p[t+i+1]) && ((t+i) < n) && (i
< 63))
       ++i;

  if(i>0)
```

```
    {
    fputc(i | 0xc0,fp);
    fputc(p[t],fp);
    t+=i;
    j+=2;
    }
   else
    {
    if(((p[t]) & 0xc0) == 0xc0)
     {
     fputc(0xc1,fp);
     ++j;
     }
    fputc(p[t++],fp);
    ++j;
    }
  }
  while(t<n);
}

long lread(FILE *fh, uc *data, long imgsize)
{
long     buf=imgsize;
 while (imgsize > 32768L)
  {

if(fread((void*)data,1,(size_t)32768L,fh)!=32
768L)     return 0;
  imgsize -= 32768L;  data += 32768L;
  }
 if (fread(data,1,(size_t)imgsize, fh) != imgsize)
return 0;
 return buf;
}

short int readpcxline(unsigned char *p,FILE
*fp,long bytes)
{
 unsigned short int n=0,i,c;
 do
 {
  c=fgetc(fp)&0xff;
  if((c&0xc0)==0xc0)
   {
    i=c&0x3f;   c=fgetc(fp);
    while(i--) p[n++]=c;
    }
  else p[n++]=c;
 }
 while(n<bytes);
 return(n);
}


short int writepcx(uc *data_out,char
*filename,long xsize,long ysize)
{
FILE *fp;
short int x,y;
unsigned char _outpalette[768],buf[1024];
PCXHEAD   pcx;
 if((fp = fopen(filename,"wb"))==NULL)
  {
  printf("Error saving %s\n",filename);
  exit(1);
  }

 for(x=0;x<256;x++)
  {
  _outpalette[x*3]  =x;
  _outpalette[x*3+1]=x;
  _outpalette[x*3+2]=x;
  }

 memset((char*)&pcx,0,128);
 pcx.manufacturer=10;
 pcx.encoding=1;
 pcx.xmin=0;
 pcx.ymin=0;
 pcx.xmax=xsize-1;
 pcx.ymax=ysize-1;
 pcx.color_planes=1;
 pcx.bytes_per_line=xsize;
 pcx.bits_per_pixel=8;
 pcx.version=5;
 fwrite((char*)&pcx,1,128,fp);

 for(y=0;y<ysize;y++)
```

```
  {
  for(x=0;x<xsize;x++)
    buf[x]=*(data_out+y*xsize+x+4);
  writepcxline(buf,fp,xsize);
  }
 fputc(0x0c,fp);
 fwrite(_outpalette,1,768,fp);
 fclose(fp);
 return(1);
}

short int writepcxmatch(uc *data_out,char
*filename,long xsize,long ysize,long mxsize,long
mysize,short int xmin,short int ymin)
{
FILE *fp;
short int x,y;
unsigned char _outpalette[768],buf[1024];
PCXHEAD   pcx;
 if((fp = fopen(filename,"wb"))==NULL)
  {
  printf("Error saving %s\n",filename);
  exit(1);
  }

 for(x=0;x<256;x++)
  {
  _outpalette[x*3]  =x;
  _outpalette[x*3+1]=x;
  _outpalette[x*3+2]=x;
  }

 memset((char*)&pcx,0,128);
 pcx.manufacturer=10;
 pcx.encoding=1;
 pcx.xmin=0;
 pcx.ymin=0;
 pcx.xmax=mxsize-1;
 pcx.ymax=mysize-1;
 pcx.color_planes=1;
 pcx.bytes_per_line=mxsize;
 pcx.bits_per_pixel=8;
 pcx.version=5;
 fwrite((char*)&pcx,1,128,fp);
 for(y=0;y<mysize;y++)
  {
  for(x=0;x<mxsize;x++)

buf[x]=*(data_out+(y+ymin)*xsize+(x+xmin)+4);
  writepcxline(buf,fp,mxsize);
  }
 fputc(0x0c,fp);
 fwrite(_outpalette,1,768,fp);
 fclose(fp);
 ysize=ysize;

 return(1);
}

float corr(unsigned char *data,long
xsize,long ysize,unsigned char
*dataref,long xr,long yr,long
*xlike_,long *ylike_)
{
int x,y;
float          R,ratio;
long
        A,B,p_,s_,C,boxsize,P,S,xlike,ylike;
short int
        xpos,ypos,xsthead,xndhead,ysthead,ynd
head,xstsearch,ystsearch;
short int  xndsearch,yndsearch;
double          cat,Bf,Cf,like;

   boxsize=xr*yr;

   p_=0;
   for(y=0;y<yr;y++)
    for(x=0;x<xr;x++)
        p_+=*(dataref+y*xr+x+4);
   p_=p_/boxsize;

   B=0;
   for(y=0;y<yr;y++)
    for(x=0;x<xr;x++)
```

```
    {
     cat=(double)*(dataref+y*xr+x+4)-p_;
     //if(cat<0) cat=0;
     B+=pow(cat,2);
    }
  Bf=sqrt((long double)B);

  xlike=0;
  ylike=0;
  like=0.0;

  ypos=0;
  xpos=0;
  {
     s_=0;
     for(y=0;y<yr;y++)
     for(x=0;x<xr;x++)

s_+=*(data+(y+ypos)*xsize+(x+xpos)+4);
     s_=s_/boxsize;

     A=C=0;
     for(y=0;y<yr;y++)
     for(x=0;x<xr;x++)
       {
        P=*(dataref+y*xr+x+4)-p_;
        //if(P<0) P=0;

S=*(data+(y+ypos)*xsize+(x+xpos)+4)-s_;
        //if(S<0) S=0;
        A+=P*S;
        C+=S*S;
       }
     Cf=sqrt((long double)C);

     R=(float)A/(Bf*Cf);

     if(R>like)
       {
        like=R;
        xlike=xpos;
        ylike=ypos;
       }
  }
  *xlike_=xlike;
  *ylike_=ylike;
  return(like);
}

void mixrgb(int color_no,int r,int g,int b)
 {
  outp(0x3c6,0xff);
  outp(0x3c8,color_no);
  outp(0x3c9,r);
  outp(0x3c9,g);
  outp(0x3c9,b);
 }

void main(short int argc,char *argv[])
{
 short int  x,y,xmin,xmax,firsty;
 FILE       *fh;
 PCXHEAD    pcx;
 PCXPALETTE pal[256];
 short int
 z2,z4,z6,z8,z5,z1,z3,z7,z9,result,resultx,resulty;
 double fresult;

 float *like,*likebak,maxlike;

 long xlike[3],ylike[3];
 long likemax,likeitem;

 union REGS r;//medit
 int temp1,temp2,ploop,xshift,yshift;

 char str1[40];

 int MAXLOOP=160,loop,likeno[6];

 if(argc!=3)
   {
   printf("\nCorrect the parameter please");
   printf("\n%s  ref.pcx  maxloop",argv[0]);
   exit(1);
   }
```

```
sprintf(nameref ,"%s",argv[1]);

MAXLOOP=atoi(argv[2]);

if((fh=fopen(nameref,"rb"))==NULL)
printf("\nFile #1 not found").exit(1);
fread((char *)&pcx,1,sizeof(PCXHEAD),fh);
fseek(fh,-769L,SEEK_END); fgetc(fh);
for(x=0;x<256;x++)

if(fread(&pal[x],1,(size_t)sizeof(PCXPALETTE),f
h) != sizeof(PCXPALETTE))
    printf("fread PALETTE error"),exit(1);
xr = pcx.xmax-pcx.xmin+1; yr = pcx.ymax-
pcx.ymin+1;
imgsize=xr*yr;
if((dataref=(uc*)calloc(imgsize+4,1))==NULL)
exit(1);
fseek(fh,128L,SEEK_SET);
for(y=0;y<yr;y++)
readpcxline(&dataref[y*xr+4],fh,xr);
dataref[0]=(xr-1)&0x00ff; dataref[1]=((xr-
1)&0xff00)>>8;
dataref[2]=(yr-1)&0x00ff; dataref[3]=((yr-
1)&0xff00)>>8;
fclose(fh);


if((like=(float*)calloc(MAXLOOP+1,sizeof(float))
)==NULL)
  {
  printf("\nMem for float like[] error\n");
  exit(1);
  }


if((likebak=(float*)calloc(MAXLOOP+1,sizeof(flo
at)))==NULL)
  {
  printf("\nMem for float likebak[] error\n");
  exit(1);
  }

if((nameing=(char
*)malloc((MAXLOOP+1)*sizeof(char)*40))==
NULL)
  {
  printf("\nMem for char *nameing[40]
error\n");
  exit(1);
  }
for(loop=1;loop<=MAXLOOP;loop++)
{
    sprintf(nameing+40*loop,"hg%d.pcx",loop);

if((fh=fopen(nameing+40*loop,"rb"))==NULL)
    {
     printf("\nFile not found");
     exit(1);
    }
    fread((char *)&pcx,1,sizeof(PCXHEAD),fh);
    fseek(fh,-769L,SEEK_END); fgetc(fh);
    for(x=0;x<256;x++)

if(fread(&pal[x],1,(size_t)sizeof(PCXPALETTE),f
h) != sizeof(PCXPALETTE))
        printf("fread PALETTE error"),exit(1);
    xsize = pcx.xmax-pcx.xmin+1; ysize =
pcx.ymax-pcx.ymin+1;
    imgsize=xsize*ysize;

if((dataingray=(uc*)calloc(imgsize+4,1))==NUL
L) exit(1);
    fseek(fh,128L,SEEK_SET);
    for(y=0;y<ysize;y++)
readpcxline(&dataingray[y*xsize+4],fh,xsize);
    dataingray[0]=(xsize-1)&0x00ff;
dataingray[1]=((xsize-1)&0xff00)>>8;
    dataingray[2]=(ysize-1)&0x00ff;
dataingray[3]=((ysize-1)&0xff00)>>8;
    fclose(fh);
    like[loop]=corr(dataingray ,xsize,ysize
,dataref,xr,yr,&xlike[0],&ylike[0]);
    free(dataingray);
    if(kbhit()) break;
}
 free(dataref);
```

```
r.w.ax = 0x0013;
int386(0x10,&r,&r);

for(x=0;x<256;x++)
 mixrgb(x,x/4,x/4,x/4);

//sort by likely value max to min 5 level
for(loop=1;loop<=MAXLOOP;loop++)
  {
  likebak[loop]=like[loop];
  }

ploop=1;
for(x=1;x<=5&&x<MAXLOOP;x++)
{
maxlike=0;
for(loop=1;loop<=MAXLOOP;loop++)
  {
   if(like[loop]>maxlike)
     {
        maxlike=like[loop];
        likeno[ploop]=loop;
     }
  }
like[likeno[ploop]]=0;
ploop++;
}
//show 5rd of maxlike from max to low
 xshift=0;
for(ploop=1;ploop<=5&&ploop<MAXLOOP;pl
oop++)
   {
        sprintf(str1,"hg%d.pcx",likeno[ploop]);

        if((fh=fopen(str1,"rb"))==NULL)
          {
          printf("\nFile not found");
          exit(1);
          }

        fread((char
*)&pcx,1,sizeof(PCXHEAD),fh);
        fseek(fh,-769L,SEEK_END);
fgetc(fh);
        for(x=0;x<256;x++)

if(fread(&pal[x],1,(size_t)sizeof(PCXPALETTE),f
h) != sizeof(PCXPALETTE))
            printf("fread PALETTE
error"),exit(1);
        xsize = pcx.xmax-pcx.xmin+1; ysize =
pcx.ymax-pcx.ymin+1;
        imgsize=xsize*ysize;
        if((dataingray=(uc*)calloc(imgsize+4,1)
)==NULL) exit(1);
        fseek(fh,128L,SEEK_SET);
        for(y=0;y<ysize;y++)
readpcxline(&dataingray[y*xsize+4],fh,xsize);
        dataingray[0]=(xsize-1)&0x00ff;
dataingray[1]=((xsize-1)&0xff00)>>8;
        dataingray[2]=(ysize-1)&0x00ff;
dataingray[3]=((ysize-1)&0xff00)>>8;
        fclose(fh);
        for(y=0;y<ysize;y++)
 //scree width is 320
memcpy(addr+y*320+xshift,dataingray+y*xsize+
4,xsize);
        free(dataingray);
        xshift+=xsize;
 }
 getch();
 //restore TEXT MODE
 r.w.ax = 0x0003;
 int386(0x10,&r,&r);

 for(x=1;x<=5&&x<MAXLOOP;x++)
 {
   printf("\n%4d
[%2.3f]",likeno[x],likebak[likeno[x]]);
 }

 free(likebak);
 free(like);
 free(nameing);
}
```

# ประวัติผู้เขียน

ชื่อ พล.ท. วาภิรมย์ มนัสรังษี

เกิดวันที่ 17 ธ.ค. 2489 ที่จังหวัด ชลบุรี

การศึกษาก่อนเข้ารับราชการ

ปี 2505 คุณวุฒิ ม.6 ของ รร.สามัญ ที่ รร. อัสสัมชัญศรีราชา

ปี 2512 คุณวุฒิ ปีที่ 4 ของ รร. ทหาร ที่ รร.จปร.

การศึกษาเมื่อเข้ารับราชการแล้ว

หลักสูตรชั้นนายพัน รุ่นที่ 30/20

รร.สธ.ทบ.หลักสูตรหลักประจำ ชุดที่ 58/23

ความรู้พิเศษ หลักสูตร โดดร่ม รุ่นที่ 49/13

หลักสูตรนักบินปีกหมุน/16

การศึกษาดูงานต่างประเทศ

สัมมนาหลักสูตร Senior International Defense Management Course

ตามโครงการ IMETP FY 90 ณ ประเทศสหรัฐอเมริกา/33

รายการแสดงการรับราชการ

| | |
|---|---|
| 1 ก.ย. 2512 | ผบ.มว.ปืนเล็ก ร้อย.อวบ.ร.21 พัน.3 รอ. |
| 22 มี.ค. 2522 | ผู้บังคับหมวดขนส่งทางอากาศ กองบินปีกหมุนที่ 1 |
| 29 ต.ค. 2522 | ประจำ รร.สธ.ทบ.สบส. |
| 2 ต.ค. 2523 | ผช.หน.ฝ่ายยุทธการ (ฝ่ายอากาศ) พล.4 |
| 24 เม.ย. 2524 | รอง ผบ.ร.21 พัน. 1 รอ. |
| 17 พ.ค.2526 | รอง เสธ.รร.จปร. |
| 1 เม.ย. 2532 | ผอ. กองส่งกำลัง รอ. |
| 1 ต.ค. 2532 | รอป.รอ. |
| 1 มิ.ย. 2538 | รอง ผอ.สนง.รอป.รอ |

ปัจจุบัน รอง ผอ.สนง.รอป.รอ