ระบบจดจำทะเบียนรถยนต์

CAR LICENSE PLATE RECOGNITION SYSTEM

โดย

นางสาวนพวรรณ ม่วงเฉย

นายนัฏฐกร ปาระชัย

อาจารย์ที่ปรึกษา

ดร.สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ ปีการศึกษา 2541

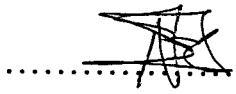
สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

เรื่อง ระบบจดจำทะเบียนรถยนต์ (CAR LICENSE PLATE RECOGNITION SYSTEM)

ผู้จัดทำ

1. นางสาวนพวรรณ ม่วงเฉย
2. นายนัฏฐกร ปาระชัย



.....

(ดร.สุรพันธ์ เอื้อไพบูลย์)

ปริญญานิพนธ์ ปีการศึกษา 2541

สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

เรื่อง ระบบจดจำทะเบียนรถยนต์ (CAR LICENSE PLATE RECOGNITION SYSTEM)

โครงการได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(ดร.สุรพันธ์ เอื้อไพบูลย์)

ระบบจดจำทะเบียนรถยนต์

นพวรรณ ม่วงเฉย
นัฏฐกร ปาระชัย
ดร.สุรพันธ์ เอื้อไพบูลย์
อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

ระบบจดจำเลขทะเบียนรถยนต์ถูกออกแบบให้สามารถประมวลผลสัญญาณภาพของรถยนต์เพื่อหาตำแหน่งและระบุเลขทะเบียนรถยนต์ได้โดยอัตโนมัติ รายงานฉบับนี้ได้นำเสนอวิธีการแยกเลขทะเบียนรถยนต์ออกจากภาพและการจดจำเลขทะเบียนรถยนต์ โดยมีขั้นตอนดังนี้คือ ในอันดับแรกจะทำการแยกวัตถุในภาพกับพื้นหลังออกจากกันด้วยวิธีการเลือกค่าเทรชโฮลที่เหมาะสม จากนั้นจะพิจารณาเพื่อเลือกวัตถุในภาพที่มีขนาดและการเรียงสอดคล้องกับขนาดและการเรียงตัวของเลขทะเบียน เลขทะเบียนที่ได้นี้จะถูกแยกออกมาเพื่อเข้าสู่กระบวนการจดจำรูปแแบบโดยใช้หลักการของโครงข่ายประสาทเทียม โดยใช้โครงสร้างสามชั้น คือ ชั้นอินพุท 35 หน่วย ชั้นซ่อน 20 หน่วย และชั้นเอาท์พุท 6 หน่วย ซึ่งโครงข่ายประสาทเทียมจะถูกฝึกหัดวิธีแบบแพร่ย้อนกลับด้วยชุดของตัวอักษรก่อนแล้ว

CAR LICENSE PLATE RECOGNITION SYSTEM

Noppawan Muangcheoy
Nattakorn Parachai
Dr.Surapan Eupaiboon
Advisor
1998

Abstract

Car license plate recognition system is able to analyze an image of a car given by a camera, locate the plate and recognize the registration numbers of a car. This report proposes the method of extracting the registration numbers using color information of the license plate. Dark objects with a light background are segmented by thresholding. Then the objects are selected according to criteria that fit the size and alignment and expected to be the registration numbers. These selected numbers will be classification by recognition based on neural networks architecture. A 3-layer neural networks was trained using varieties of input images for the best performance.

สารบัญ

สารบัญรูป

บทที่ 1
บทนำ

ในปัจจุบัน รถยนต์มีจำนวนเพิ่มมากขึ้น ทำให้เป็นการไม่สะดวกและสิ้นเปลืองบุคลากร ในการบันทึกข้อมูลทะเบียนรถยนต์เหล่านั้นในงานต่าง ๆ เช่น การเก็บค่าผ่านทางด่วน การให้เช่าสถานที่จอดรถ เป็นต้น จึงได้มีแนวความคิดที่จะพัฒนาระบบจดจำทะเบียนรถยนต์โดยอัตโนมัติขึ้นโดยใช้หลักการเปลี่ยนข้อมูลที่อยู่ในลักษณะของรูปภาพ (Image Data) ให้อยู่ในรูปของการจัดเก็บข้อมูลแบบตัวหนังสือ (Text File) เพื่อนำข้อมูลที่ได้ไปใช้กับระบบฐานข้อมูล (Database) หรือประยุกต์ใช้กับงานอื่น ๆ ได้ โดยอาศัยหลักการประมวลผลภาพ (Image Processing) และการจดจำรูปแบบ (Pattern Recognition) เป็นหลักสำคัญในการออกแบบระบบ

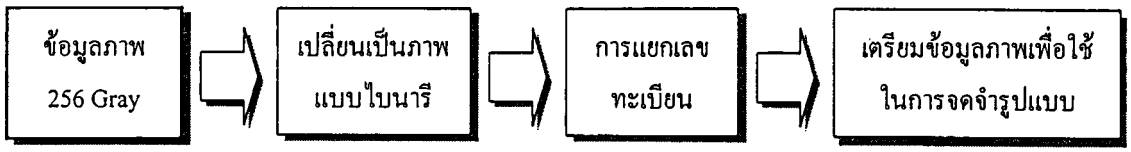
1.1 วัตถุประสงค์ของโครงงาน

- เพื่อศึกษาหลักการประมวลผลภาพ
- เพื่อศึกษาหลักการการจดจำรูปแบบ
- เพื่อศึกษาการเขียนโปรแกรมประยุกต์ใช้งานด้วยเดลไฟ

1.2 ขอบเขตของโครงงาน

ระบบจดจำป้ายทะเบียนรถยนต์ที่นำเสนอนี้ จะนำข้อมูลภาพของทะเบียนรถยนต์เป็นข้อมูลเพื่อใช้ในการหาตำแหน่งของหมายเลขทะเบียนรถยนต์โดยอาศัยขั้นตอนการประมวลผลภาพ เช่น การกำจัดสัญญาณรบกวน การทำไบนาไรเซชัน (Image Binarization) การจำแนกภาพ (Image Segmentation) แล้วนำผลลัพธ์ที่ได้เข้าสู่กระบวนการจดจำรูปแบบของหมายเลขทะเบียนรถยนต์ เป็นการทำให้คอมพิวเตอร์สามารถรู้จักและเข้าใจภาพได้ (Recognition)

รายงานฉบับนี้ ได้เสนอวิธีการแยกหมายเลขทะเบียนรถยนต์จากข้อมูลภาพเพื่อเตรียมข้อมูลสำหรับขั้นตอนการจดจำรูปแบบ โดยนำข้อมูลภาพด้านหน้าของรถยนต์ที่มีการเก็บข้อมูลแบบบิตแมปเกรย์สเกล 256 ระดับ (256 Gray Scale) ขนาด 320×220 พิกเซล มาใช้เป็นข้อมูลในการวิเคราะห์ เพื่อแยกเฉพาะส่วนของเลขทะเบียนออกจากข้อมูลภาพทั้งหมด โดยมีขั้นตอนต่าง ๆ ดังรูปที่ 1.1



รูปที่ 1.1 ขั้นตอนการทำงานของโปรแกรม

หลังจากผ่านขั้นตอนของการแยกหมายเลขทะเบียนรถยนต์แล้ว ข้อมูลภาพของหมายเลขทะเบียนรถยนต์จะถูกนำเข้าสู่กระบวนการจดจำรูปแบบ เพื่อทำการประมวลผลข้อมูลและให้ผลลัพธ์ออกมาเป็นค่าของตัวเลขหรือตัวอักษร ในรายงานฉบับนี้ได้เสนอวิธีการจดจำตัวอักษรโดยใช้หลักการของทฤษฎีโครงข่ายประสาทเทียม (Neural Network) ด้วยวิธีแพร่ย้อนกลับ ซึ่งมีรายละเอียดการทำงานดังจะอธิบายต่อไป

1.3 การประยุกต์ใช้งานของระบบจดจำป้ายทะเบียนรถยนต์

- **ระบบเก็บค่าใช้บริการทางด่วน**

ระบบจดจำเลขทะเบียนรถยนต์สามารถนำไปประยุกต์ใช้กับฐานข้อมูลของระบบเก็บค่าใช้บริการทางด่วน โดยเจ้าของรถยนต์จะมีบัญชีเพื่อหักค่าบริการทางด่วน เมื่อรถยนต์ผ่านเข้าไปใช้บริการทางด่วน ระบบจดจำเลขทะเบียนรถยนต์จะประมวลผลเพื่อระบุหมายเลขทะเบียนรถยนต์ แล้วจัดการกับข้อมูลของรถยนต์คันนั้นในฐานข้อมูล เช่น การหักบัญชีค่าบริการทางด่วนจากเจ้าของรถยนต์โดยอัตโนมัติ ซึ่งทำให้สามารถลดจำนวนพนักงานเก็บค่าทางด่วนและเพิ่มความสะดวกสบายให้กับผู้ใช้บริการทางด่วน

- **ระบบป้องกันการโจรกรรมรถยนต์**

ในสถานที่ที่มีความเสี่ยงต่อการโจรกรรมรถยนต์ เช่น ในศูนย์การค้า สามารถใช้ระบบจดจำเลขทะเบียนรถยนต์ เพื่อตรวจเช็คการเข้าออกของรถยนต์ โดยเมื่อมีรถยนต์เข้ามาระบบจะระบุเลขทะเบียนรถยนต์ และออกบัตรจอดรถที่มีหมายเลขทะเบียนตรงกัน ดังนั้นรถยนต์จะสามารถออกไปได้เมื่อมีบัตรตรงกับหมายเลขทะเบียนรถยนต์เท่านั้น

นอกจากนี้ยังสามารถประยุกต์ใช้ร่วมกับงานของเจ้าหน้าที่ตำรวจในการติดตามรถยนต์ที่ต้องการ เช่น รถยนต์ที่ถูกโจรกรรม หรือรถยนต์ที่ผู้กระทำความผิดใช้ โดยติดตั้งระบบตามด่านตรวจต่างๆ เพื่อให้สามารถติดตามรถได้อย่างรวดเร็ว

- **ระบบควบคุมการเข้าออกสถานที่สำคัญ**

ในสถานที่สำคัญ เช่น เขตทหารจะไม่สามารถให้รถยนต์ที่ไม่ได้รับอนุญาตเข้าไปในสถานที่นั้นได้ ดังนั้นในระบบควบคุมการเข้าออกสถานที่จะใช้ระบบจดจำเลขทะเบียนรถยนต์เพื่อตรวจสอบหมายเลขทะเบียนของรถยนต์นั้นว่าตรงกับหมายเลขทะเบียนของรถยนต์ในระบบฐานข้อมูลที่สามารถให้เข้าไปในสถานที่นั้นหรือไม่

บทที่ 2
การประมวลผลภาพ

2.1 การประมวลผลภาพเชิงตัวเลข (Digital Image Processing)

การประมวลผลภาพเชิงตัวเลข หมายถึง การนำภาพที่พบทั่วไปมาประมวลผลด้วยเครื่องคอมพิวเตอร์ โดยภาพที่นำมาประมวลผลด้วยเครื่องคอมพิวเตอร์นี้จะถูกแทนที่ด้วยตัวเลขให้อยู่ในรูปของเมตริกซ์ แต่ภาพที่ได้โดยส่วนมากแล้วจะเป็นภาพที่ได้จากตัวรับสัญญาณ ซึ่งอยู่ในรูปของฟังก์ชัน $f(x,y)$ ที่ต่อเนื่องในระนาบสองมิติ (คือแกน X และแกน Y) โดยจะเป็นสัดส่วนกับความสว่างหรือความเข้มของภาพ ที่ตำแหน่ง (x,y) ซึ่งเรียกว่า ระดับสีเทา (Gray Level)

2.1.1 การแทนภาพด้วยข้อมูลแบบดิจิตอล

ภาพข้อมูลแบบดิจิตอล (Digital Image) เป็นภาพที่ถูกแปลงมาจากอนาลอก อยู่ในรูปของตัวเลข โดยภาพอนาลอกถูกแบ่งเป็นพื้นที่สี่เหลี่ยมเล็ก ๆ ที่เรียกว่า พิกเซล (Pixel) ในแต่ละพิกเซล จะถูกระบุตำแหน่งโดย (x,y) และค่าระดับสีเทาของพิกเซล โดยเราสามารถแปลงภาพเป็นข้อมูลแบบดิจิตอลได้ โดยมีขั้นตอนและวิธีการดังนี้

เมื่อเรานำสัญญาณอนาลอกที่ต้องการประมวลผลมาผ่านส่วนที่เรียกว่า ดิจิไตเซอร์ (Digitizor) ซึ่งจะมีหน้าที่ในการเปลี่ยนสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอล อุปกรณ์ส่วนนี้ได้แก่ กล้องโทรทัศน์ดิจิไตเซอร์ จากนั้นทำการควอนไตซ์ (Quantizing) เพื่อที่จะประมวลสัญญาณด้วยระบบคอมพิวเตอร์ ฟังก์ชันของภาพ $f(x,y)$ จะถูกทำให้เป็นสัญญาณไม่ต่อเนื่องทั้งระนาบของภาพ ซึ่งเราเรียกว่า การสุ่มภาพ (Image Sampling) ของฟังก์ชันที่ได้เรียกว่า การควอนไตเซชั่นระดับสีเทา (Gray Level Quantization) ก็จะได้ข้อมูลที่เป็นดิจิตอล

สมมุติว่าสัญญาณภาพต่อเนื่อง $f(x,y)$ ถูกดิจิไตซ์ในระนาบ X และ Y เป็นช่วงเท่าๆ กัน เราสามารถจัด $f(x,y)$ ให้อยู่ในรูปของเมตริกซ์ ขนาด N×N ได้ดังสมการที่ 2.1

$$f(x,y) = \begin{matrix} f(0,0) & f(0,1) & f(0,2) \dots f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) \dots f(1,N-1) \\ . & . & . \quad . \\ . & . & . \quad . \\ f(N-1,0) & f(N-1,1) & f(N-1,2) \quad f(N-1,N-1) \end{matrix} \qquad \dots(2.1)$$

ซึ่งทางขวาของสมการ จะเรียกได้ว่า ภาพดิจิตอล และทุกๆ สมาชิกของเมตริกซ์ จะเรียกว่าพิกเซล จากขบวนการสร้างภาพดิจิตอลข้างต้น จะเห็นได้ว่าเราสามารถทราบขนาดของความละเอียดของภาพ N×N พิกเซล และจำนวนระดับของเกรย์สเกล ในทางปฏิบัติการทำควอนไตเซชั่นในระบบภาพดิจิตอล จะมีค่าดังสมการที่ 2.2

$$B = N \times N \times M \quad \text{บิต} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

เมื่อ B = ขนาดของข้อมูลภาพที่เป็นดิจิตอล

G = จำนวนของเกรย์สเกล ที่ต้องการใช้ในการเก็บข้อมูลภาพ

M = จำนวนบิตที่ใช้ในการแทนข้อมูลภาพ 1 พิกเซล

โดย M สามารถหาได้จาก

$$G = 2^M$$

2.1.2 ลักษณะการจัดเก็บข้อมูลภาพแบบดิจิตอล

โดยทั่วไปแล้ว ข้อมูลภาพจะมีค่าความเข้มตั้งแต่ 2 ระดับขึ้นไป แต่ที่ใช้กันมากจะใช้กันที่ค่าระดับความเข้มของจุดภาพเท่ากับ 256 ระดับ ซึ่งจะทำให้ค่าของจุดภาพอยู่ในช่วง (0-255) โดยใช้เนื้อที่การเก็บข้อมูลภาพขนาด 1 ไบต์ หรือ 8 บิต สำหรับข้อมูล 1 จุดภาพ ($2^8 = 256$) ในกรณีที่ต้องการภาพที่มีความละเอียดของระดับความเข้มสูงๆ อาจจะต้องการจำนวนบิตสำหรับการเก็บข้อมูลมากกว่า 8 บิต คืออาจจะเป็น 16 หรือ 24 บิต โดยค่าความเข้มของจุดภาพจะเท่ากับ 2^{16} และ 2^{24} โดยจะแยกให้เห็นชัดเจนดังนี้

1. ภาพ 2 ระดับ คือ มีเพียงแค่จุดขาวกับจุดดำเท่านั้น โดยแต่ละจุดภาพเป็นข้อมูลขนาด 1 บิต

2. ภาพ 16 ระดับ คือ ในแต่ละจุดภาพจะมีขนาดของข้อมูล 4 บิต ซึ่งทำให้สามารถแสดงได้16 ระดับสี หรือ 16 เกรย์สเกล ขึ้นอยู่กับภาพนั้นเป็นภาพสีหรือภาพขาวดำ

3. ภาพ 256 ระดับ คือ ในแต่ละจุดภาพจะมีขนาดของข้อมูล 8 บิต ซึ่งทำให้สามารถแสดงภาพได้ 256 ระดับสี หรือ 256 เกรย์สเกล ขึ้นอยู่กับภาพนั้นเป็นภาพสีหรือภาพขาวดำ

4. ภาพทรูคัลเลอร์ (True color) คือในแต่ละจุดภาพจะมีขนาดของข้อมูล 24 บิต ทำให้สามารถแสดงผลภาพได้เหมือนภาพจริงที่สุด เพราะสามารถแสดงสีได้ถึง 16,777,216 สี ภาพทรูคัลเลอร์สามารถแสดงได้เฉพาะภาพสีเท่านั้น ไม่สามารถแสดงผลภาพขาวดำได้

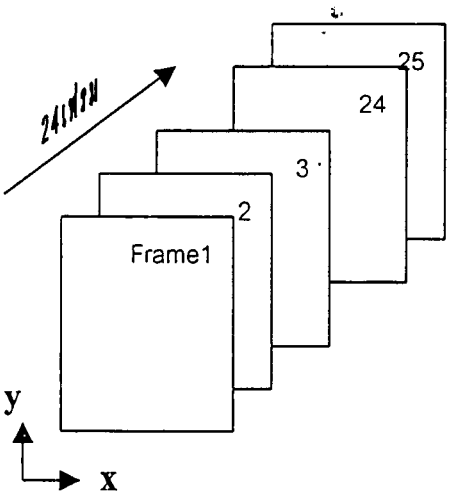
การแสดงภาพนี้ใช้วิธีตั้งค่าของแม่สีในตารางสี โดยอาจเลือกสีเป็นแบบ 16 สี จาก 64 สี หรือ 16 สี จาก 262,144 สี หรือ 256 สี จาก 262,144 สี ขึ้นอยู่กับโหมดการแสดงผล สำหรับทรูคัลเลอร์ จะไม่มีการเลือกสี แสดงผลโดยการส่งค่าสี RGB ผ่าน D/A สีละ 8 บิต ออกไป ความแตกต่างของการแสดงภาพสีและขาวดำ คือ ภาพขาวดำจะต้องตั้งให้แม่สีทั้งสามสี มีค่าเท่ากัน เนื่องจาก VGA กำหนดให้แม่สีแต่ละสีใช้ได้เพียง 64 ระดับเท่านั้น หากต้องการให้เห็นจริงทั้ง 256 ระดับ ต้องแสดงในโหมดทรูคัลเลอร์ แล้วให้ RGB มีค่าเท่ากัน ซึ่งในโหมดนี้ จะสามารถใช้ รีจีสเตอร์ ได้ 8 บิต สำหรับแต่ละแม่สี

โดยทั่วไปวิธีการประมวลผลภาพเชิงตัวเลขที่ทำให้คอมพิวเตอร์สามารถรู้จักวัตถุภายในภาพได้นั้น พอจะแบ่งได้สองระดับด้วยกันคือ การประมวลผลภาพในระดับต่ำ (Low-Level Image Processing) และการประมวลผลภาพในระดับสูง (High-Level Image Processing) การประมวลผลในระดับต่ำจะเป็นการประมวลผลเชิงตัวเลขเกือบทั้งหมด เพื่อหาตัวแปรต่างๆ มาอธิบายข้อมูลภาพ โดยมีจุดประสงค์เพื่อนำตัวแปรเหล่านั้นไปใช้ในการประมวลผลระดับสูงต่อไป โดยทั่วไปแล้วการประมวลผลภาพระดับต่ำจะประกอบไปด้วย การประมวลผลภาพก่อน (Preprocessing) การกำจัดสัญญาณรบกวนหรือการทำให้ภาพคมชัด การหาขอบภาพ เป็นต้น

การประมวลผลในระดับสูง เป็นการนำผลลัพธ์หรือสัญลักษณ์ที่ได้จากการประมวลผลระดับต่ำมาตีความหรือประมวลผลเพื่อให้คอมพิวเตอร์สามารถรู้จักและเข้าใจภาพได้สำหรับความแตกต่างของการประมวลผลภาพระดับต่ำและการประมวลผลภาพระดับสูงนั้นคือ ข้อมูลที่นำมาใช้ในการประมวลผลภาพ โดยที่การประมวลผลภาพระดับต่ำจะใช้ค่าความสว่างของจุดโดยตรง ส่วนการประมวลผลภาพระดับสูงนั้นข้อมูลภาพที่นำมาประมวลผลจะถูกแสดงในรูปของสัญลักษณ์ ซึ่งสัญลักษณ์เหล่านี้จะแสดงถึงสิ่งต่างๆ ที่มีอยู่ในภาพ เช่น ขนาดของวัตถุ รูปร่าง และความสัมพันธ์กันระหว่างวัตถุภายในภาพ

2.2 สัญญาณข้อมูลภาพจากดิจิตอลวิดีโอ

การส่งสัญญาณข้อมูลภาพจากวิดีโอ จะมีลักษณะการส่งที่เป็นลำดับภาพเดี่ยวหรือเฟรม (Frame) ที่ฉายต่อเนื่องกันดังรูปที่ 2.1 เช่น ภาพยนตร์ใช้อัตรา 24 เฟรมต่อวินาที หรือ วิดีโอระบบ NTSC จะส่งด้วย อัตราเร็ว 30 เฟรมต่อวินาที โดยดิจิตอลวิดีโอแต่ละเฟรมจะเป็นข้อมูลภาพดิจิตอลในลักษณะของเมตริกซ์ (Matrix) ซึ่งแต่ละจุดเรียกว่า พิกเซล (Pixel) มีค่าของระดับความเข้มสี โดยทั่วไปจะใช้เกรย์สเกลสากลที่มีค่าตัวเลขตั้งแต่ 0 ถึง 255 โดย 0 แทนความมืดมากที่สุด ส่วนตัวเลข 255 จะแทนความสว่างมากที่สุดซึ่งจะกล่าวในหัวข้อต่อไป



รูปที่ 2.1 แผนภาพตัวอย่างการส่งสัญญาณวิดีโออัตรา 24 เฟรมต่อวินาที

2.3 ไฟล์ข้อมูลภาพชนิดบิตแมป

2.3.1 รูปแบบของไฟล์ข้อมูลภาพชนิดบิตแมป

รูปแบบของไฟล์ข้อมูลภาพชนิดบิตแมป เป็นฟอร์แมตของวินโดว์บิตแมป (Bitmap) ซึ่งเป็นมาตรฐานสำหรับไฟล์กราฟิกบนวินโดว์ ซึ่งจะใช้ในการตัดต่อ หรือสำเนาภาพต่างๆ ลงบนคลิปบอร์ด (Clipboard) เมื่อเวลาจัดเก็บไฟล์ที่มีสกุล BMP ซึ่งเป็นฟอร์แมตนี้ยังสามารถใช้เป็นวอลเปเปอร์ (Wallpaper) ของวินโดว์ได้อีกด้วย

2.3.2 โครงสร้างของไฟล์ข้อมูลภาพชนิดบิตแมป

โครงสร้างของไฟล์ข้อมูลภาพชนิดบิตแมป จะประกอบด้วย 3 ส่วนคือ

1. ข้อมูลเฮดเดอร์ (Header)
2. ข้อมูลจานสี (Palette)
3. ข้อมูลภาพ (Data)

1. ข้อมูลเฮดเดอร์ คือข้อมูลที่อยู่บริเวณส่วนหัวของไฟล์ ซึ่งประกอบด้วยข้อมูลที่บอกรายละเอียดต่างๆของภาพ เช่น ความกว้าง ความยาวของภาพ จำนวนสี จำนวนบิต ความละเอียด เป็นต้น ซึ่งโครงสร้างของ เฮดเดอร์จะแสดงในภาคผนวก

2. ข้อมูลจานสี คือ ข้อมูลที่บอกถึงชุดของจานสี ที่เกิดจากการผสมแม่สีทั้งสาม คือ แดง เขียว และน้ำเงิน มาผสมกันได้เป็นสีต่างๆ ตามจำนวนสีของภาพ เช่น รูปขนาด 4 บิต จะมี 16 ระดับสี รูปขนาด 8 บิต จะมีขนาด 256 ระดับสี เป็นต้น ซึ่งถ้ามีจำนวนสีน้อยๆ ก็จะมีการเก็บค่าจานสีนี้ลง

ไฟล์ไปด้วย แต่ถ้าเป็นรูปประเภท 24 บิตจะไม่มีค่าจานสี แต่จะใช้วิธีการเก็บค่าแม่สีทั้งสามลงไปเป็นข้อมูลแทนเพราะถ้าเก็บค่าจานสี ที่มีถึง 16.7 ล้านสีลงไปด้วยจะเปลืองพื้นที่มาก ข้อแตกต่างที่สำคัญของบิตแมปขนาดนี้ คือ ไฟล์บิตแมป จะเก็บค่าของจานสี ชุดละ 4 ไบต์ แต่ก็ใช้แค่ 3ไบต์ เช่นกัน คือ แดง เขียว และน้ำเงิน อย่างละ 1 ไบต์

3. ข้อมูลภาพ คือ ข้อมูลสีของภาพแต่ละจุดที่มาประกอบกันเป็นรูปภาพ ซึ่งค่าที่เก็บนี้จะเป็นค่าที่ใช้ในการชี้ตาราง Palette หมายเลขอะไร เช่น จุดแรกมีค่าเป็น 10 ก็ให้ไปเปิดตาราง Palette หมายเลข 10 สมมุติว่าของแม่สีเป็น R = 0 G = 0 และB = 100 ก็จะได้จุดนี้เป็นสีน้ำเงิน ซึ่งถ้าเป็นในกรณีของรูป 24 บิต จะเป็นการอ่านข้อมูลขึ้นมา 3 ค่า เป็นค่าของแม่สี RGB แล้วนำไปผสมบนจอภาพแทน

2.3.3 การจัดเก็บไฟล์ข้อมูลชนิดบิตแมป

การเก็บไฟล์ข้อมูลภาพชนิดบิตแมป มีการเก็บอยู่ 2 แบบ คือ

- **แบบบีบอัดข้อมูล**
  - RLE 4 เป็นการบีบอัดข้อมูลแบบ Run-length Encoder แบบ 4 บิต
  - RLE 8 เป็นการบีบอัดข้อมูลแบบ Run-length Encoder แบบ 8 บิต
- **แบบไม่ได้บีบอัดข้อมูล**

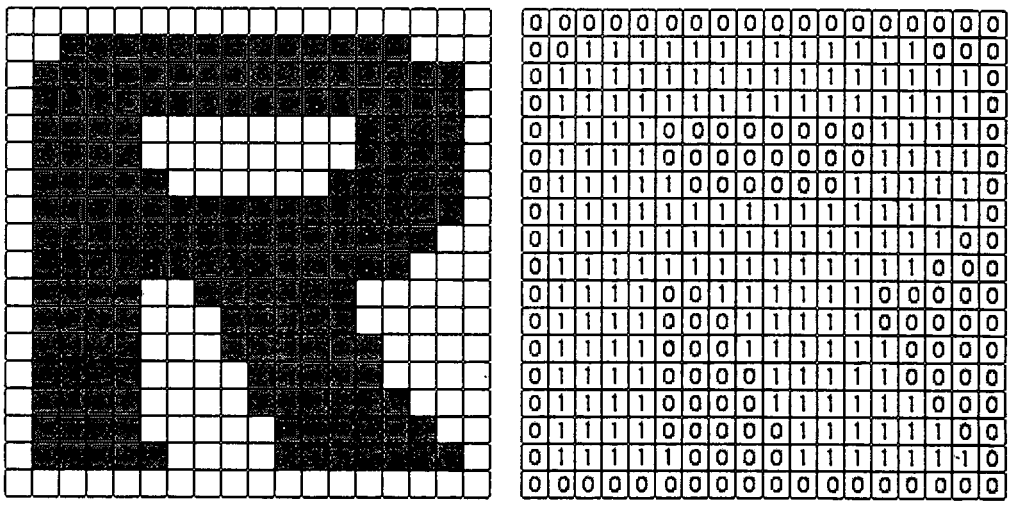
  เป็นการเก็บข้อมูลจริงของสีของพิกเซล ซึ่งทำให้ขนาดของไฟล์ค่อนข้างใหญ่ แต่จะทำการแสดงภาพได้เร็วกว่า เพราะไม่ต้องเสียเวลาในการคลายข้อมูล

2.4 การสร้างภาพไบนารี

อุปกรณ์ที่มีความสามารถในการแสดงผลได้แค่ 2 ระดับ หรือ 2 สี คือ สีขาวกับสีดำยังมีการใช้กันอย่างแพร่หลาย เช่น เครื่องพิมพ์ (Printer) เครื่องโทรสาร (Fax) จอภาพแสดงผลแบบโมโนโครม (Monochrome Monitor) เป็นต้น เนื่องจากอุปกรณ์เหล่านี้เป็นอุปกรณ์ ที่มีราคาถูก ดังนั้นการที่จะแสดงผลหรือพิมพ์รูปภาพที่มีระดับความเข้มของภาพหลายระดับซึ่งมีมากกว่าความสามารถในการแสดงผลของอุปกรณ์เหล่านั้นที่มีเพียงแค่ 2 ระดับเท่านั้น

จะเห็นได้ว่าการที่จะแก้ปัญหาการแสดงผลภาพที่มีความเข้มหลายระดับบนอุปกรณ์ที่สามารถแสดงผลได้ 2 ระดับนั้น จะต้องทำการแปลงข้อมูลภาพให้เป็นภาพไบนารี (Binary Image) ก่อน ซึ่งการสร้างภาพไบนารี นั่นก็หมายถึงการแปลงข้อมูลภาพที่มีระดับความเข้มหลายระดับ (Multi Level Image)ให้เป็นภาพที่มีระดับความเข้มเพียง 2 ระดับ นั่นคือ 1 จุด ภาพมีได้ 2 ค่าเท่านั้น คือ 0 กับ 1 โดยจุดภาพที่แทนด้วย 1 จะหมายถึงจุดภาพที่มีสีดำ ส่วนจุดที่แทนด้วย 0 จะหมายถึงจุดภาพที่มีสีขาว เมื่อทำการแปลงเป็นภาพไบนารีแล้วจึงนำภาพนั้นไปแสดงผลบนอุปกรณ์เหล่านั้น จะเห็นได้ว่า

การแปลงข้อมูลภาพหลายระดับเป็นภาพไบนารีจึงมีความจำเป็นและมีประโยชน์มากในการแสดงผลภาพที่มีระดับความเข้มของภาพหลายระดับบนอุปกรณ์ที่มีความสามารถในการแสดงผลได้ 2 ระดับ สำหรับประโยชน์อีกประการหนึ่งในการแปลงข้อมูลภาพนั้นเป็นภาพไบนารีคือการลดเนื้อที่การเก็บข้อมูลภาพจะใช้เนื้อที่ในการเก็บ 8 บิต เมื่อสร้างเป็นภาพไบนารีแล้วสามารถลดลงได้ถึง 8 เท่า นั่นคือ 1 จุดภาพจะใช้เนื้อที่ในการเก็บ 1 บิต อีกทั้งยังสามารถนำไปประยุกต์ใช้งานได้อย่างแพร่หลาย เช่นนำไปประยุกต์ใช้ในการวิเคราะห์เอกสารในขั้นตอนที่เรียกว่าการประมวลผลขั้นต้น (Preprocessing) เป็นต้น



รูปที่ 2.2 ภาพแบบไบนารีและข้อมูลของแต่ละพิกเซล

ในการสร้างภาพไบนารี สามารถทำได้โดยใช้เทคนิคการทำเทรชโฮล (Thresholding Technique) โดยพิจารณาว่าจุดภาพใดควรจะเป็นจุดขาวหรือจุดดำ จะกระทำโดยการเปรียบเทียบระหว่างจุดภาพเริ่มต้นกับค่าคงที่ค่าหนึ่งที่เรียกว่า "ค่าเทรชโฮล" (Threshold Value) เทคนิคนี้ใช้กันมากในกรณีที่ข้อมูลภาพมีลักษณะแตกต่างกันระหว่างวัตถุ (Object) และพื้นหลัง (Background) โดยค่าของจุดภาพใดๆ ที่มีค่าน้อยกว่าค่าเทรชโฮลจะถูกกำหนดให้เป็น 1 (จุดดำ) และถ้าค่าของจุดภาพใดๆ ที่มีค่ามากกว่า หรือเท่ากับค่าเทรชโฮลจะถูกเปลี่ยนให้เป็น 0 (จุดขาว) ซึ่งการทำงานสามารถแสดงได้ดังสมการ ที่ 2.3

$$b(x,y) = \begin{cases} 1 ; g(x,y) < Thr \\ 0 ; g(x,y) \geq Thr \end{cases} \qquad \text{.....(2.3)}$$

b(x,y) ข้อมูลภาพผลลัพธ์ เป็นภาพไบนารี

g(x,y) ข้อมูลภาพอินพุท ที่มีระดับความเข้ม 0 ถึง L ระดับ

Thr ค่าเทรชโฮล เป็นค่าคงที่ ที่อยู่ระหว่าง 0 ถึง L ระดับ

0 จุดดำ

0 จุดขาว

โดยที่ L คือระดับความเข้มของจุดภาพสูงสุด

ในการสร้างภาพไบนารีโดยใช้เทคนิคเทรชโฮลเพื่อให้ได้ผลลัพธ์ที่ได้เหมาะสมและคมชัด สิ่งที่สำคัญที่สุดคือ ค่าเทรชโฮล เนื่องจากถ้าเลือกค่าเทรชโฮลที่ไม่เหมาะสม (ค่าเทรชโฮลที่มีค่าน้อยเกินไปหรือมากเกินไป) ภาพที่ได้อาจจะไม่เหมาะสม ขาดความคมชัดและรายละเอียดบางส่วนขาดหายไป กล่าวคือภาพที่ได้อาจจะมืดเกินไป (จุดดำมากเกินไป) หรือสว่างเกินไป (จุดขาวมากเกินไป) หรือภาพที่ได้มีสิ่งรบกวน (Noise) เกิดขึ้น อันเป็นผลทำให้ภาพผลลัพธ์ที่ได้ไม่สวยงามเท่าที่ควร ดังนั้นปัญหาของการสร้างภาพไบนารีโดยวิธีเทรชโฮลนี้คือ ทำอย่างไรจึงจะสามารถคำนวณหาค่าเทรชโฮลที่เหมาะสมสำหรับแต่ละภาพที่จะนำมาทำการสร้างภาพไบนารี ซึ่งมีวิธีการคำนวณหาค่าเทรชโฮลหลายวิธี โดยแต่ละวิธีเหมาะสมกับลักษณะการทำงานที่แตกต่างกันไป เช่นการหาค่าเทรชโฮลโดยการกำหนดค่าล่วงหน้า (Preassigned Threshold Value) การหาค่าเทรชโฮล จากค่ากลาง (Mid-Range Threshold Value) แต่ละวิธีอธิบายได้ดังนี้

การหาค่าเทรชโฮลโดยการกำหนดค่าล่วงหน้า (Preassigned Threshold Value) การหาค่าเทรชโฮลโดยวิธีการกำหนดล่วงหน้า นี้เป็นวิธีที่ง่ายที่สุด เป็นการคำนวณค่าเทรชโฮลโดยการกำหนดเองจากผู้ใช้ ซึ่งการกำหนดนี้จะขึ้นอยู่กับประสบการณ์ของผู้ใช้นั้นๆ โดยการ เลือกค่าคงที่ค่าหนึ่ง ซึ่งเรียกค่านั้นว่า ค่าเทรชโฮล โดยค่าที่เลือกมานี้จะเป็นค่าที่อยู่ระหว่างค่าต่ำสุดและค่าสูงสุด ของระดับความเข้มของข้อมูลภาพอินพุต เช่นภาพข้อมูลอินพุตมีเกรย์สเกล 256 ระดับ จะมีค่าเกรย์สเกลได้ตั้งแต่ 0 – 255 เมื่อเลือกค่าเทรชโฮลได้แล้วสามารถสร้างภาพไบนารีได้โดยสมการ 2.5

การหาค่าเทรชโฮลจากค่ากลาง (Mid – Range Threshold Value) การหาค่าเทรชโฮลโดยพิจารณาจากค่ากลาง เป็นการหาค่าเทรชโฮลที่แตกต่างจากการหาค่าเทรชโฮลวิธีแรก สำหรับวิธีนี้จะเป็นการคำนวณหาค่าเทรชโฮลโดยอัตโนมัติโดยไม่ต้องให้ผู้ใช้เป็นผู้กำหนด โดยการหาค่าเทรชโฮลวิธีนี้ได้อาศัยการคำนวณพื้นฐานทางสถิติในเรื่องของการหาค่ากลางหรือค่าเฉลี่ย (Mean) มาประยุกต์ใช้ ค่าเทรชโฮล ที่คำนวณได้จะเป็นค่าที่ได้จากค่ากึ่งกลางที่อยู่ระหว่างค่าระดับความเข้มสูงสุด

(Maximum Level) และค่าระดับความเข้มต่ำสุด(Minimum Level) ของข้อมูลภาพอินพุต สำหรับการคำนวณหาค่ากึ่งกลางนี้สามารถคำนวณได้จากสมการที่ 2.4

$$Thr = \frac{Maximum(g(x,y)) + Minimum(g(x,y))}{2} \quad \text{.....(2.4)}$$

โดยที่ Thr ค่าเทรชโฮล

$g(x,y)$ ข้อมูลภาพอินพุต ที่มีระดับความเข้ม 0 ถึง L ระดับ

MAXIMUM(g(x,y)) ค่าสูงสุดเกรย์สเกลของข้อมูลอินพุต

MINIMUM(g(x,y)) ค่าต่ำสุดเกรย์สเกลของข้อมูลอินพุต

เมื่อทำการคำนวณค่าเทรชโฮลได้แล้ว ก็สามารถสร้างภาพไบนารีได้โดยนำค่าเทรชโฮลที่ได้มาแทนค่าในสมการ 2.3

การหาค่าเทรชโฮลจากค่าเฉลี่ยเลขคณิต หาได้จากสมการที่ 2.5

$$Thr = \frac{\sum_{i=0}^{N\times N} gi(x,y)}{N \times N} \quad \text{.....(2.5)}$$

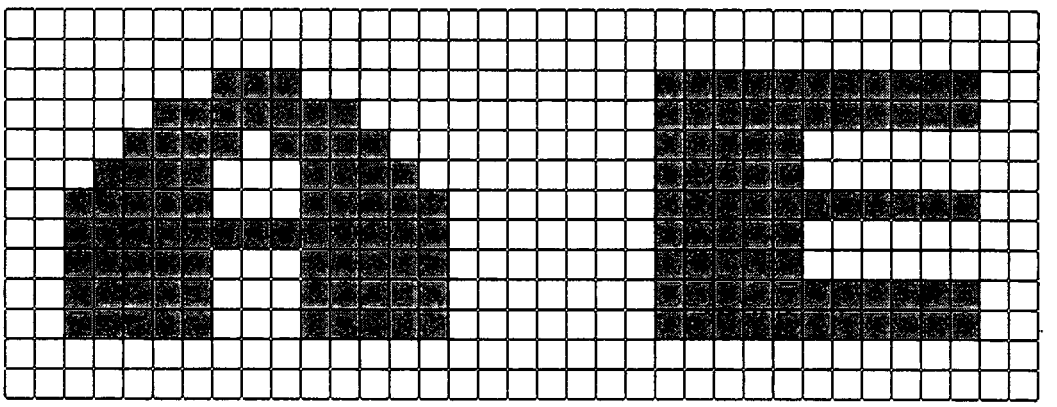
2.5 การแยกวัตถุจากภาพ (Segmentation)

กระบวนการสำคัญอีกขั้นตอนหนึ่งในการประมวลผลภาพเบื้องต้นก่อนที่จะไปสู่ขั้นตอนการจดจำรูปแบบ ก็คือกระบวนการแยกวัตถุออกจากพื้นหลัง ซึ่งในที่นี้จะเป็นการแยกข้อมูลภาพที่เป็นตัวอักษรออกจากข้อมูลภาพทั้งหมด โดยแยกออกมาทีละตัวอักษรเพื่อนำไปเข้าสู่กระบวนการจดจำรูปแบบซึ่งสามารถประมวลผลได้ทีละหนึ่งตัวอักษรเท่านั้น

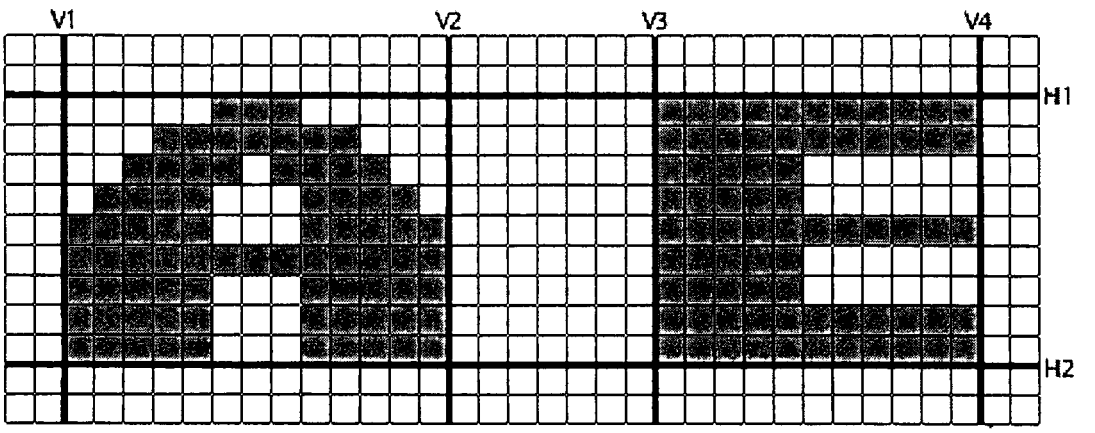
2.5.2 การแยกภาพด้วยการพิจารณาการต่อเนื่องของข้อมูล

เมื่อรับข้อมูลภาพที่ได้จากการเปลี่ยนข้อมูลเป็นรูปแบบไบนารี ที่มีค่า 0 กับ 1 เรียบร้อยแล้ว ซึ่งข้อมูล 0 จะแทน ส่วนที่เป็นพื้นหลัง และ 1 แทนส่วนที่เป็นตัวอักษร หลักการเบื้องต้นคือการหาค่าพิกเซลที่เป็น 0 ที่ต่อเนื่องกันตลอดแนวตั้ง และแนวนอนทำให้ได้ขนาดของกรอบ (Block) ข้อมูลภาพวัตถุที่มีขนาดต่างๆ กันจากนั้นจะพิจารณาเลือกขนาดของกรอบที่ต้องการจากความแตกต่างของจำนวนพิกเซล ความสูง ความกว้าง และตำแหน่ง เป็นต้น ซึ่งจะได้กรอบของตัวอักษรที่ต้องการ

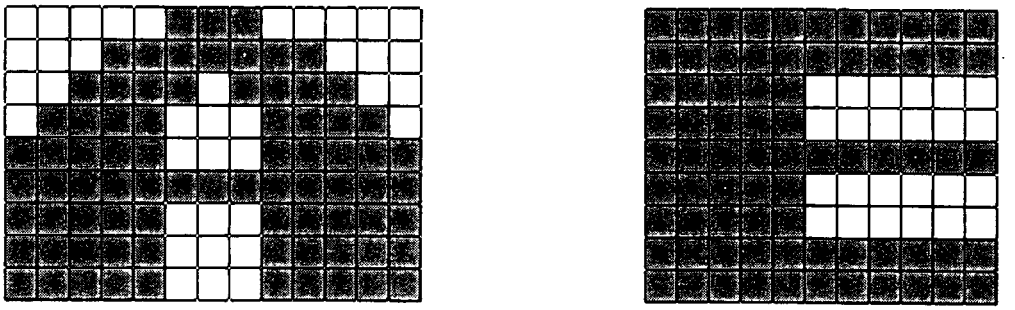
ตัวอย่างการแยกข้อมูลภาพที่มีตัวอักษร 2 ตัว เพื่อให้เห็นแนวทางในกระบวนการแยกเป็นขั้นตอนง่ายๆ ดังนี้ จากข้อมูลภาพ ดังรูปที่ 2.3 จะพิจารณาค่าพิกเซลที่เป็น 0 ที่ต่อเนื่อง ตลอดระยะทางระหว่างระยะขอบเขตระยะด้านบนและระยะด้านล่างของแต่ละบรรทัด (สแกนตามแนวตั้ง) แล้วหาขอบเขตระยะด้านหน้าและด้านหลังของแต่ละตัวอักษร (สแกนตามแนวนอน) เป็นขอบเขตของแต่ละตัวอักษร เพื่อใช้ในการประมวลผลต่อไป ดังแสดงในรูปที่ 2.4 จะได้ระยะ V1, V2, V3, V4 เป็นขอบเขตของระยะด้านหน้าและด้านหลังของตัวอักษร และจะได้ระยะ H1, H2 เป็นขอบเขตของระยะด้านบนและระยะด้านล่างของตัวอักษรตามลำดับ นั่นคือ เมื่อพิจารณาขนาด ความกว้าง และความสูงในช่วงที่ยอมรับ และความสัมพันธ์ของความกว้าง (V2 - V1) ที่มีค่ามากกว่าความสูง (H2 – H1) ซึ่งเป็นคุณสมบัติของตัวอักษรที่ต้องการ ก็จะสามารถแยกตัวอักษรที่ต้องการจากข้อมูลภาพทั้งหมดได้ดังรูปที่ 2.5



รูปที่ 2.3 รูปแสดงข้อมูลภาพแบบไบนารี



รูปที่ 2.4 แสดงตัวแปรที่ใช้ในการแยกตัวอักษรออกทีละตัวอักษร

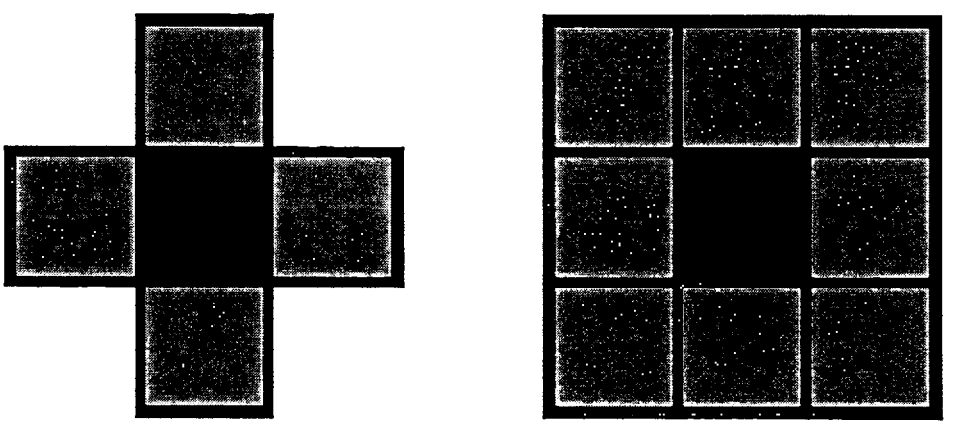


รูปที่ 2.5 แสดงผลของการแยกตัวอักษร

2.5.2 การแยกภาพด้วยวิธี Region Labeling

ในการจำแนกภาพโดยวิธีนี้ได้ถือว่าบริเวณที่อยู่ข้างเคียงเป็นบริเวณที่สำคัญมาก จุดภาพที่อยู่ข้างเคียงกันมักจะมีคุณสมบัติทางสถิติที่คล้ายกันหรือใกล้เคียงกันสำหรับจุดรอบข้างที่มาเชื่อมต่อกัน ในวิธีนี้จะทำการพิจารณาภาพบริเวณย่อยๆ จำนวนมาก จากนั้นพื้นที่ที่ติดกันจะถูกนำมาพิจารณาถึงความเป็นเนื้อเดียวกันร่วมกัน การรวมตัวกันจะสิ้นสุดลงเมื่อพื้นที่ข้างเคียงไม่สามารถพิจารณาถึงความเป็นเนื้อเดียวกันได้ แต่ถ้าจุดภาพที่อยู่ใกล้เคียงกันนั้นตรวจสอบแล้วไม่อยู่ในเกณฑ์การรวม จุดภาพนั้นจะไม่ถูกรวมเข้าไปในส่วนนั้นของภาพแต่จะถูกเลือกให้เป็นจุดเริ่มต้นส่วนอื่นๆ ต่อไป และหลังจากที่จุดภาพทุกจุดได้รวมตัวกันเป็นกลุ่มเรียบร้อยแล้ว

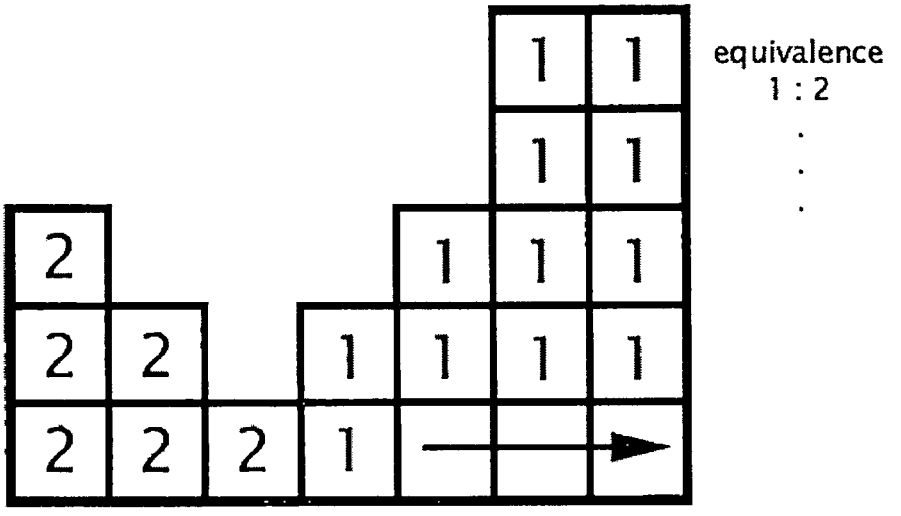
ในกรณีนี้จะกล่าวถึงภาพที่มีวัตถุในภาพมาก วิธีการที่จะแยกแต่ละวัตถุออกจากกันจะทำได้โดยพิจารณาจากการติดกันของพิกเซลที่เป็น 1 โดยสามารถพิจารณาได้ดังนี้



รูปที่ 2.6 การพิจารณาพิกเซลที่ติดกัน (ก) ติดกันแบบ 4 จุด (ข) ติดกันแบบ 8 จุด

- การติดกันแบบ 4 จุด จะพิจารณา 4 พิกเซล รอบข้างทางด้านแนวนอน และ แนวตั้ง แสดงดังรูปที่ 2.6 (ก)
- การติดกันแบบ 8 จุด จะพิจารณา 8 พิกเซล รอบข้างทางด้านแนวนอน และ แนวตั้ง แสดงดังรูปที่ 2.6 (ข)

วิธีการแยกวัตถุแบบ Region Labeling นี้ จะพิจารณาภาพแบบไบนารีเฉพาะพิกเซลที่มีค่า 1 ทีละแถวจากซ้ายไปขวาและจากบนลงล่าง ซึ่งเมื่อพิจารณาที่พิกเซลใดพิกเซลหนึ่ง ถ้าพิกเซลแถวบนและทางซ้ายซึ่งผ่านการกำหนด Label แล้ว มีค่าไม่เป็น 1 ก็จะกำหนด Label ใหม่ให้กับพิกเซลนี้ ถ้าพิกเซลใดพิกเซลหนึ่งมีค่าเป็น 1 ก็จะกำหนด Label เหมือนกับพิกเซลข้างเคียงที่เป็น 1 แต่ถ้าในกรณีที่มีพิกเซลข้างเคียงมีค่า 1 มากกว่า 1 พิกเซล และแต่ละพิกเซลมี Label ต่างกัน ก็จะกำหนด Label ให้กับพิกเซลที่พิจารณาอยู่เหมือนกับ Label ของพิกเซลใดพิกเซลหนึ่ง และบันทึก Label ที่สมมูลกันไว้ ทำเช่นนี้ไปเรื่อยๆ จนครบทุกพิกเซล แล้วจึงเปลี่ยน Label ที่สมมูลกันให้เหมือนกัน ซึ่งจะสามารถทราบถึงความแตกต่างของแต่ละวัตถุในภาพ โดยดูจาก Label ที่ต่างกัน



รูปที่ 2.7 การทำ Region Labeling

บทที่ 3
โครงข่ายประสาทเทียม

3.1 ทฤษฎีโครงข่ายประสาทเทียม

นิวรอลเน็ตเวิร์คเป็นความพยายามอีกอย่างของมนุษย์ในการที่จะลอกเลียนแบบธรรมชาติ โดยทั่วไปแล้วนิวรอลเน็ตเวิร์คจะประกอบขึ้นจากหน่วยย่อยที่มีหน้าที่การทำงานคล้ายคลึงกับการทำงานของมนุษย์ จากนั้นหน่วยย่อยเหล่านี้จะถูกจัดเชื่อมต่อกันเป็นโครงข่ายในลักษณะที่สัมพันธ์กับกายวิภาคของสมองหรือบางครั้งอาจจะไม่สัมพันธ์ก็ได้ และโดยการเชื่อมต่อกันในโครงข่ายนี้ทำให้นิวรอลเน็ตเวิร์คสามารถแสดงพฤติกรรมมากมายที่มีการตอบสนองคล้ายคลึงกับระบบประสาทในสิ่งมีชีวิตยกตัวอย่างเช่นความสามารถในการที่จะเรียนรู้จากประสบการณ์ การจัดกลุ่มของข่าวสารจากตัวอย่างที่ให้หรือจับลักษณะเด่นของอินพุทที่ประกอบจากข้อมูลที่ไม่สมบูรณ์ เป็นต้น

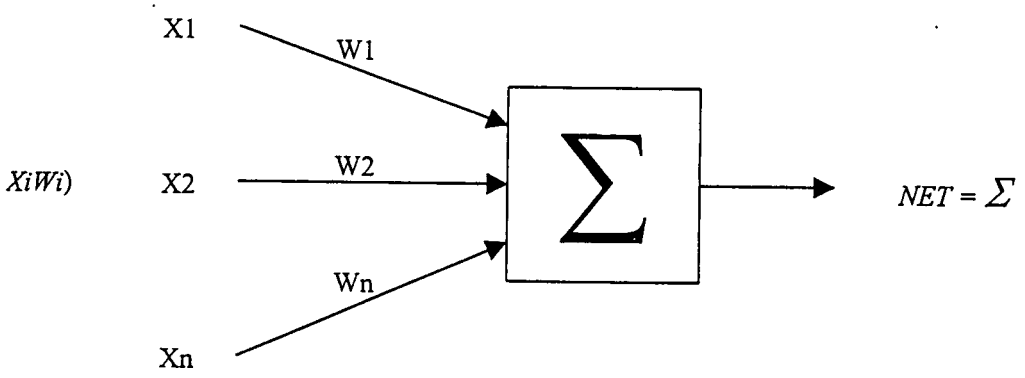
การศึกษาและสิ่งประดิษฐ์ทางด้านนิวรอลเน็ตเวิร์คได้ถูกนำเสนอออกมาทั้งในรูปแบบของฮาร์ดแวร์และซอฟแวร์ โดยผลิตภัณฑ์ที่เป็นฮาร์ดแวร์มักเรียกว่า Neurocomputer ซึ่งจะทำการจำลองการทำงานของเซลประสาทลงบนวงจรทางไฟฟ้าหรือทางออปติกจากนั้นอาจถูกนำมาใช้ร่วมกับคอมพิวเตอร์ฯรรมดาได้ในลักษณะของ Co–processor ซึ่งสามารถเรียกใช้เหมือนกับเป็นโปรแกรมย่อยอันหนึ่งในระบบได้หรือในอีกแนวทางหนึ่งจะอยู่ในรูปของซอฟแวร์โดยสร้างโปรแกรมเพื่อจำลองเซลประสาทและจุดเชื่อมต่อของเซลประสาทเหล่านั้นลงบนคอมพิวเตอร์ เพื่อศึกษาพฤติกรรมของมัน

โครงข่ายนิวรอลเน็ตเวิร์คถ้าพิจารณาความสามารถพิเศษของนิวรอลเน็ตเวิร์ค ดังเช่นความสามารถในการเรียนรู้จากประสบการณ์หรืออีกหลายๆ อย่าง จะเห็นว่าความสามารถเหล่านั้นสามารถทได้จากวิชาการทางด้านปัญญาประดิษฐ์ AI (Artificial Intelligence) ซึ่งจะสามารถเรียนรู้และจดจำ ซึ่งมีกระบวนการคล้ายคลึงกับกระบวนการทางสมองของมนุษย์ โดยเฉพาะทางด้านระบบผู้เชี่ยวชาญ (Expert System) ซึ่งประสบความสำเร็จอย่างมากในหลายที่ แต่ก็ประสบปัญหาความยุ่งยากในการสร้างความสามารถสำหรับการเรียนรู้จากประสบการณ์ไม่น้อย จนหลายคนมองว่านิวรอลเน็ตเวิร์คอาจเข้ามาแทนที่ AI แต่ที่จริงแล้วควรนำเทคนิคทั้งสองแบบมาใช้ร่วมกันได้เป็นระบบโดยที่แต่ละตัวทำงานในส่วนที่ตนเองถนัดมากกว่า

ความแตกต่างระหว่างคอมพิวเตอร์ธรรมดากับนิวรอลเน็ตเวิร์ค อีกอย่างหนึ่งคือในคอมพิวเตอร์ธรรมดาเมื่อต้องการให้การทำงานอย่างใดสามารถทำได้โดยการสร้างโปรแกรมแต่ในทางตรงกันข้ามนิวรอลเน็ตเวิร์คเมื่อต้องการให้การทำงานอย่างใดสามารถทำได้โดยการสอน

(Training) โดยการฝึกสอนนี้จะทำให้นิวรอลเน็ตเวิร์คนำบทเรียนที่ได้จากการฝึกเข้าไปเก็บในโครงสร้างของนิวรอล และจากนั้นจะถูกเรียกกลับขึ้นมาใช้เมื่ออยู่ในระหว่างประมวลผลข้อมูล

การออกแบบสร้างประสาทเทียมมีสมมติฐานจากโครงข่ายประสาทชีวภาพ ซึ่งมีความนำสัญญาณไฟฟ้าเคมีต่างกัน โมดูลโครงข่ายประสาทเทียมจึงต้องมีการถ่วงน้ำหนักก่อนนำไปใช้งาน เรียกว่า ไซแนปติกเวจท์ ปริมาณข้อมูลจะถูกนำมารวมกัน และตัดสินใจด้วยระดับความสนใจของนิวรอล (Activation Level) แล้วส่งเอาท์พุท(Output) ออกไปยังนิวรอลอื่นๆ



รูปที่ 3.1 ส่วนของนิวรอล 1 หน่วย

จากรูปที่ 3.1 เป็นไดอะแกรมที่จำลองจากแนวความคิดของเซลล์สมองทางชีวภาพ สัญญาณอินพุทคือ X1,X2,...,Xn จะถูกป้อนเข้ามายังนิวรอล เปรียบได้กับสัญญาณไฟฟ้าเคมีเข้ามายังไซแนปส์ของเซลล์ประสาท ค่าอินพุทเหล่านี้จะคูณด้วยค่าถ่วงน้ำหนัก (weight) ที่มีค่าตั้งแต่ 0.0-1.0 ผลรวมของสัญญาณทั้งหมดจะส่งออกมาโดย

$$NET = X1W1+X2W2+ \ldots +XnWn \quad \ldots\ldots\ldots\ldots 3.1$$

หรืออาจเขียนเป็นรูปแบบของเวคเตอร์ได้ดังนี้

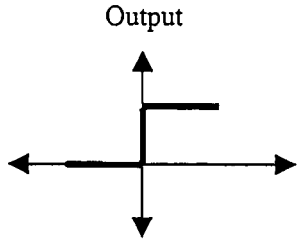
$$NET = \Sigma(XiWi) \quad \ldots\ldots\ldots\ldots 3.2$$

โดย X = เวคเตอร์อินพุท

W = เวคเตอร์น้ำหนัก

NET = เวคเตอร์เอาท์พุท

จากนั้นจะถูกตัดสินใจโดยหน่วยเซลล์ด้วยฟังก์ชันการตัดสินใจ (Activation Function)และได้ค่าเอ่าท์พุทออกมา



รูปที่ 3.2 ฟังก์ชันเส้นตรง

$$OUT = F(NET) \quad \cdots\cdots\cdots 3.3$$

ฟังก์ชันตัดสินใจอาจเป็น hard linear function โดย

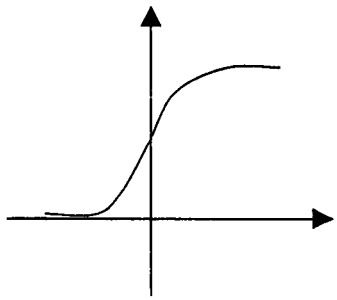
OUT = *1* ถ้า NET > T

= *0* ถ้า NET มีค่าเป็นกรณีอื่น

T เป็นค่าคงที่เทรชโฮล (Threshold) หรืออาจเป็นฟังก์ชันอื่นๆที่เลียนแบบคุณสมบัติไม่คงที่ของเซลล์ประสาทได้ดีกว่า และใช้เป็นฟังก์ชันให้กับโครงข่ายทั่วไปได้ ฟังก์ชันตัดสินใจที่นิยมใช้กันทั่วไป คือ สแควชชิ่งฟังก์ชัน (Squashing function) หรือซิกมอยด์ฟังก์ชัน (Sigmoid function) ซึ่งมีรูปร่างคล้าย

$$f(x) = \frac{1}{1+\exp(-x)} \quad \cdots\cdots\cdots 3.4$$

ตัว "S" และมีสมการทางคณิตศาสตร์ดังนี้

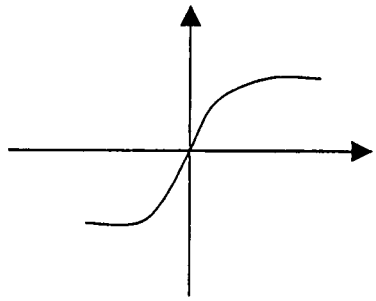


รูปที่ 3.3 ซิกมอยด์ ฟังก์ชัน(Sigmoid function)

การใช้ซิกมอยด์ฟังก์ชัน จะทำให้เทรชโฮลฟังก์ชันมีลักษณะ ฟังก์ชันไม่เป็นเชิงเส้นทำให้ได้ค่าอินพุทที่ไวต่อสัญญาณขนาดเล็กๆ และเฉลี่ยต่อสัญญาณแรงๆ (คือสัญญาณไปทางบวกเพียงเล็กน้อยก็จะมีค่าเอาท์พุทใกล้เคียง "1" และสัญญาณเป็นลบค่าต่ำๆ ใกล้เคียง "0" ขณะเดียวกัน สัญญาณบวกขนาดแรงๆก็ยังคงให้ค่าเอาท์พุทใกล้เคียง "1" และสัญญาณลบค่ามากๆก็ยังคงให้เอาท์พุทใกล้เคียงศูนย์ได้ คือมีคุณสมบัติ non linear gain นั่นเอง ซึ่ง คลอสสรี (Grossberg,1973) พบว่าสามารถแก้ปัญหา Noise saturation dilemma ได้ และทำให้นิวรอลสามารถทำงานได้กว้างขวางขึ้น

นอกจากนี้ยังมีฟังก์ชันอื่นๆอีกคือ ไฮเปอร์โบลิกฟังก์ชัน (Hyperbolic function) ซึ่งมีลักษณะคล้ายกับ logistic function คือ

$$OUT = \tanh(X) \qquad \text{.....3.4}$$



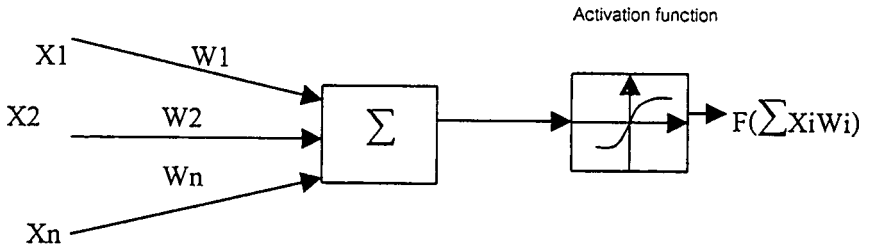
รูป 3.4 hyperbolic tangent function

3.2 โครงข่ายประสาทเทียมแบบชั้นเดียว

(Single Layer Artificial Neural Network)

ถึงแม้ว่าลักษณะของนิวรอลเพียง 1 หน่วย ที่ประกอบด้วยฟังก์ชันบางอย่างได้ แต่ประสิทธิภาพที่เพิ่มขึ้นจากการทำงานมาจากการรวมนิวรอลหลาย ๆ หน่วยเข้าด้วยกัน

ลักษณะรูปแบบอย่างง่ายที่สุดของโครงข่ายนิวรอล จะจัดกลุ่มของนิวรอลมีลักษณะ 1 ชั้นดังแสดงดังรูปที่ 3.5

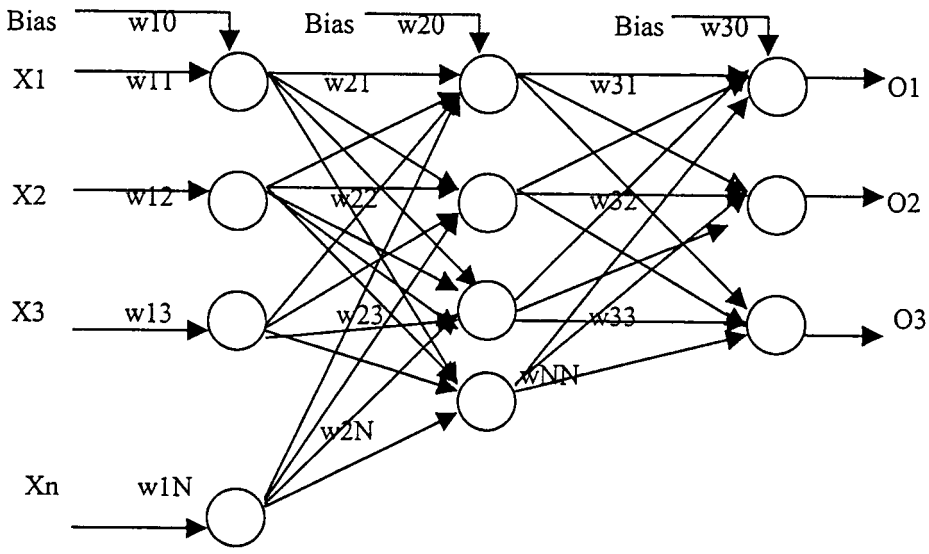


รูปที่ 3.5 ลักษณะของนิวรอล 1 หน่วย ที่ประกอบด้วยฟังก์ชันกระตุ้น

3.3 โครงข่ายประสาทเทียมแบบหลายชั้น

(Multilayer Artificial Neural Network)

ซึ่งประกอบด้วยอินพุทเลเยอร์ (Input Layer) เอาท์พุทเลเยอร์ (Output Layer) และชั้นซ่อน (Hidden Layer) จำนวนตั้งแต่ 1 ชั้นขึ้นไป ทำให้ความสามารถของโครงข่ายดีขึ้น คือสามารถแก้ปัญหาที่ไม่เป็นเชิงเส้นได้ ปัญหาที่มักเกิดขึ้นจากการออกแบบโครงข่ายประสาทแบบหลายชั้นจะเป็นในแง่ของการกำหนดขนาดจำนวนหน่วยในแต่ละชั้น จำนวนชั้นซ่อนที่เหมาะสม และลักษณะการเชื่อมของแต่ละหน่วย แต่ละชั้น



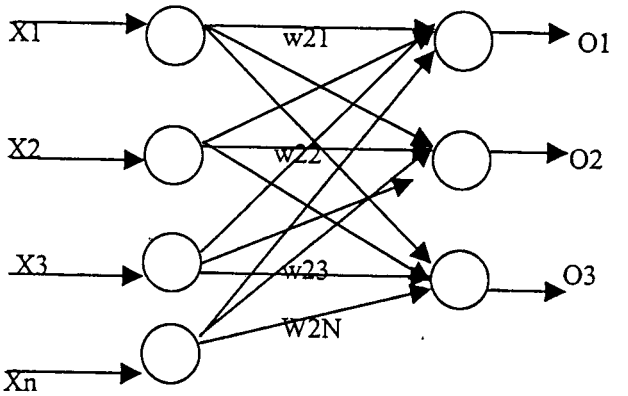
รูป 3.6 โครงข่ายประสาทเทียมแบบหลายชั้น(Multilayer Artificial Neural Network)

3.4 การฝึกสอนให้กับโครงข่าย (Training Algorithm)

เทรนนิ่งอัลกอริทึมถูกจัดเป็น 2 ประเภท คือ แบบควบคุม (Supervise training) และ แบบอิสระ(Unsupervised) โดยเทรนนิ่งแบบควบคุมต้องการข้อมูล 2 ชุดคือ อินพุท กับ ชุดเป้าหมาย เรียกเทรนนิ่งแพร์ (Training pairs) เช่น การเรียนรู้แบบแพร่ย้อนกลับ ส่วนการเทรนนิ่งแบบอิสระสร้างขึ้นโดยเปรียบเทียบการทำงานของสมองที่ไม่จำเป็นต้องมีผู้มาคอยคิดค่าเป้าหมายให้ก่อน โครงข่ายจะรับเพียงค่าอินพุทเข้าไปเท่านั้น เทรนนิ่งอัลกอริทึมจะเปลี่ยนแปลงค่าถ่วงน้ำหนักของโครงข่าย เพื่อสร้างเอาท์พุทที่มั่นคง และเมื่อค่าอินพุทมีการเปลี่ยนแปลงไปเพียงเล็กน้อย โครงข่ายจะยังสามารถชี้ได้ว่าเป็นข้อมูลเดิม เนื่องจากเอาท์พุทของโครงข่ายไม่มีการกำหนดมาก่อน ส่วนใหญ่จะถูกแปรรูปให้เข้าใจได้ จึงไม่สามารถใช้ตัดสินใจปัญหาที่มีความยากได้ มักใช้ในงานง่ายๆ เช่น การเปรียบเทียบเอกลักษณ์ รูปภาพ รูปแบบที่สัมพันธ์กันระหว่างอินพุท-เอาท์พุท

3.5 ขั้นตอนการเรียนรู้แบบแพร่ย้อนกลับ (Backpropagation Learning)

การฝึกสอนให้กับโครงข่ายประสาทแบบหลายชั้นใช้การเรียนรู้แบบแพร่ย้อนกลับ โดยโครงข่ายจะรับค่าอินพุท X และค่าเป้าหมาย (Target) ไว้ เมื่อคำนวณค่าเอาท์พุทในโครงข่ายแล้วจะเปรียบเทียบค่าผิดพลาดของเอาท์พุทที่คำนวณได้กับค่าเป้าหมาย



รูป 3.7 ไดอะแกรมของ Backpropagation Neural Network แบบ 2 ชั้น

Error = *Target-Output*

ค่าผิดพลาดนี้จะนำไปใช้ปรับค่าถ่วงน้ำหนักของโครงข่ายเพื่อให้สอดคล้องกับค่าเป้าหมาย การเรียนรู้จะทำซ้ำไปเรื่อยๆจนโครงข่ายสามารถแยกแยะข้อมูลได้

ขั้นตอนการฝึกสอนสามารถสรุปเป็นขั้นๆได้ดังต่อไปนี้

1. สร้างโครงข่ายนิวรอลที่มีจำนวนอินพุท และเอาท์พุทของปัญหาส่วนจำนวนชั้นซ่อนอาจเริ่มที่ค่ามากพอสมควรก่อน
2. ให้ค่าเริ่มต้นแก่ค่าถ่วงน้ำหนักในโครงข่าย โดยควรมีค่าในช่วง –1 ถึง 1
3. คำนวณค่า $\sum XiWi = Ui$ สำหรับแต่ละนิวรอล j ของชั้นซ่อน
4. คำนวนค่ากระตุ้น (Activation) สำหรับแต่ละนิวรอล j ของชั้นซ่อน $Yj=(Ui)$ โดยฟังก์ชันที่นิยมใช้คือ sigmoid function $f(x) = 1/[1+\exp(-x)]$
5. คำนวณค่า $\sum YiWi = Vi$ สำหรับแต่ละนิวรอล k ของชั้นเอาท์พุท
6. คำนวนค่ากระตุ้น ของแต่ละนิวรอล k โดย $OUT = Zk = f(Vi)$
7. นำค่า Zk มาเปรียบเทียบกับเป้าหมายถ้าค่าผิดพลาดที่คำนวณได้น้อยกว่าระดับที่กำหนดไว้ ก็จบการเรียนรู้ มิฉะนั้นก็ทำต่อไป
8. คำนวณค่าปรับน้ำหนักสำหรับแต่ละค่าถ่วงน้ำหนัก

   8.1 สำหรับเส้นเชื่อมชั้นซ่อนกับชั้นเอาท์พุท

   $$\Delta Wj = \alpha \delta k Yj$$

   โดยที่ $\delta j = (\text{Target } k - Zk) f'(Vk)$

   8.2 สำหรับเส้นเชื่อมชั้นอินพุทกับชั้นซ่อน

   $$\Delta Wij = \alpha \delta j Xi$$

   โดยที่ $\delta j = \sum \delta k Wjk f'(Uj)$

   ค่า α : อัตราการเรียนรู้ (Learning rate) มีค่าในช่วง 0 ถึง 1
9. ปรับค่าน้ำหนักเส้นเชื่อมจากนิวรอล r ไปนิวรอล s จะได้

   $$Wrs(new) = Wrs(old) + \Delta Wrs$$
10. กลับไปทำข้อ 3

การปรับความเร็วในการเรียนรู้อาจใช้การปรับค่า Gain ของ sigmoid function ในส่วนดีกรีของ exponential การปรับอัตราการเรียนรู้ ขนาดจำนวนหน่วยในชั้นซ่อน และการเพิ่มโมเมนตัมแฟก

เตอร์เพื่อคำนวณทิศทางการปรับค่าถ่วงน้ำหนัก นอกจากนี้ยังเลือกใช้อัตราการเรียนรู้เป็นฟังก์ชันที่มีความเหมาะสมได้อีกด้วย

ฟังก์ชัน Sigmoid เป็นที่นิยมในการใช้เทรนนิ่งเนื่องจากมีความใกล้เคียงกับฟังก์ชันขั้นบันไดและสามารถคำนวณค่าอนุพันธ์ได้ง่ายอีกด้วย โดยอยู่ในรูปนิพจน์ของตัวฟังก์ชันเอง

$$f'(x) = f(x)[1-f(x)]$$

และจากข้อ 8 มีการใช้กฎของเดลต้าคือกฎนี้ใช้ปรับน้ำหนักโดยมีลักษณะดังนี้

$$\delta = T-A$$

โดยที่ T เป็นค่าเอาท์พุทที่ต้องการ

A เป็นค่าเอาท์พุทที่ออกมาจริง

δ เป็นค่าผลต่างของ 2 ค่าข้างต้น

หลังจากได้ค่าของเดลตาแล้วค่านี้จะถูกนำไปคูณค่ากับค่าอินพุท แต่ละอินพุทเพื่อนำไปใช้ในการปรับค่าน้ำหนักให้เหมาะสม นอกจากนั้นยังต้องถูกนำไปคูณกับค่าอัตราการเรียนรู้เพื่อเพิ่มประสิทธิภาพในการฝึกสอนให้เร็วขึ้นหรือช้าลงซึ่งเขียนเป็นสูตรได้ดังนี้

$$\Delta W_{ij} = \alpha \delta_j X_i$$

โดยที่ ΔW_{ij} หมายถึงค่าที่ต้องถูกนำมาใช้ในการปรับน้ำหนักของแต่ละค่าน้ำหนัก

X_i คือค่าอินพุทเวคเตอร์

α_j คือค่าอัตราการเรียนรู้

ดังนั้นสูตรการปรับค่าน้ำหนักสามารถเขียนได้ดังนี้

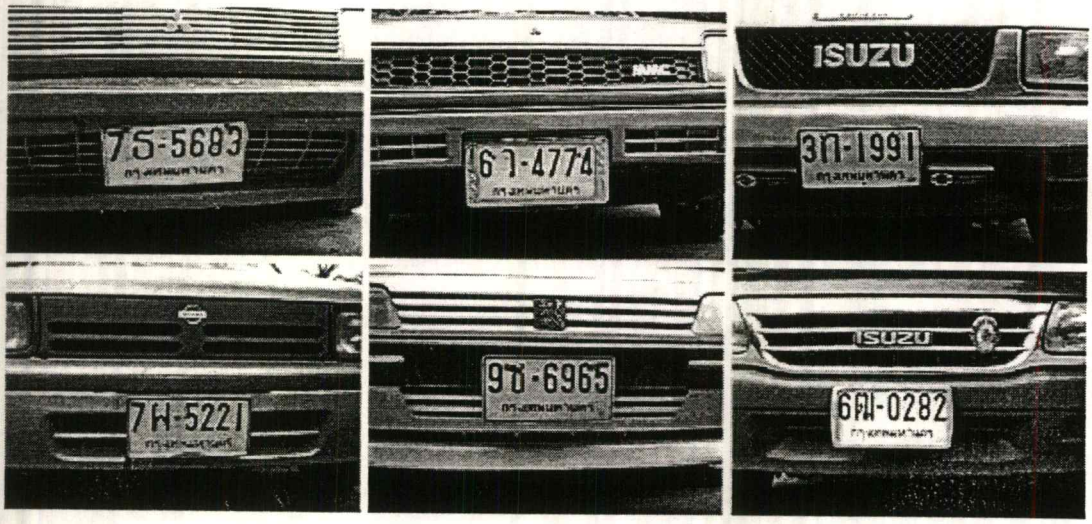
$$W_{ij}(n+1) = W_{ij}(n) + \Delta W_{ij}$$

โดยที่ ΔW_{ij} หมายถึงค่าที่ต้องถูกนำมาใช้ในการปรับน้ำหนักของแต่ละค่าน้ำหนัก

$W_{ij}(n)$ เป็นค่าน้ำหนักหลักการปรับ

$W_{ij}(n+1)$ เป็นค่าน้ำหนักก่อนการปรับ

รูปที่ 4.3 ตัวอย่างภาพที่ถ่ายระยะไกล



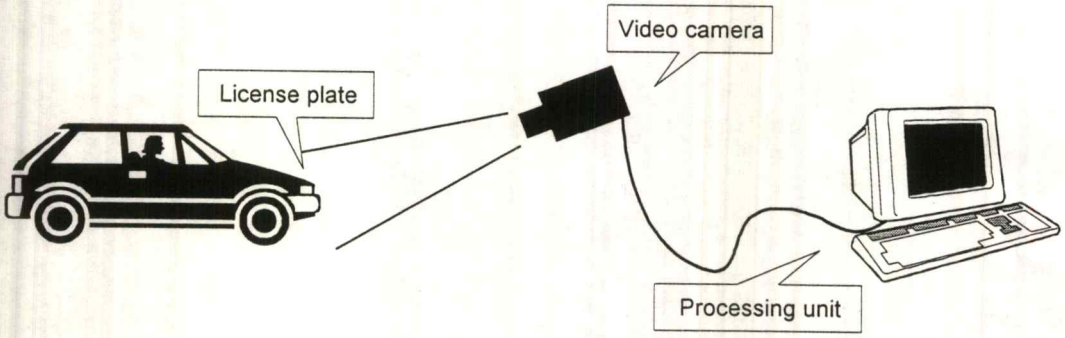
รูปที่ 4.4 ตัวอย่างภาพที่ถ่ายระยะใกล้

รูปที่ 4.5 ตัวอย่างป้ายทะเบียนรถยนต์

บทที่ 4
หลักการทำงานของระบบ

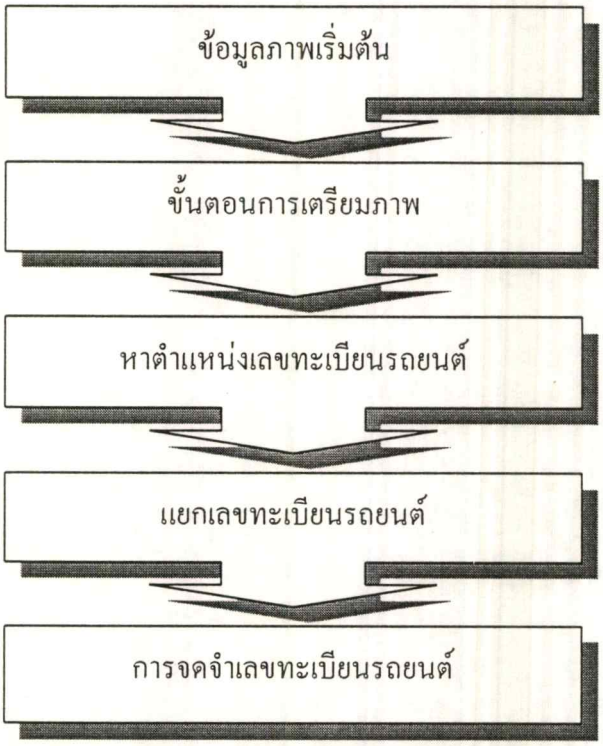
4.1 โครงสร้างของระบบจดจำเลขทะเบียนรถยนต์

ระบบจดจำเลขทะเบียนรถยนต์นี้ ประกอบด้วยส่วนสำหรับรับภาพ และส่วนของโปรแกรมเพื่อประมวลผลภาพ ในส่วนของการรับภาพจะประกอบด้วยกล้องวิดีโอสำหรับจับภาพทางด้านหน้าของรถยนต์ สัญญาณจากกล้องวิดีโอจะต่อผ่านการ์ดวิดีโอเพื่อแปลงสัญญาณจากกล้องให้เป็นสัญญาณดิจิตอลแล้งส่งให้กับคอมพิวเตอร์เพื่อทำการประมวลผล เมื่อรถยนต์ผ่านอุปกรณ์ตรวจจับด้วยแสงคอมพิวเตอร์จะสั่งจับภาพในขณะนั้นมา 1 เฟรม และเก็บข้อมูลภาพแบบบิตแมปเกรย์สเกล 256 ระดับ และนำภาพที่ได้ไปประมวลในส่วนโปรแกรมต่อไป



รูปที่ 4.1 โครงสร้างของระบบ

ในส่วนของโปรแกรมประมวลผลภาพนั้นมีขั้นตอนในการแยกหมายเลขทะเบียนออกจากข้อมูลภาพทั้งหมด ดังแสดงในรูปที่ 4.2 โดยมีขั้นตอนการทำงานของโปรแกรมดังนี้คือในขั้นแรกจะเป็นขั้นตอนของการเตรียมภาพโดยภาพเกรย์สเกล 256 ระดับจะถูกแปลงให้เป็นภาพแบบไบนารีนารีด้วยวิธีการเลือกค่าเทรชโฮล ซึ่งจะเหลือข้อมูลภาพเพียงสองระดับ คือ 0 และ 1 โดย 0 จะเป็นพื้นหลัง ส่วน 1 จะเป็นวัตถุในภาพ จากนั้นจะทำการกำจัดสัญญาณรบกวนที่เกิดขึ้น แล้วจึงแยกวัตถุออกจากกัน แล้วจึงพิจารณาแต่ละวัตถุว่าน่าจะเป็นเลขทะเบียนหรือไม่ โดยอาศัยคุณสมบัติที่สังเกตได้ เช่น ขนาด การเรียงตัว เป็นต้น หลังจากขั้นตอนนี้จะได้ตำแหน่งของเลขทะเบียนรถยนต์ แล้วจึงแยกเลขทะเบียนออกจากภาพเพื่อเข้าสู่ขั้นตอนการจดจำรูปแบบในลำดับต่อไป



รูปที่ 4.2 ขั้นตอนการทำงานของโปรแกรม

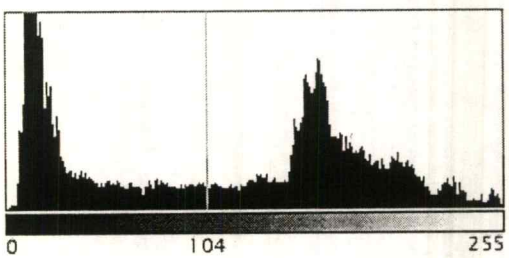
4.2 การเตรียมข้อมูลภาพ

4.2.1 การแปลงข้อมูลภาพไบนารี

ในขั้นตอนแรก จะเป็นการเตรียมข้อมูลภาพเพื่อใช้ในการประมวลผล เนื่องจากระบบยังไม่สามารถทำงานในเวลาตริงได้ ดังนั้นเพื่อความสะดวกจะใช้ข้อมูลภาพที่เก็บเป็นไฟล์บิตแมปไว้แล้วแทนข้อมูลภาพจากกล้องวิดีโอ โดยจะใช้ภาพที่ถ่ายทางด้านหน้าของรถยนต์ ข้อมูลภาพที่ได้นี้จะถูกเก็บเป็นไฟล์บิตแมปแบบเกรย์สเกล 256 ระดับ ตัวอย่างของภาพที่ได้แสดงดังรูป 4.3 และรูปที่ 4.4

ถ้าพิจารณาจากลักษณะของแผ่นป้ายทะเบียนรถยนต์ที่สังเกตได้ดังรูปที่ 4.5 จะพบว่าโดยทั่วไปแผ่นป้ายทะเบียนรถยนต์จะมี 2 สี คือสีขาวซึ่งเป็นสีของพื้นหลังและสีดำเป็นสีของเลขทะเบียนรถยนต์ เมื่อพิจารณาจากตัวอย่างของภาพที่ถ่ายด้านหน้าของรถยนต์ จะเห็นว่าสีของเลขทะเบียนรถยนต์กับพื้นหลัง มีความแตกต่างอย่างชัดเจน กล่าวคือ สีของเลขทะเบียนรถยนต์จะเข้มกว่าพื้นหลัง และนอกจากนี้ เลขทะเบียนรถยนต์แต่ละตัวจะแยกออกจากกันเป็นตัวเดี่ยวๆ จึงสามารถใช้ลักษณะของป้ายทะเบียนรถดังที่กล่าวมาแล้วเพื่อทำการแยกเลขทะเบียนรถยนต์แต่ละตัวออกจากกันได้ ด้วยการแยกข้อมูลในภาพออกเป็นส่วนของสีขาวซึ่งเป็นพื้นหลังของป้ายทะเบียน และส่วนของสีดำซึ่งเป็นส่วนของเลขทะเบียน

รูปที่ 4.6 ภาพตัวอย่างของด้านหน้ารถยนต์



รูปที่ 4.7 ฮิสโทแกรมของภาพในรูปที่ 4.6

รูปที่ 4.6 แสดงตัวอย่างภาพของแผ่นป้ายทะเบียนรถยนต์ที่เป็นภาพบิตแมปแบบเกรย์สเกล 256 ระดับ โดยมีค่าตั้งแต่ 0 ถึง 255 พิกเซลที่สว่างที่สุดจะมีค่าเท่ากับ 255 หรือพิกเซลสีขาว ส่วนพิกเซลที่มีความเข้มมากที่สุดหรือสว่างน้อยที่สุดหรือพิกเซลสีดำจะมีค่าเท่ากับ 0 เมื่อสังเกตฮิสโตแกรมแสดงจำนวนพิกเซลที่ค่าความสว่างต่างๆ ดังรูปที่ 4.7 จะสังเกตเห็นว่าจะมีจำนวนพิกเซลหนาแน่นที่ 2

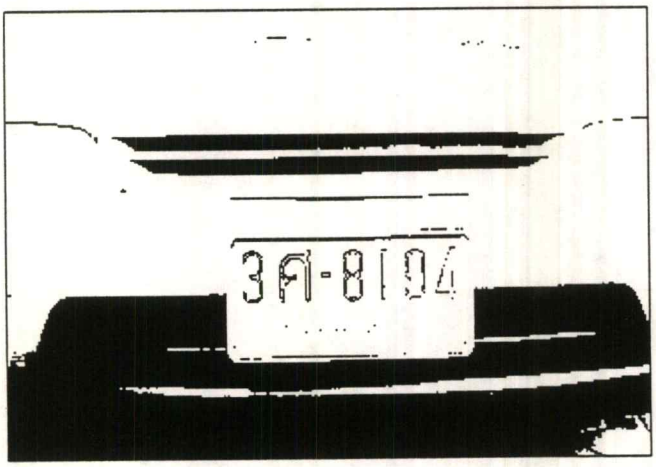
ช่วง คือช่วงที่มีค่าความสว่างต่ำ จะเป็นส่วนของเลขทะเบียนรถและส่วนอื่นๆ ในภาพที่ค่อนข้างมืด และอีกช่วงหนึ่งที่มีค่าความสว่างสูง ซึ่งจะเป็นส่วนสีพื้นของป้ายทะเบียน

เพื่อทำให้ข้อมูลภาพมีรูปแบบที่เหมาะสมในการประมวลผล จึงต้องทำการแปลงข้อมูลของภาพให้เป็นภาพไบนารี คือ มีระดับสีเพียง 2 ระดับได้แก่สีดำเป็น 1 และสีขาวเป็น 0 วิธีการแปลงข้อมูลของภาพให้เป็นไบนารีนั้น ทำได้โดยการเลือกค่าเทรชโฮลที่เหมาะสม โดยข้อมูลภาพที่มีค่าระดับความสว่างสูงกว่าค่าเทรชโฮลจะถูกปรับให้เป็น 0 หรือสีขาว ส่วนค่าระดับความสว่างที่มีค่าต่ำกว่าค่าเทรชโฮลจะถูกปรับให้เป็น 0 หรือสีดำ จากฮิสโตแกรมในรูป 4.7 ถ้าเลือกค่าเทรชโฮลเท่ากับ 104 จะได้ภาพที่ผ่านการแปลงเป็นภาพไบนารีดังแสดงในรูปที่ 4.8

รูปที่ 4.8 ภาพที่ได้จากการแปลงเป็นไบนารี

จากรูป 4.8 ภาพที่เป็นภาพไบนารีแล้วจะได้เลขทะเบียนแยกออกจากพื้นหลังของป้ายทะเบียนเป็นตัวเดี่ยวๆ อย่างชัดเจน เนื่องมาจากการเลือกค่าเทรชโฮลที่เหมาะสม แต่อาจมีบางกรณีที่เลือกค่าเทรชโฮลสูงหรือต่ำเกินไป จะทำให้ไม่สามารถแยกเลขทะเบียนได้ทุกตัวหรือสูญเสียเลขทะเบียนบางตังไป ในรูปที่ 4.7 แสดงภาพที่เลือกค่าเทรชโฮลเท่ากับ 155 ซึ่งมีค่าสูงเกินไป ทำให้เลข 4 เชื่อมติดกับขอบของป้ายทะเบียน และส่วนพื้นหลังของป้ายทะเบียนไม่แยกเป็นสีขาวอย่างชัดเจน จะทำให้ไม่สามารถแยกเลข 4 ออกจากส่วนอื่นของภาพได้ สำหรับในรูปที่ 4.8 แสดงภาพที่เลือกค่าเทรชโฮลเท่ากับ 40 ซึ่งมีค่าต่ำเกินไป จะเห็นว่าพิกเซลที่เป็นส่วนประกอบของเลข 9 และเลข 4 จะไม่เชื่อมติดกัน ซึ่งจะทำให้สูญเสียเลขทะเบียนทั้งสองตัวนี้ได้ ดังนั้นต้องเลือกค่าเทรชโฮลที่เหมาะสมที่สุดในการแปลงเป็นภาพไบนารีเพื่อให้เกิดความเสียหายของข้อมูลน้อนที่สุด

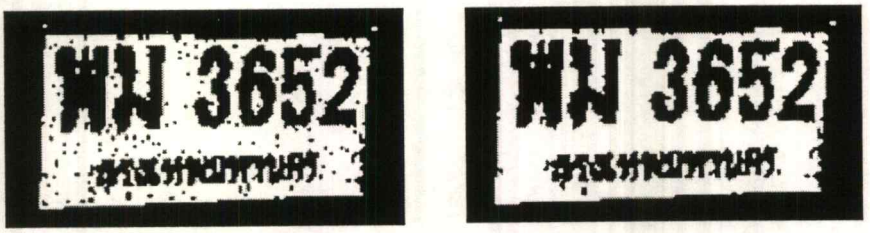
รูปที่ 4.9 แสดงการเลือกค่าเทรชโฮลสูงเกินไป



รูปที่ 4.10 แสดงการเลือกค่าเทรชโฮลต่ำเกินไป

4.2.2 การกำจัดสิ่งรบกวน

ในขั้นตอนของการแปลงเป็นภาพไบนารีนั้น อาจทำให้เกิดพิกเซลสีดำเดี่ยวๆ จำนวนมาก เช่น ในรูปที่ 4.11 (ก) ซึ่งเป็นสิ่งรบกวน (Noise) ทำให้เกิดผลต่อการประมวลผลในขั้นต่อไปได้ ดังนั้นจึงต้องกำจัดพิกเซลที่เป็บสิ่งรบกวนนี้ออกจากภาพ โดยการหาตำแหน่งของพิกเซลสีดำที่ไม่มีพิกเซลข้างเคียงเลย แล้วเปลี่ยนให้เป็นพิกเซลสีขาวแทน ผลของการกำจัดพิกเซลที่เป็นสิ่งรบกวนออกแสดงในรูปที่ 4.11 (ข)



(ก) (ข)

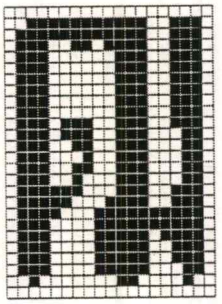
รูปที่ 4.11 การกำจัดสิ่งรบกวนในภาพ

(ก) ภาพที่เกิดสิ่งรบกวนซึ่งเกิดจากการแปลงเป็นภาพไบนารี

(ข) ภาพที่กำจัดสิ่งรบกวนออกแล้ว

4.2.3 การหาตำแหน่งของเลขทะเบียนรถยนต์

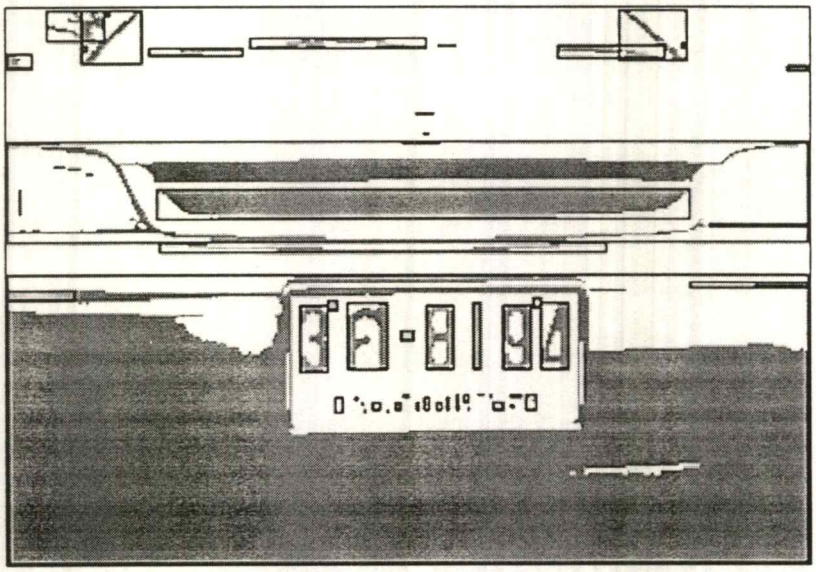
หลังจากที่ภาพผ่านการแปลงให้เป็นไบนารีแล้ว สามารถหาตำแหน่งของเลขทะเบียนรถยนต์ได้ โดยพิจารณาว่าเลขทะเบียนแต่ละตัวคือกลุ่มของพิกเซลสีดำหรือพิกเซลที่มีค่าเป็น 1 ติดกัน ดังนั้นภาพที่ผ่านการแปลงเป็นไบนารีแล้วนี้จะมีกลุ่มของพิกเซลดังกล่าวอยู่หลายกลุ่มทั้งที่เป็นเลขทะเบียนและส่วนอื่น ๆ ของภาพ



รูปที่ 4.12 กลุ่มของพิกเซลติดกันที่รวมกันเป็นเลขทะเบียน

ในการรวมพิกเซลที่มีค่า 1 ติดกัน ใช้หลักการของ Region labeling หรือการกำหนด label ให้กับทุก ๆ พิกเซล โดยพิจารณาทีละพิกเซลตั้งแต่แถวบนสุดจนถึงแถวล่างสุด และจากซ้ายไปขวาในแต่ละแถว เมื่อพิจารณาพิกเซลที่มีค่า 1 ที่ตำแหน่งใดตำแหน่งหนึ่ง พิกเซลที่อยู่แถวบนและพิกเซลทางด้านซ้ายจะถูกพิจารณามาก่อนหน้านี้แล้ว ถ้ามีพิกเซลใด ๆ ของแถวบนและทางซ้ายมีค่า 1 ด้วยก็จะกำหนด label ให้กับพิกเซลที่กำลังพิจารณาอยู่เหมือนกับพิกเซลนั้น ถ้ากรณีที่มีพิกเซลของแถวบนและทางซ้ายมีค่า 1 มากกว่า 1 พิกเซลและมี label ไม่เท่ากัน ก็จะบันทึก label ที่สมมูลกันไว้

เมื่อทำการกำหนด label ให้กับทุก ๆ พิกเซลแล้ว จะทำการพิจารณาแต่ละพิกเซลอีกครั้งหนึ่ง เพื่อเปลี่ยน label ที่สมมูลกันให้เหมือนกัน เมื่อเสร็จสิ้นขั้นตอนนี้แล้ว แต่ละส่วนในรูปก็จะประกอบด้วยพิกเซลที่มี label เหมือนกัน และจะถูกแทนด้วยบล็อกของแต่ละ label



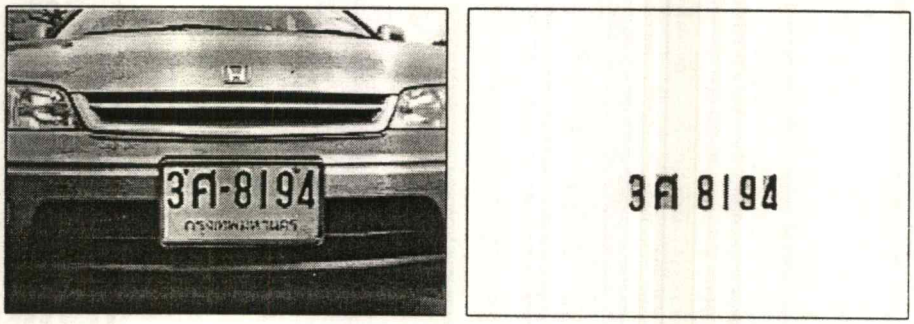
รูปที่ 4.13 การกำหนดบล็อกให้กับพิกเซลที่มี label เหมือนกัน

4.2.4 การแยกเลขทะเบียนออกจากภาพ

หลังจากที่แต่ละส่วนในภาพถูกแทนด้วยบล็อกแล้ว ขั้นตอนต่อไปจะพิจารณาคุณสมบัติของแต่ละบล็อกเพื่อหาบล็อกที่เป็นเลขทะเบียน โดยพิจารณาจากคุณสมบัติของแต่ละบล็อกดังต่อไปนี้

1. ความสูง
2. ความกว้าง
3. ตำแหน่ง

ในขั้นแรก บล็อกที่ไม่ได้ขนาดซึ่งอาจเกิดจากภาพของรถยนต์จะถูกกำจัดโดยพิจารณาจากความสูงและความกว้างของบล็อก โดยจากความสูงและความกว้างของบล็อกต้องมีขนาดจำกัดอยู่ในช่วงหนึ่ง และจากความกว้างของบล็อกต้องน้อยกว่าความสูงของบล็อกด้วย จากนั้นจะสังเกตจากตำแหน่งของบล็อก โดยบล็อกที่เป็นเลขทะเบียนจะต้องมีตำแหน่งเรียงกันตามแนวแกนนอน ตัวอย่างของภาพที่ผ่านการพิจารณาเพื่อแยกเลขทะเบียนแสดงดังรูปที่ 4.14

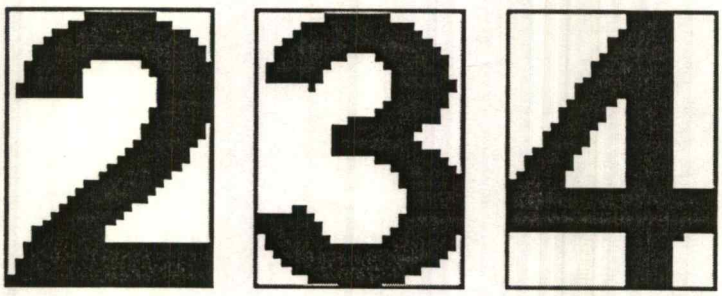


รูปที่ 4.14 รูปที่แยกเลขทะเบียนแล้ว

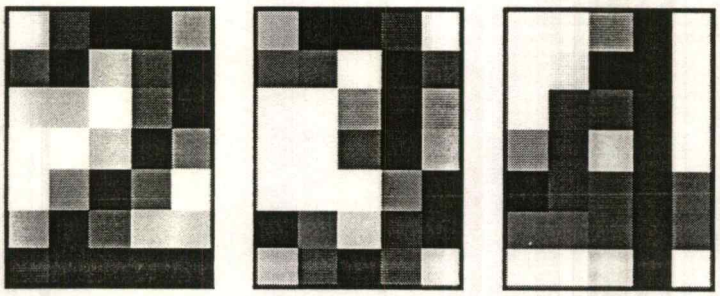
ตัวเลขทะเบียนที่แยกออกมาได้นี้จะเป็นข้อมูลสำหรับส่วนจดจำเลขทะเบียนในลำดับต่อไป

4.3 การจดจำเลขทะเบียนรถยนต์

หลังจากที่เลขทะเบียนรถยนต์ถูกแยกออกจากกันแต่ละตัวแล้ว ก็จะนำข้อมูลของตัวเลขทะเบียนแต่ละตัวเข้าสู่กระบวนการจดจำรูปแบบตัวอักษร โดยใช้หลักการของโครงข่ายประสาทเทียม ซึ่งผ่านกระบวนการฝึกหัดมาก่อนหน้านี้แล้ว สำหรับการเตรียมข้อมูลให้กับโครงข่ายประสาทเทียมนั้น ข้อมูลของตัวเลขทะเบียนรถยนต์แต่ละตัวจะถูกทำให้มีขนาดเท่ากัน และจะถูกแบ่งออกเป็นบล็อกหรือพื้นที่สี่เหลี่ยมเล็กๆ ข้อมูลของแต่ละพื้นที่จะมีค่าระหว่าง 0-1 ซึ่งจะได้มาจากเปอร์เซ็นต์ของจำนวนพิกเซลสีดำในแต่ละพื้นที่ บล็อกที่มีพิกเซลสีดำทั้งหมดจะมีค่าเป็น 1 บล็อกที่มีพิกเซลสีขาวทั้งหมดจะมีค่าเป็น 0 ถ้าในบล็อกมีพิกเซลสีดำอยู่ครึ่งหนึ่งจะมีค่าเป็น 0.5 รูปที่ 4.15 แสดงตัวอย่างของตัวเลขที่ผ่านการแยกออกจากข้อมูลภาพ เมื่อผ่านการแบ่งออกเป็นพื้นที่สี่เหลี่ยมเล็กๆ และแทนค่าของแต่ละพื้นที่ด้วยเปอร์เซ็นต์ของจำนวนพิกเซลสีดำในแต่ละพื้นที่ จะได้ข้อมูลดังแสดงในรูปที่ 4.16



รูปที่ 4.15 แสดงตัวอย่างของตัวเลขที่ผ่านการแยกออกจากข้อมูลภาพ



รูปที่ 4.16 การแทนแต่ละพื้นที่ด้วยเปอร์เซ็นต์ของจำนวนของพิกเซลสีดำ

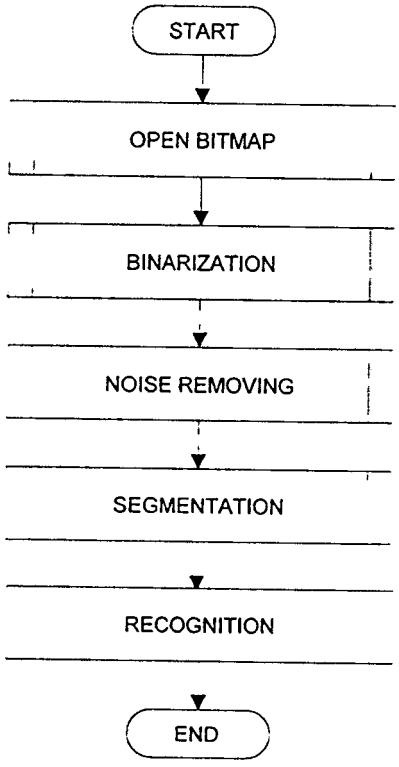
โครงข่ายประสาทเทียมที่ใช้ในกระบวนการจดจำตัวเลขทะเบียนนี้จะถูกฝึกหัดด้วยข้อมูลที่กำหนดไว้ก่อนแล้ว และเมื่อผ่านการฝึกหัดแล้วโครงข่ายประสาทเทียมก็จะสามารถจดจำลักษณะของตัวเลขและตัวหนังสือแต่ละตัว โดยอยู่ในรูปของค่าถ่วงน้ำหนักสำหรับแต่ละข้อมูลและสามารถตัดสินใจได้ว่าตังเลขทะเบียนนั้นเป็นตัวอะไร

บทที่ 5
โครงสร้างของโปรแกรม

ในส่วนของโปรแกรมที่ใช้ประมวลผลภาพในระบบจดจำทะเบียนรถยนต์นี้ เขียนขึ้นด้วย Delphi 4 ซึ่งอยู่บนพื้นฐานของภาษาปาสคาล โดยในการเขียนโปรแกรมเพื่อทำการประมวลผลภาพนี้ ได้แบ่งออกเป็น ส่วนของโปรแกรมหลัก และส่วนของโปรแกรมย่อยเพื่อแยกการประมวลผลออกเป็นส่วน ๆ

5.1 โปรแกรมหลัก

ส่วนของโปรแกรมหลัก จะมีขั้นตอนในการเรียกใช้โปรแกรมย่อยดังแสดงในรูปที่ 5.1

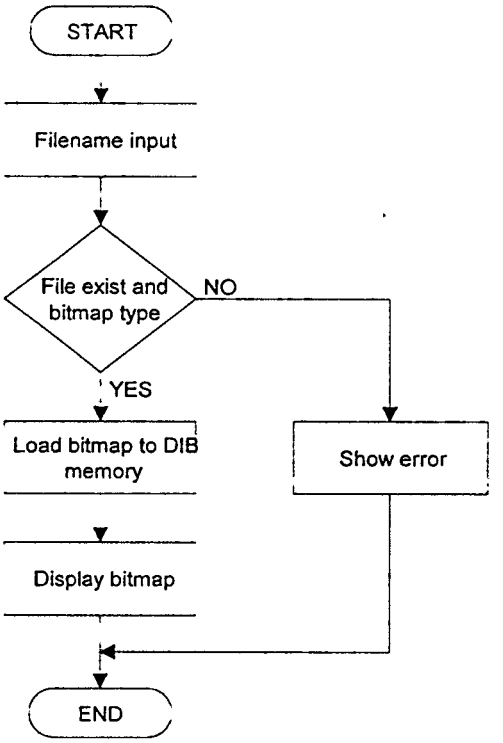


รูปที่ 5.1 แผนภาพการทำงานของโปรแกรมหลัก

ส่วนของโปรแกรมหลักจะเป็นการเรียกโปรแกรมย่อยต่างๆ เพื่อทำการประมวลผลภาพตามลำดับการทำงาน สำหรับการทำงานของโปรแกรมย่อยแต่ละโปรแกรมจะอธิบายในหัวข้อต่อไป

5.2 โปรแกรมเปิดไฟล์ภาพบิตแมป

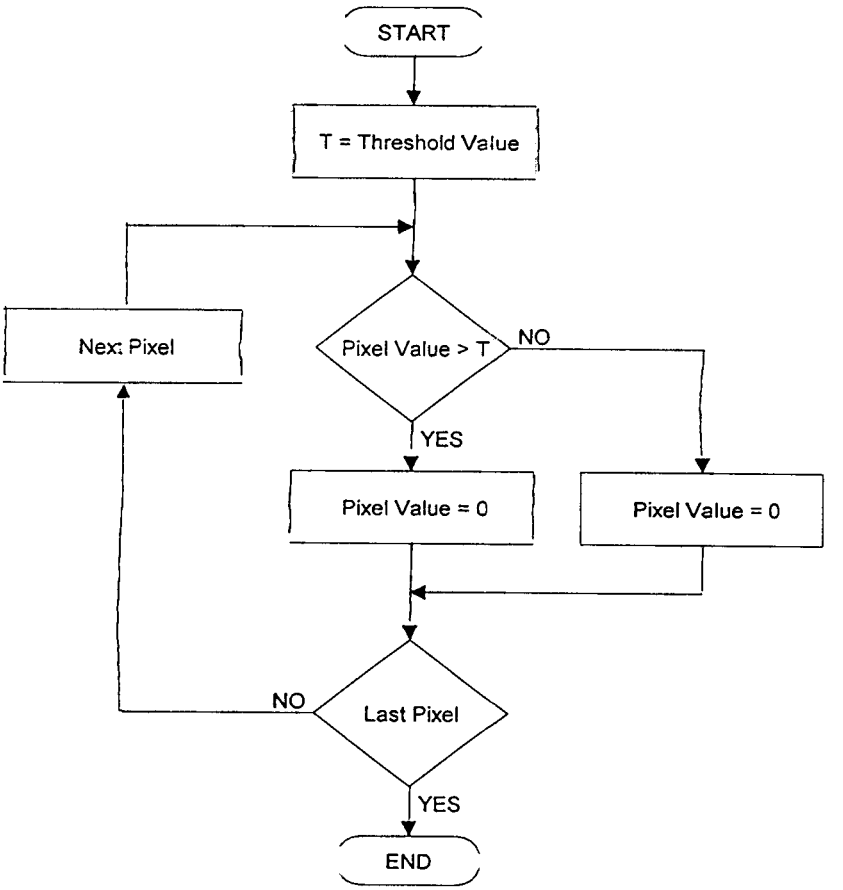
ในการเปิดไฟล์ภาพบิตแมปเพื่อนำข้อมูลภาพมาประมวลผลในขั้นต่างๆ จะนำข้อมูลจากไฟล์ภาพแบบบิตแมปลงในหน่วยความจำ แล้วเข้าไปจัดการกับข้อมูลภาพในหน่วยความจำโดยตรง ซึ่งเรียกลักษณะของข้อมูลภาพในหน่วยความจำนี้เป็น Device Independent Bitmap หรือ DIB



รูปที่ 5.2 แผนภาพการเปิดไฟล์ภาพบิตแมป

5.3 โปรแกรมแปลงเป็นภาพไบนารี

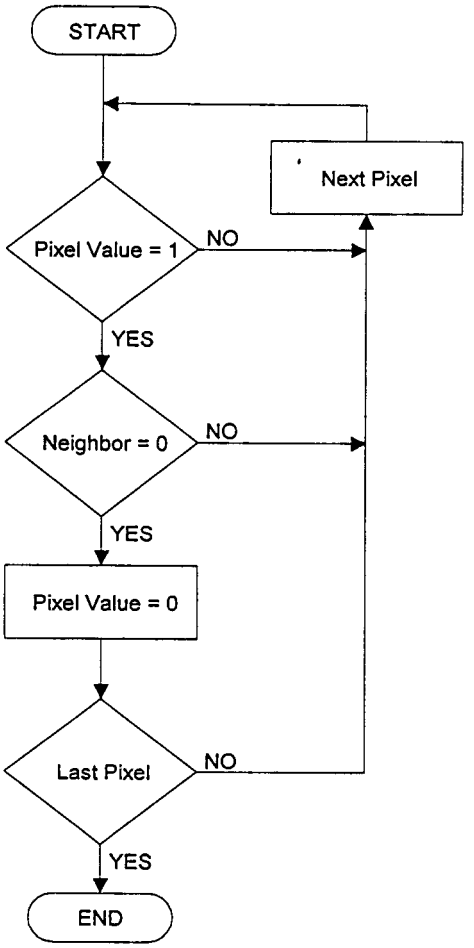
ก่อนที่จะนำภาพไปประมวลผล ต้องทำการแปลงภาพเกรย์สเกล 256 ระดับเป็นภาพแบบไบนารีก่อน โดยจะใช้วิธี Threshold โดยมีขั้นตอนดังรูปที่ 5.3



รูปที่ 5.3 แผนภาพการทำงานของการแปลงเป็นภาพไบนารี

5.4 โปรแกรมกำจัดสิ่งรบกวน

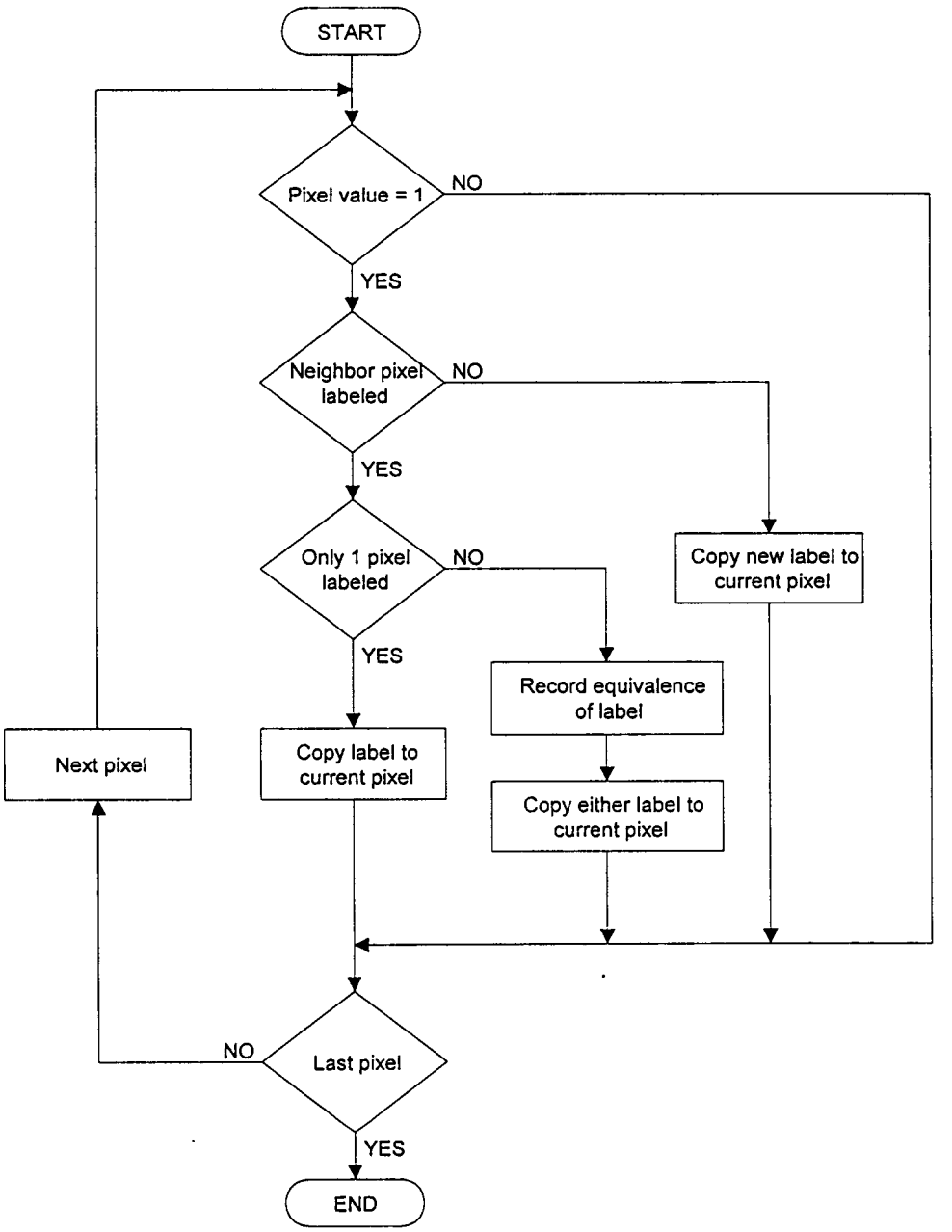
เมื่อภายผ่านการแปลงเป็นภาพแบบไบนารีแล้ว อาจมีพิกเซลเดี่ยวๆ เกิดขึ้น ซึ่งเป็นสิ่งรบกวนของระบบ จึงต้อนผ่านขั้นตอนการกำจัดสิ่งรบกวนก่อน โดยมีขั้นตอนแสดงดังแผนภาพในรูปที่ 5.4



รูปที่ 5.4 แผนภาพการทำงานของโปรแกรมกำจัดสัญญาณรบกวน

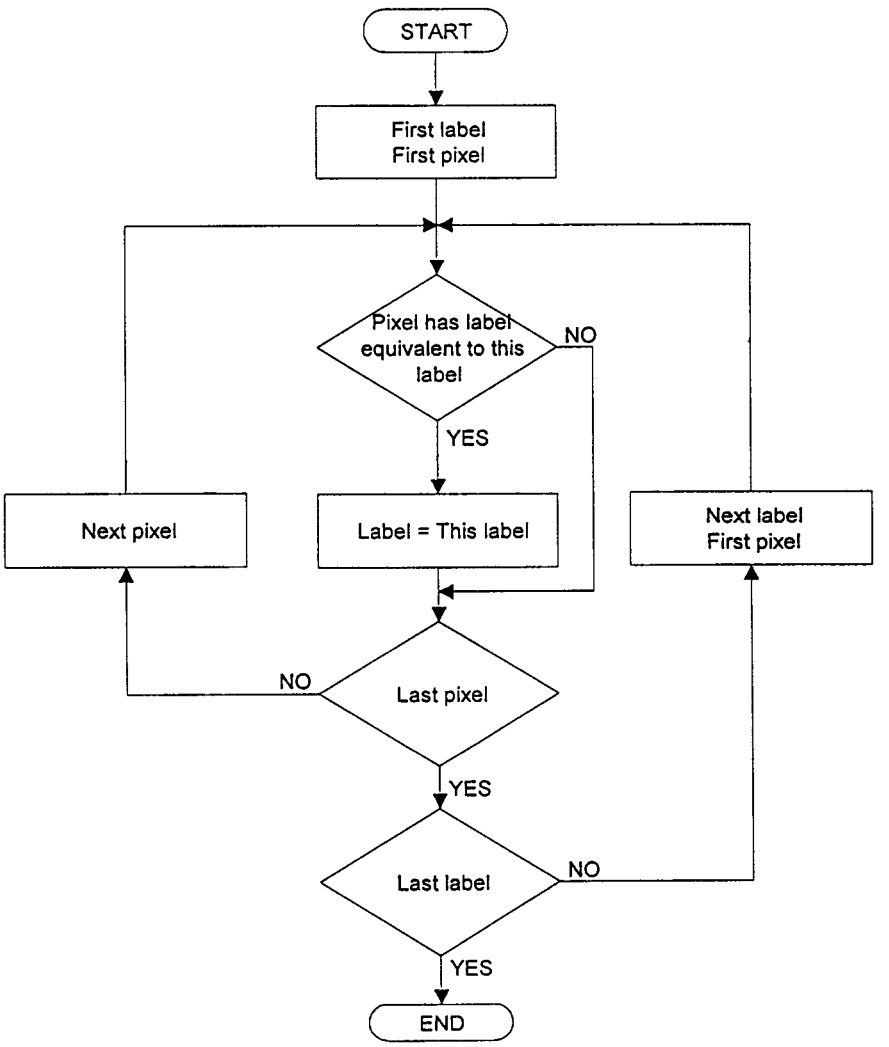
5.5 โปรแกรมแยกเลขทะเบียนรถยนต์

ในส่วนของโปรแกรมแยกเลขทะเบียนรถยนต์ จะประกอบด้วยส่วนย่อยๆ อีก 4 ส่วน ได้แก่ การทำ Region Labeling การรวมพิกเซลที่มี Label ที่สมมูลกันเป็นบล็อกเดียวกัน การพิจารณาขนาดของบล็อก และการหาบล็อกที่เรียงตัวกันในแนวนอน



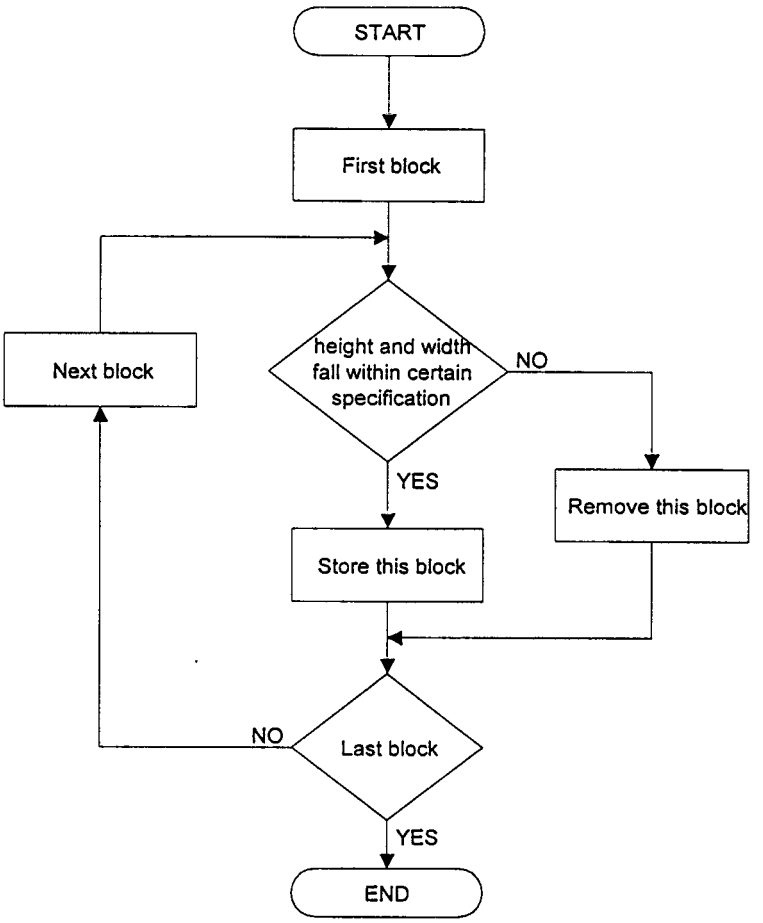
รูปที่ 5.5 แผนภาพการทำงานของ Region Labeling

การทำงานของ Region Labeling จะพิจารณาเฉพาะพิกเซลที่มีค่าเป็น 1 ถ้าพิกเซลที่พิจารณามีพิกเซลข้างเคียงเป็น 1 ด้วย โดยถ้ามีพิกเซลก่อนหน้านี้ซึ่งถูกพิจารณาแล้วติดกับที่พิจารณา 1 พิกเซลก็ให้ label ของพิกเซลที่พิจารณาอยู่เหมือนกับพิกเซลนั้น แต่ถ้าพิกเซลข้างเคียง มากกว่าหนึ่งพิกเซลและมี label ไม่เท่ากันก็ให้ ค่า label ใด label หนึ่งกับพิกเซลที่พิจารณานั้นและบันทึก label ที่สมมูลย์กันไว้



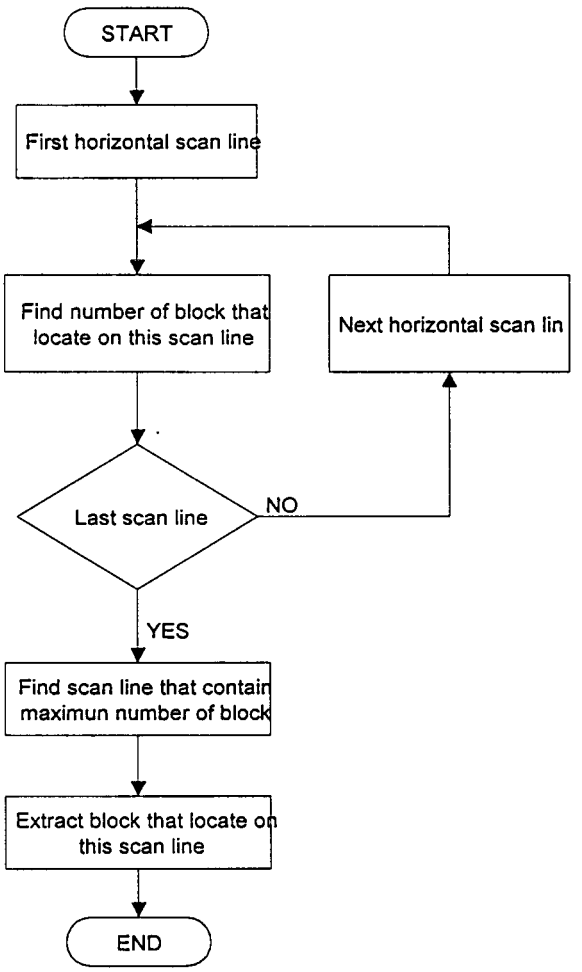
รูปที่ 5.6 แผนภาพแสดงการรวมพิกเซลที่มี Label สมมูลกันเป็นบล็อกเดียวกัน

การรวมพิกเซลที่มี Label สมมูลกันเป็นบล็อกเดียวกัน จะพิจารณาแต่ละ Label ว่ามีพิกเซลใดบ้างที่มี label สมมูลกับ label นี้ ซึ่งถ้าสมมูลกันจะ เปลี่ยน label ของ พิกเซลนั้นๆ เป็น label เดียวกัน



รูปที่ 5.7 แผนภาพแสดงการพิจารณาขนาดของบล็อก

พิจารณาขนาดของบล็อก โดยใช้ความกว้างและความสูงแต่ละบล็อกถ้ามีขนาดในช่วงที่กำหนดจะนำบล็อกนั้นๆ ไปแสดงผลต่อไป

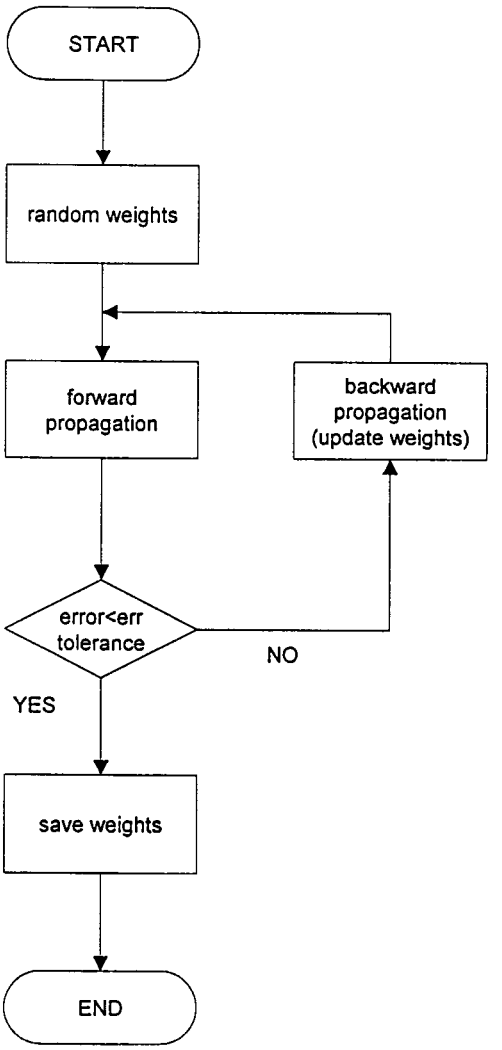


รูปที่ 5.8 แผนภาพแสดงการพิจารณาการเรียงตัวของบล็อกในแนวนอน

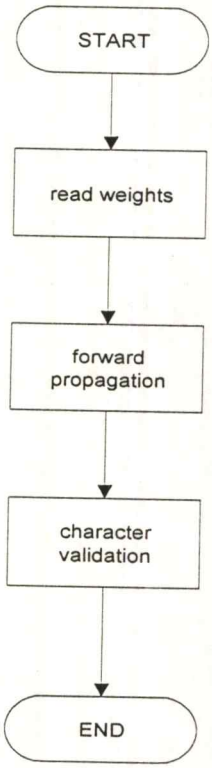
พิจารณาการเรียงตัวของบล็อกในแนวนอนการเรียงตัวของบล็อกตามแนวนอน โดยสแกนตามแนวนอน เปรียบเทียบหาแนวที่ประกอบด้วยจำนวนบล็อกมากที่สุด เมื่อได้แนวที่มีจำนวนมากที่สุดแล้วจะใช้ค่าบล็อกที่วางตัวในแนวนั้นมาพิจารณาต่อไป

5.6 โปรแกรมจดจำเลขทะเบียนรถยนต์

ในส่วนของขั้นตอนการจดจำรูปแบบ จะมีการทำงาน 2 ส่วน คือ โปรแกรมสำหรับฝึกหัดโครงข่ายและโปรแกรมที่ใช้จดจำรูปแบบโดยอาศัยค่าถ่วงน้ำหนักที่ได้จากโปรแกรมฝึกหัดโครงข่าย



รูปที่ 5.9 โปรแกรมสำหรับฝึกหัดโครงข่าย



รูปที่ 5.9 โปรแกรมที่ใช้จดจำรูปแบบ

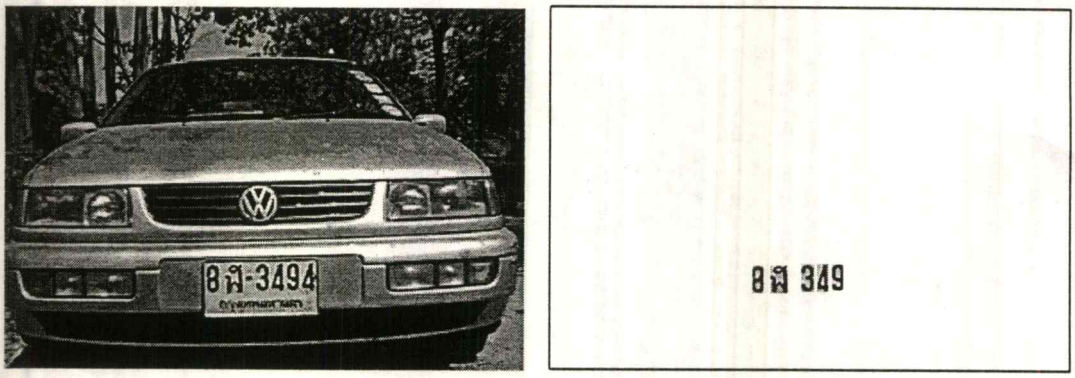
บทที่ 6

การทดลองและผลการทดลอง

6.1 การทดลองส่วนของการเตรียมข้อมูลภาพ

สำหรับการทดลอง จะทำการศึกษาปัจจัยต่างๆ ที่มีผลต่อการทำงานของโปรแกรมเพื่อเป็นแนวทางในการออกแบบและปรับปรุงแก้ไขระบบต่อไป โดยในการทดลองจะใช้ภาพตัวอย่างที่ถ่ายทางด้านหน้าของรถยนต์และจัดเก็บในรูปแแบบของบิตแมปเกรย์สเกล 256 ระดับ ขนาด 320×250 พิกเซล

ในการทดลองเบื้องต้น จะทดลองใช้โปรแกรมเพื่อแยกเลขทะเบียนออกจากภาพ โดยในขั้นตอนการแปลงเป็นภาพไบนารีจะใช้ค่าเทรชโฮลเท่ากับ 128 ซึ่งจากผลการทำงานของโปรแกรมพบว่ามีทั้งภาพที่สามารถแยกเลขทะเบียนออกจากภาพได้ครบทุกตัว และหลายภาพที่แยกเลขทะเบียนได้ไม่ครบทุกตัว และบางภาพไม่สามารถแยกเลขทะเบียนออกมาได้เลย



รูปที่ 6.1 ภาพที่แยกเลขทะเบียนได้ไม่ครบทุกตัว

เมื่อพิจารณาภาพที่ไม่สามารถแยกเลขทะเบียนได้ทุกตัว เช่น ภาพตัวอย่างในรูปที่ 6.1 พบว่าไม่สามารถแยกเลข 4 ออกมาได้ เมื่อพิจารณาภาพหลังจากผ่านการแปลงเป็นภาพไบนารีแล้วจะพบว่าตัวเลขทะเบียนที่ไม่สามารถแยกออกจากภาพได้เกิดจากการเชื่อมติดกันของตัวเลขทะเบียนกับขอบป้ายทะเบียนดังแสดงในรูปที่ 6.2 (ก) ซึ่งเป็นผลมาจากการเลือกค่าเทรชโฮลสูงเกินไป เมื่อลองเปลี่ยนค่าเทรชโฮลเป็นค่าเฉลี่ยและค่ามัธยฐานของระดับเกรย์สเกลของทุกพิกเซลแทนดังรูปที่ 6.2 (ข) และรูปที่ 6.2 (ค) จะสังเกตเห็นว่าขอบของป้ายทะเบียนจะถูกกำจัดไป ทำให้แยกเลข 4 ออกได้ ดังนั้น ค่า

เฉลี่ยและค่ามัธยฐานนี้สามารถเป็นตัวเลือกสำหรับเป็นค่าเทรชโฮลในการแปลงให้เป็นภาพไบนารีได้ ในการทดลองต่อไปจะเป็นการสังเกตผลของการเลือกค่าเฉลี่ยและค่ามัธยฐานเป็นค่าเทรชโฮล

(ก) (ข) (ค)

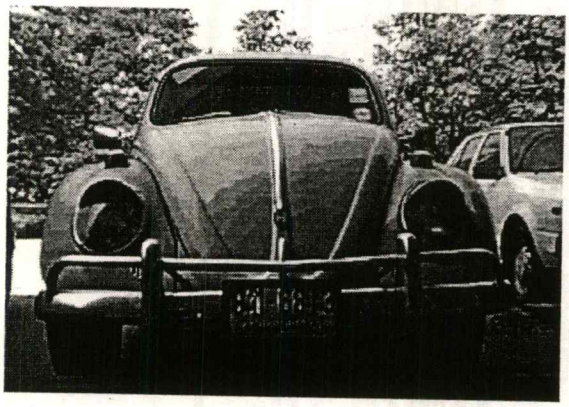
รูปที่ 6.2 ภาพไบนารีเมื่อเลือกค่าเทรชโฮลด้วยค่าต่างๆ

(ก) 128

(ข) ค่าเฉลี่ย =106

(ค) ค่ามัธยฐาน = 102

ในการทดลองต่อไปจะใช้ข้อมูลชุดเดียวกันกันการทดลองแรก แต่เปลี่ยนค่าเทรชโฮลจากค่า 128 เป็นค่าเฉลี่ยและค่ามัธยฐาน ซึ่งผลการทดลองที่ได้คือ ภาพที่ใช้ค่าเทรชโฮลเท่ากับค่าเฉลี่ยและค่ามัธยฐาน จะสามารถแยกตัวอักษรได้ดีกว่าการใช้ค่า 128 ในการทดลองแรก และการใช้ค่าเฉลี่ยจะให้ผลดีกว่าการใช้ค่ามัธยฐาน แต่ยังมีภาพอีกหลายภาพที่ไม่สามารถแยกเลขทะเบียนออกมาจากภาพได้ เมื่อพิจารณาภาพในกลุ่มนี้ พบว่าส่วนใหญ่เป็นภาพที่ถ่ายในระยะไกล ซึ่งทำให้พื้นที่ของแผ่นป้ายทะเบียนมีน้อยเมื่อเทียบกับพื้นที่ทั้งภาพ ทำให้ส่วนประกอบอื่นๆ ของภาพมีผลต่อค่าเฉลี่ยและค่ามัธยฐานได้มาก และสภาพของแสงภายนอกอาจทำให้ภาพมืดหรือสว่างเกินไป นอกจากนี้การที่ป้ายทะเบียนรถยนต์ที่ใส่กรอบหนาและมีสีเข้ม จะทำให้ตัวเลขทะเบียนเชื่อมติดกับกรอบได้ ความสกปรกของแผ่นป้ายทะเบียนก็สามารถทำให้เลขทะเบียนเชื่อมติดกัน ทำให้ไม่สามารถแยกเลขทะเบียนออกจากกันได้



รูปที่ 6.3 ภาพของรถยนต์ที่บริเวณป้ายทะเบียนรถมืดเกินไป

รูปที่ 6.4 แผ่นป้ายทะเบียนที่มีเลขทะเบียนติดกับกรอบ

เมื่อทำการทดลองอีกครั้ง โดยเลือกเฉพาะภาพระยะใกล้ที่มีเลขทะเบียนกับพื้นหลังแยกจากกันโดยสามารถสังเกตุได้ด้วยตาเปล่า และภาพไม่มืดหรือสว่างจนเกินไป ใช้ค่าเฉลี่ยเป็นค่าเทรชโฮล จะสามารถแยกเลขทะเบียนได้ทุกตัวในเกือบทุกๆ ภาพ จะมีบางภาพที่มีการเอียงของแผ่นป้ายทะเบียนรถมากเกินไป ซึ่งจะทำให้ในส่วนของโปรแกรมที่พิจารณาการเรียงตัวกันทางแนวนอนทำงานผิดพลาดได้

จากการทดลองทั้งหมดนั้นสามารถสรุปข้อจำกัดของภาพที่โปรแกรมสามารถแยกเลขทะเบียนรถออกจากภาพได้ กล่าวคือ สามารถสังเกตุความแตกต่างระหว่างเลขทะเบียนกับพื้นหลังได้อย่างชัดเจน ภาพต้องไม่มืดหรือสว่างจนเกินไป และแผ่นป้ายทะเบียนไม่เอียงมากนัก

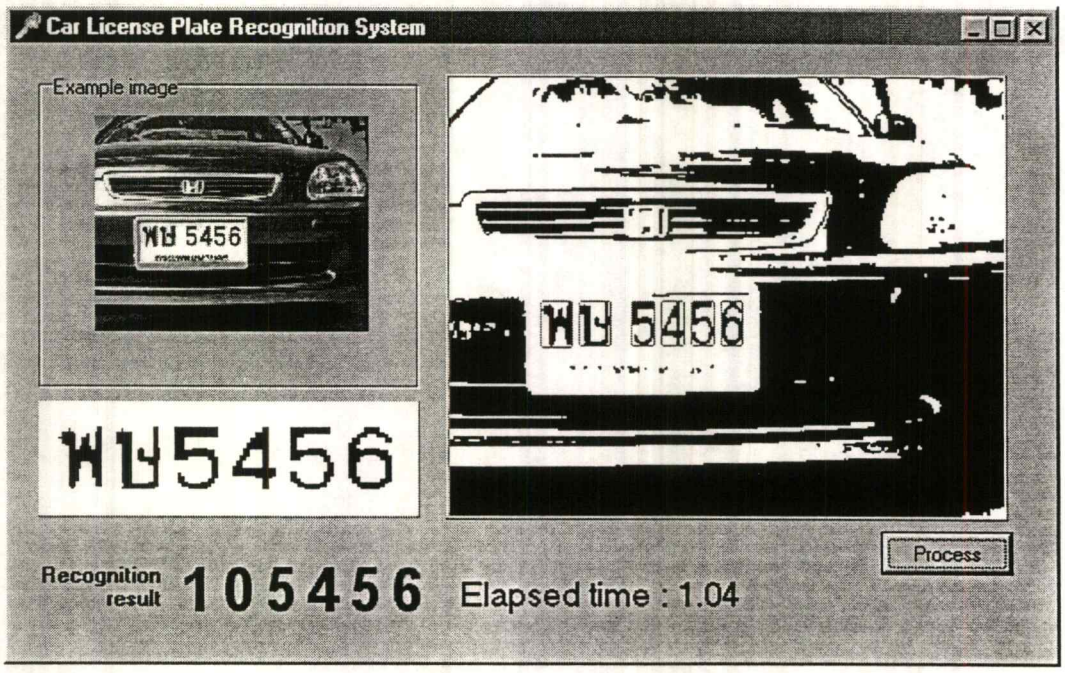
รูปที่ 6.5 ตัวอย่างภาพของแผ่นป้ายทะเบียนที่ระบบสามารถแยกเลขทะเบียนได้อย่างถูกต้อง

6.2 การทดลองส่วนของการจดจำรูปแบบหมายเลขทะเบียนรถยนต์

การทดลองเพื่อศึกษาถึงผลการเรียนรู้ของโครงข่ายนิวรอลแบบพร็อบพาเกชั่นในแง่ของการทดลอง กับข้อมูลอินพุทที่เปลี่ยนแปลงไปหลายๆ ค่า ว่าการเรียนรู้ที่ได้มีความยืดหยุ่นต่อสภาพแวดล้อมมากน้อยเพียงใด ขึ้นกับการแก้ปัญหาของการเรียนรู้รวมทั้งเวลาในการประมวลผลและค่าของเอาท์พุทที่ได้จากการเรียนรู้นั้นมีการผิดพลาดมากน้อยเพียงใดโดยจะพิจารณาทั้งค่าของเอาท์พุทจริงแต่ละตัวเมื่อเทียบกับค่าอินพุทที่ถูกต้องและค่าของร้อยละความถูกต้องสมบูรณ์ของทั้งแผ่นป้ายทะเบียนเพื่อพิจารณาความเป็นไปได้ในการใช้งานและเป็นแนวทางในการออกแบบพัฒนาและแก้ไขต่อไป

จากการทดลองจะใช้โครงข่ายสมองเทียมที่มีจำนวน 3 ชั้น ได้แก่ ชั้นอินพุท 35 หน่วย ชั้นซ่อน 20 หน่วย และชั้นเอาท์พุท 6 หน่วย พบว่าถ้าใช้ตัวอักษรที่ใช้ในการฝึกหัดโครงข่ายจำนวนมากขึ้น จะทำให้จำนวนของตัวอักษรที่วิเคราะห์ผิดพลาดน้อยลง หรือสามารถทำการวิเคราะห์ได้ถูกต้องมากขึ้น แต่ในการใช้จำนวนตัวอักษรมาก ก็จะทำให้ต้องใช้เวลาในการฝึกหัดมากขึ้นไปด้วย

ในขั้นตอนการทดลองนี้ ใช้ข้อมูลภาพทั้งหมด 57 ภาพ สามารถแยกเลขทะเบียนทุกตัวออกจากภาพได้ 66.67% และจากจำนวนตัวอักษรที่แยกออกมาได้ทั้งหมด สามารถจดจำตัวอักษรได้ถูกต้องเป็นจำนวน 65.52% ของจำนวนตัวอักษรที่แยกได้ทั้งหมด



รูปที่ 6.6 รูปแสดงโปรแกรมจดจำเลขทะเบียนรถยนต์

จากผลการทดลอง จะเห็นว่าประสิทธิภาพของโปรแกรมการจดจำทะเบียนรถยนต์ยังไม่ดีมากนัก ทั้งนี้เมื่อกลับไปพิจารณาข้อมูลภาพที่ใช้ในการทดลอง พบว่ามีหลายภาพที่มีลักษณะของแสงและเงาบนทะเบียนรถยนต์ส่งผลให้ไม่สามารถแยกเลขทะเบียนออกมาได้ และบางภาพมีสาเหตุมาจากความสมบูรณ์ของแผ่นป้ายทะเบียนเองด้วย

นอกจากนี้แล้ว ชุดของรูปแบบอักษรที่ใช้ในขั้นตอนการฝึกหัดโครงข่ายเป็นชุดตัวเลขและตัวหนังสือของฟอนท์ Browallia ซึ่งยังมีความแตกต่างจากเลขทะเบียนรถยนต์อยู่มาก จึงทำให้ประสิทธิภาพของการจดจำรูปแบบไม่ดีเท่าที่ควร

บทที่ 7
สรุปและวิจารณ์

7.1 ส่วนของการเตรียมข้อมูล

จากการทดลองเพื่อหาปัจจัยที่ส่งผลกระทบต่อการทำงานของระบบในขั้นตอนการแยกเลขทะเบียนรถยนต์ ทำให้ทราบถึงสาเหตุต่างๆ ที่ทำให้เกิดข้อผิดพลาดในการทำงานโดยทำให้ไม่สามารถแยกส่วนของหมายเลขทะเบียนได้อย่างสมบูรณ์ และได้ข้อจำกัดในการใช้งานระบบโดยระบบจะทำงานได้ไม่สมบูรณ์ถ้าข้อมูลภาพมีลักษณะต่างๆ คือ

- รถนั้นไม่มีป้ายทะเบียน
- ส่วนของเลขทะเบียนมีกรอบบัง หรือเชื่อมติดกับส่วนของหมายเลขทะเบียน
- เลขทะเบียนมีสีซีด ทำให้ไม่สามารถแยกส่วนของหมายเลขทะเบียนออกจากพื้นหลังได้
- เลขทะเบียนมีคราบสกปรกทำให้เลขทะเบียนเชื่อมติดกัน
- สภาพแสงภายนอก ตัวอย่างเช่น สภาวะที่มีฝนตก ความมืด ความสว่าง มีแสงสะท้อนที่แผ่นป้ายทะเบียนสูงทำให้ไม่สามารถแยกส่วนของหมายเลขทะเบียนออกจากพื้นหลังได้
- สิ่งแวดล้อมภายนอก ตัวอย่างเช่น ต้นไม้ ส่วนประกอบอื่น ๆ ของรถยนต์ ถนน เป็นต้น ซึ่งจะส่งผลทำให้ค่าเทรชโฮลไม่เหมาะสม
- ความเอียงของรูป ซึ่งอาจทำให้ส่วนของโปรแกรมทำงานผิดพลาด

ดังนั้นในทางปฏิบัติสามารถนำข้อจำกัดดังกล่าวเป็นสิ่งสำคัญในการออกแบบระบบที่ใช้งานจริง โดยเฉพาะในส่วนการรับข้อมูลภาพและการประมวลผลขั้นต้น ซึ่งต้องออกแบบให้ครอบคลุมถึงข้อจำกัดทุกๆ ข้อ เพื่อให้ระบบสามารถทำงานได้อย่างถูกต้องและแม่นยำ

นอกจากนี้ในขั้นตอนของการแปลงภาพเป็นภาพไบนารีด้วยการเลือกค่าเทรชโฮลที่เหมาะสมนั้น จะพบว่าค่าเทรชโฮลที่ได้จากการหาค่าเฉลี่ยเลขคณิตมีความถูกต้องของผลลัพธ์มากกว่าการใช้ค่าเทรชโฮลจากค่ามัธยฐาน หรือค่ากลางของข้อมูล (128) ดังนั้นจะเลือกค่าเฉลี่ยเลขคณิตมาใช้ขั้นตอนการสร้างภาพไบนารี เพื่อเป็นข้อมูลในการแยกเลขทะเบียนต่อไป ในบางกรณีค่าเทรชโฮลที่ใช้นี้อาจไม่เหมาะสม (ค่ามากเกินไปหรือน้อยเกินไป) ทำให้เกิดข้อผิดพลาดในขั้นตอนต่อไปอันเนื่องมาจากรายละเอียดของข้อมูลบางส่วนขาดหายไป ซึ่งอาจทำให้ไม่สามารถแยกเลขทะเบียนได้อย่างถูกต้อง ดังนั้นจึงต้องมีการปรับปรุงวิธีการต่อไป โดยอาศัยข้อมูลย้อนกลับจากขั้นตอนการแยกเลขทะเบียนแล้วปรับค่าเทรชโฮลจนกระทั่งได้ค่าที่เหมาะสมที่สุด

ผลการทดลองและข้อจำกัดต่างๆ เหล่านี้ จะเป็นประโยชน์อย่างยิ่งในการออกแบบเพื่อที่จะทำให้ระบบสามารถทำงานได้อย่างถูกต้อง และได้ผลลัพธ์ตามที่ต้องการ สามารถนำผลลัพธ์ที่ได้นี้เข้าสู่กระบวนการจดจำรูปแบบต่อไป ซึ่งเป็นขั้นตอนสำคัญที่จะทำให้ระบบใช้งานได้อย่างมีประสิทธิภาพ

7.2 ส่วนของการจดจำเลขทะเบียนรถยนต์

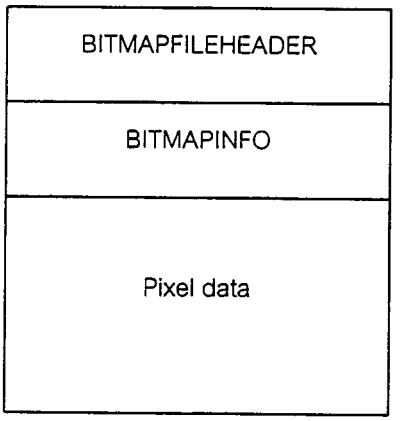
จากผลการทดลองของโปรแกรมในส่วนของการจดจำเลขทะเบียนรถยนต์ จะเห็นว่าประสิทธิภาพของกระบวนการจดจำ จะขึ้นอยู่กับขั้นตอนการฝึกหัดโครงข่าย โดยเฉพาะชุดของตัวอักษรที่ใช้ในการฝึกหัดโครงข่าย ถ้าชุดของตัวอักษรที่ใช้มีความใกล้เคียงกับเลขทะเบียนมากขึ้นเท่าไหร่ ความสามารถในการจดจำเลขทะเบียนรถยนต์ก็จะยิ่งเพิ่มสูงขึ้น

นอกจากนี้ กระบวนการเตรียมข้อมูลเพื่อการจดจำรูปแบบ ก็มีผลต่อประสิทธิภาพของระบบเป็นอย่างมาก ในวิธีที่นำเสนอนี้ ข้อมูลที่ได้จะขึ้นอยู่กับลักษณะทางภาพของภาพเป็นอย่างมาก โดยการผ่านกระบวนการเตรียมข้อมูลตั้งแต่การแปลงเป็นภาพไบนารี และการแยกเลขทะเบียน อาจทำให้ต้องสูญเสียข้อมูลบางส่วนที่เป็นลักษณะสำคัญของตัวเลขทะเบียนแต่ละตัวไปได้ เช่น ความหนาบางของตัวเลขทะเบียนตัวเดียวกันในภาพที่ความเข้มต่างกันอาจจะไม่เท่ากัน ทำให้เมื่อแปลงเป็นข้อมูลสำหรับกระบวนการจดจำแล้วมีความแตกต่างกันมาก ทำให้กระบวนการจดจำรูปแบบต้องอาศัยการฝึกหัดโครงข่ายที่ใช้ข้อมูลอินพุทที่หลากหลายมากยิ่งขึ้นของตัวอักษรแต่ละตัว

ภาคผนวก ก

โครงสร้างของไฟล์ข้อมูลภาพชนิดบิตแมป

ไฟล์ข้อมูลภาพชนิดบิตแมป มีโครงสร้างดังรูปที่ ก-1 โดยแบ่งออกเป็น 3 ส่วน ได้แก่ Bitmap File Header เป็นส่วนที่บอกข้อมูลของไฟล์ Bitmap Information เป็นส่วนที่แสดงขนาดและข้อมูลสีของภาพ และส่วนสุดท้ายคือ Pixel Data เป็นส่วนที่เก็บข้อมูลสีของแต่ละพิกเซล ซึ่งจะมีการเก็บข้อมูลตามขนาดของสี ถ้ามีสีมากจะทำให้ภาพที่ต้องการจัดเก็บมีขนาดใหญ่ไปด้วย



รูป ก-1 โครงสร้างของ Bitmap File

BITMAPFILEHEADER

Byte	Data	Detail
1-2	**File type**	Must be ASCII text "BM"
3-6	Size of file	In double word (32-bit integer)
7-10	Reserved for future	Must be zero
11-14	Byte offset to bitmap data	Offset from Bitmap File Header

ตาราง ก-1 แสดงข้อมูลใน Bitmap File Header

BITMAPINFO

โครงสร้างของ BITMAPINFO เขียนเป็นโครงสร้างได้ดังนี้

```
typedef struct tagBITMAPINFO { // bmi
    BITMAPINFOHEADER        bmiHeader;
    RGBQUAD                 bmiColors[1];
} BITMAPINFO;
```

BITMAPINFO ประกอบด้วย 2 ส่วนคือ BITMAPINFOHEADER เป็นส่วนที่บอกขนาดและข้อมูลสีของภาพบิตแมป และส่วนของ RGBQUAD ซึ่งจะเก็บค่าตารางสีสำหรับเทียบค่าจากค่าของแต่ละพิกเซล

BITMAPINFOHEADER

โครงสร้างของ BITMAPINFOHEADER สามารถเขียนได้ดังนี้

```
typedef struct tagBITMAPINFOHEADER{ // bmih
    DWORD       biSize;
    LONG        biWidth;
    LONG        biHeight;
    WORD        biPlanes;
    WORD        biBitCount
    DWORD       biCompression;
    DWORD       biSizeImage;
    LONG        biXPelsPerMeter;
    LONG        biYPelsPerMeter;
    DWORD       biClrUsed;
    DWORD       biClrImportant;
} BITMAPINFOHEADER;
```

โดยแต่ละฟิลด์จะมีความหมายดังนี้

biSize จำนวนไบต์ของ Header file

biwidth, biHeight บอกขนาดของภาพในความกว้าง และความสูงในหน่วยพิกเซล

biplanes เป็น 1 เสมอ

biBitCount คือจำนวนบิตต่อ 1 พิกเซล

biCompression แสดงการบีบอัดข้อมูล

ถ้ามีค่า BI_RGB ไฟล์เป็นแบบ ไม่มีการบีบอัดข้อมูล

RLE 4 เป็นการบีบอัดข้อมูลแบบ Run-length Encoder แบบ 4 บิต

RLE 8 เป็นการบีบอัดข้อมูลแบบ Run-length Encoder แบบ 8 บิต

BI_BITFIELDS เป็นการบีบอัดข้อมูลแบบ มีตารางสี

biSizeImage บอกขนาดของไฟล์

biXPelsPerMeter ความยาวแนวนอน มีหน่วยเป็นพิกเซลต่อเมตร

biYPelsPrMeter ความยาวแนวตั้ง มีหน่วยเป็นพิกเซลต่อเมตร

biClrUsed เป็นจำนวนสีในตารางสี ที่จะถูกชี้ด้วยค่าพิกเซลในบิตแมป

biClrImportant เป็นเลขที่แสดงว่าข้อมูลสีมีความสำคัญในการแสดงผลของบิตแมปถ้าเป็น

ศูนย์แสดงว่าทุกสีมีความสำคัญ

RGBQUAD

มีโครงสร้างเขียนได้ดังนี้

```
typedef struct tagRGBQUAD { // rgbq
  BYTE   rgbBlue;
  BYTE   rgbGreen;
  BYTE   rgbRed;
  BYTE   rgbReserved;
} RGBQUAD;
```

RGBQUAD จะเป็นโครงสร้างที่แสดงความเข้มของสีแดง เขียว และน้ำเงิน โดยมีความหมายของแต่ละฟิลด์ดังนี้

rgbBlue แสดงความเข้มของสีน้ำเงิน

rgbGreen แสดงความเข้มของสีเขียว

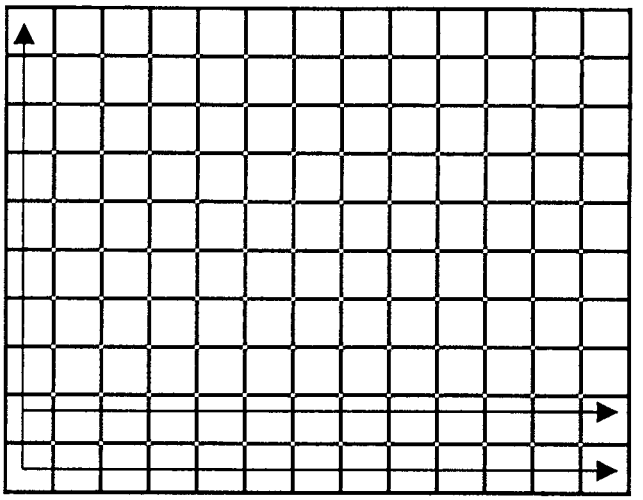
rgbRed แสดงความเข้มของสีแดง

rgbReserved ต้องมีค่าเป็น 0

ในส่วนของ bmiColors ของโครงสร้าง BITMAPINFO จะประกอบด้วยอาเรย์ของ RGBQUAD เพื่อเป็นตารางเทียบสีของข้อมูลในแต่ละพิกเซล

Pixel Data

เป็นส่วนที่เก็บข้อมูลสีของแต่ละพิกเซลของภาพ โดยข้อมูลแรกจะเป็นค่าสีของพิกเซลที่อยู่แถวล่างสุดที่ตำแหน่งซ้ายสุด ข้อมูลลำดับต่อไปจะเรียงไปทางขวาจากแถวล่างจนถึงแถวบนสุด



รูปที่ ก-2 แสดงการเก็บข้อมูลของแต่ละพิกเซล

ภาคผนวก ข

การเขียนโปรแกรมเพื่อรับข้อมูลภาพจากกล้องวิดีโอ

ในขั้นตอนของการรับภาพจากกล้องวิดีโอ จะเรียกใช้ฟังก์ชันสำหรับจับภาพจาก Video for Window หรือ VFW class ของ Window95 โดยเขียนโปรแกรมเพื่อติดต่อกับกล้องวิดีโอได้ดังนี้

```
procedure OpenVideo;
begin
  video := CreateCaptureWindow('',WS_VISIBLE+WS_CHILD+WS_BORDER,
                                  0,0,200,200,MainForm.handle,0);
  capDriverConnect(video, 0);
  capPreviewRate(video, 66);
  capPreview(video, True);
end;
```

เมื่อต้องการข้อมูลภาพจากวิดีโอ ต้องเรียก Callback Function ซึ่งจะให้ค่าตำแหน่งของหน่วยความจำที่เก็บโครงสร้างของ Video header data block (VIDEOHDR) อยู่ ซึ่งมีลักษณะโครงสร้างดังนี้

```
type
  PVIDEOHDR              = ^TVIDEOHDR;
  TVIDEOHDR              = record
    LpData          :PBYTE; // pointer to locked data buffer
    dwBufferLength  :DWORD; // Length of data buffer
    dwBytesUsed     :DWORD; // Bytes actually used
    dwTimeCaptured  :DWORD; // Milliseconds from start of stream
    dwUser          :DWORD; // for client's use
    dwFlags         :DWORD; // assorted flags
    dwReserved      :array[0..3] of DWORD; // reserved for driver
  end;
```

ส่วนสำคัญของโครงสร้าง VIDEOHDR คือ LpData ซึ่งจะให้ค่าชี้ตำแหน่งของข้อมูลภาพ ดังนั้นเมื่อต้องการข้อมูลภาพ จะต้องเรียกใช้ Callback Function ดังนี้

```
capSetCallbackOnFrame(video, VideoFrameCallback);
```

โดยต้องกำหนดฟังก์ชัน video Frame Callback ไว้ก่อนเพื่อทำการจัดการกับข้อมูลภาพได้

```
function VideoFrameCallback(window: Hwnd; lpVHdr: PvideoHDR):
         DWORD; stdcall;
begin
  // Process video frame data here by lpVHdr that return pointer
  // to DIB data
end;
```

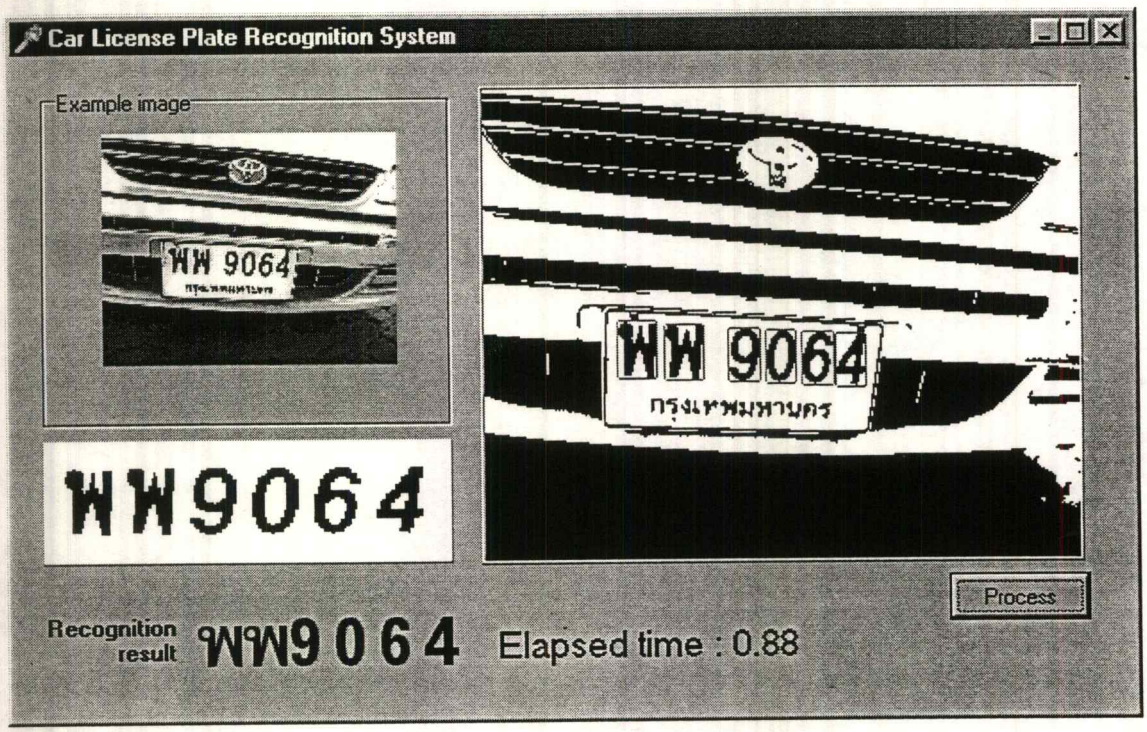
เมื่อต้องการข้อมูลของ BITMAPINFO เขียนโปรแกรมได้ดังนี้

```
var
  dwSize: DWORD;
  bmi: PBITMAPINFO;

  dwSize := capGetVideoFormatSize(video);
  bmi := allocmem(dwSize);
  capGetVideoFormat(video,bmi,dwSize);
```

ภาคผนวก ค

Program Source Code

1. โปรแกรมจดจำทะเบียนรถยนต์




```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, DIBitmap, ExtCtrls, Math;

type
  TForm1 = class(TForm)
    Button2: TButton;
    Elapsed: TLabel;
    Image2: TImage;
    Bevel2: TBevel;
    res1: TLabel;
    res2: TLabel;
    res3: TLabel;
    res4: TLabel;
    res5: TLabel;
    res6: TLabel;
    Timer1: TTimer;
    Panel1: TPanel;
```

```
    Digit1: TImage;
    Digit2: TImage;
    Digit4: TImage;
    Digit5: TImage;
    Digit3: TImage;
    Digit6: TImage;
    ImName: TLabel;
    GroupBox1: TGroupBox;
    Image1: TImage;
    Label1: TLabel;
    procedure Button2Click(Sender: TObject);
    procedure Main(fn:string);
    procedure FormCreate(Sender: TObject);
    function  GetTime(start:Boolean;var time:Word):Word;
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

type
  output = record
    dig:array[0..5] of real;
  end;

var
  Form1      :TForm1;
  index      :Integer=0;
  digit:array[0..5] of TImage;
  DIB:TDIBitmap;
  result:array[0..5] of output;
  res:array[0..5]of TLabel;
  Dir:string;
  FN   :String;
implementation

uses Unit2, layer;

{$R *.DFM}

function BinToStr(bin:array of real):String;
var
  acc:string[6];
  i:integer;
begin
  acc := '000000';
  for i := 0 to 5 do
  begin
     if bin[i]>= 0.7 then
       acc[i+1] := '1'
     else if bin[i]<= 0.3 then
       acc[i+1] := '0'
     else
       acc[i+1] := '?';
  end;

  result := acc;
end;

function BinToInt(bin:array of real):integer;
var
  i:integer;
  int:integer;
begin
  int:=0;
  for i := 0 to 5 do
```

```
    begin
       int := int+ Trunc(Round(bin[i])*IntPower(2,5-i));
    end;

    result := int;
  end;

  function IntToChar(int:integer):Char;
  begin
    case int of
    0:Result := '0';
    1:Result := '1';
    2:Result := '2';
    3:Result := '3';
    4:Result := '4';
    5:Result := '5';
    6:Result := '6';
    7:Result := '7';
    8:Result := '8';
    9:Result := '9';
    10:Result := #161;
    11:Result := #162;
    12:Result := #164;
    13:Result := #166;
    14:Result := #167;
    15:Result := #168;
    16:Result := #169;
    17:Result := #170;
    18:Result := #171;
    19:Result := #172;
    20:Result := #173;
    21:Result := #174;
    22:Result := #175;
    23:Result := #176;
    24:Result := #177;
    25:Result := #178;
    26:Result := #179;
    27:Result := #180;
    28:Result := #181;
    29:Result := #182;
    30:Result := #183;
    31:Result := #184;
    32:Result := #185;
    33:Result := #186;
    34:Result := #187;
    35:Result := #188;
    36:Result := #189;
    37:Result := #190;
    38:Result := #191;
    39:Result := #192;
    40:Result := #193;
    41:Result := #194;
    42:Result := #195;
    43:Result := #197;
    44:Result := #199;
    45:Result := #200;
    46:Result := #201;
    47:Result := #202;
    48:Result := #203;
    49:Result := #204;
    50:Result := #205;
    51:Result := #206;
    else Result := '-';
    end;
  end;

  function CharToBin(ch:Char):string;
```

```
begin
  case ch of
  '0':Result := '000000';
  '1':Result := '000001';
  '2':Result := '000010';
  '3':Result := '000011';
  '4':Result := '000100';
  '5':Result := '000101';
  '6':Result := '000110';
  '7':Result := '000111';
  '8':Result := '001000';
  '9':Result := '001001';
  #161:Result := '001010';
  #162:Result := '001011';
  #164:Result := '001100';
  #166:Result := '001101';
  #167:Result := '001110';
  #168:Result := '001111';
  #169:Result := '010000';
  #170:Result := '010001';
  #171:Result := '010010';
  #172:Result := '010011';
  #173:Result := '010100';
  #174:Result := '010101';
  #175:Result := '010110';
  #176:Result := '010111';
  #177:Result := '011000';
  #178:Result := '011001';
  #179:Result := '011010';
  #180:Result := '011011';
  #181:Result := '011100';
  #182:Result := '011101';
  #183:Result := '011110';
  #184:Result := '011111';
  #185:Result := '100000';
  #186:Result := '100001';
  #187:Result := '100010';
  #188:Result := '100011';
  #189:Result := '100100';
  #190:Result := '100101';
  #191:Result := '100110';
  #192:Result := '101111';
  #193:Result := '101000';
  #194:Result := '101001';
  #195:Result := '101010';
  #197:Result := '101011';
  #199:Result := '101100';
  #200:Result := '101101';
  #201:Result := '101110';
  #202:Result := '101111';
  #203:Result := '110000';
  #204:Result := '110001';
  #205:Result := '110010';
  #206:Result := '110011';
  else Result := '111111';
  end;
end;

function TForm1.GetTime(start:Boolean;var time:Word):Word;
var
  Present: TDateTime;
  Hour, Min, Sec, MSec: Word;
begin
  Present:= Now;
  DecodeTime(Present, Hour, Min, Sec, MSec);
  if start then
    time := Hour*1000*60*60 + Min*1000*60 + Sec*1000 + MSec
```

```
    else
      time := Hour*1000*60*60 + Min*1000*60 + Sec*1000 + MSec - time;

    Result := time;
  end;

  procedure TForm1.Main(fn: string);
  const
    WEIGHT_FILE = 'weight.dat';
  var
    net:network;
    x,y,i,j,p:integer;
    input:array[0..87]of real;
    input_count:integer;
    acc:integer;
  begin
      DIB := TDIBitmap.Create;
      DIB.Source := fn;
      DIB.Binarize;
      DIB.RemoveNoise;
      DIB.Labeling;

      DIB.Display(Image2.Canvas.Handle,Image2.Width,Image2.Height);
      Image2.Repaint;

      for p := 0 to DIB.digit_count-1 do
      begin
        StretchDIBits(digit[p].Canvas.Handle,0,0,digit[p].width,digit
  [p].height,
                      DIB.digit[p].x1,DIB.digit[p].y1,DIB.digit[p].x2-
  DIB.digit[p].x1+1,DIB.digit[p].y2-DIB.digit[p].y1+1,
                      DIB.ptrBits,DIB.ptrBitmapInfo^,
                      DIB_RGB_COLORS, SRCCOPY);
        digit[p].Repaint;

        input_count := 0;

        for j := Trunc(digit[p].Height/5)-1 downto 0  do
        begin
          for i := 0 to Trunc(digit[p].width/5)-1 do
          begin
            acc := 0;
            for y := 0 to 4 do
            begin
              for x := 0 to 4 do
              begin
                if GetPixel(digit[p].Canvas.Handle,x+i*5,y+j*5) =
  clBlack then
                  inc(acc);
              end;
            end;
            input[input_count] := acc/25;
            inc(input_count);
          end;
        end;

        net := Network.Create;
        net.GetLayerInfo;
        net.SetupNetwork;
        net.ReadWeights(WEIGHT_FILE);
        net.FillBuffer(input);
        net.ForwardProp;
        net.GetOutput(result[p].dig);

        res[p].Caption := IntToChar(BinToInt(result[p].dig));
        res[p].visible := True;
        digit[p].visible := True;
```

```
      net.Destroy;
    end;

    for i := 5 downto p do
    begin
      res[i].Visible := False;
      digit[i].visible := False;
    end;

    DIB.Destroy;
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  time    :Word;
begin
  Timer1.Enabled := False;
  GetTime(true,time);
  Main(FN);
  Elapsed.Caption := 'Elapsed time : '+FloatToStr(GetTime
(false,time)/1000);
  Timer1.Enabled := True;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Dir := GetCurrentDir;
  digit[0] := digit1;
  digit[1] := digit2;
  digit[2] := digit3;
  digit[3] := digit4;
  digit[4] := digit5;
  digit[5] := digit6;

  res[0] := res1;
  res[1] := res2;
  res[2] := res3;
  res[3] := res4;
  res[4] := res5;
  res[5] := res6;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
const
  LIST = 'list.txt';
var
  F    :TextFile;
  i    :Integer;
begin
  AssignFile(F,LIST);
  Reset(F);
  try
    for i := 0 to index do
      Readln(F,FN);
    if FN = '' then
    begin
      index := 0;
      Reset(F);
      Readln(F,FN);
    end;

    ImName.Caption := ExtractFileName(FN);
    FN := Dir+FN;
    Image1.Picture.LoadFromFile(FN);
    Image1.Repaint;
    inc(index);
```

```
    finally
      CloseFile(F);
    end;
  end;

  end.



  unit DIBitmap;

  interface

  uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs,
    StdCtrls, FileCtrl, ExtCtrls, dib32c, Math, OleCtrls;

  type
    box_ = record
      x1:Integer;
      x2:Integer;
      y1:Integer;
      y2:Integer;
    end;

    TDIBitmap = class(TDIB)
    private
    public
      digit:     array[0..5] of box_;
      digit_count:Integer;
      procedure  Binarize;
      procedure  RemoveNoise;
      procedure  Labeling;
      procedure  Display(DC:HDC;w,h:integer);
    end;

  implementation

  procedure Swap(var a,b:integer);
  var
    temp:integer;
  begin
    temp := a;
    a:=b;
    b:=temp;
  end;

  procedure  TDIBitmap.Binarize;
  var
    x,y: integer;
    data: PByte;
    Mean: real;
  begin
    mean := 0;

    for y := 0 to Height-1 do
    begin
      for x := 0 to Width-1 do
      begin
        data := pointer(longint(ptrBits) + x + y*width);
        mean := mean + data^/80000;
      end;
    end;

    if mean < 65 then
```

```
      mean := 120;

    for y := 0 to Height-1 do
    begin
      for x := 0 to Width-1 do
      begin
        data := pointer(longint(ptrBits) + x + y*width);
        if data^ < mean then
          data^ := 0
        else
          data^ := 255;
      end;
    end;
  end;

  procedure  TDIBitmap.RemoveNoise;
  var
    x,y,neighborCount: integer;
    data,neighbor: PByte;
  begin
    for y := 0 to Height-1 do
    begin
      for x := 0 to Width-1 do
      begin
        neighborCount := 0;

        neighbor := pointer(longint(ptrBits) + (x-1) + (y+1)*width);
        neighborCount := neighborCount + round(neighbor^/255);
        neighbor := pointer(longint(ptrBits) + (x)   + (y+1)*width);
        neighborCount := neighborCount + round(neighbor^/255);
        neighbor := pointer(longint(ptrBits) + (x+1) + (y+1)*width);
        neighborCount := neighborCount + round(neighbor^/255);
        neighbor := pointer(longint(ptrBits) + (x-1) + (y)*width);
        neighborCount := neighborCount + round(neighbor^/255);
        neighbor := pointer(longint(ptrBits) + (x+1) + (y)*width);
        neighborCount := neighborCount + round(neighbor^/255);
        neighbor := pointer(longint(ptrBits) + (x-1) + (y-1)*width);
        neighborCount := neighborCount + round(neighbor^/255);
        neighbor := pointer(longint(ptrBits) + (x)   + (y-1)*width);
        neighborCount := neighborCount + round(neighbor^/255);
        neighbor := pointer(longint(ptrBits) + (x+1) + (y-1)*width);
        neighborCount := neighborCount + round(neighbor^/255);

        data := pointer(longint(ptrBits) + x + y*width);
        if neighborCount > 6 then
          data^ := 255
        else if neighborCount = 0 then
          data^ := 0;
      end;
    end;
  end;

  procedure  TDIBitmap.Labeling;
  type
    rect = record
      x1:Integer;
      x2:Integer;
      y1:Integer;
      y2:Integer;
      Centroid:Integer;
      Size:boolean;
    end;
  var
    x,y,i,j,t,index,centroid_n,n_dc: integer;
    data          :PByte;
    equi,group    :Array[0..1000]of integer;
    found,first   :Boolean;
```

```
  a             :Longint;
  box           :array[0..1000]of rect;
  hough         :array[0..288]of integer;
  p1,p2         :TPoint;
  tData         :pByte;
  Dc            :HDC;
  B,C,D,E       :PByte;
  pLabel        :PByte;
  ptrLabel      :PByte;
begin
  index := 0;
  ptrLabel := AllocMem(Width*Height);

  for y := 0 to Height-1 do
  for x := 0 to Width-1 do
  begin
    data := pointer(longint(ptrBits) + x + y*Width);
    plabel :=  pointer(longint(ptrLabel) + x + y*Width);
    if data^ = 0 then
    begin
      B := pointer(longint(ptrLabel) + (x-1) + (y)*Width);
      C := pointer(longint(ptrLabel) + (x)   + (y-1)*Width);
      D := pointer(longint(ptrLabel) + (x+1) + (y-1)*Width);
      E := pointer(longint(ptrLabel) + (x-1) + (y-1)*Width);

      // at right edge
      if ((x=Width-1)and(y<>0)) then
      begin
        if (E^<>0) then
          pLabel^ := E^

        else if ((B^=0) and (C^=0)) then
        begin
          inc(index);
          plabel^ := index;
          equi[index] := index;
        end

        else if ((B^<>0) xor (C^<>0)) then
        begin
          if (B^<>0) then
            plabel^ := B^
          else
            plabel^ := C^;
        end

        else if ((B^<>0) and (C^<>0)) then
        begin
          if (B^=C^) then
            plabel^ := B^
          else
          begin
            plabel^ := C^;
            if equi[B^]<=equi[C^] then
            begin
              equi[plabel^] := equi[B^];
              equi[C^] := equi[B^];
            end
            else
            begin
              equi[plabel^] := equi[C^];
              equi[B^] := equi[C^];
            end;
          end;//else
        end;
      end
```

```
// at any coordination
else if ((x<>0) and (y<>0)) then
begin
  if ((D^<>0)and(E^<>0)) then
  begin
    if D^=E^ then
      plabel^ := D^
    else
    begin
      plabel^ := D^;
      if equi[E^]<=equi[D^] then
      begin
        equi[plabel^] := equi[E^];
        equi[D^] := equi[E^];
      end
      else
      begin
        equi[plabel^] := equi[D^];
        equi[E^] := equi[D^];
      end;
    end;
  end

  else if ((D^<>0)xor(E^<>0)) then
  begin
    if E^<>0 then
      plabel^ := E^
    else
    begin
      if B^=0 then
        plabel^ := D^
      else if B^=D^ then
        plabel^ := B^
      else if B^<>D^ then
      begin
        plabel^ := D^;
        if equi[B^]<=equi[D^] then
        begin
          equi[plabel^] := equi[B^];
          equi[D^] := equi[B^];
        end
        else
        begin
          equi[plabel^] := equi[D^];
          equi[B^] := equi[D^];
        end;
      end;
    end;
  end

  else if ((D^=0)and(E^=0)) then
  begin
    if ((B^=0) and (C^=0)) then
    begin
      inc(index);
      plabel^ := index;
      equi[index] := index;
    end

    else if ((B^<>0) xor (C^<>0)) then
    begin
      if (B^<>0) then
        plabel^ := B^
      else
        plabel^ := C^;
    end
```

```
          else if ((B^<>0) and (C^<>0)) then
          begin
            if (B^=C^) then
              pLabel^ := B^
            else
            begin
              pLabel^ := C^;
              if equi[B^]<=equi[C^] then
              begin
                equi[pLabel^] := equi[B^];
                equi[C^] := equi[B^];
              end
              else
              begin
                equi[pLabel^] := equi[C^];
                equi[B^] := equi[C^];
              end;
            end;
          end;
        end

        // at first row or first column
        else if ((x=0) xor (y=0)) then
        begin
          //at first row
```

```
unit layer;

interface

uses
  Windows, SysUtils;

type
  TLayer = class(TObject)
    protected
      num_inputs    :integer;
      num_outputs   :integer;
    public
      inputs        :array[0..66]of real;
      outputs       :array[0..66]of real;
    end;

  TInputLayer = class(TLayer)
    private
      noise_factor  :real;
      orig_outputs  :array[0..66]of real;
    public
      constructor Create(i:integer;o:integer);
      destructor  Destroy; override;
      procedure   CalOut; virtual;
      procedure   SetNF(NF:real);
    end;

  TOutputLayer = class(TLayer)
    protected
      weights                 :array[0..1320]of real;
      output_errors           :array[0..1320]of real;
      back_errors             :array[0..1320]of real;
      expected_values         :array[0..1320]of real;
      cum_deltas              :array[0..1320]of real;
      past_deltas             :array[0..1320]of real;
```

```
    public
      constructor   Create(i:integer;o:integer);
      destructor    Destroy; override;
      procedure     CalcOut;
      procedure     CalcError(var error:real);
      procedure     RandomizeWeights;
      procedure     UpdateWeights(const beta:real;const alpha:real);
      procedure     UpdateMomentum;
      procedure     WriteWeights(layer:integer;WeightFilename:String);
      procedure     ReadWeights(layer:integer;WeightFilename:String);
    end;

  THiddenLayer = class(TOutputLayer)
    public
      procedure CalcError;
    end;

  Network = class
    private
      LayerSize             :array[0..2]of integer;
      buffer                :array[0..1320]of real;
      InputLayer            :TInputLayer;
      HiddenLayer           :THiddenLayer;
      OutputLayer           :TOutputLayer;
    public
      procedure  GetLayerInfo;
      procedure  SetupNetwork;
      procedure  RandomizeWeights;
      procedure  FillBuffer(input:array of real);
      procedure  SetupPattern(buffer_index:integer);
      procedure  SetNF(NF:real);
      procedure  ForwardProp;
      procedure  BackwardProp(var error_last_pattern:real);
      procedure  UpdateWeights(learning_rate:real;momentum:real);
      procedure  UpdateMomemtum;
      procedure  WriteWeight(WeightFilename:String);
      procedure  WriteOutput(OutputFilename:String;pattern:integer);
      procedure  ReadWeights(WeightFilename:String);
      procedure  GetOutput(var output:array of real);
    end;

implementation

function RandomWeight(first_time:Boolean): Real;
var num: Integer;
begin
  if first_time then Randomize;
  num := Random(100);
  result := (2*(num/100))-1;
end;

function squash(input:real):real;
begin
  if input < -50 then
    result := 0
  else if input > 50 then
    result := 1
  else
    result := 1/(1+exp(-input));
end;

constructor TInputLayer.Create(i:integer;o:integer);
begin
  inherited Create;
  num_inputs := i;
  num_outputs := o;
end;
```

```
destructor TInputLayer.Destroy;
begin
  inherited Destroy;
end;

procedure TInputLayer.CalOut;
var
  i:integer;
begin
  for i := 0 to num_outputs-1 do
  begin
    outputs[i] := orig_outputs[i]*(1+noise_factor*RandomWeight
(false));
  end;
end;

procedure TInputLayer.SetNF(NF:real);
begin
  noise_factor := NF;
end;

constructor TOutputLayer.Create(i: integer; o: integer);
begin
  inherited Create;
  num_inputs := i;
  num_outputs := o;
end;

destructor TOutputLayer.Destroy;
begin
  inherited Destroy;
end;

procedure TOutputLayer.CalcOut;
var
  i,j,k: integer;
  acc: real;
begin
  acc := 0;
  for j := 0 to num_outputs-1 do
  begin
    for i := 0 to num_inputs-1 do
    begin
      k := i*num_outputs;
      if weights[k+j]*weights[k+j] > 1000000 then
      begin
        raise Exception.Create('Weights are blowing up.');
        Exit;
      end;
      outputs[j] := weights[k+j]*inputs[i];
      acc := acc + outputs[j];
    end;
    outputs[j] := squash(acc);
    acc := 0;
  end;
end;

procedure TOutputLayer.CalcError(var error:real);
var
  i,j,k: integer;
  acc:real;
  total_error:real;
begin
  acc := 0;
  total_error := 0;
```

```
    for j:= 0 to num_outputs-1 do
    begin
      output_errors[j] := (expected_values[j]-outputs[j]);
      total_error := total_error + output_errors[j];
    end;

    error := total_error;

    for i:= 0 to num_inputs-1 do
    begin
      k := i*num_outputs;
      for j := 0 to num_outputs-1 do
      begin
        back_errors[i] := weights[k+j]*output_errors[j];
        acc := acc + back_errors[i];
      end;
      back_errors[i] := acc;
      acc := 0;
      back_errors[i] := back_errors[i]*(inputs[i]*(1-inputs[i]));
    end;
  end;

  procedure TOutputLayer.RandomizeWeights;
  const
    FIRST_TIME = True;
    NOT_FIRST_TIME = False;
  var
    i,j,k: integer;
    discard:real;
  begin
    discard := RandomWeight(FIRST_TIME);
    for i := 0 to num_inputs-1 do
    begin
      k := i*num_outputs;
      for j := 0 to num_outputs-1 do
      begin
        weights[k+j] := RandomWeight(NOT_FIRST_TIME);
      end;
    end;
  end;

  procedure TOutputLayer.UpdateWeights(const beta:real;const
  alpha:real);
  var
    i,j,k : integer;
    delta:real;
  begin
    for i:= 0 to num_inputs-1 do
    begin
      k := i*num_outputs;
      for j := 0 to num_outputs-1 do
      begin
        delta := beta*output_errors[j]*inputs[i] + alpha*past_deltas
  [k+j];
        weights[k+j] := weights[k+j] + delta;
        cum_deltas[k+j] := cum_deltas[k+j] + delta;
      end;
    end;
  end;

  procedure TOutputLayer.UpdateMomentum;
  var
    i,j,k:integer;
    temp: real;
  begin
    for i := 0 to 50 do
    begin
```

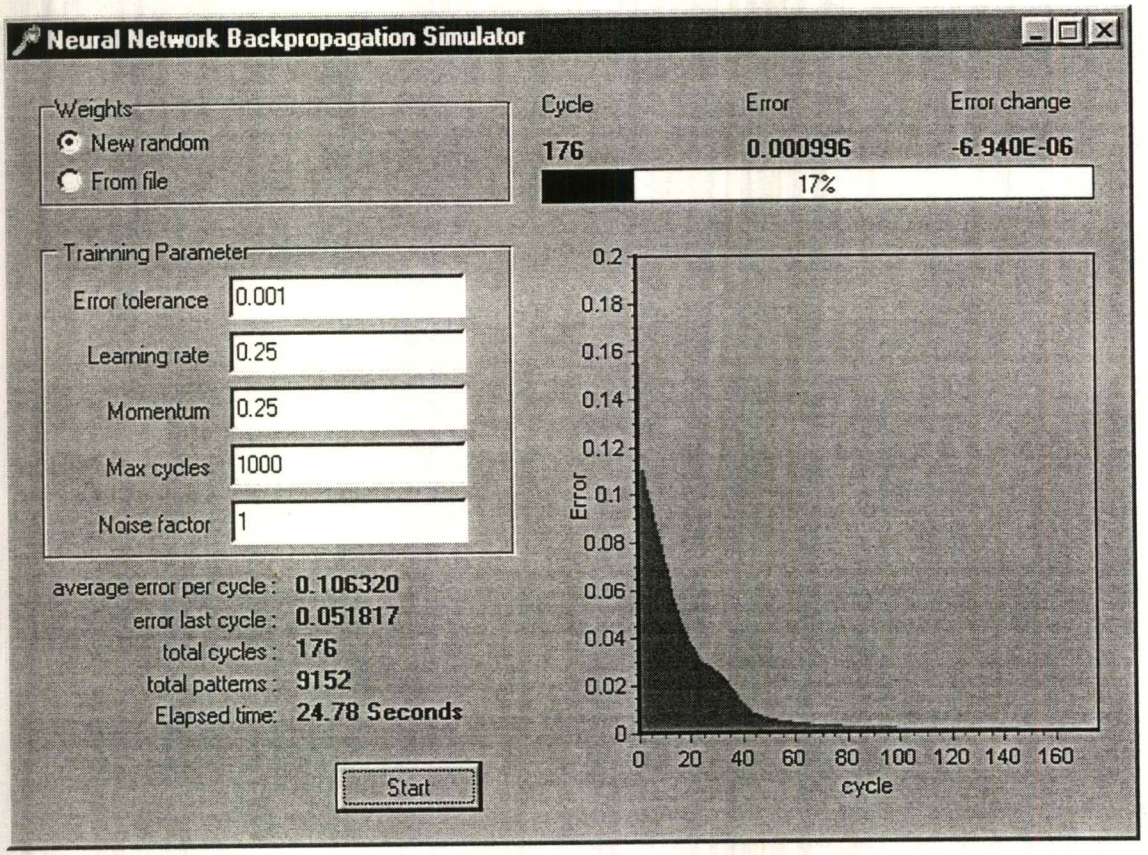
```
      temp := past_deltas[i];
      past_deltas[i] := cum_deltas[i];
      cum_deltas[i] := temp;
    end;

    for i := 0 to num_inputs-1 do
    begin
      k := i*num_outputs;
      for j := 0 to num_outputs-1 do
        cum_deltas[k+j] := 0
    end;
  end;

  procedure
  TOutputLayer.WriteWeights(layer:integer;WeightFilename:String);
  var
    i,j,k:integer;
    weight_file:textfile;
  begin
    Assign(weight_file,WeightFilename);
    if layer = 0 then
      Rewrite(weight_file)
    else
      Append(weight_file);
    for i := 0 to num_inputs-1 do
    begin
      Write(weight_file,layer,' ');
      k := i*num_outputs;
      for j := 0 to num_outputs-1 do
      begin
        Write(weight_file,weights[k+j],' ');
  //      Write(weight_file,FloatToStrf(weights[k+j],ffFixed,17,4),' ');
      end;
      Writeln(weight_file);
    end;
    Close(weight_file);
  end;

  procedure
  TOutputLayer.ReadWeights(layer:integer;WeightFilename:String);
  var
    i,j,k:integer;
    weight_file:textfile;
  begin
    Assign(weight_file,WeightFilename);
    Reset(weight_file);
    while not EOF(weight_file) do
    begin
      Read(weight_file,j);
      if j=layer then
        break
      else
        Readln(weight_file);
    end;//while

    for j := 0 to num_outputs-1 do
      Read(weight_file,weights[j]);

    Readln(weight_file);

    for i:=1 to num_inputs-1 do
    begin
      Read(weight_file,layer);
      k := i*num_outputs;
      for j:= 0 to num_outputs-1 do
        Read(weight_file,weights[k+j]);
    end;
```

```
  Close(weight_file);
end;

procedure THiddenLayer.CalcError;
var
  i,j,k: integer;
  acc:real;
begin
  acc := 0;
  for i:= 0 to num_inputs-1 do
  begin
    k := i*num_outputs;
    for j := 0 to num_outputs-1 do
    begin
      back_errors[i] := weights[k+j]*output_errors[j];
      acc := acc + back_errors[i];
    end;
    back_errors[i] := acc;
    acc := 0;
    back_errors[i] := back_errors[i]*(inputs[i]*(1-inputs[i]));
  end;
end;

////////////////////////////////////////////////////////////////////////
//////////
//
//
//                              NETWORK
//
//
//
////////////////////////////////////////////////////////////////////////
//////////
procedure Network.GetLayerInfo;
var
  TrainFile:TextFile;
  i:integer;
begin
  LayerSize[0] := 35;
  LayerSize[1] := 20;
  LayerSize[2] := 6;
end;

procedure Network.SetupNetwork;
var
  i,j,k:integer;
begin
  InputLayer := TInputLayer.Create(0,LayerSize[0]);
  OutputLayer := TOutputLayer.Create(LayerSize[1],LayerSize[2]);
  HiddenLayer := THiddenLayer.Create(LayerSize[0],LayerSize[1]);
end;

procedure Network.RandomizeWeights;
begin
  HiddenLayer.RandomizeWeights;
  OutputLayer.RandomizeWeights;
end;

procedure Network.FillBuffer(input:array of real);
var
  i,count:integer;
  ins:integer;
begin
  ins := InputLayer.num_outputs;
  for i := 0 to ins-1 do
    InputLayer.outputs[i] := input[i];
end;
```

```
procedure Network.SetupPattern(buffer_index:integer);
var
  i,k: integer;
  ins,outs: integer;
begin
  ins := InputLayer.num_outputs;
  outs := OutputLayer.num_outputs;
  k := buffer_index*(ins+outs);
  for i := 0 to ins-1 do
    InputLayer.outputs[i] := buffer[k+i];
  for i := 0 to outs-1 do
    OutputLayer.expected_values[i] := buffer[k+i+ins];
end;

procedure Network.SetNF(NF:real);
begin
  InputLayer.SetNF(NF);
end;

procedure Network.ForwardProp;
var
  i:integer;
begin
  for i := 0 to InputLayer.num_outputs-1 do
    HiddenLayer.inputs[i] := InputLayer.outputs[i];
  HiddenLayer.CalcOut;
  for i := 0 to HiddenLayer.num_outputs-1 do
    OutputLayer.inputs[i] := HiddenLayer.outputs[i];
  OutputLayer.CalcOut;
end;


procedure Network.BackwardProp(var error_last_pattern:real);
var
  i:integer;
begin
  OutputLayer.CalcError(error_last_pattern);
  for i := 0 to HiddenLayer.num_outputs-1 do
    HiddenLayer.output_errors[i] := OutputLayer.back_errors[i];
  HiddenLayer.CalcError;
end;

procedure Network.UpdateWeights(learning_rate:real;momentum:real);
begin
  HiddenLayer.UpdateWeights(learning_rate,momentum);
  OutputLayer.UpdateWeights(learning_rate,momentum);
end;

procedure Network.UpdateMomemtum;
begin
  HiddenLayer.UpdateMomentum;
  OutputLayer.UpdateMomentum;
end;


procedure Network.WriteWeight(WeightFilename:String);
var
  WeightFile:textfile;
begin
  HiddenLayer.WriteWeights(0,WeightFilename);
  OutputLayer.WriteWeights(1,WeightFilename);
end;

procedure Network.WriteOutput(OutputFilename:String;pattern:integer);
var
  i :integer;
```

```
  OutputFile:textfile;
begin
  Assign(OutputFile,OutputFilename);
  if pattern = 0 then
    Rewrite(OutputFile)
  else
    Append(OutputFile);
  Write(OutputFile,pattern+1,' ');
  for i := 0 to OutputLayer.num_outputs-1 do
  begin
    Write(OutputFile,FloatToStrf(OutputLayer.expected_values
[i],ffFixed,18,1),' ');
  end;
  for i := 0 to OutputLayer.num_outputs-1 do
  begin

Write(OutputFile,FloatToStrf(OutputLayer.outputs[i],ffFixed,18,5),'
');
  end;
  Writeln(OutputFile);
  Close(OutputFile);
end;

procedure Network.ReadWeights(WeightFilename:String);
begin
  HiddenLayer.ReadWeights(0,WeightFilename);
  OutputLAyer.ReadWeights(1,WeightFilename);
end;

procedure Network.GetOutput(var output:array of real);
var
  i:integer;
begin
  for i := 0 to OutputLayer.num_outputs-1 do
    output[i] :=  OutputLayer.Outputs[i];
end;

end.
```

2. โปรแกรมฝึกหัดโครงข่าย



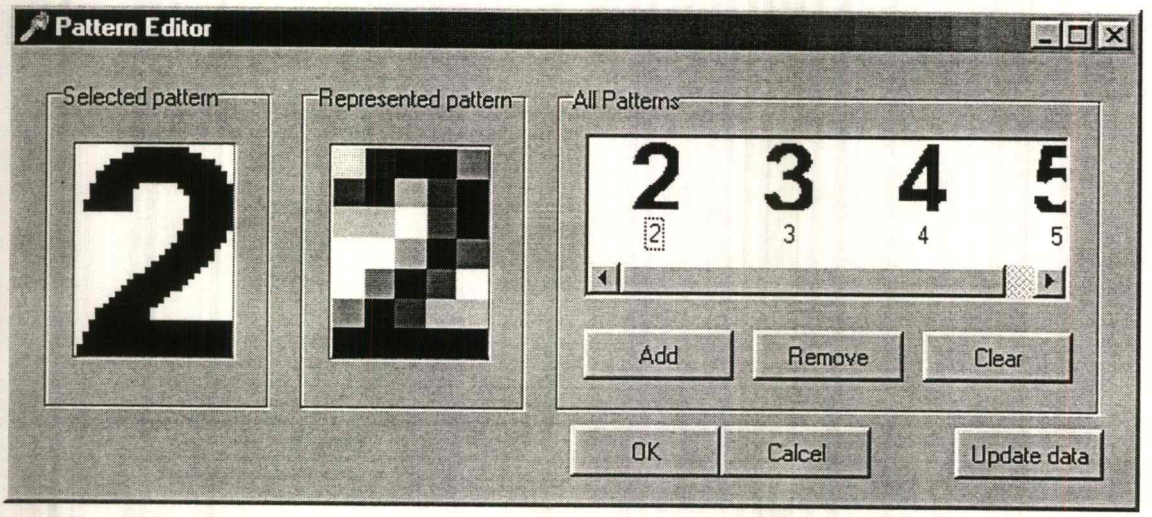

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  Db, Grids, DBGrids, DBTables, ToolWin, ComCtrls, DBClient,
  StdCtrls, Gauges, ExtCtrls, Menus, OleCtrls, TeEngine, Series,
TeeProcs,
  Chart;

type
  TMainForm = class(TForm)
    StartBtn: TButton;
    Gauge1: TGauge;
    IndLbl3: TLabel;
    IndLbl2: TLabel;
    IndLbl1: TLabel;
    Label21: TLabel;
    RadioGroup2: TRadioGroup;
    Parameter: TGroupBox;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label24: TLabel;
    Label14: TLabel;
```

```
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Label18: TLabel;
    Label17: TLabel;
    Label16: TLabel;
    Label1: TLabel;
    SumLb1: TLabel;
    SumLb2: TLabel;
    SumLb3: TLabel;
    SumLb4: TLabel;
    Chart1: TChart;
    Series1: TAreaSeries;
    Label3: TLabel;
    Label4: TLabel;
    Label2: TLabel;
    Label5: TLabel;
    procedure Train;
    procedure Edit3Exit(Sender: TObject);
    procedure Edit2Exit(Sender: TObject);
    procedure Edit1Exit(Sender: TObject);
    procedure StartBtnClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  MainForm: TMainForm;
  PageIndex: Integer;
  clock:comp=0;
  TimeStamp:TTimeStamp;
implementation

uses layer;

{$R *.DFM}

procedure TMainForm.Train;
const
  TRAINING_FILE = 'train.dat';
  WEIGHT_FILE = 'weight.dat';
  OUTPUT_FILE = 'output.dat';
var
  error_tolerance,error_per_pattern:  real;
  learning_rate,total_error:real;
  momentum:real;
  error_last_cycle,error_last_pattern:real;
  total_cycles,max_cycles:  Integer;
  total_patterns: integer;
  vectors_in_buffer:  integer;
  net:  network;
  pattern: integer;
  acc: real;
  NF,newNF:real;
begin
  Gauge1.Visible := True;
  Repaint;

  error_tolerance := StrToFloat(Edit1.Text);
  learning_rate := StrToFloat(Edit2.Text);
  momentum := StrToFloat(Edit3.Text);
  max_cycles := StrToInt(Edit4.Text);
  NF := StrToFloat(Edit5.Text);
```

```
  Gauge1.MaxValue := max_cycles;
  Gauge1.Progress := 0;
  Series1.Clear;
//////////////
  Series1.AddXY(0,0.2,'',clRed);
//////////////

  total_error := 0;
  error_last_cycle := 0;
  error_last_pattern := 10;
  error_per_pattern := 100;
  total_cycles := 0;
  total_patterns := 0;

  newNF := NF;

  net := Network.Create;
  net.GetLayerInfo(TRAINING_FILE);
  net.SetupNetwork;
  if RadioGroup2.ItemIndex = 0 then
    net.RandomizeWeights
  else
    net.ReadWeights(WEIGHT_FILE);
  net.SetNF(newNF);
  vectors_in_buffer := net.FillBuffer(TRAINING_FILE);
  //main loop
  while (error_per_pattern > error_tolerance)and
  (total_cycles < max_cycles) do
  begin
    error_last_cycle := 0;

    for pattern := 0 to vectors_in_buffer-1 do
    begin
      net.SetupPattern(pattern);
      inc(total_patterns);
      net.ForwardProp;
      net.BackwardProp(error_last_pattern);
      error_last_cycle := error_last_cycle+Sqr(error_last_pattern);
      net.UpdateWeights(learning_rate,momentum);
      net.WriteOutput(OUTPUT_FILE,pattern);
    end;//for

    error_per_pattern := Sqrt(error_last_cycle)/pattern;
    total_error := total_error+error_last_cycle;
    inc(total_cycles);

    if total_cycles > 0.7*max_cycles then newNF := 0
    else if total_cycles > 0.5*max_cycles then newNF := 0.25*NF
         else if total_cycles > 0.3*max_cycles then newNF := 0.5*NF
              else if total_cycles > 0.1*max_cycles then newNF :=
0.75*NF;
    net.SetNF(newNF);

    IndLbl1.Caption := IntToStr(total_cycles);
    IndLbl2.Caption := FloatToStrf(error_per_pattern,ffFixed,18,6);
    IndLbl3.Caption := FloatToStrf(error_per_pattern-
acc,ffExponent,4,2);
//////////////////////////////
    with Series1 do
    begin
      AddXY(total_cycles,error_per_pattern,'',clTeeColor);
    end;
//////////////////////////////
    Repaint;

    gauge1.AddProgress(1);
```

```
    acc := error_per_pattern;
  end;  //while
  SumLb1.Caption :=
FloatToStrf(Sqrt(total_error)/total_cycles,ffFixed,18,6);
  SumLb2.Caption := FloatToStrf(Sqrt(error_last_cycle),ffFixed,18,6);
  SumLb3.Caption := IntToStr(total_cycles);
  SumLb4.Caption := IntToStr(total_patterns);
  net.WriteWeight(WEIGHT_FILE);
  net.Destroy;
end;

procedure TMainForm.Edit3Exit(Sender: TObject);
begin
  if (StrToFloat(Edit3.text)>1000000) then
  begin
    try raise Exception.Create('Please enter in the max_cycles less
than 1,000,000,try 100 to start');
    finally Edit3.Text := '100';end;
  end;
end;

procedure TMainForm.Edit2Exit(Sender: TObject);
begin
  if (StrToFloat(Edit2.text)<0.01)
  or (StrToFloat(Edit2.text)>1.0) then
  begin
    try raise Exception.Create('Please enter in the learning_rate
between 0.01 to 1.0,try 0.5 to start');
    finally Edit2.Text := '0.5';end;
  end;
end;

procedure TMainForm.Edit1Exit(Sender: TObject);
begin
  if (StrToFloat(Edit1.text)<0.001)
  or (StrToFloat(Edit1.text)>100) then
  begin
    try raise Exception.Create('Please enter in the error_tolerance
between 0.001 to 100.0,try 0.1 to start');
    finally Edit1.Text := '0.1';end;
  end;
end;

procedure TMainForm.StartBtnClick(Sender: TObject);
var
  Present: TDateTime;
  Hour, Min, Sec, MSec: Word;
  time: Word;
begin
  Present:= Now;
  DecodeTime(Present, Hour, Min, Sec, MSec);
  time := Hour*1000*60*60 + Min*1000*60 + Sec*1000 + MSec;

  Train;

  Present:= Now;
  DecodeTime(Present, Hour, Min, Sec, MSec);
  time := (Hour*1000*60*60 + Min*1000*60 + Sec*1000 + MSec) - time;
  Label4.Caption := FloatToStr(time/1000) + ' Seconds';

end;

end.
```

3. โปรแกรมเตรียมข้อมูลสำหรับการฝึกหัดโครงข่าย




```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, ExtCtrls, ComCtrls, ImgList, Math;

type
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    OpenDialog1: TOpenDialog;
    ImageList1: TImageList;
    Image1: TImage;
    Bevel1: TBevel;
    OKBtn: TButton;
    CalcelBtn: TButton;
    UpdateBtn: TButton;
    GroupBox2: TGroupBox;
    ListView1: TListView;
    AddBtn: TButton;
    RemoveBtn: TButton;
    ClearBtn: TButton;
    GroupBox3: TGroupBox;
    Bevel2: TBevel;
    Image2: TImage;
    procedure AddBtnClick(Sender: TObject);
    procedure OKBtnClick(Sender: TObject);
    procedure CalcelBtnClick(Sender: TObject);
    procedure RemoveBtnClick(Sender: TObject);
    procedure ClearBtnClick(Sender: TObject);
    procedure UpdateBtnClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure DisplayData(index:integer);
    procedure ListView1SelectItem(Sender: TObject; Item: TListItem;
      Selected: Boolean);

  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;

  pattern_data = record
    fn :string[255];
    expected_value :Char;
  end;

var
  Form1: TForm1;
  pattern_array: array[0..54]of pattern_data;
  pattern_index: integer;
  im:TimageList;
  dir:string;
implementation

uses Unit2, dib32c;

{$R *.DFM}

function CharToBin(ch:Char):string;
begin
  case ch of
  '0':Result := '0 0 0 0 0 0';
  '1':Result := '0 0 0 0 0 1';
  '2':Result := '0 0 0 0 1 0';
  '3':Result := '0 0 0 0 1 1';
  '4':Result := '0 0 0 1 0 0';
  '5':Result := '0 0 0 1 0 1';
  '6':Result := '0 0 0 1 1 0';
  '7':Result := '0 0 0 1 1 1';
  '8':Result := '0 0 1 0 0 0';
  '9':Result := '0 0 1 0 0 1';
  #161:Result := '0 0 1 0 1 0';
  #162:Result := '0 0 1 0 1 1';
  #164:Result := '0 0 1 1 0 0';
  #166:Result := '0 0 1 1 0 1';
  #167:Result := '0 0 1 1 1 0';
  #168:Result := '0 0 1 1 1 1';
  #169:Result := '0 1 0 0 0 0';
  #170:Result := '0 1 0 0 0 1';
  #171:Result := '0 1 0 0 1 0';
  #172:Result := '0 1 0 0 1 1';
  #173:Result := '0 1 0 1 0 0';
  #174:Result := '0 1 0 1 0 1';
  #175:Result := '0 1 0 1 1 0';
  #176:Result := '0 1 0 1 1 1';
  #177:Result := '0 1 1 0 0 0';
  #178:Result := '0 1 1 0 0 1';
  #179:Result := '0 1 1 0 1 0';
  #180:Result := '0 1 1 0 1 1';
  #181:Result := '0 1 1 1 0 0';
  #182:Result := '0 1 1 1 0 1';
  #183:Result := '0 1 1 1 1 0';
  #184:Result := '0 1 1 1 1 1';
  #185:Result := '1 0 0 0 0 0';
  #186:Result := '1 0 0 0 0 1';
  #187:Result := '1 0 0 0 1 0';
  #188:Result := '1 0 0 0 1 1';
  #189:Result := '1 0 0 1 0 0';
  #190:Result := '1 0 0 1 0 1';
  #191:Result := '1 0 0 1 1 0';
  #192:Result := '1 0 0 1 1 1';
  #193:Result := '1 0 1 0 0 0';
  #194:Result := '1 0 1 0 0 1';
  #195:Result := '1 0 1 0 1 0';
  #197:Result := '1 0 1 0 1 1';
```

```
  #199:Result := '1 0 1 1 0 0';
  #200:Result := '1 0 1 1 0 1';
  #201:Result := '1 0 1 1 1 0';
  #202:Result := '1 0 1 1 1 1';
  #203:Result := '1 1 0 0 0 0';
  #204:Result := '1 1 0 0 0 1';
  #205:Result := '1 1 0 0 1 0';
  #206:Result := '1 1 0 0 1 1';
  else Result := '1 1 1 1 1 1';
  end;
end;

procedure TForm1.AddBtnClick(Sender: TObject);
begin
  if OpenDialog1.Execute then
  begin
    if GetData(pattern_array[pattern_index].expected_value) then
    begin
      pattern_array[pattern_index].fn := OpenDialog1.Filename;

Image1.Picture.Bitmap.LoadFromFile(pattern_array[pattern_index].fn);
      DisplayData(pattern_index);
      ImageList1.Add(Image1.Picture.Bitmap,nil);

      with ListView1.Items.Add do
      begin
        Caption := pattern_array[pattern_index].expected_value;
        ImageIndex := ImageList1.Count-1;
      end;

      inc(pattern_index);
      UpdateBtn.Enabled := True;
    end;
  end;
end;

procedure TForm1.OKBtnClick(Sender: TObject);
const
  PATTERN_FILENAME = '\pattern.dat';
var
  pattern_file: TextFile;
  i : integer;
begin
  AssignFile(pattern_file,dir+PATTERN_FILENAME);
  Rewrite(pattern_file);
  for i := 0 to pattern_index-2 do
  begin
    Writeln(pattern_file,pattern_array[i].expected_value+'
'+pattern_array[i].fn);
  end;
  Write(pattern_file,pattern_array[pattern_index-1].expected_value+'
'+pattern_array[pattern_index-1].fn);
  CloseFile(pattern_file);
  UpdateBtnClick(Sender);
  Close;
end;

procedure TForm1.DisplayData(index:integer);
var
  DIB:TDIB;
  i,j,x,y,acc,d:integer;
  data:pbyte;
begin
  if pattern_array[index].fn<>'' then
  begin
    DIB := TDIB.Create;
    DIB.Source := pattern_array[index].fn;
```

```
    for j := 0 to Trunc(DIB.Height/5)-1 do
    for i := 0 to Trunc(DIB.Width/5)-1 do
    begin
      acc := 0;
      for y := 0 to 4 do
      for x := 0 to 4 do
      begin
        data :=
pointer(longint(DIB.ptrBits)+i*5+j*5*(DIB.Width+3)+x+y*(DIB.Width+3));
        if data^=255 then
          inc(acc);
      end;

      for y := 0 to 4 do
      for x := 0 to 4 do
      begin
        data :=
pointer(longint(DIB.ptrBits)+i*5+j*5*(DIB.Width+3)+x+y*(DIB.Width+3));
        data^ := Trunc(acc/25*255);
      end;
    end;

    StretchDIBits(Image2.Canvas.Handle, 0, 0, Image2.Width+acc-acc,
Image2.Height,
                  0, 0, DIB.Width, DIB.Height,
                  DIB.ptrBits, DIB.ptrBitmapInfo^,
                  DIB_RGB_COLORS, SRCCOPY);
    Image2.Repaint;
  end;
end;

procedure TForm1.CalcelBtnClick(Sender: TObject);
begin
  Close;
end;

procedure TForm1.RemoveBtnClick(Sender: TObject);
var
  i : integer;
begin
  dec(pattern_index);
  for i := ListView1.Selected.Index to pattern_index-1 do
  begin
    pattern_array[i] := pattern_array[i+1]
  end;
  pattern_array[i].fn := '';
  pattern_array[i].expected_value := #0;


  ListView1.Items.Delete(ListView1.Selected.Index);
  RemoveBtn.Enabled := False;
  UpdateBtn.Enabled := True;
end;


procedure TForm1.ClearBtnClick(Sender: TObject);
var
  i :integer;
begin
  ListView1.Items.Clear;
  for i := 0 to pattern_index-1 do
    with pattern_array[i] do
    begin
      fn := '';
      expected_value := #0;
    end;
```

```
  pattern_index := 0;
  UpdateBtn.Enabled := True;
end;

function IntToBin(int:integer): string;
var
  i:integer;
  bin:string;
begin
  bin := '';
  for i := 5 downto 0 do
  begin
    if IntPower(2,i)>int then
      bin := bin + '0 '
    else
    begin
      bin := bin + '1 ';
      int := int - Trunc(IntPower(2,i));
    end;
  end;
  result := bin;
end;

procedure TForm1.UpdateBtnClick(Sender: TObject);
const
  TRAIN_FILENAME = '\training.dat';
var
  index,i,j,k,x,y,z,acc:integer;
  DIB:TDIB;
  train_file:textfile;
  data:PByte;
begin
  try
    AssignFile(train_file,dir+'\'+TRAIN_FILENAME);
    Rewrite(train_file);

    for index := 0 to pattern_index-1 do
    begin
      DIB := TDIB.Create;
      DIB.Source := pattern_array[index].fn;

      for j := 0 to Trunc(DIB.Height/5)-1 do
      for i := 0 to Trunc(DIB.Width/5)-1 do
      begin
        acc := 0;
        for y := 0 to 4 do
        for x := 0 to 4 do
        begin
          data := pointer(longint(DIB.ptrBits)+i*5+j*5*
(DIB.Width+3)+x+y*(DIB.Width+3));
          if data^=255 then
            inc(acc);
        end;

        Write(train_file, FloatToStrf(1-acc/25,ffFixed,18,2),' ');
      end;

Writeln(train_file,CharToBin(pattern_array[index].expected_value));
    end;//for index
  finally
    CloseFile(train_file);
    UpdateBtn.Enabled := False;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
const
```

```
  PATTERN_FILENAME = '\pattern.dat';
var
  pattern_file: TextFile;
  NumRead,i : integer;
  acc:Char;
begin
  try
    dir := GetCurrentDir;
    OpenDialog1.InitialDir := dir+'\images\';
    AssignFile(pattern_file,dir+'\'+PATTERN_FILENAME);
    Reset(pattern_file);
    pattern_index := 0;
    while not EOF(pattern_file) do
    begin
      Read(pattern_file, pattern_array[pattern_index].expected_value);
      Read(pattern_file, acc);
      Readln(pattern_file, pattern_array[pattern_index].fn);

Image1.Picture.Bitmap.LoadFromFile(pattern_array[pattern_index].fn);
      DisplayData(pattern_index);
      ImageList1.Add(Image1.Picture.Bitmap,nil);
      with ListView1.Items.Add do
      begin
        Caption := pattern_array[pattern_index].expected_value;
        ImageIndex := ImageList1.Count-1;
      end;
      inc(pattern_index);
    end;
  finally
    CloseFile(pattern_file);
  end;
end;

procedure TForm1.ListView1SelectItem(Sender: TObject; Item: TListItem;
  Selected: Boolean);
begin
  if pattern_array[ListView1.Items.IndexOf(Item)].fn<>'' then
  begin
    Image1.Picture.LoadFromFile(pattern_array[ListView1.Items.IndexOf
(Item)].fn);

    DisplayData(ListView1.Items.IndexOf(Item));

    RemoveBtn.Enabled := True;
  end;
end;

end.




unit dib32c;

interface

uses
  Windows, SysUtils, Classes, Graphics;

type
  tDIB = class
    protected
      pDIB    : PBitmapInfo;
      MemSize   : longint;
```

```
    function Get_biSize : integer;
    function Get_DIBSize : longint;

    procedure ReadFromFile(const fn : string);
    function Get_width : longint;
    function Get_height : longint;
    function Get_ncolors : longint;
    function Get_ptr_to_RGBQuad : PRGBQuad;
    function Get_ptr_to_bits : pointer;

  public
    constructor Create;
    destructor Destroy; override;

    property Source         : string  write ReadFromFile;
    property Width          : longint read Get_width;
    property Height         : longint read Get_height;
    property NumColors      : longint read Get_ncolors;
    property ptrRGBQuad     : PRGBQuad read Get_ptr_to_RGBQuad;
    property ptrBits        : pointer read Get_ptr_to_bits;
    property ptrBitmapInfo : PBitmapInfo read pDIB;

    function Create_Palette(const palSysFlags,palFlags : byte):
hPalette;
    function Create_IdentPal256(const palFlags: byte): hPalette;
  end;

implementation

constructor tDIB.Create;
begin
  inherited Create;
  MemSize := 0;
end;

destructor tDIB.Destroy;
begin
  if MemSize <> 0 then
    freemem(pDIB,MemSize);

  inherited Destroy;
end;

{
  tDIB.ReadFromFile: loads a bmp file and returns a pointer to the DIB
in
                     memory. Also returns the size of the memory image
so
                     that the memory can be freed.
}
procedure tDIB.ReadFromFile(const fn : string);
var
  stream: TMemoryStream;
begin
  with TMemoryStream.Create do
  begin
    LoadFromFile(fn);
    MemSize := size - sizeof(TBitmapFileHeader);
    pDIB := AllocMem(MemSize);
    seek(sizeof(TBitmapFileHeader), 0);
    read(pDIB^, MemSize);
    free;
  end;

end; {ReadFromFile}
```

```
//  tDIB.Get_width: returns the width of a dib
function tDIB.Get_width : longint;
begin
  result := pDIB^.bmiHeader.biwidth;
end; {Get_width}


//  tDIB.Get_height: returns the height of a dib
function tDIB.Get_height : longint;
begin
  result := pDIB^.bmiHeader.biheight
end; {dib_height}


//  tDIB.Get_ncolors: returns the number of colors in a DIB
function tDIB.Get_ncolors : longint;
begin
  if pDIB^.bmiHeader.biClrUsed > 0 then
    result := pDIB^.bmiHeader.biClrUsed
  else
    result := 1 shl pDIB^.bmiHeader.biBitCount;

end; {Get_ncolors}


//  tDIB.Get_biSize: returns the number of bytes in a DIB header
including the
//                   color table.
function tDIB.Get_biSize : integer;
begin
  result := sizeof(TBitmapInfoHeader) + (Get_ncolors * sizeof
(TRGBQuad));
end; {dib_biSize}


//  tDIB.Get_DIBSize: returns the number of bytes in a DIB.
function tDIB.Get_DIBSize : longint;
begin
  result := Get_biSize + pDIB^.bmiHeader.biSizeImage;
end; {Get_DIBSize}


//  tDIB.Get_ptr_to_RGBQuad: returns a pointer to the first RGBQuad
structure
function tDIB.Get_ptr_to_RGBQuad : PRGBQuad;
begin

  {set a pointer to the DIB palette table as an offset from the DIB
ptr}
  result := pointer(longint(pDIB) + sizeof(TBitmapInfoHeader));

end; {Get_ptr_to_RGBQuad}


// tDIB.Get_ptr_to_bits: returns a pointer to the pixel data of the
DIB
function tDIB.Get_ptr_to_bits : pointer;
begin
  result := pointer(longint(pDIB) + Get_biSize);
end; {Get_ptr_to_bits}


// tDIB.Create_Palette: creates a palette from the DIB's color table
and
//                      returns a handle.
function tDIB.Create_Palette( const palSysFlags,palFlags: byte):
hPalette;
```

```
var
  DstPal: PLogPalette;
  Colors, n: Integer;
  Size: Longint;
  DC: HDC;
  Focus: HWND;
  Pal: array[0..15] of TPaletteEntry;
  SysPalSize: Integer;
  I: Integer;

begin
  Result := 0;
  Colors := Get_ncolors;
  if Colors <= 2 then Exit;

  Size := SizeOf(TLogPalette) + ((Colors - 1) *
SizeOf(TPaletteEntry));
  DstPal := AllocMem(Size);
  try
    FillChar(DstPal^, Size, 0);
    with DstPal^ do
    begin
      palNumEntries := Colors;
      palVersion := $300;
      Focus := GetFocus;
      DC := GetDC(Focus);
      try
        SysPalSize := GetDeviceCaps(DC, SIZEPALETTE);
        if (Colors = 16) and (SysPalSize >= 16) then
        begin
          { Ignore the disk image of the palette for 16 color bitmaps
use
            instead the first 8 and last 8 of the current system
palette }
          GetSystemPaletteEntries(DC, 0, 8, palPalEntry);
          I := 8;
          GetSystemPaletteEntries(DC, SysPalSize - I, I, palPalEntry
[I]);
        end
        else
          { Copy the palette for all others (i.e. 256 colors) }
          {$R-}
          for N := 0 to Colors - 1 do
          begin
            palPalEntry[N].peRed   := pDIB^.bmiColors[N].rgbRed;
            palPalEntry[N].peGreen := pDIB^.bmiColors[N].rgbGreen;
            palPalEntry[N].peBlue  := pDIB^.bmiColors[N].rgbBlue;
            if (N < 10) or (N > 245) then
              palPalEntry[N].peFlags := palSysFlags
            else
              palPalEntry[N].peFlags := palFlags;
          end;
          {$R+}
      finally
        ReleaseDC(Focus, DC);
      end;
    end;
    Result := CreatePalette(DstPal^);
  finally
    FreeMem(DstPal, Size);
  end;
end; {Create_Palette}


// tDIB.Create_IdentPal256: returns a handle to the identity palette
of a DIB
function tDIB.Create_IdentPal256(const palFlags : byte): hPalette;
```

```
var
  DstPal: PLogPalette;
  n: Integer;
  Size: Longint;
  DC: HDC;
  Focus: HWND;

begin
  Result := 0;

  Size := SizeOf(TLogPalette) + (256 * SizeOf(TPaletteEntry));
  DstPal := AllocMem(Size);
  try
    FillChar(DstPal^, Size, 0);
    with DstPal^ do
    begin
      palNumEntries := 256;
      palVersion := $300;
      Focus := GetFocus;
      DC := GetDC(Focus);
      {$R-}
      try
          {use the system palette entries for 0-9, and 246-255}
          GetSystemPaletteEntries(DC, 0, 10, palPalEntry);
          N := 246;
          GetSystemPaletteEntries(DC, 246, 10, palPalEntry[N]);

          {use the entries from 0-236 of the color table and map them
to
           entries 10-245 in the new logical palette}
          for N := 10 to 245 do
          begin
            palPalEntry[N].peRed   := pDIB^.bmiColors[N-10].rgbRed;
            palPalEntry[N].peGreen := pDIB^.bmiColors[N-10].rgbGreen;
            palPalEntry[N].peBlue  := pDIB^.bmiColors[N-10].rgbBlue;
             palPalEntry[N].peFlags := palFlags;
          end;
      finally
        ReleaseDC(Focus, DC);
      end;
    {$R-}
    end;
    Result := CreatePalette(DstPal^);
  finally
    FreeMem(DstPal, Size);
  end;
end; {Create_IdentPal256}

end.
```

เอกสารอ้างอิง

1. Kenneth R. Catleman, "Digital Image Processing," Englewood Cliffs, NJ : Prentice Hall, 1996, pp. 478-481.
2. Lawrence O'Gorman, Rangachar Kasturi, "Document Image Analysis," Los Alamitos, CA : IEEE Computer Society Press, 1995, pp. 24-26.
3. Valluru B. Rao, Hayagriva V. Rao, "C++ Neural Networks and Fuzzy Logic, " New York, MIS Press, 1993, pp.101-147.

กิตติกรรมประกาศ

สำหรับความสำเร็จสมบูรณ์ของรายงานฉบับนี้ เกิดขึ้นได้ด้วยความกรุณาของท่านอาจารย์ ดร.สุรพันธ์ เอื้อไพบูลย์ ที่ได้ให้คำปรึกษาแนะนำ ให้แนวความคิดที่เป็นประโยชน์ต่อโครงงาน และให้กำลังใจมาโดยตลอด ขอขอบพระคุณอาจารย์ทุกท่านที่ได้ให้ความรู้ในด้านต่างๆ และขอขอบคุณเพื่อนๆ ทุกคนที่ได้ให้กำลังใจและช่วยเหลือในการจัดทำรายงานฉบับนี้ขึ้น

สุดท้ายนี้ ขอขอบพระคุณบิดา มารดา ที่ได้ให้การสนับสนุนและส่งเสริมมาโดยตลอด

นางสาวนพวรรณ ม่วงเฉย

นายนัฏฐกร ปาระชัย