ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชา วิศวกรรมศาสตร์คอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประมวลผลภาพสำหรับรายงานสภาพการจราจรบนทางด่วน

(Image Processing for Highway Traffic Report)

ผู้จัดทำ

1. นายสมพร แซ่ตั้ง รหัสประจำตัว 36013177

2. นายสุรศักดิ์ สุขบำรุง รหัสประจำตัว 36013182

.....อาจารย์ที่ปรึกษา

(.....นายบุญวัฒน์ อัตชู.....)

การประมวลผลภาพสำหรับรายงานสภาพการจราจรบนทางด่วน

Image Processing for Highway Traffic Report

โดย

นายสมพร แซ่ตั้ง 36013177

นายสุรศักดิ์ สุขบำรุง 36013182

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา

วิศวกรรมศาสตร์

ภาควิชาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

การประมวลผลภาพสำหรับรายงานสภาพการจราจรบนทางด่วน

นายสมพร แซ่ตั้ง
นายสุรศักดิ์ สุขบำรุง

ผศ.ดร.บุญวัฒน์ อัตชู อาจารย์ที่ปรึกษา
ปีการศึกษา 2538

บทคัดย่อ

งานวิจัยนี้เป็นการประมวลผลภาพสำหรับการรายงานสภาพจราจรบนทางด่วน โดยเริ่มจากการรับสัญญาณภาพที่ได้จากกล้องวีดีโอบนทางด่วน ซึ่งเป็นสัญญาณอนาล็อก (Analog signal) มาทำการเปลี่ยนเป็นสัญญาณดิจิตอล (Digital signal) โดยใช้การ์ดดิจิไทเซอร์ (Digitizer Card) ซึ่งใส่อยู่ในแผ่นวงจรหลักของเครื่องคอมพิวเตอร์ ซึ่งเป็นข้อมูลภาพแบบดิจิตอลมีความแตกต่างของระดับสีเทา (Gray scale) 256 ระดับ โดยทำการจัดเก็บเป็นเฟรมต่อเนื่องกันเป็นไฟล์กราฟฟิกอยู่ในฟอร์แมตของไฟล์ BMP ขนาด 300*300 Pixel แล้วจัดเก็บลงฮาร์ดดิสก์ ต่อจากนั้นคอมพิวเตอร์ซึ่งจะมีโปรแกรมการวิเคราะห์การประมวลผลภาพก็จะอ่านข้อมูลภาพแบบดิจิตอลที่ถูกจัดเก็บลงฮาร์ดดิสก์ มาทำการวิเคราะห์ด้วยวิธีการต่างๆ ตามทฤษฎีการประมวลผลภาพโดยอัตโนมัติ เช่น การหาขอบภาพ, การตัดค่า threshold, การติดตามขอบภาพ และการกำจัดส่วนที่ไม่ต้องการ (Noise)

ผลลัพธ์ที่ต้องการคือ รายงานสภาพการจราจรบนทางด่วน ซึ่งใช้เงื่อนไขการเคลื่อนที่ของรถยนต์เป็นตัวตัดสินใจสภาพการจราจรบนทางด่วน โดยจะแสดงผลการวิเคราะห์สภาพการจราจรบนทางด่วนในลักษณะข้อความทางจอแสดงผล (Display monitor)

Image Processing for Highway Traffic Report

Mr. Somporn Saetang

Mr. Surasak Sukbumrung

Dr. Boonwat Attachoo Advisor

1995

Abstract

This tesis presents highway traffic report using image processing. The traffic status was capture by video camera and then was feed to computer. Video signal was changed to digital signal by digitizer card. 300*300 pixels of digital image with 256 gray level was recorded into harddisk frame by frame. Then the image processing theory was applied to digital image such as edge detection, histograme, noise reduction etc. to detect movement of cars. The analysis result will be show in text mode on display monitor.

สารบัญ

สารบัญ (ต่อ)

สารบัญภาพประกอบ

สารบัญภาพประกอบ (ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

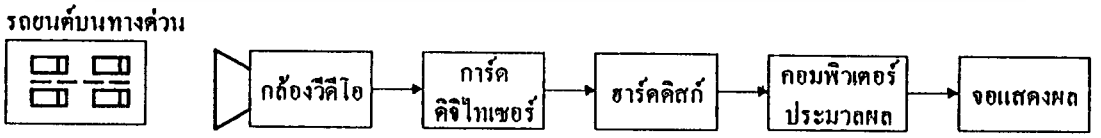
1.1 วัตถุประสงค์ของงานวิจัย

1. ศึกษาทฤษฎีการประมวลผลภาพด้วยคอมพิวเตอร์
2. นำภาพที่บันทึกจากกล้องวีดีโอบนทางด่วนมาประมวลผล
3. ผลที่ได้จากการประมวลผลนำมาสรุปด้วยคอมพิวเตอร์ แล้วรายงานออกมาในลักษณะข้อความทางจอแสดงผล

1.2 ลักษณะและความสำคัญของงานวิจัย

เริ่มจากการนำสัญญาณภาพที่ได้จากกล้องวีดีโอบนทางด่วน ซึ่งเป็นสัญญาณอนาล็อก (Analog signal) มาทำการเปลี่ยนเป็นสัญญาณดิจิตอล (Digital signal) โดยใช้การ์ดดิจิไทเซอร์ (Digitizer Card) ซึ่งใส่อยู่ในแผ่นวงจรหลักของเครื่องคอมพิวเตอร์ ซึ่งจะทำหน้าที่แปลงข้อมูลภาพเป็นข้อมูลภาพแบบดิจิตอลมีความแตกต่างของระดับสีเทา (Gray scale) 256 ระดับ โดยทำการจัดเก็บเป็นเฟรมต่อเนื่องกันเป็นไฟล์กราฟฟิกอยู่ในฟอแมตของไฟล์ BMP ขนาด 300*300 Pixel แล้วจัดเก็บลงฮาร์ดดิสก์ ต่อจากนั้นคอมพิวเตอร์ซึ่งจะมีโปรแกรมการวิเคราะห์การประมวลผลภาพก็จะอ่านข้อมูลภาพแบบดิจิตอลที่ถูกจัดเก็บลงฮาร์ดดิสก์ มาทำการวิเคราะห์ด้วยวิธีการต่างๆ ตามทฤษฎีการประมวลผลภาพโดยอัตโนมัติ

ผลลัพธ์ที่ต้องการคือ รายงานสภาพการจราจรบนทางด่วน ซึ่งใช้เงื่อนไขความเร็วของรถยนต์เป็นตัวตัดสินใจสภาพการจราจรบนทางด่วน โดยจะแสดงผลการวิเคราะห์สภาพการจราจรบนทางด่วนในลักษณะข้อความทางจอแสดงผล (Display monitor) ซึ่งจะมีการเปลี่ยนแปลงข้อความผลลัพธ์ของสภาพการจราจรบนทางด่วนในช่วงระยะห่างของเวลาพอสมควร เช่น 1 นาที เปลี่ยนแปลงครั้งหนึ่ง เป็นต้น



รูปที่ 1.1 แสดงแผนภาพลักษณะขั้นตอนการทำงานของงานวิจัย

1.3 ขอบเขตและข้อกำหนดของงานวิจัย

- งานวิจัยนี้จะทำการประมวลผลภาพเฉพาะ ในช่วงเวลาที่มีค่าของความแตกต่างของระดับสีเทา (Gray -scale) ระหว่างรถยนต์กับถนนพอสมควร เช่น ช่วงเวลาเช้า ถึงช่วงเย็น

- ลักษณะภาพถ่ายจากกล้องวีดีโอต้องถ่ายจากมุมสูงด้านบน เพราะรถยนต์ที่อยู่ภายในขอบเขตของภาพที่ต้องการประมวลผลจะต้องไม่ติดกัน คือรถยนต์คันหน้าบังรถยนต์คันหลัง

- ใช้การประมวลผลสภาพการจราจรบนทางด่วน โดยคอมพิวเตอร์เพียงจุดเดียวเท่านั้น คือไม่มีการต่อเชื่อมโยงเป็นระบบเครือข่าย

- ในการรายงานสภาพการจราจรบนทางด่วนจะแสดงผลการวิเคราะห์ในลักษณะข้อความของสภาพการจราจรบนทางด่วนทางจอแสดงผลเท่านั้น

- การเปลี่ยนแปลงค่าของการรายงานสภาพการจราจรบนทางด่วน มีช่วงห่างของระยะเวลาพอสมควร เช่น 1 นาที เปลี่ยนแปลงค่าของการรายงานสภาพการจราจรบนทางด่วนครั้งหนึ่ง

- ใช้เงื่อนไขความเร็วของรถยนต์เป็นตัวตัดสินใจสภาพการจราจรบนทางด่วนเท่านั้น

1.4 ประโยชน์ของงานวิจัยที่คาดว่าจะได้รับ

ประโยชน์ของงานวิจัยที่คาดว่าจะได้รับคือ เป็นประยุกต์การใช้ทฤษฎีการประมวลผลภาพด้วยคอมพิวเตอร์ โดยนำภาพที่บันทึกจากกล้องวีดีโอบนทางด่วนมาประมวลผล เพื่อใช้เป็นแนวทางในการนำไปใช้ในการรายงานสภาพการจราจรบนทางด่วนโดยอัตโนมัติ ซึ่งอาจทำให้มีความสะดวกรวดเร็วในการรายงานสภาพการจราจรบนทางด่วน และลดจำนวนของเจ้าหน้าที่ซึ่งในปัจจุบันจะต้องมาคอยควบคุมดูสภาพการจราจรบนจอแสดงผลด้วยตนเองแล้วจึงรายงานสภาพการจราจรตลอดเวลา

บทที่ 2

ทฤษฎีและความรู้พื้นฐาน

2.1 ไฟล์ข้อมูลภาพกราฟิกชนิด BMP

2.1.1 รูปแบบของไฟล์ข้อมูลภาพชนิด BMP

รูปแบบของไฟล์ข้อมูลภาพชนิด BMP เป็นฟอร์แมตของวินโดว์ Bitmapped ซึ่งเป็นมาตราฐานสำหรับไฟล์กราฟิกบนวินโดว์ ซึ่งใช้ในการตัดต่อหรือสำเนาภาพต่าง ๆ ลงบนโปรแกรม Clipboard เมื่อเวลาจัดเก็บไฟล์ที่มีสกุลเป็น BMP ซึ่งฟอร์แมตนี้ยังสามารถใช้เป็น Wallpaper ของวินโดว์ได้อีกด้วย

2.1.2 โครงสร้างของไฟล์ข้อมูลภาพชนิด BMP

โครงสร้างของไฟล์ BMP จะประกอบด้วย 3 ส่วน คือ

1. ข้อมูล header
2. ข้อมูล palette
3. ข้อมูลภาพ

1. ข้อมูล header คือ ข้อมูลที่อยู่บริเวณส่วนหัวของไฟล์ ซึ่งประกอบไปด้วยข้อมูลที่บอกรายละเอียดต่าง ๆ ของภาพ เช่น ความกว้าง ความยาวของภาพ จำนวนสี จำนวนบิต ความละเอียด เป็นต้น

2. ข้อมูล palette คือ ข้อมูลที่บอกถึงชุดของจานสี (palette) ที่เกิดจากการผสมแม่สีทั้งสาม คือ RED,GREEN,BLUE มาผสมกันได้เป็นสีต่าง ๆ ตามจำนวนสีของภาพ เช่น รูปขนาด 4 บิต จะมี 16 สี รูป 8 บิต จะมี 256 สี เป็นต้น ซึ่งถ้ามีจำนวนสีน้อย ๆ ก็จะมีการเก็บค่า palette นี้ลงไฟล์ไปด้วย แต่ถ้าเป็นรูปประเภท 24 บิต จะไม่มีค่า palette แต่จะใช้วิธีการเก็บค่าแม่สีทั้งสามลงไปเป็นข้อมูลแทนเพราะถ้าเก็บค่า palette ที่มีถึง 16.7 ล้านสีลงไปด้วย จะเปลืองพื้นที่มาก ข้อแตกต่างที่สำคัญของ BMP คือ ไฟล์ BMP จะเก็บค่าของ palette ชุดละ 4 ไบต์แต่ก็ใช้แค่ 3 ไบต์ เช่นกัน คือ RED,GREEN,BLUE อย่างละ 1 ไบต์

3. ข้อมูลภาพ คือ ข้อมูลสีของภาพแต่ละจุดบนจอภาพที่มาประกอบกันเป็นรูปภาพ ซึ่งค่าที่เก็บนี้จะเป็นค่าที่ใช้ในการชี้ตาราง palette หมายเลขอะไร เช่น จุดแรกมีค่าเป็น 10 ก็ไปเปิดตาราง palette หมายเลข 10 สมมุติว่า ได้ความเข้มของแม่สีเป็น R = 0, G = 0, B = 100 ก็จะได้จุดนี้เป็นสีน้ำเงิน ซึ่งถ้าเป็นในกรณีของรูป 24 บิต จะเป็นการอ่านข้อมูลขึ้นมา 3 ค่าเป็นค่าของแม่สี RGB แล้วนำไปผสมบนจอแทน

2.1.3 การจัดเก็บไฟล์ข้อมูลภาพชนิด BMP

การเก็บไฟล์ข้อมูลภาพชนิด BMP มีการเก็บอยู่ 2 แบบ คือ

1. แบบบีบอัดข้อมูล

1.1 RLE 4 เป็นการบีบอัดข้อมูลแบบ Run-length Encoder แบบ 4 บิต

1.2 RLE 8 เป็นการบีบอัดข้อมูลแบบ Run-length Encoder แบบ 8 บิต

2. แบบไม่ได้บีบอัดข้อมูล

เป็นการเก็บข้อมูลจริง ๆ เลย ซึ่งทำให้ขนาดของไฟล์ค่อนข้างใหญ่ แต่จะทำการแสดงภาพได้เร็วกว่า เพราะไม่ต้องเสียเวลาในการคลายข้อมูล

2.1.4 โครงสร้างของ Header ไฟล์ข้อมูลภาพชนิด BMP

โครงสร้าง Header ของ BMP สามารถเขียนเป็นโครงสร้างข้อมูลแบบภาษาซี ได้ดังนี้

```
typedef  struct  {
        char    id[2];
        long    filesize;
        int     reserve[2];
        long    headersize;
        long    infosize;
        long    width;
        long    depth;
        int     biplanes;
        int     bits;
        long    bicompression;
        long    bisizeimage;
        long    bixpelspermeter;
        long    biypelspermeter;
        long    biclrused;
        long    clrimportant;
} BMPHEADER;
```

โดยแต่ละฟิลด์มีความหมายดังนี้

ฟิลด์ id เป็นรหัสที่ใช้เช็คว่าเป็นไฟล์ BMP หรือไม่ โดยปกติจะให้ค่าเป็น "BM"

ฟิลด์ filesize บอกขนาดของไฟล์

ฟิลด์ reserve สงวนไว้ 2 ไบต์

ฟิลด์ headsize จำนวนไบต์ของ header file

ฟิลด์ infosize บอกขนาดของส่วนที่เป็น information ซึ่งจะมีค่าเป็น 28H เสมอ

ฟิลด์ width,depth บอกขนาดของภาพในความกว้าง และความสูงในหน่วยจุด (pixel)

ฟิลด์ biplanes เป็น 1 เสมอ

ฟิลด์ bits คือ จำนวนบิตต่อ 1 จุด

ฟิลด์ bicompress ถ้าเป็น 0L ก็คือ ไฟล์นี้เป็นแบบ uncompressed
ส่วนฟิลด์ที่เหลือนั้นไม่ค่อยได้ใช้ และ ไม่ค่อยมีความสำคัญมากนัก

2.2 ความรู้ข้อมูลภาพแบบดิจิตอล

ในการประมวลผลสัญญาณภาพด้วยระบบคอมพิวเตอร์ จำเป็นต้องเปลี่ยนข้อมูลหรือสัญญาณภาพที่อยู่ในรูปของอนาลอก ที่อาจจะได้มาจากกล้องวีดิโอ หรือ ภาพที่อยู่บนจอภาพก็ตาม ให้เป็นข้อมูลภาพแบบดิจิตอลเพื่อประโยชน์ในการคำนวน และประมวลผลได้ง่ายขึ้น

2.2.1 ความหมายและนิยามของภาพแบบดิจิตอล

ภาพ (IMAGE) ในเชิงคณิตศาสตร์จะหมายถึง ฟังก์ชัน 2 มิติ f(x,y) โดยที่ x และ y เป็นแกนพิกัดในระนาบ 2 มิติ ค่าฟังก์ชัน f(x,y) จะเป็นสัดส่วนกับความสว่างหรือความเข้มของภาพ ที่ตำแหน่ง (x,y) ซึ่งเราเรียกว่า ระดับสีเทา (GRAY LEVEL)

ภาพ 2 มิติที่แทนด้วยฟังก์ชัน f(x,y) โดย x และ y เป็นแกนในระนาบของภาพค่าของฟังก์ชันที่จุด (x,y) ต้องไม่เป็นศูนย์ และมีค่าน้อยกว่าค่าอนันต์ (finite) นั่นคือ

$$0 < f(x,y) < R$$

โดยธรรมชาติของแสง ซึ่งจะต้องมีแหล่งกำเนิดแสงและส่วนที่สะท้อนของแสง ดังนั้นเราสามารถแยกฟังก์ชัน f(x,y) ออกเป็น 2 ส่วนคือ อลูมินชั่นคอมโพเนนท์ และ รีเฟล็กแทนท์คอมโพเนนท์ จะได้ว่า

$$f(x,y) = i(x,y) * r(x,y)$$

เมื่อ

$$0 < i(x,y) < R$$

และ

$$0 < r(x,y) < 1$$

จากฟังก์ชันต่าง ๆ ที่กล่าวมาแล้วจะเห็นว่า ฟังก์ชันการสะท้อนถูกจำกัดขอบเขตระหว่าง 0 (ซึ่ง หมายถึง การดูดซึมสมบูรณ์) และ 1 (ซึ่งหมายถึง การสะท้อนโดยสมบูรณ์) ธรรมชาติของ i(x,y) ขึ้นอยู่กับแหล่งกำเนิดแสง ในขณะที่ r(x,y) ขึ้นอยู่กับวัตถุที่สะท้อนแสงมาเข้าตา

ดังนั้น ความเข้มของภาพที่จุด (x,y) เราเรียกว่า ระดับสีเทา (GRAY LEVEL)

2.2.2 การแทนภาพด้วยข้อมูลแบบดิจิตอล

ภาพข้อมูลแบบดิจิตอล (digital image) เป็นภาพที่ถูกแปลงมาจากภาพอนาลอก อยู่ในรูปตัวเลข โดยภาพอนาลอกถูกแบ่งเป็นพื้นที่สี่เหลี่ยมเล็ก ๆ ที่เรียกว่า pixel ในแต่ละ pixel จะถูกระบุตำแหน่งโดย (x,y) และค่าระดับสีเทาของ pixel

โดยเราสามารถแปลงภาพเป็นข้อมูลแบบดิจิตอลได้โดยมีขั้นตอนและวิธีการดังนี้

เมื่อเรานำสัญญาณอนาลอกที่ต้องการประมวลผลมาผ่านส่วนที่เรียกว่า ดิจิไตเซอร์ ซึ่งจะมีหน้าที่ในการเปลี่ยนสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอล อุปกรณ์ส่วนนี้ได้แก่ กล้องโทรทัศน์ดิจิไตเซอร์ จากนั้นทำการควอนไตซิ่ง (quantizing) เพื่อที่จะประมวลสัญญาณภาพด้วยระบบคอมพิวเตอร์ ฟังก์ชันของภาพ f(x,y) จะถูกทำให้เป็นสัญญาณไม่ต่อเนื่อง ทั้งระนาบของภาพ ซึ่งเราเรียกว่า การสุ่มภาพ (image sampling) ของฟังก์ชันที่ได้เรียกว่า การควอนไตเซชั่นระดับสีเทา (gray level quantizion) ก็จะได้ข้อมูลภาพที่เป็นดิจิตอล

สมมุติว่าสัญญาณภาพต่อเนื่อง f(x,y) ถูกดิจิไตซ์ในระนาบ x,y เป็นช่วงเท่า ๆ กัน เราสามารถจัด f(x,y) ให้อยู่ในรูปของเมตริกซ์ ขนาด N * N ได้ดังสมการ

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & \cdots & f(0,N-1) \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ f(N-1,0) & f(N-1,1) & f(N-1,2) & \cdots & f(N-1,N-1) \end{bmatrix}$$

ซึ่งทางขวาของสมการ จะเรียกได้ว่า ภาพดิจิตอล และทุก ๆ สมาชิกของเมตริกซ์ จะเรียกว่า pixel จากขบวนการสร้างภาพดิจิตอลข้างต้น จะเห็นว่า เราสามารถทราบขนาดของความละเอียดของภาพ N*N pixels และจำนวนระดับของ gray level ในทางปฏิบัติการทำควอนไตเซชั่นในระบบภาพดิจิตอล จะมีค่าเป็น

$$B = N*N*M \quad \text{bits}$$

เมื่อ B = ขนาดของข้อมูลภาพที่เป็นข้อมูลดิจิตอล

G = จำนวนของระดับ gray level ที่ต้องการใช้ในการเก็บข้อมูลภาพ

M = จำนวน bits ที่ใช้ในการแทนข้อมูลภาพ 1 pixel

โดย M สามารถหาได้จาก

$$G = 2^M$$

2.2.3 ลักษณะการจัดเก็บข้อมูลภาพแบบดิจิตอล

1. ภาพ 2 ระดับ คือ มีเพียงแค่จุดขาวกับจุดดำเท่านั้น โดยแต่ละจุดภาพเป็นข้อมูลขนาด 1 BIT

2. ภาพ 16 ระดับ คือ ในแต่ละจุดภาพจะมีขนาดของข้อมูล 4 BIT ซึ่งทำให้สามารถแสดงภาพได้ 16 ระดับสี หรือ 16 ระดับ GRAY LEVEL ขึ้นอยู่กับภาพนั้นเป็นภาพสีหรือภาพขาวดำ

3. ภาพ 256 ระดับ คือ ในแต่ละจุดภาพจะมีขนาดของข้อมูล 8 BIT ซึ่งทำให้สามารถแสดงภาพได้ 256 ระดับสี หรือ 256 ระดับ GRAY LEVEL ขึ้นอยู่กับภาพนั้นเป็นภาพสีหรือภาพขาวดำ

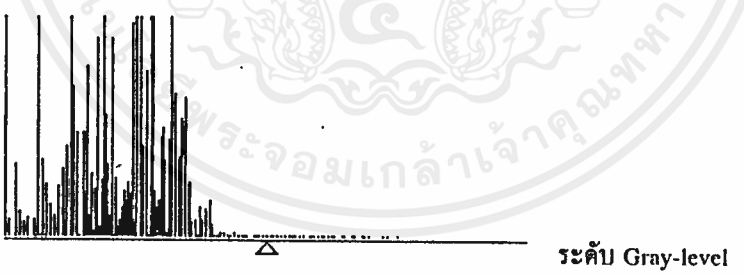
4. ภาพ TRUE COLOR คือ ในแต่ละจุดภาพจะมีขนาดของข้อมูล 24 BIT ทำให้สามารถแสดงผลภาพได้เหมือนภาพจริงที่สุด เพราะสามารถแสดงสีได้ถึง 16,777,216 สี ภาพ TRUE COLOR สามารถแสดงผลได้เฉพาะภาพสีเท่านั้น ไม่สามารถแสดงผลภาพขาวดำได้

การแสดงภาพนี้ใช้วิธี ตั้งค่าของแม่สีในตารางสี โดยอาจเลือกสีเป็นแบบ 16 สี จาก 64 สี หรือ 16 สี จาก 262,144 สี หรือ 256 สี จาก 262,144 สี ขึ้นอยู่กับ MODE การแสดงผล สำหรับ TRUE COLOR จะไม่มีการเลือกสี แสดงผลโดยการส่งค่าสี RGB ผ่าน D/A สีละ 8 BIT ออกไปเลย ความแตกต่างของการแสดงภาพสีและภาพขาวดำ คือ ภาพขาวดำจะต้องตั้งให้แม่สีทั้ง 3 สี มีค่าเท่ากัน เนื่องจาก VGA กำหนดให้แม่สีแต่ละสีใช้ REGISTER 6 BIT ทำให้แต่ละแม่สีแสดงผลได้เพียง 32 ระดับเท่านั้น ยังผลให้เราแสดงภาพ 256 ระดับให้เห็นได้เพียง 64 ระดับเท่านั้น หากต้องการให้เห็นจริงทั้ง 256 ระดับ ต้องแสดงใน MODE TRUE COLOR แล้วให้ RGB มีค่าเท่ากัน ซึ่งใน MODE นี้ จะสามารถใช้ REGISTER ได้ 8 BIT สำหรับแต่ละแม่สี

2.3 ทฤษฎีพื้นฐานของการประมวลผลภาพที่นำมาใช้

2.3.1 Histogram

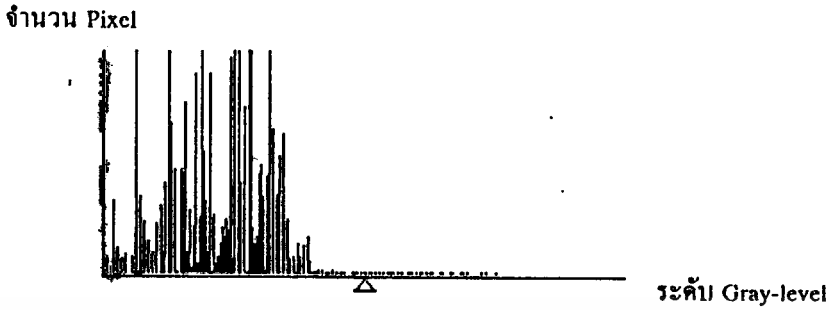
Histogram คือกราฟที่แสดงความสัมพันธ์ระหว่างจำนวน Pixel ของแต่ละระดับ Gray-level ของภาพ เช่น ภาพ 256 สี จะมีค่า Gray-level ที่มีค่าตั้งแต่ Gray-level = 0 ถึง Gray-level = 255



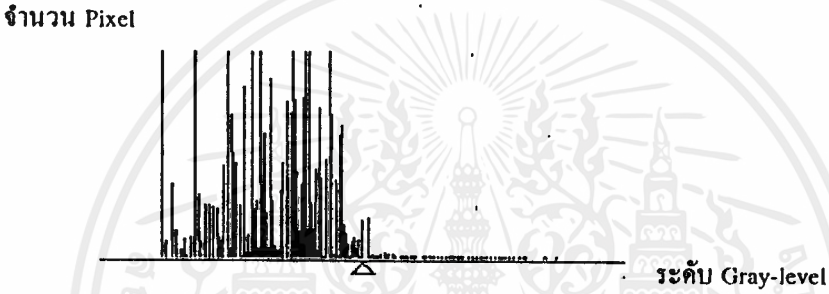
รูปที่ 2.1 แสดง Histogram ที่ได้จากภาพตัวอย่าง

2.3.2 Brightness

การปรับ Brightness ของภาพคือการปรับความสว่างของภาพให้มีค่าความสว่างมากขึ้นหรือลดลง ซึ่งสามารถทำได้โดยปรับหรือ Shift ค่าระดับของ Gray-level ของภาพทั้งหมดทุกค่าด้วยค่าคงที่ค่าหนึ่งที่เราต้องการ อาจเป็นค่าบวก (ความสว่างมากขึ้น) หรือค่าลบ (ความสว่างลดลง) ก็ได้ ดังรูปที่แสดงซึ่งเป็นค่าบวก



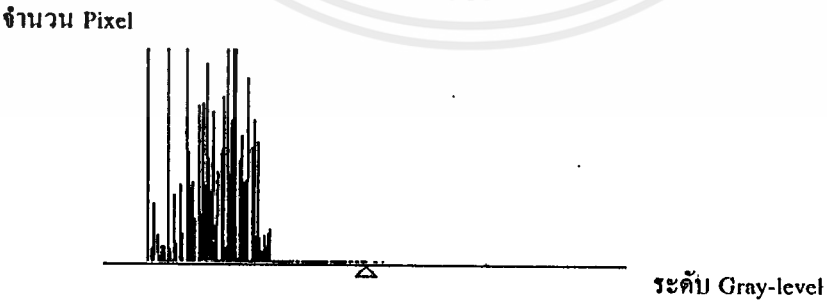
รูปที่ 2.2 แสดง Histogram ก่อนปรับค่า Brightness



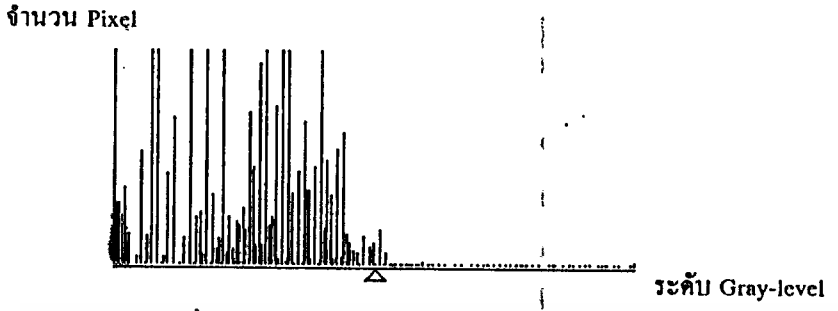
รูปที่ 2.3 แสดง Histogram หลังปรับค่า Brightness เพิ่มขึ้น

2.3.3 Contrast

การปรับ Contrast ของภาพคือการปรับความคมชัดของภาพให้มีค่าความคมชัดมากขึ้นหรือลดลง ซึ่งภาพมีค่าความคมชัดมากขึ้นระดับ Gray-level จะมีลักษณะรวมตัวกัน ส่วนภาพมีค่าความคมชัดลดลงระดับ Gray-level จะมีลักษณะกระจายออก ดังรูปที่แสดง



รูปที่ 2.4 แสดง Histogram ของภาพที่มีค่าความคมชัดมากขึ้น



รูปที่ 2.5 แสดง Histogram ของภาพที่มีค่าความคมชัดน้อยลง

2.3.4 Thresdhold

การปรับค่า Threshold นี้คือ ลักษณะการแยก (Split) ระดับ Gray-level ออกเป็น 2 ค่า คือค่าสูงสุด (Gray-level = 255) และค่าต่ำสุด (Gray-level = 0) ตามที่เรากำหนด โดยการเปรียบเทียบค่า Threshold ที่ตั้งไว้ว่ามีค่ามากกว่าหรือมีค่าน้อยกว่า คือ ถ้าจุดภาพนั้นมีระดับ Gray-level มากกว่าค่า Threshold ก็เปลี่ยนให้จุดภาพนั้นมีระดับ Gray-level = 255 และถ้าจุดภาพนั้นมีระดับ Gray-level น้อยกว่าค่า Threshold ก็เปลี่ยนให้จุดภาพนั้นมีระดับ Gray-level = 0

รูปที่ 2.6 แสดง Histogram หลังจากปรับค่า Threshold

2.3.5 การหาขอบภาพโดยหลักการของ SOBEL

การหาขอบภาพโดยวิธีการของ SOBEL จะเป็นวิธีการพิจารณาหาขอบภาพเป็นส่วน ๆ ไปจนครบตามพื้นที่ที่ต้องการหาขอบภาพ โดยมีหลักการดังนี้ คือ จะใช้วินโดวส์ขนาด 3*3 เป็นขอบเขตในการหาความแตกต่างของข้อมูลภาพภายในวินโดวส์นี้ โดยความแตกต่างของข้อมูลภาพหมายถึงจะให้ความสนใจกับความแตกต่างที่เกิดจากการเปลี่ยนจากสีขาวเป็นสีดำ หรือ จากสีดำเป็นสีขาวของข้อมูลภาพ บริเวณนั้นก็จะได้ขอบของข้อมูลภาพออกมา ซึ่งโดยทั่วไปแล้ว การหาขอบภาพโดยวิธีของ SOBEL นี้จะแยกการหาขอบภาพเป็น 2 ทิศทาง คือ การหาขอบภาพทางแนวนอน และ การหาขอบภาพทางแนวตั้ง ซึ่งการหาขอบภาพทั้ง 2 ทิศทางจะมี ค่าประจำตำแหน่งของแต่ละช่องของวินโดวส์ที่ใช้ในการพิจารณาดังนี้

การหาขอบภาพทางแนวตั้ง

1	0	-1
2	0	-2
1	0	-1

การหาขอบภาพทางแนวนอน

-1	-2	-1
0	0	0
1	2	1

โดยหลักการของการหาขอบภาพด้วยวิธีการของ SOBEL มีดังนี้ คือ

- ทำการรวมค่าที่อยู่ในขอบเขตของวินโดวส์ โดยสมการที่ใช้ในการรวมค่าเป็นดังนี้

การหาขอบภาพทางแนวนอน

$$S(x,y) = \left| \begin{array}{l} -S(x-1,y-1) - 2S(x,y-1) - S(x+1,y-1) + \\ S(x-1,y+1) + 2S(x,y+1) + S(x+1,y+1) \end{array} \right|$$

การหาขอบภาพทางแนวตั้ง

$$S(x,y) = \left| \begin{array}{l} S(x-1,y-1) + 2S(x-1,y) + S(x-1,y+1) + \\ -S(x+1,y-1) - 2S(x+1,y) - S(x+1,y+1) \end{array} \right|$$

เมื่อ x คือ พิกัดทางแนวนอนของข้อมูลรูปภาพ (1 - 298)

y คือ พิกัดทางแนวตั้งของข้อมูลรูปภาพ (1 - 298)

เช่น จากข้อมูลภาพ รูปที่ 2.7 (การหาขอบภาพทางแนวตั้ง) ถ้าต้องการหาผลรวมของข้อมูลตำแหน่ง ที่ (x = 1, y = 1) จะได้ดังนี้

$$S(1,1) = S(0,0) + 2S(0,1) + S(0,2) + -S(2,0) - 2S(2,1) - S(2,2)$$

$$= 1+2(1)+1-1-2(1)-1$$

$$= 0$$

ดังนั้น $S(1,1) = 0$ ดังปรากฏในรูปที่ 2.8

1	0	-1
2	0	-2
1	0	-1

1	1	1	2	2	2	1	1
1	2	1	200	201	200	2	1
1	1	1	201	200	200	1	2
1	1	2	200	201	201	2	2
2	2	1	201	201	200	1	1
1	1	1	200	200	200	1	1
2	1	1	201	201	200	2	2
1	1	1	2	2	2	1	1

รูปที่ 2.7 แสดงการหาผลรวมของข้อมูลภาพตำแหน่งที่ (x = 1, y = 1)

x	x	x	...
x	0	...	...
x	...	...	...
...	...	...	...

รูปที่ 2.8 แสดงผลรวมของข้อมูลภาพตำแหน่งที่ (x = 1, y = 1)

หมายเหตุ x หมายถึง ข้อมูลภาพที่ไม่สามารถนำมาประมวลผลได้ (ไม่นำมาใช้ในการพิจารณา)

- ทำการรวมค่าของข้อมูลภาพดังที่ได้กล่าวมาแล้วข้างต้นจนครบทุกจุดภาพ โดยจากข้อมูลภาพที่ยกตัวอย่างมาข้างต้น เมื่อทำการรวมค่าของข้อมูลภาพจนครบทุกจุดภาพ จะมีข้อมูลปรากฏดังรูปที่ 2.9

x	x	x	x	x	x	x	x
x	0	597	598	3	598	597	x
x	1	797	797	3	796	794	x
x	3	797	797	4	797	795	x
x	3	796	798	3	798	796	x
x	2	797	798	2	797	795	x
x	1	600	600	2	598	596	x
x	x	x	x	x	x	x	x

รูปที่ 2.9 แสดงผลลัพธ์การรวมค่าของข้อมูลภาพครบทุกจุดภาพ

- ทำการปรับค่าของข้อมูลที่ได้จากการรวมค่าของข้อมูลภาพ โดยถ้าข้อมูลที่ได้มีค่าเกิน 255 เราจะให้ค่าของข้อมูลมีค่าสูงสุดเท่ากับ 255 เท่านั้น โดยจากข้อมูลภาพที่ยกตัวอย่างมาข้างต้น เมื่อทำการปรับค่าของข้อมูลภาพจนครบทุกจุดภาพ จะมีข้อมูลปรากฏดังรูปที่ 2.10

x	x	x	x	x	x	x	x
x	0	255	255	3	255	255	x
x	1	255	255	3	255	255	x
x	3	255	255	4	255	255	x
x	3	255	255	3	255	255	x
x	2	255	255	2	255	255	x
x	1	255	255	2	255	255	x
x	x	x	x	x	x	x	x

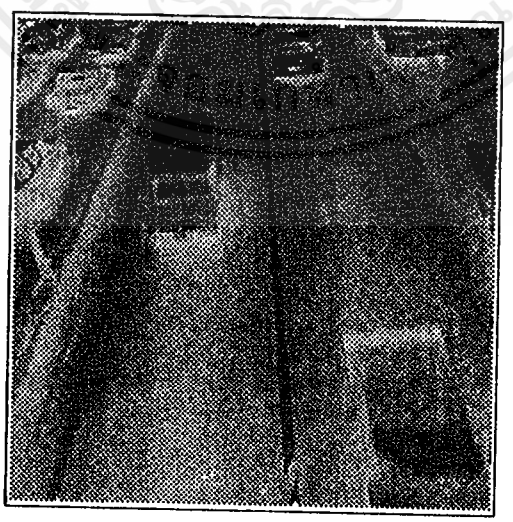
รูปที่ 2.10 แสดงผลลัพธ์การปรับค่าของข้อมูลภาพ

- กำหนดค่า threshold เพื่อใช้เป็นตัวกำหนดการทำให้ภาพเป็นภาพแบบไบนารี (คือข้อมูลภาพมีค่าอยู่เพียง 2 ค่า คือ 0 กับ 1 เท่านั้น) โดยจากข้อมูลภาพที่ยกตัวอย่างมาข้างต้น เมื่อทำการกำหนดค่า threshold = 100 จะมีข้อมูลปรากฏดังรูปที่ 2.11

x	x	x	x	x	x	x	x
x	0	1	1	0	1	1	x
x	0	1	1	0	1	1	x
x	0	1	1	0	1	1	x
x	0	1	1	0	1	1	x
x	0	1	1	0	1	1	x
x	0	1	1	0	1	1	x
x	x	x	x	x	x	x	x

รูปที่ 2.11 แสดงผลลัพธ์ของการทำ Ver. Sobel เมื่อกำหนดค่า threshold = 100

ตัวอย่างของข้อมูลภาพที่ใช้จริงในการหาขอบภาพโดยวิธีการของ SOBEL



รูปที่ 2.12 แสดงรูปต้นแบบ

รูปที่ 2.13 แสดงผลลัพธ์ที่ได้จากการหาขอบทางแนวตั้ง

2.3.6 การติดตามรอยขอบของภาพ

ข้อมูลภาพที่จะนำมาประมวลผลด้วยเทคนิคนี้จะต้องอยู่ในรูปของข้อมูลไบนารี่ (Binary Image) นั่นคือ จุดภาพจะแสดงด้วยตัวเลข "0" กับ "1" เท่านั้น โดยจุดภาพที่มีค่าเป็น "1" แทนจุดดำหรือจุดที่เป็นส่วนของรูปภาพ และจุดภาพที่มีค่าเป็น "0" แทนจุดขาวหรือจุดที่เป็นพื้นหลัง

การทำงานของเทคนิคการติดตามรอยขอบของภาพ เป็นการเดินไต่ไปตามขอบระหว่างส่วนที่เป็นรูปภาพ กับส่วนที่เป็นพื้นหลัง โดยจะตรวจกวาดไปทุก ๆ จุดภาพ (Pixel) โดยจะเริ่มจากจุดมุมซ้ายบนของข้อมูลภาพ ตรวจกวาดไปในทิศทางจากซ้ายไปขวา และเลื่อนจากบนลงล่าง

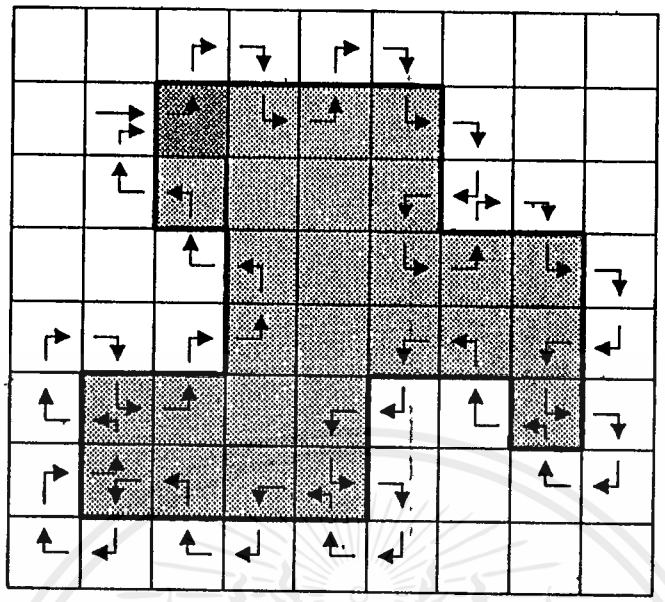
เมื่อตรวจกวาดมาพบจุดภาพใด ๆ ที่มีค่าของจุดภาพเป็น "1" ก็จะเปลี่ยนลักษณะการเคลื่อนที่ไปยังจุดภาพจุดถัดไปเสียใหม่ โดยมีเงื่อนไขของการเคลื่อนที่ดังต่อไปนี้

1. ถ้าจุดที่อยู่ปัจจุบันเป็นจุดของภาพหรือมีค่าของจุดเป็น "1" ให้เลี้ยวซ้าย แล้วก้าวเดินไปข้างหน้าไปยังจุดถัดไป

2. ถ้าจุดที่อยู่ปัจจุบันเป็นพื้นเบื้องหลังหรือมีค่าของจุดเป็น "0" ให้เลี้ยวขวา แล้วก้าวเดินไปข้างหน้าไปยังจุดถัดไป

3. การเคลื่อนที่จะสิ้นสุดลง เมื่อจุดที่อยู่ปัจจุบันเป็นจุดเดียวกันกับจุดเริ่มต้น

ดังรูปที่แสดงจะแสดงลักษณะการทำงานของเทคนิคการติดตามรอยขอบภาพ ซึ่งจะแสดงการเคลื่อนที่ไปตามจุดต่างๆ ที่เป็นขอบของภาพ เริ่มจากจุดที่แรเงาไว้ซึ่งเป็นจุดของภาพจุดแรกที่ตรวจกวาดมาพบ การเคลื่อนที่จะเป็นไปตามเงื่อนไขที่กำหนด เมื่อมีการเคลื่อนที่วนกลับมาถึงจุดที่เป็นจุดเริ่มต้น ก็จะทราบจุดที่เป็นขอบของภาพได้ทั้งหมด



รูปที่ 2.14 แสดงลักษณะการติดตามรอยขอบของภาพ

2.3.7 Noise Removing

เป็นวิธีที่ใช้ในการกำจัดข้อมูลภาพส่วนที่ไม่ต้องการ (noise) ออกจากข้อมูลภาพที่นำมาประมวลผล โดยใช้หลักการดังนี้ คือ

- ใช้วินโดวส์ขนาด 3*3 เป็นขอบเขตที่ใช้ทำการรวมค่าที่อยู่ในขอบเขตของวินโดวส์ โดยค่าประจำตำแหน่งของแต่ละช่องในวินโดวส์มีค่าดังนี้

a1	a2	a3
a4	a5	a6
a7	a8	a9

1	1	1
1	1	1
1	1	1

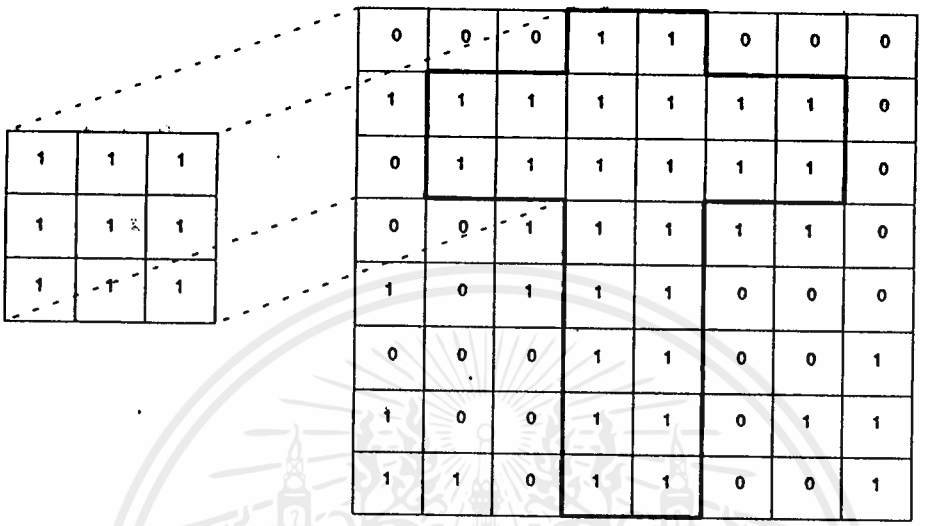
รูปที่ 2.15 แสดงค่าประจำตำแหน่งแต่ละช่องในวินโดวส์

- ทำการรวมค่าที่อยู่ในขอบเขตของวินโดวส์ โดยสมการที่ใช้ในการรวมค่าเป็นดังนี้

$$
\begin{aligned}
S(x,y) = {} & S(x-1,y-1) + S(x-1,y) + S(x-1,y+1) + \\
& S(x,y-1) + S(x,y) + S(x,y+1) + \\
& S(x+1,y-1) + S(x+1,y) + S(x+1,y+1)
\end{aligned}
$$

เมื่อ x คือ พิกัดทางแนวนอนของข้อมูลรูปภาพ (1 - 298)

y คือ พิกัดทางแนวตั้งของข้อมูลรูปภาพ (1 - 298)



1	1	1
1	1	1
1	1	1

0	0	0	1	1	0	0	0
1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	1	1	1	1	1	0
1	0	1	1	1	0	0	0
0	0	0	1	1	0	0	1
1	0	0	1	1	0	1	1
1	1	0	1	1	0	0	1

รูปที่ 2.16 แสดงการหาผลรวมของข้อมูลภาพตำแหน่งที่ (x = 1, y = 1)

เช่น จากข้อมูลภาพ รูปที่ 2.16 ถ้าต้องการหาผลรวมของข้อมูลตำแหน่งที่ (x = 1, y = 1) จะได้ดังนี้

$$S(1,1) = S(0,0) + S(0,1) + S(0,2) + S(1,0) + S(1,1) + S(1,2) + S(2,0) + S(2,1) + S(2,2)$$

$$= 0+0+0+1+1+1+0+1+1$$

$$= 5$$

ดังนั้น $S(1,1) = 5$ ดังปรากฏในรูปที่ 2.17

x	x	x	...
x	5	...	...
x	...	...	...
...	...	...	...

รูปที่ 2.10 **รูปที่ 2.17 แสดงผลรวมของข้อมูลภาพตำแหน่งที่ (x = 1, y = 1)**

- ทำการรวมค่าของข้อมูลภาพดังที่ได้กล่าวมาแล้วข้างต้นจนครบทุกจุดภาพ โดยจากข้อมูลภาพที่ยกตัวอย่างมาข้างต้น เมื่อทำการรวมค่าของข้อมูลภาพจนครบทุกจุดภาพ จะมีข้อมูลปรากฏดังรูปที่ 2.18

x	x	x	x	x	x	x	x
x	5	7	8	8	7	4	x
x	6	8	9	9	9	6	x
x	5	7	9	8	8	5	x
x	3	5	8	7	5	4	x
x	3	4	7	6	4	4	x
x	3	4	6	6	4	4	x
x	x	x	x	x	x	x	x

รูปที่ 2.18 แสดงผลลัพธ์การรวมค่าของข้อมูลภาพครบทุกจุดภาพ

- กำหนดค่า threshold เพื่อใช้เป็นตัวกำหนดการกำจัดข้อมูลส่วนที่ไม่ต้องการ (noise)

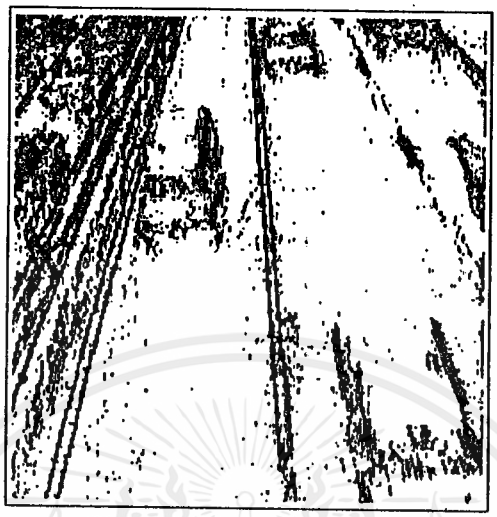
โดยจากข้อมูลภาพที่ยกตัวอย่างมาข้างต้น เมื่อทำการกำหนดค่า threshold = 6 จะมีข้อมูลปรากฏดังรูปที่ 2.19

x	x	x	x	x	x	x	x
x	0	1	1	1	1	0	x
x	1	1	1	1	1	1	x
x	0	1	1	1	1	0	x
x	0	0	1	1	0	0	x
x	0	0	1	1	0	0	x
x	0	0	1	1	0	0	x
x	x	x	x	x	x	x	x

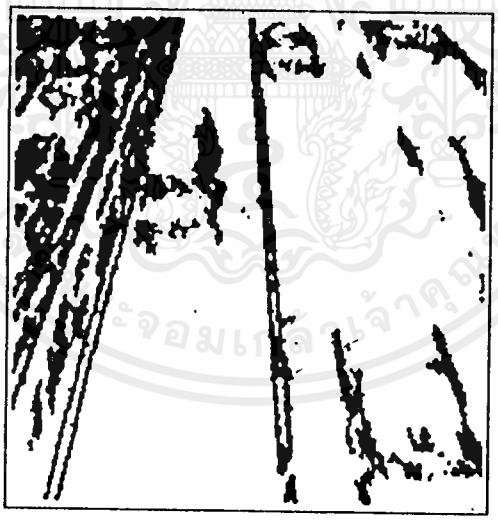
รูปที่ 2.19 แสดงผลลัพธ์ของการทำ Noise Removing เมื่อกำหนดค่า threshold = 6

หมายเหตุ x หมายถึง ข้อมูลภาพที่ไม่สามารถนำมาประมวลผลได้ (ไม่นำมาใช้ในการพิจารณา)

ตัวอย่างของข้อมูลภาพที่ใช้จริงในการทำ Noise Removing



รูปที่ 2.20 ข้อมูลภาพก่อนทำ Noise Removing



รูปที่ 2.21 ข้อมูลภาพหลังทำ Noise Removing

บทที่ 3

การคำนวณและการสร้าง

3.1 รายละเอียดและขั้นตอนการดำเนินงาน

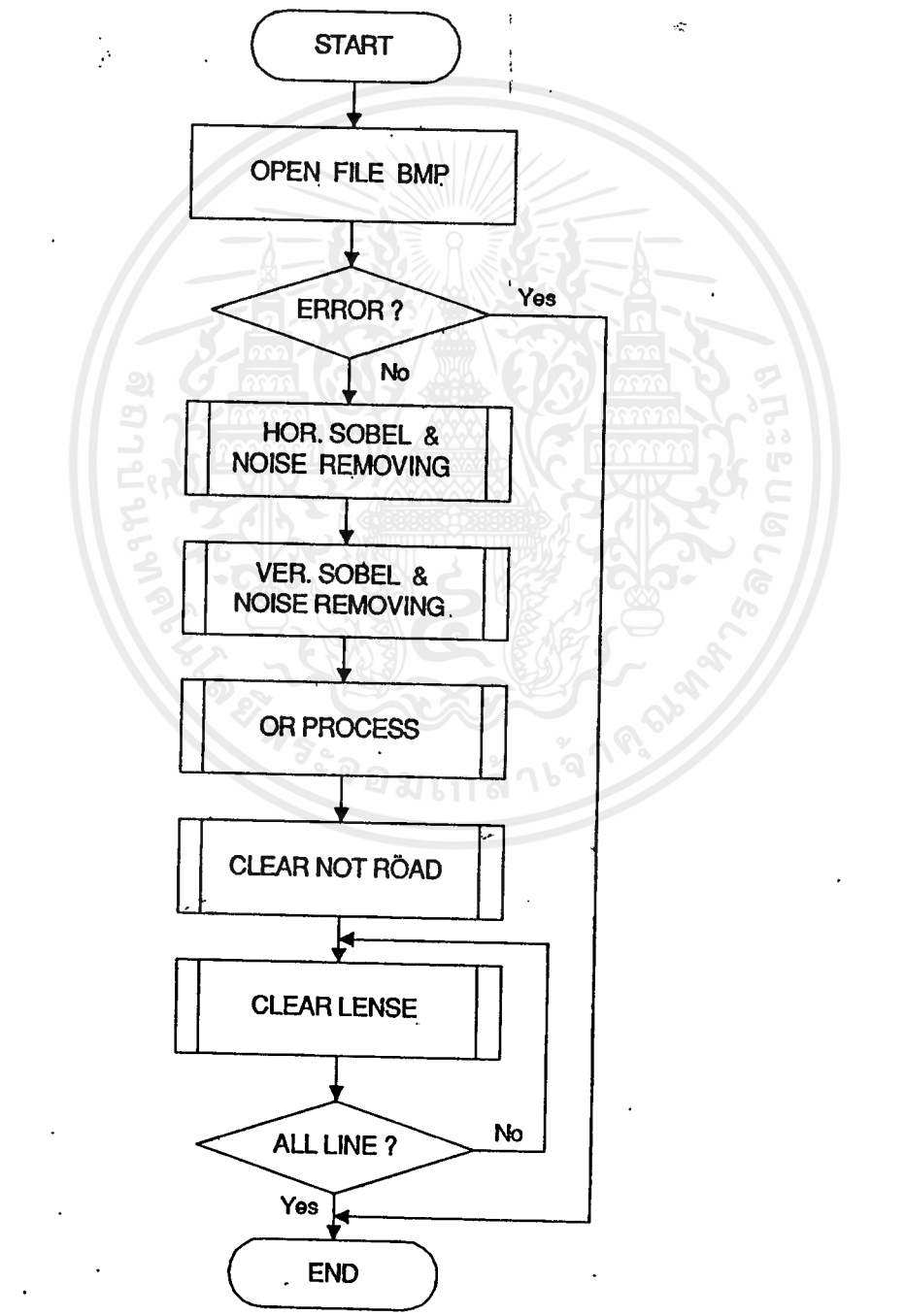
1. ถ่ายภาพรถยนต์บนทางด่วนด้วยกล้องวีดีโอเก็บลงม้วนเทป
2. การแปลงสัญญาณภาพเป็นข้อมูลแบบดิจิตอลในไฟล์ฟอร์แมต BMP
   - ต่อสายสัญญาณจากกล้องวีดีโอ (video) เข้ากับการ์ดวีดีโอ
   - ใช้ software ของการ์ดวีดีโอแปลงสัญญาณภาพวีดีโอ เป็นไฟล์ข้อมูลภาพแบบ . AVI
   - ใช้ software เปิดไฟล์ข้อมูลภาพ . AVI ที่ละเฟรมต่อเนื่องกัน ซึ่งเราจะต้องจับภาพ (capture) ภาพ . AVI ในแต่ละเฟรมที่ต่อเนื่องกันลง clipboard และจัดเก็บเป็นข้อมูลภาพแบบ . BMP โดยมีขนาด 300*300 Pixel เป็น Gray-level 256 ระดับ
   - โดยการ จัดเก็บเป็นข้อมูลภาพแบบ . BMP นั้นการตั้งชื่อของไฟล์ข้อมูลภาพจะต้องเป็นตัวเลขที่เรียงลำดับต่อเนื่องกันไป เช่น ไฟล์ข้อมูลภาพจะต้องเริ่มจาก 1,2,3,4,..... ,n ตามลำดับ
3. เขียนโปรแกรมประมวลผลภาพกับไฟล์ข้อมูลภาพที่จัดเตรียมไว้แล้วตามหลักการที่วางแผนไว้คือ
   - การหาความแตกต่างระหว่างรถยนต์กับถนน
   - การตรวจสอบรถยนต์ที่ต้องการ
   - การคำนวณหาความเร็วของรถยนต์
4. สรุปผลสภาพการจราจรบนทางด่วนที่ได้ทางจอแสดงผล

3.2 หลักการที่ใช้ในการประมวลผล

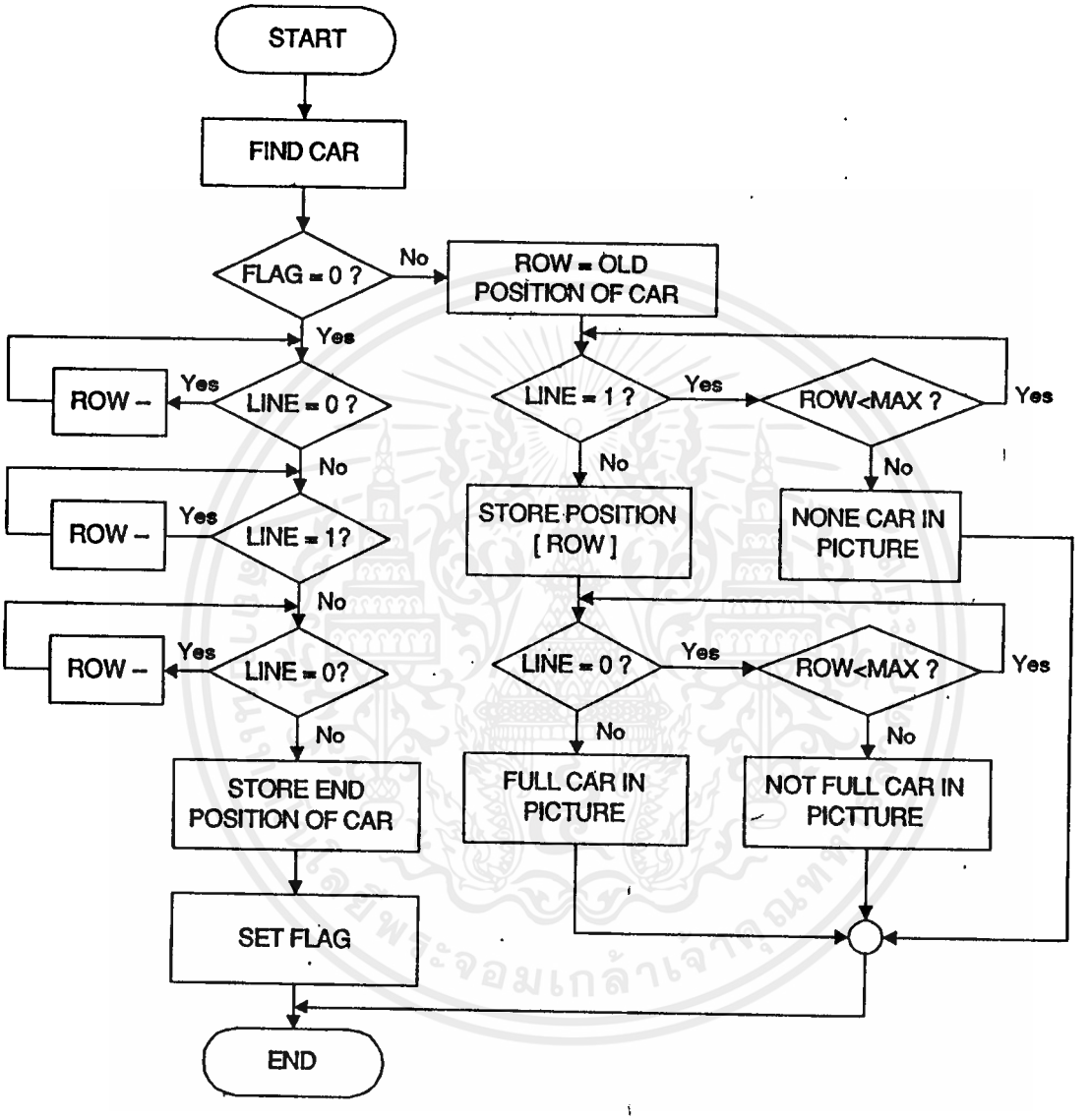
3.2.1 การหาความแตกต่างระหว่างรถยนต์กับถนน

การแยกรถยนต์ออกจากถนน เริ่มจากใช้ SOBEL เพื่อหาขอบภาพทางแนวนอนของข้อมูลภาพทั้งภาพ เมื่อเสร็จแล้วจะมีส่วนของข้อมูลที่มีลักษณะเป็นจุด ๆ ซึ่งเราต้องใช้ NOISE REMOVING ในการกำจัดจุดที่ไม่ต้องการเพื่อเป็นการลดสิ่งที่ไม่ต้องการออกไปก่อนเป็นบ้างส่วน จากนั้นก็จะได้ขอบภาพทางแนวนอนซึ่งกำจัดส่วนที่ไม่ต้องการออกแล้ว ต่อไปก็นำข้อมูลภาพเดียวกันมาใช้ SOBEL เพื่อหาขอบภาพทางแนวตั้งของข้อมูลภาพทั้งภาพเช่นเดียวกันเมื่อเสร็จแล้วเราก็ใช้ NOISE REMOVING ในการกำจัดสิ่งที่ไม่ต้องการออกไปก่อนเป็นบ้างส่วน จากนั้นก็จะได้ขอบภาพทางแนวตั้งที่กำจัดส่วนที่ไม่ต้องการออกแล้ว และนำผลลัพธ์จากการหาขอบภาพทางแนวนอนมาทำการ OR กับผลลัพธ์จากการหาขอบภาพทางแนวตั้งก็จะได้ขอบภาพที่สมบูรณ์ จากนั้นทำการเคลียร์ข้อมูลภาพในส่วนที่ไม่ใช่ถนน และ เคลียร์ข้อมูลภาพในส่วนที่เป็นเลนซ์ เพื่อให้ข้อมูลภาพเหลือแต่สิ่งที่คาดว่าน่าจะเป็นรถยนต์ จากนั้นใช้ EDGE FOLLOW ไล่ขอบภาพ ที่ละเลนซ์ ถ้าขนาดของการวนครบรอบเป็น

ไปตามที่กำหนดไว้จะถือว่าเป็นรถยนต์ ก็จะทำการกำหนดข้อมูลของรถยนต์ให้มีค่าเป็น "1" แต่ถ้าขนาดของการวนครบรอบไม่เป็นไปตามที่กำหนดไว้จะถือว่าเป็นส่วนที่ไม่ต้องการ จะกำหนดข้อมูลของชุดข้อมูลนี้เป็น "0" ดังนั้นเมื่อเสร็จตามขั้นตอนที่กล่าวมาข้อมูลภาพจะเป็นข้อมูลแบบไบนารีที่มีกลุ่มของข้อมูลเป็น "1" หมายถึง เป็นรถยนต์ ส่วนกลุ่มของข้อมูลเป็น "0" หมายถึง เป็นถนนและส่วนที่ไม่สนใจในการประมวลผล เพื่อให้ง่ายต่อการทำความเข้าใจสามารถดูได้จากแผนผังการทำงาน (flowchart)

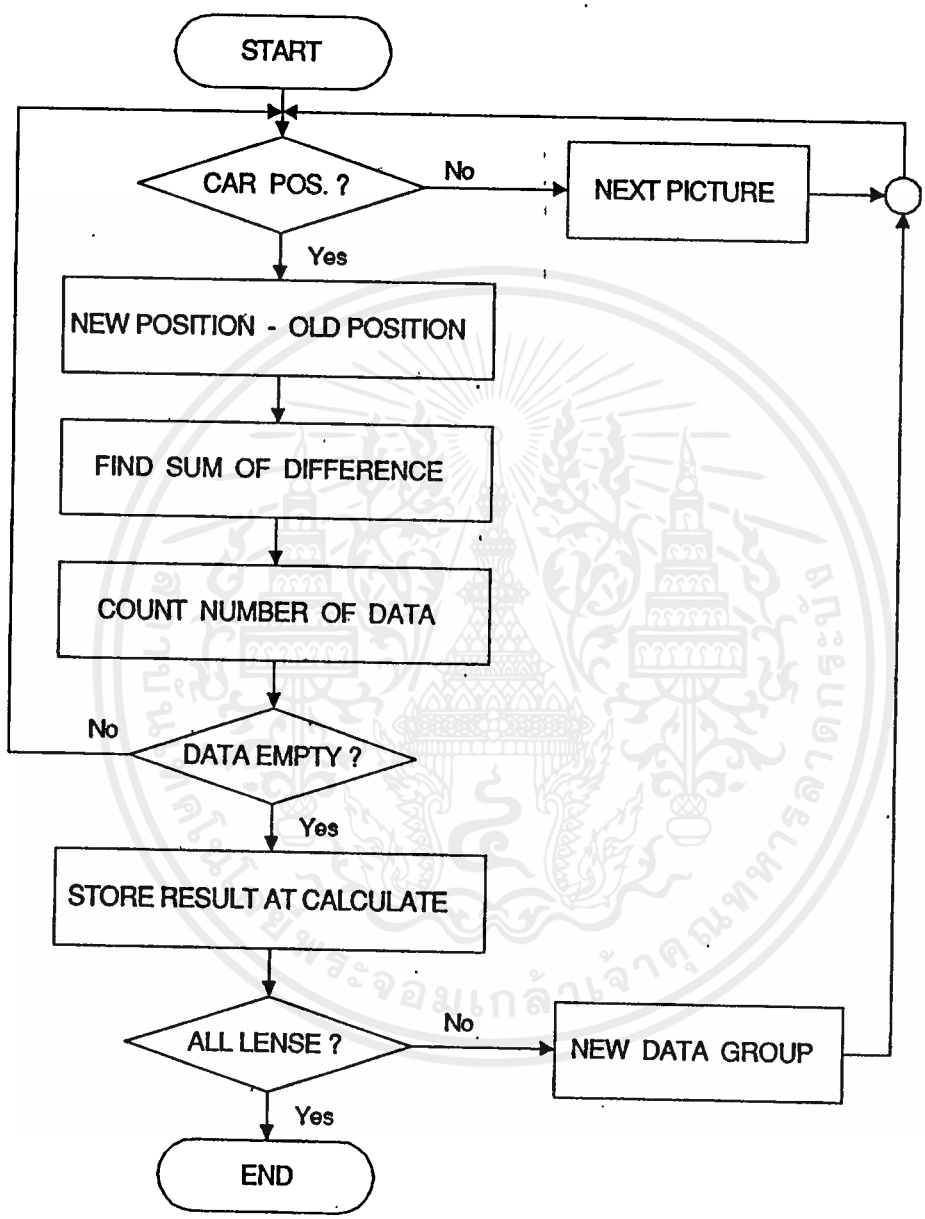


3.2.2 การตรวจสอบรถยนต์ที่ต้องการ



การตรวจหารถยนต์คันที่จะใช้ในการพิจารณาจะแยกหาในแต่ละเลนซ์ โดยที่จะใช้รถยนต์คันแรกที่สมบูรณ์ในแต่ละเลนซ์ (เริ่มพิจารณาจากขอบภาพที่มีทิศทางสวนกับการวิ่งของรถยนต์) เป็นคันที่จะใช้ในการประมวลผล ถ้าในภาพไม่มีรถยนต์คันที่สมบูรณ์อยู่เลยก็จะทำการเปิดแฟ้มข้อมูลภาพถัดไปเรื่อย ๆ เมื่อได้รถยนต์คันที่สมบูรณ์แล้วก็จะมีการเก็บตำแหน่งของท้ายรถยนต์ของแต่ละเลนซ์ไว้ แล้วก็ทำการประมวลผลแบบเดียวกันนี้กับภาพถัดไปจนกว่าจะมีรถยนต์คันใดที่ใช้ในการพิจารณาพ้นขอบภาพไปก็จะหยุดการประมวลผลกับแฟ้มข้อมูลภาพ ซึ่งการหยุดการประมวลผลกับแฟ้มข้อมูลภาพอาจเกิดจากแฟ้มข้อมูลภาพที่ใช้ในการประมวลผลหมดก่อนที่จะมีรถยนต์ที่ใช้ในการพิจารณาคันใดพ้นขอบภาพไป เพื่อให้ง่ายต่อการทำความเข้าใจสามารถดูได้จากแผนผังการทำงาน (flowchart)

3.2.3 การคำนวณหาความเร็วของรถยนต์



การคำนวณหาความเร็วจะนำข้อมูลตำแหน่งของท้ายรถยนต์ที่ได้เก็บไว้ก่อนแล้วตอนประมวลผลกับแฟ้มข้อมูลภาพ โดยการประมวลผลเพื่อหาความเร็วจะทำที่ละเลนซ์โดยจะใช้ตำแหน่งที่ได้จากท้ายรถยนต์คันเดียวกันของแต่ละภาพในการประมวลผล โดยนำตำแหน่งของท้ายรถยนต์จากข้อมูลที่มีอยู่ในลำดับท้ายสุดลบด้วยตำแหน่งของท้ายรถยนต์จากข้อมูลที่มีอยู่ในลำดับรองลงมา ก็จะได้ความแตกต่างของจำนวนจุดภาพของรถยนต์ระหว่างแฟ้มข้อมูลภาพ แล้วหาความแตกต่างของจำนวนจุดภาพของรถยนต์ระหว่างแฟ้มข้อมูลภาพ ต่อไปจน

หมดข้อมูล แล้วก็นำความแตกต่างของจำนวนจุดภาพของรถยนต์ที่หามาได้มาทำการหาความ แตกต่างของจำนวนจุดภาพของรถยนต์โดยเฉลี่ย แล้วใช้ค่าแตกต่างของจำนวนจุดภาพของรถยนต์โดยเฉลี่ยนี้ในการแสดงผล โดยมีความหมายว่า ความเร็วของรถยนต์โดยเฉลี่ยจะมีค่าเท่ากับ ความแตกต่างของจำนวนจุดภาพของรถยนต์โดยเฉลี่ยต่อหนึ่งหน่วยเวลา เช่น ค่าแตกต่างของจำนวนจุดภาพของรถยนต์โดยเฉลี่ย เป็น 10 มีความหมายว่า ความเร็วของรถยนต์โดยเฉลี่ยจะมีค่าเป็น 10 จุดภาพต่อหนึ่งหน่วยเวลา เพื่อให้ง่ายต่อการทำความเข้าใจสามารถดูได้จากแผนผังการทำงาน (flowchart)

บทที่ 4

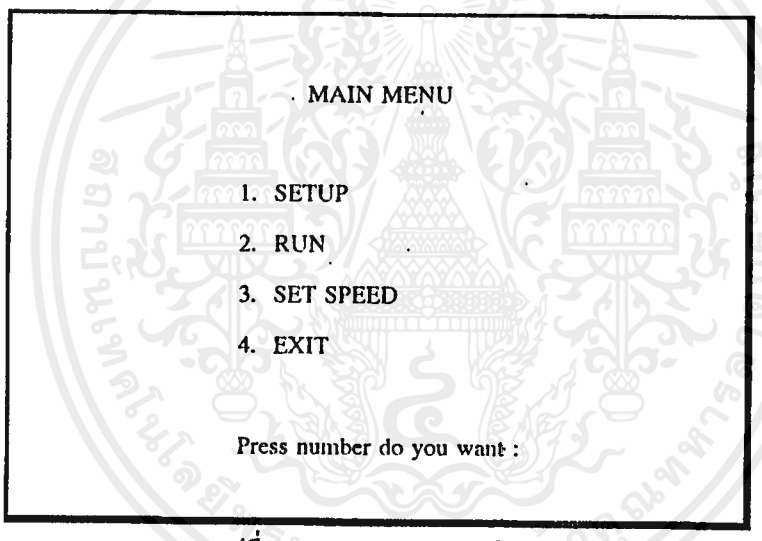
การทดลองและผลการทดลอง

4.1 การใช้งานโปรแกรมการประมวลผลภาพ

1. เมื่ออยู่ที่ DOS PROMPT และอยู่ใน PROJECT SUB DIRECTORY พิมพ์

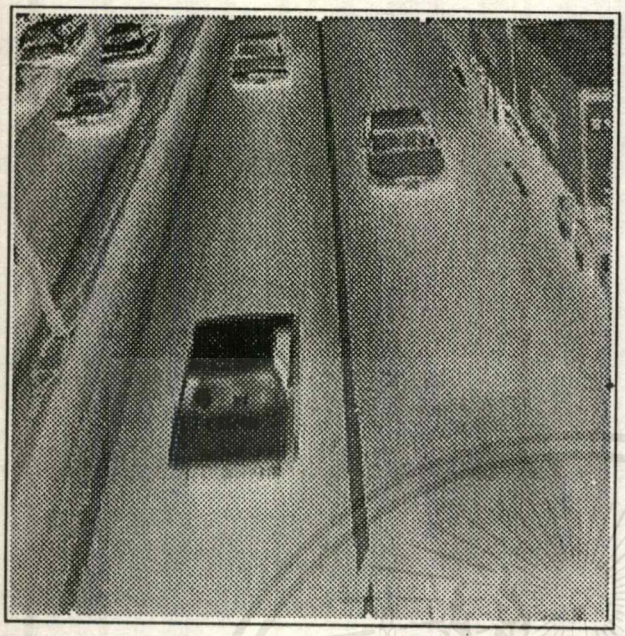
   c:\project\> proj <enter>

2. จะเห็น MAIN MENU จะปรากฏดังรูป



รูปที่ 4.1 แสดงเมนูหลักของโปรแกรม

3. กำหนดบริเวณที่จะทำการประมวลผลในไฟล์รูปภาพ
โดยเลือกหัวข้อที่ 1 [SETUP] จะปรากฏดังรูป



รูปที่ 4.2 แสดงภาพหลังการ SETUP

เมื่อเลือกหัวข้อที่ 1 จะมีรูปที่ต้องการจะประมวลผลปรากฏบนจอภาพเพื่อที่ให้ผู้ใช้ได้ทำการกำหนดขอบเขตที่ต้องการประมวลผล เมื่อทำการกำหนดขอบเขตที่ต้องการประมวลผลแล้วจะได้ผลลัพธ์ปรากฏบนจอภาพดังรูปที่ 4.2

โดยมีวิธีการและขั้นตอนการกำหนดขอบเขตที่ต้องการดังนี้

3.1 การควบคุมเคอร์เซอร์

โปรแกรมมีการควบคุมเคอร์เซอร์อยู่ 2 ลักษณะ คือ

1. การควบคุมเคอร์เซอร์แบบละเอียด มีคีย์ที่ใช้อยู่ 4 คีย์ คือ

คีย์ลูกศรชี้ขวา	เลื่อนเคอร์เซอร์ไปทางขวาหนึ่งตำแหน่ง
คีย์ลูกศรชี้ซ้าย	เลื่อนเคอร์เซอร์ไปทางซ้ายหนึ่งตำแหน่ง
คีย์ลูกศรชี้ล่าง	เลื่อนเคอร์เซอร์ไปข้างล่างหนึ่งตำแหน่ง
คีย์ลูกศรชี้บน	เลื่อนเคอร์เซอร์ไปข้างบนหนึ่งตำแหน่ง

ตารางที่ 4.1 แสดงคีย์ที่ใช้ในการควบคุมเคอร์เซอร์แบบละเอียด

2. การควบคุมเคอร์เซอร์แบบหยาบ มีคีย์ที่ใช้อยู่ 4 คีย์ คือ

d	เลื่อนเคอร์เซอร์ไปทางขวาขึ้สิบตำแหน่ง
e	เลื่อนเคอร์เซอร์ไปข้างบนขึ้สิบตำแหน่ง
s	เลื่อนเคอร์เซอร์ไปทางซ้ายขึ้สิบตำแหน่ง
x	เลื่อนเคอร์เซอร์ไปข้างล่างขึ้สิบตำแหน่ง

ตารางที่ 4.2 แสดงคีย์ที่ใช้ในการควบคุมเคอร์เซอร์แบบหยาบ

3.2 การกำหนดและยกเลิกตำแหน่ง

โปรแกรมมีการกำหนดคีย์ที่ใช้ 2 คีย์ คือ

enter	กำหนดตำแหน่งที่ต้องการ
c	ยกเลิกตำแหน่งที่ต้องการ

ตารางที่ 4.3 แสดงคีย์ที่ใช้ในการกำหนดและยกเลิกตำแหน่ง

3.3 การออกจากส่วน SETUP ทำได้โดยคีย์

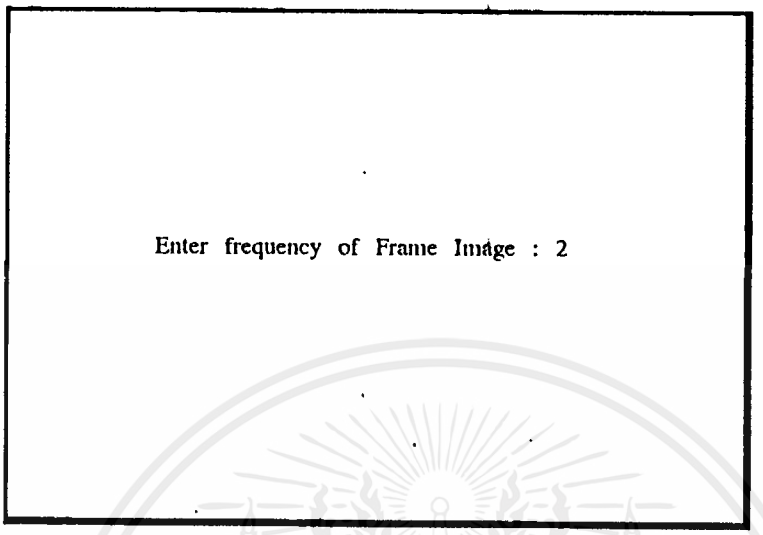
โปรแกรมมีการกำหนดคีย์ที่ใช้ คือ

q	จบการกำหนดขอบเขตที่สนใจ

ตารางที่ 4.4 แสดงคีย์ที่ใช้ในการออกจากส่วน SETUP

4. กำหนดระยะห่างของภาพต่อไปที่จะทำการประมวลผล

โดยเลือกหัวข้อที่ 3 [SET SPEED] จะปรากฏดังรูป

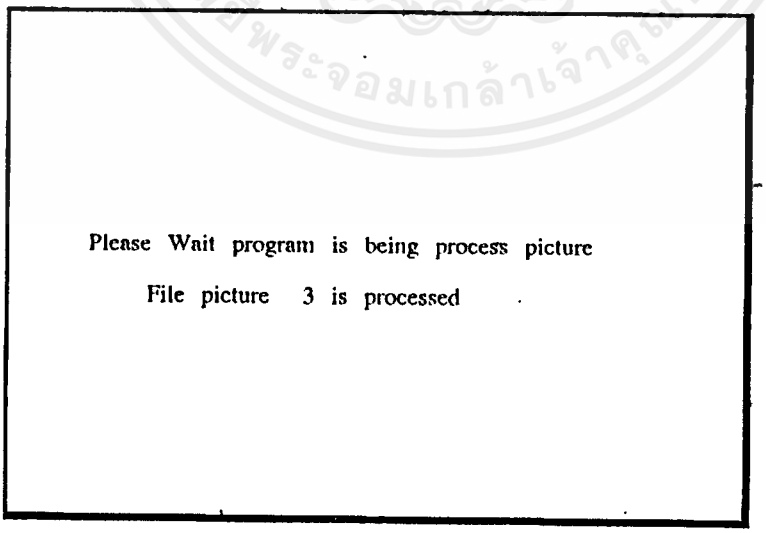


รูปที่ 4.3 แสดงภาพหลังการ SET SPEED

เช่น ถ้ากำหนดระยะห่างของภาพต่อไปเป็น 2 โปรแกรมจะทำการประมวลผลเริ่มจากไฟล์ที่ 1,3,5,7...

หมายเหตุ ถ้ามีการกำหนดระยะห่างของภาพต่อไปไว้ก่อนแล้วและไม่ต้องการเปลี่ยนระยะห่างของภาพต่อไปที่จะทำการประมวลผลก็ไม่จำเป็นต้องกำหนดอีก

5. ให้โปรแกรมทำการประมวลผล โดยเลือกหัวข้อที่ 2 [RUN] จะปรากฏดังรูป

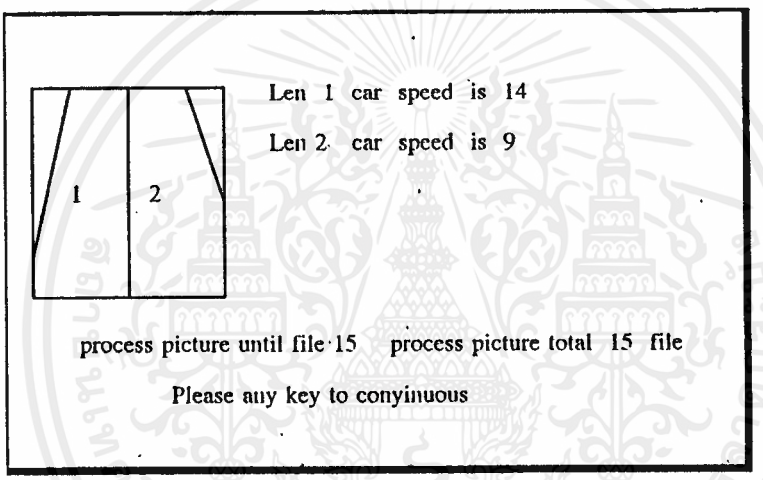


รูปที่ 4.4 แสดงภาพขณะประมวลผล

โดยในระหว่างที่โปรแกรมทำการประมวลผลจะมีการแสดงข้อความบอกว่าตอนนี้โปรแกรมทำการประมวลผลถึงไฟล์รูปภาพที่เท่าไดแล้ว ดังเช่นรูปข้างล่าง

หมายเหตุ ถ้าต้องการให้โปรแกรมทำการประมวลผลอีกครั้งโดยบริเวณขอบเขตที่สนใจยังเป็นบริเวณเดิม และไม่ต้องการเปลี่ยนระยะห่างของภาพต่อไปที่จะทำการประมวลผลเมื่อโปรแกรมกลับมาที่เมนูหลักก็สามารถเลือกหัวข้อที่ 2 [RUN] เพื่อให้โปรแกรมประมวลผลอีกครั้งได้เลย

โดยเมื่อโปรแกรมประมวลผลเสร็จก็จะมีการรายงานผลการประมวลผลดังในรูป ซึ่งจะมีรายละเอียดดังนี้ จะรายงานความเร็วของแต่ละเลนซ์เทียบกับหน่วยเวลา มีการแสดงรูปย่อของรูปที่ทำการ SETUP เพื่อทราบว่าเลนซ์ที่รายงานผลอยู่ในภาพบริเวณใด และยังรายงานว่าโปรแกรมทำการประมวลผลทั้งหมดกี่ไฟล์ภาพ และประมวลผลถึงไฟล์ข้อมูลภาพที่เท่าใดอีกด้วย



รูปที่ 4.5 แสดงผลของการประมวลผล

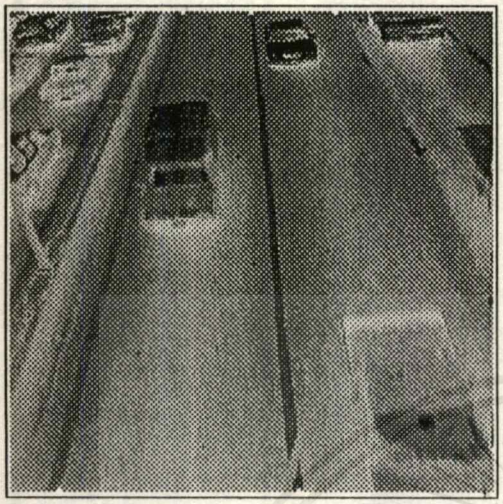
6. ออกจากโปรแกรม โดยเลือกหัวข้อที่ 4 [EXIT]
เมื่อจบการทำงานของโปรแกรมก็จะกลับมาที่ DOS PROMPT เหมือนก่อนการเรียกใช้โปรแกรม

4.2 ขั้นตอนการทดลอง

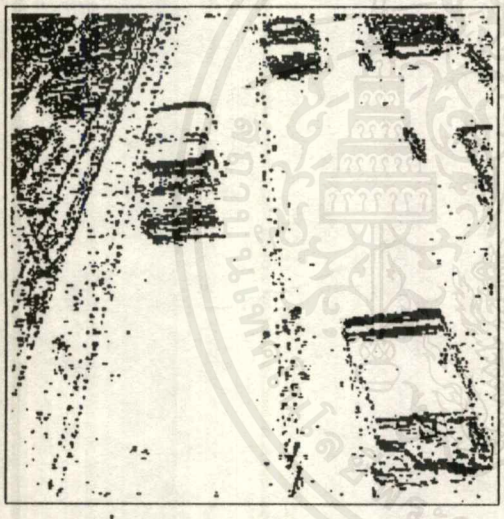
หลังจากที่ได้ศึกษาการใช้โปรแกรมเรียบร้อยแล้ว ต่อไปเป็นการทดลองกับภาพทดสอบซึ่งมีทั้งหมด 3 ชุด โดยมีขั้นตอนการทดลองตามลำดับดังนี้

1. จากเมนูหลักเลือก SETUP เพื่อทำการกำหนดตำแหน่งขอบเขตของเลนซ์และถนน
2. หลังจากเสร็จขั้นตอนที่ 1 แล้ว เลือก SET SPEED ของไฟล์ที่จะทำการอ่านตามต้องการ
3. เลือก RUN จากเมนูหลัก ในขั้นตอนนี้โปรแกรมจะประมวลผลการทำงานโดยอัตโนมัติจะไม่เห็นผลลำดับขั้นตอนการทำงาน จนกระทั่งได้ผลลัพธ์ที่ต้องการ
แสดงลำดับขั้นตอนการทำงานทุกขั้นตอนของการประมวลผล ซึ่งได้แสดงไว้แล้วดังรูปข้างล่างนี้
4. เลือก EXIT เพื่อออกจากการทำงาน

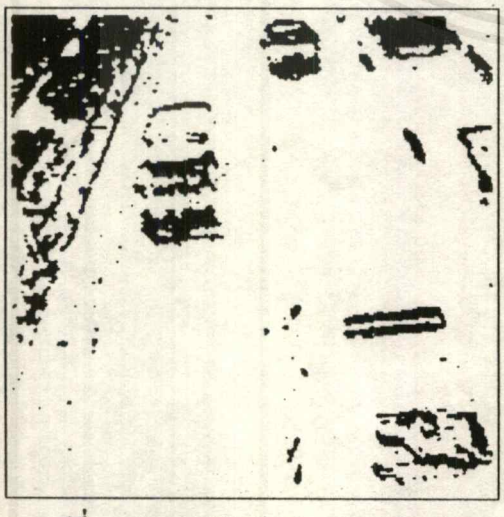
ชุดภาพที่ 1



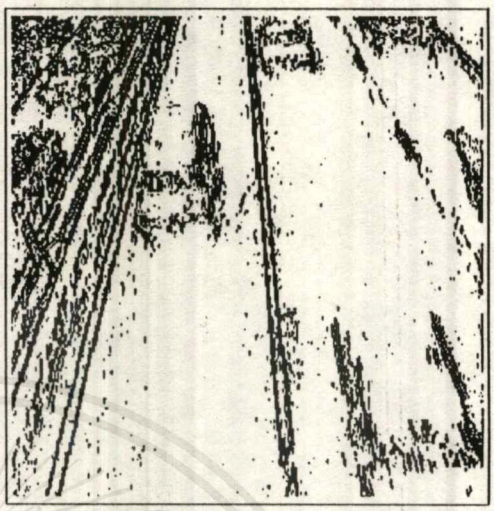
รูปที่ 4.6 แสดงรูปต้นแบบ



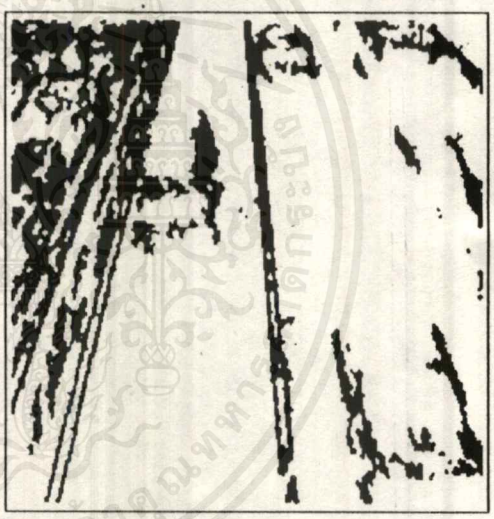
รูปที่ 4.7 แสดงหาขอบทางแนวนอน



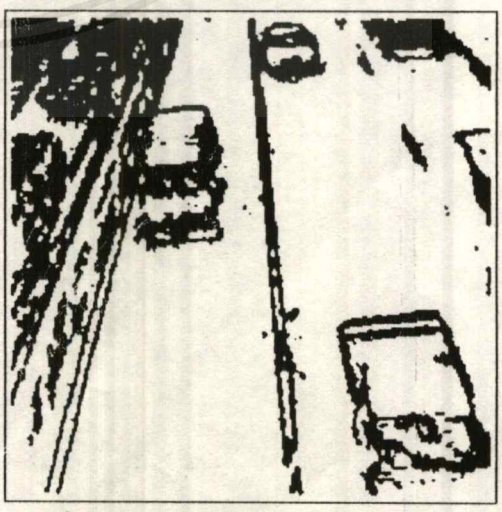
รูปที่ 4.8 แสดงกำจัด Noise ทางแนวนอน



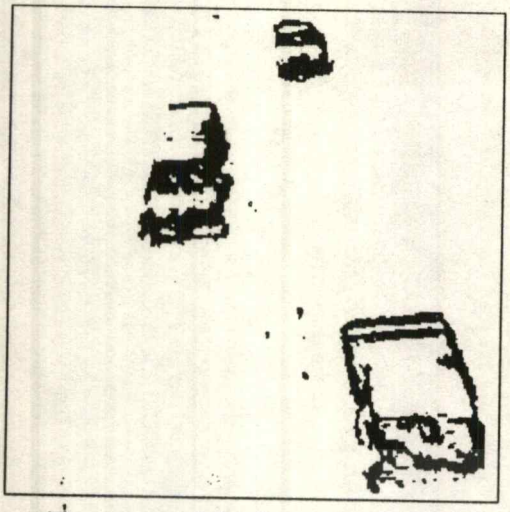
รูปที่ 4.9 แสดงหาขอบทางแนวตั้ง



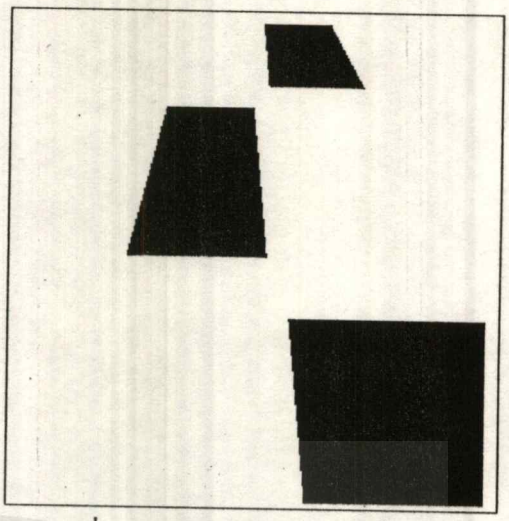
รูปที่ 4.10 แสดงกำจัด Noise ทางแนวตั้ง



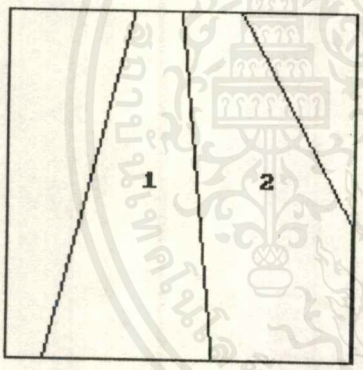
รูปที่ 4.11 แสดง OR กันแนวนอนกับแนวตั้ง



รูปที่ 4.12 แสดงกำจัดเส้นเลนซ์และขอบถนน



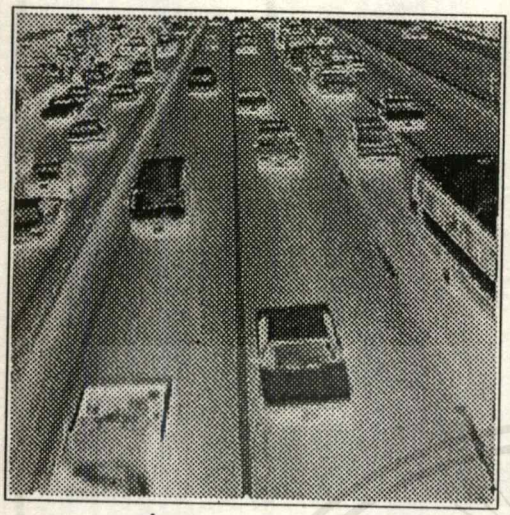
รูปที่ 4.13 แสดงลักษณะรถยนต์ที่หาได้



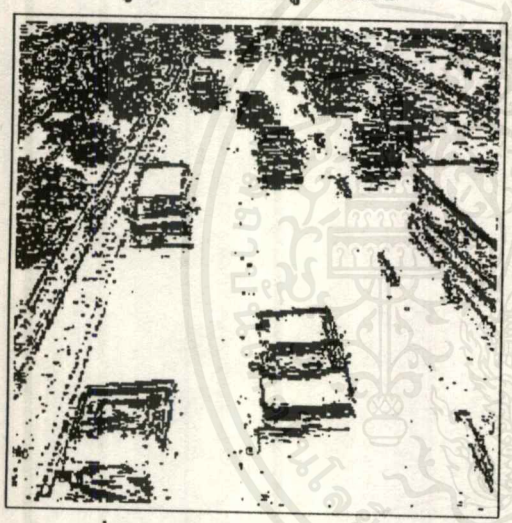

```
len 1 car is avg speed  8

len 2 car is avg speed  2

process picture until file  29
process picture total  29  file

Please any key to continues.
```

รูปที่ 4.14 แสดงผลลัพธ์ของการประมวลผลภาพทั้งหมด

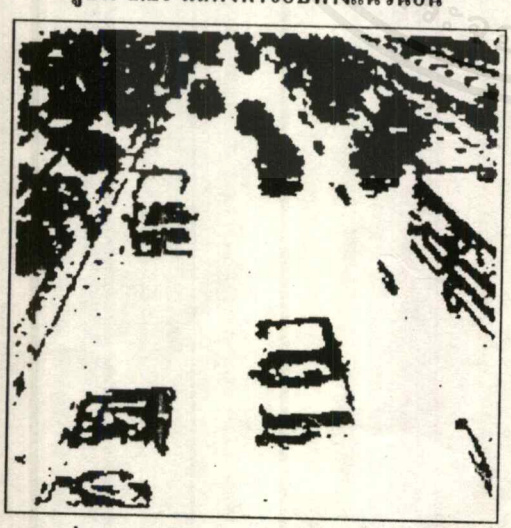
ชุดภาพที่ 2



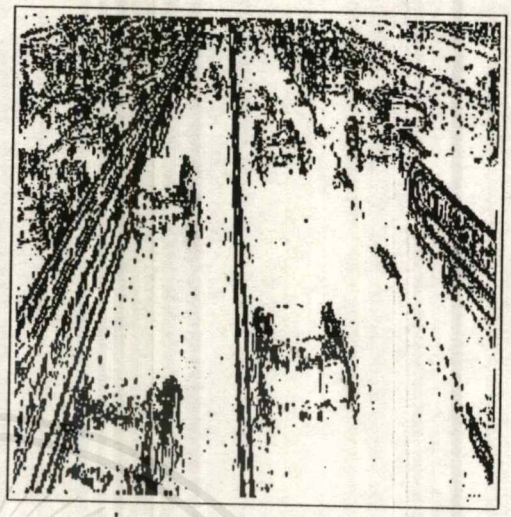
รูปที่ 4.15 แสดงรูปต้นแบบ



รูปที่ 4.16 แสดงหาขอบทางแนวนอน

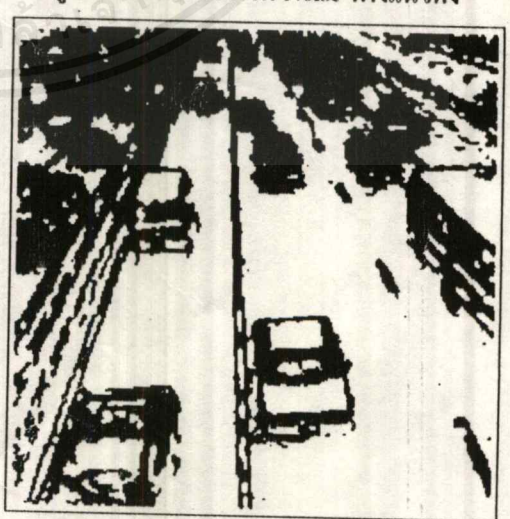


รูปที่ 4.17 แสดงกำจัด Noise ทางแนวนอน

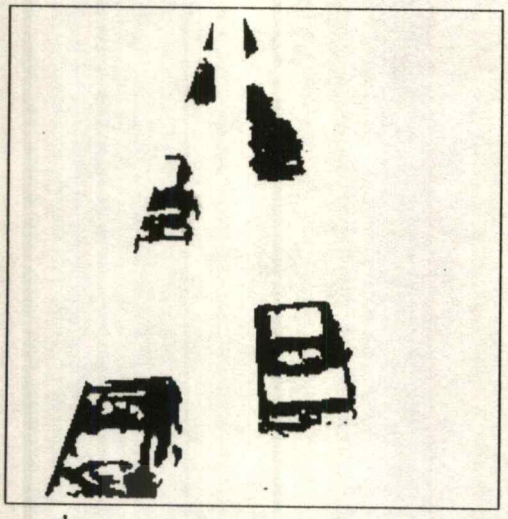


รูปที่ 4.18 แสดงหาขอบทางแนวตั้ง

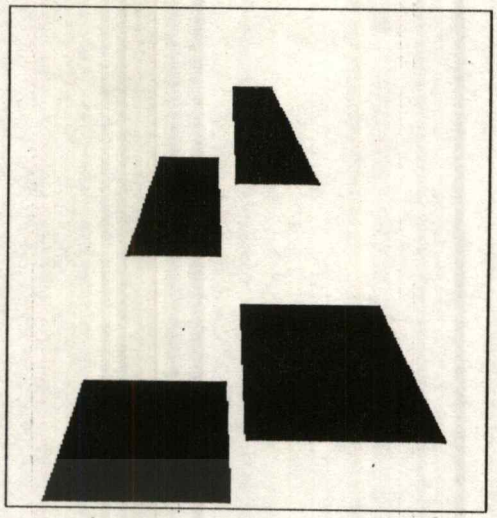
รูปที่ 4.19 แสดงกำจัด Noise ทางแนวตั้ง



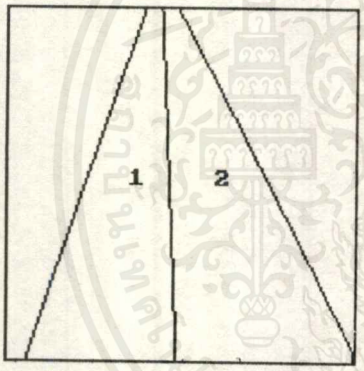
รูปที่ 4.20 แสดง OR กันแนวนอนกับแนวตั้ง



รูปที่ 4.21 แสดงกำจัดเส้นเลนซ์และขอบถนน



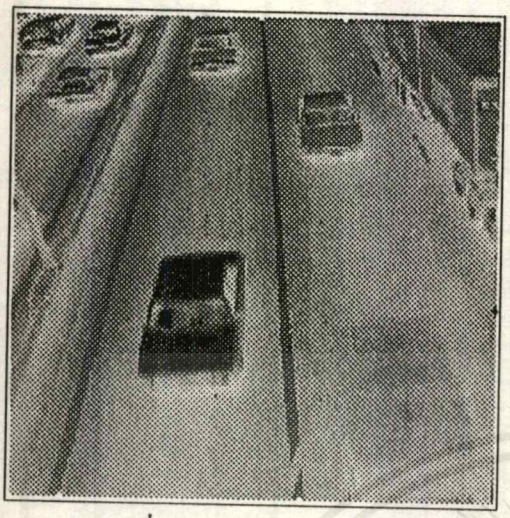
รูปที่ 4.22 แสดงลักษณะรถยนต์ที่หาได้



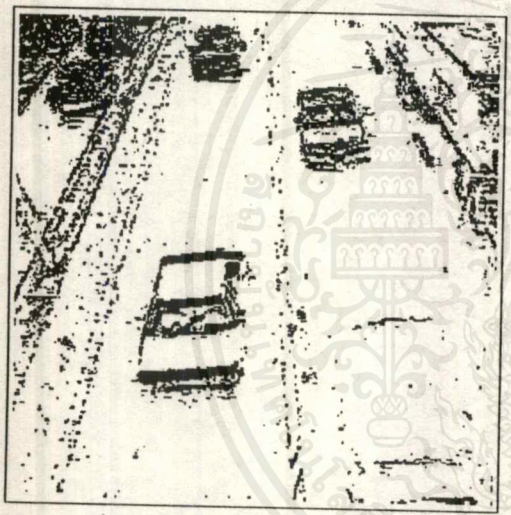

```
len 1 car is avg speed  2

len 2 car is avg speed  5

process picture until file  20
process picture total  20  file

Please any key to continues.
```

รูปที่ 4.23 แสดงผลลัพธ์ของการประมวลผลภาพทั้งหมด

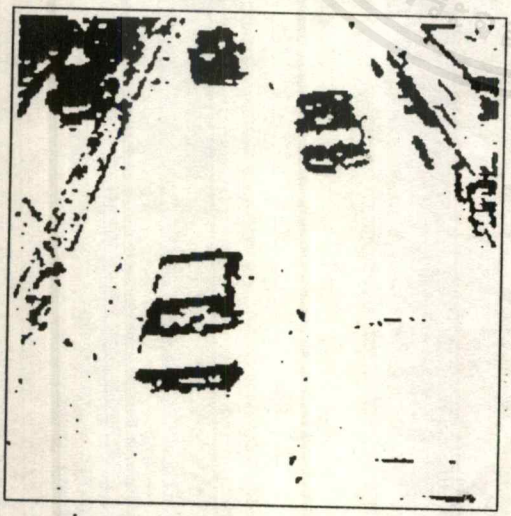
ชุดภาพที่ 3



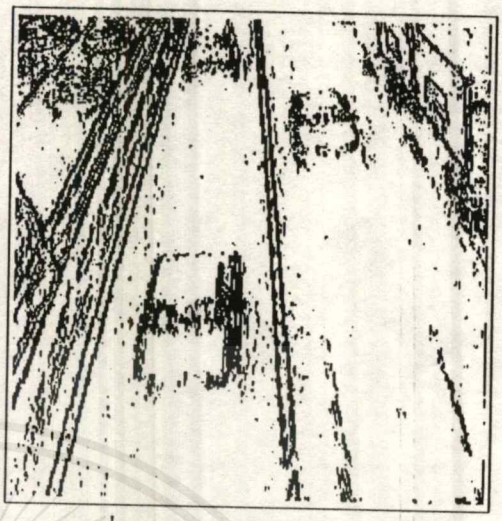
รูปที่ 4.24 แสดงรูปต้นแบบ



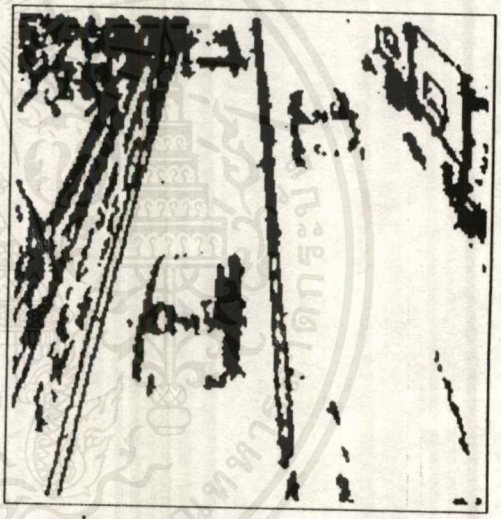
รูปที่ 4.25 แสดงหาขอบทางแนวนอน



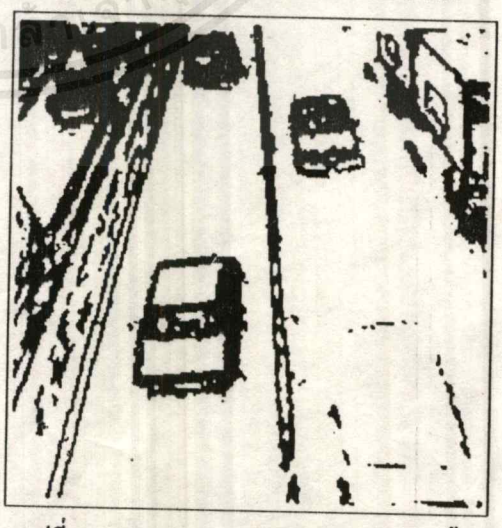
รูปที่ 4.26 แสดงกำจัด Noise ทางแนวนอน



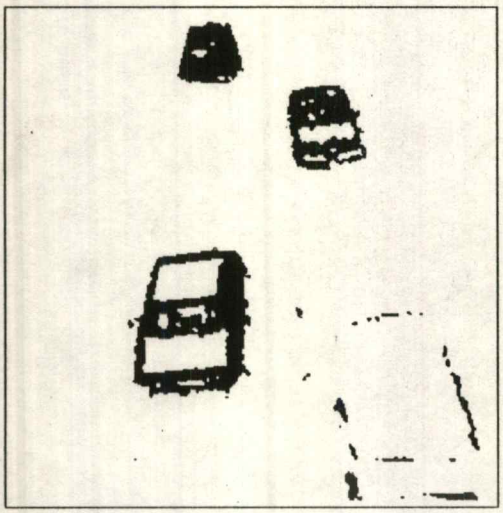
รูปที่ 4.27 แสดงหาขอบทางแนวตั้ง



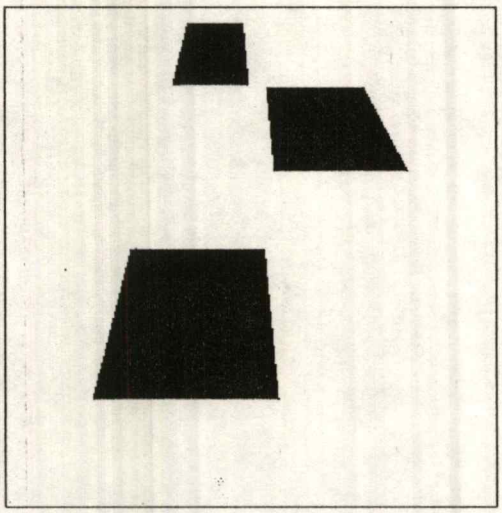
รูปที่ 4.28 แสดงกำจัด Noise ทางแนวตั้ง



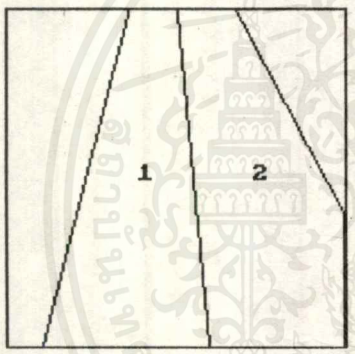
รูปที่ 4.29 แสดง OR กันแนวนอนกับแนวตั้ง



รูปที่ 4.30 แสดงกำจัดเส้นเลนซ์และขอบถนน



รูปที่ 4.31 แสดงลักษณะรถยนต์ที่หาได้




```
len 1 car is avg speed  10

len 2 car is avg speed  3

process picture until file  16
process picture total  16  file

Please any key to continues.
```

รูปที่ 4.32 แสดงผลลัพธ์ของการประมวลผลภาพทั้งหมด

4.3 ผลการทดลอง

ผลการทดลองผลงานวิจัยการประมวลผลภาพสำหรับรายงานสภาพการจราจรบนทางด่วน (Image Processing for Highway Traffic Report) สามารถทำงานได้ผลดีเป็นที่น่าพอใจพสมควร คือสามารถที่จะรายงานสภาพการจราจรได้ถูกต้องภายใต้ข้อกำหนดของการทำงาน

ระยะเวลาที่ใช้ในการประมวลผลภาพจะมีค่าไม่เท่ากันในแต่ละชุดภาพ เพราะขึ้นอยู่กับความเร็วของรถยนต์ที่วิ่งอยู่ภายในขอบเขตของภาพที่เราทำการวิเคราะห์อยู่ คือถ้ารถยนต์วิ่งช้าโปรแกรมการประมวลผลภาพก็จะต้องอ่านจำนวนเฟรมภาพที่ต่อเนื่องไปจนกระทั่งรถยนต์เกินขอบเขตของภาพที่เราสนใจซึ่งก็จะใช้เวลาการประมวลผลมากขึ้นด้วย

การหาความแตกต่างระหว่างรถยนต์กับถนน โดยใช้หลักการการขอบภาพโดยวิธีของ SOBEL โดยการหาขอบทางแนวตั้ง และแนวนอนของภาพ ซึ่งทั้งสองจะต้องมีการกำจัด Noiseและตัดค่า Thresdhold เรียบร้อยแล้ว ซึ่งจะเป็นภาพไบนารี (Binary Image) ต่อจากนั้นนำภาพที่ได้จากทั้งสองมาทำการออ (OR) กันจะได้ขอบของภาพซึ่งคือรถยนต์บนถนนอย่างมีประสิทธิภาพ ซึ่งก็จะมีข้อจำกัดอยู่ที่ว่าถ้าหากระดับความแตกต่างของระดับสีเทา (Gray-scale) ของรถยนต์มีค่าที่ใกล้เคียงกันมากๆ เช่น รถยนต์ที่วิ่งบนทางด่วนช่วงเวลากลางคืนก็อาจทำให้การประมวลผลผิดพลาดได้ คืออาจไม่สามารถหาความแตกต่างระหว่างรถยนต์กับถนนได้ ก็จะทำให้ไม่สามารถประมวลผลภาพ

บทที่ 5

บทวิจารณ์และสรุป

5.1 สรุป

สรุปผลงานวิจัยการประมวลผลภาพด้วยคอมพิวเตอร์ สำหรับรายงานสภาพการจราจรบนทางด่วน (Image Processing for Highway Traffic Report) ที่ได้ทดสอบการทำงานสามารถที่จะทำการวิเคราะห์ได้ผลดีเป็นที่น่าพอใจพอสมควร คือสามารถที่จะรายงานสภาพการจราจรได้ถูกต้องภายใต้ข้อกำหนดของงานวิจัย

การหาความแตกต่างระหว่างรถยนต์กับถนน ซึ่งเป็นจุดที่สำคัญมากของงานวิจัยนี้ โดยงานวิจัยนี้ใช้หลักการหาขอบภาพโดยวิธีของการ SOBEL ซึ่งได้ทำการทดสอบด้วยวิธีต่างๆ มากมาย แต่ท้ายที่สุดที่ทดสอบว่าได้ผลดีที่สุดคือ การหาขอบภาพที่เน้นขอบภาพทางแนวตั้ง และนำผลที่ได้มาทำการกำจัดส่วนที่ไม่ต้องการ (Noise) โดยใช้การตัดค่า Thresdhold ที่เหมาะสม ต่อจากนั้นการหาขอบภาพที่เน้นขอบภาพทางแนวนอน โดยใช้ภาพต้นแบบเดียวกัน และนำผลที่ได้มาทำการกำจัดส่วนที่ไม่ต้องการ (Noise) โดยใช้การตัดค่า Thresdhold ที่เหมาะสม เช่นเดียวกัน ซึ่งภาพที่ได้ทั้งสองเป็นข้อมูลแบบไบนารี (Binary Image) ต่อจากนั้นนำภาพที่ได้ของทั้งสองมาทำการออ (OR) กันจะได้ขอบของภาพซึ่งคือรถยนต์บนถนนอย่างมีประสิทธิภาพ

ข้อจำกัดของวิธีการหาขอบคือ ถ้าหากระดับความแตกต่างของระดับสีเทา (Gray-scale) ของรถยนต์มีค่าที่ใกล้เคียงกันมากๆ เช่น รถยนต์ที่วิ่งบนทางด่วนช่วงเวลากลางคืนก็อาจทำให้การหาขอบไม่มีประสิทธิภาพได้ ซึ่งก็อาจทำให้การประมวลผลผิดพลาดได้ คือไม่สามารถหาความแตกต่างระหว่างรถยนต์กับถนนได้นั่นเอง ซึ่งก็ทำให้การวิเคราะห์ผลในลักษณะของงานวิจัยนี้ไม่สามารถทำได้เช่นเดียวกัน

ในทางปฏิบัติการที่จะนำระบบการรายงานสภาพการจราจรบนทางด่วนโดยวิธีการด้านการประมวลผลภาพไปใช้งานได้จริงยังคงต้องมีการปรับปรุงแนวคิดการทำงานอีกโดยเฉพาะช่วงเวลากลางคืนที่ที่มีแสงไม่เพียงพอ จะทำให้ไม่สามารถหาความแตกต่างระหว่างรถยนต์กับถนนได้ ท้ายนี้คิดว่าในอนาคตคงมีผู้ที่สนใจงานทางด้านการประมวลผลภาพนี้ สามารถคิดค้นวิธีการที่ดีและปรับปรุงสามารถนำไปใช้งานในการรายงานสภาพการจราจรบนทางด่วนได้จริงๆ ก็จะเป็นประโยชน์ต่อประเทศชาติอย่างมหาศาลต่อไป

5.2 แนวทางในการพัฒนาระบบในอนาคต

สำหรับผู้ที่สนใจในงานด้านการประมวลผลภาพสำหรับการรายงานสภาพการจราจรบนทางด่วน สามารถที่จะนำระบบที่ได้ทำการวิจัยนี้ไปเป็นแนวทางและปรับปรุงประสิทธิภาพการทำงานให้ดียิ่งขึ้นได้ โดยมีแนวทางที่น่าสนใจคือ

1. ปรับปรุงวิธีการหรือหาแนวความคิดที่ดีกว่ามาใช้ในการประมวลผลภาพของรถยนต์บนถนน โดยเฉพาะในช่วงเวลากลางคืน

2. ปรับปรุงวิธีการให้สามารถวิเคราะห์หาเลนได้โดยอัตโนมัติ

3. ปรับปรุงส่วนการใช้งานของผู้ใช้ให้ดียิ่งขึ้น

4. ควรมีการ์ดทำหน้าที่จัดเก็บข้อมูลภาพที่ได้จากกล้องวีดีโอโดยอัตโนมัติ

5. การพัฒนาให้ระบบสามารถแจ้งข่าวสารสภาพการจราจรบนทางด่วน เช่น ข้อความ ภาพ และเสียง ผ่านเครือข่ายต่างๆ ได้

5.3 ข้อจำกัดของระบบ

1. โปรแกรมทำงานบน DOS ได้อย่างเดียว

2. ประมวลผลได้เฉพาะไฟล์ BMP เท่านั้น

3. ข้อมูลภาพที่ใช้เป็น GRAYSCAL 256 ระดับ

4. ขนาดของข้อมูลภาพที่ใช้ได้คือ 300 PIXEL * 300 PIXEL

5. ไฟล์ข้อมูลภาพที่จะวิเคราะห์ต้องอยู่ที่ C:\PROJECT\IMAGE> เสมอ

6. ประมวลผลได้มากที่สุด 4 เลนซ์

7. ประมวลผลเฉพาะเวลาที่มีค่าความแตกต่างของระดับสีเทา (Gray scal) ของรถยนต์กับถนนพอสมควร

8. ต้องให้ผู้ใช้กำหนดขอบเขตของถนนและเลนซ์มาให้

9. การเปลี่ยนแปลงค่าของการรายงานสภาพการจราจรบนทางด่วนมีช่วงห่างของระยะเวลาพอสมควร เช่น 1 นาที เปลี่ยนแปลงค่าของการรายงานสภาพการจราจรบนทางด่วนครั้งหนึ่ง

ภาคผนวก ก

ตารางแสดงฟังก์ชันคีย์ที่ใช้งานของโปรแกรม

ฟังก์ชันคีย์ (FUNCTION KEYS)

หน้าที่ของคีย์ที่ใช้งานในส่วนของการกำหนดขอบเขต (หัวข้อที่ 1 [SETUP]) มีดังนี้

คีย์	หน้าที่
คีย์ลูกศรชี้ขวา	เลื่อนเคอร์เซอร์ไปทางขวาหนึ่งตำแหน่ง
คีย์ลูกศรชี้ซ้าย	เลื่อนเคอร์เซอร์ไปทางซ้ายหนึ่งตำแหน่ง
คีย์ลูกศรชี้ล่าง	เลื่อนเคอร์เซอร์ไปข้างล่างหนึ่งตำแหน่ง
คีย์ลูกศรชี้บน	เลื่อนเคอร์เซอร์ไปข้างบนหนึ่งตำแหน่ง
enter	กำหนดตำแหน่งที่ต้องการ
d	เลื่อนเคอร์เซอร์ไปทางขวาสิบตำแหน่ง
e	เลื่อนเคอร์เซอร์ไปข้างบนสิบตำแหน่ง
s	เลื่อนเคอร์เซอร์ไปทางซ้ายสิบตำแหน่ง
x	เลื่อนเคอร์เซอร์ไปข้างล่างสิบตำแหน่ง
c	ยกเลิกตำแหน่งที่ต้องการ
q	จบการกำหนดขอบเขตที่สนใจ

ภาคผนวก ข

ตารางแสดงข้อความผิดพลาดของโปรแกรม

ข้อผิดพลาด (ERROR MESSAGES)

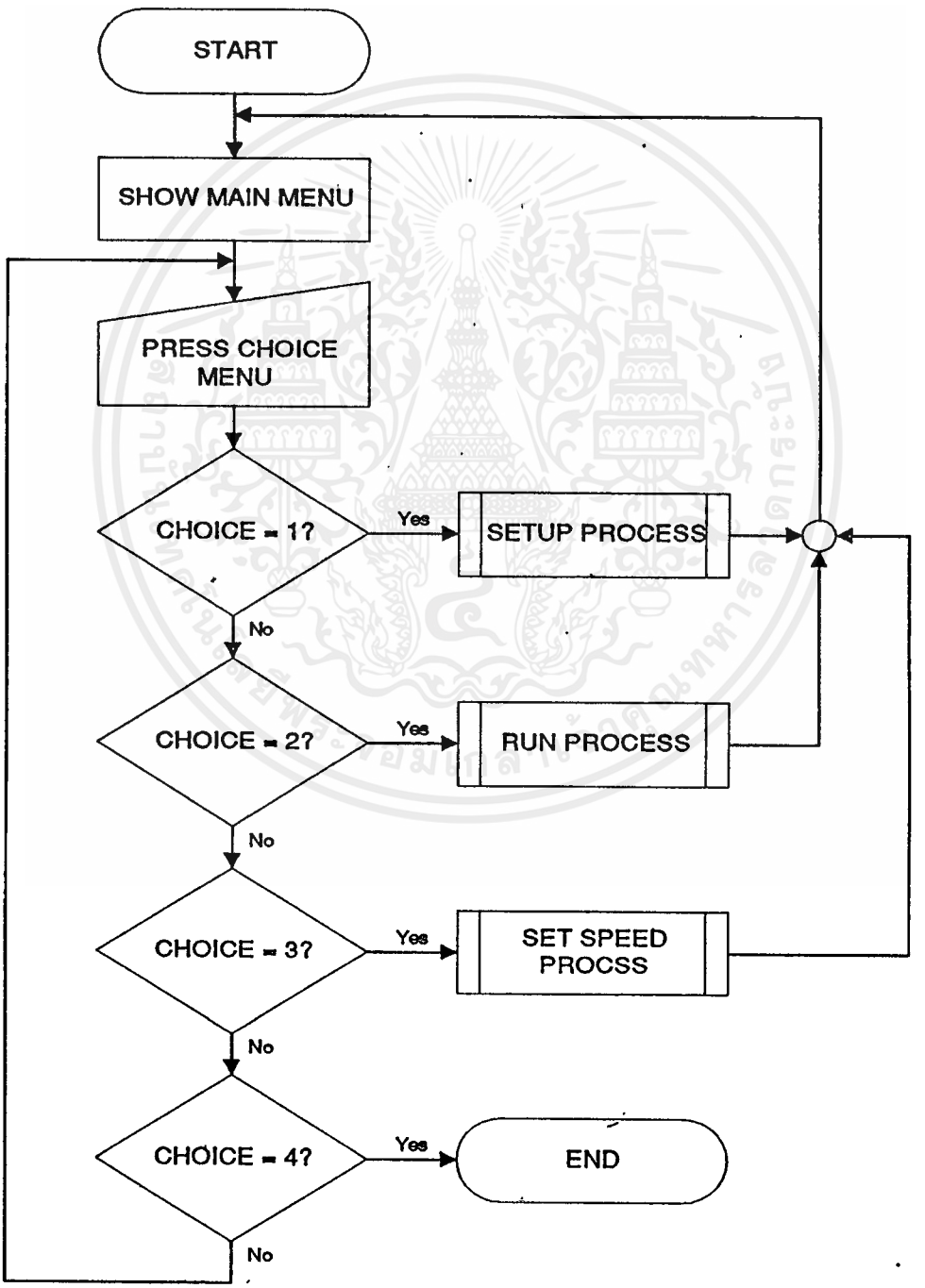
สรุปข้อผิดพลาดที่เกิดขึ้นในส่วนของการกำหนดขอบเขต (หัวข้อที่ 1 [SETUP]) มีดังนี้

ข้อความที่แสดงความผิดพลาด	สาเหตุของความผิดพลาด
This position Duplicate!.	ตำแหน่งที่จะทำการกำหนดซ้ำกับตำแหน่งเดิม
This position Not Mark!.	ทำการลบตำแหน่งโดยที่ตำแหน่งนั้นยังไม่ได้กำหนด
Don't Mark data Overflow!.	ไม่สามารถกำหนดตำแหน่งต่อไปได้อีก
Mark position Not Complete!.	การกำหนดตำแหน่งยังไม่สมบูรณ์
Don't Mark position that edit!.	มีการแก้ไขตำแหน่งโดยไม่มีการกำหนดตำแหน่งใหม่
Don't Initial Mark position!.	ไม่มีการกำหนดตำแหน่งเพื่อใช้ในการประมวลผล

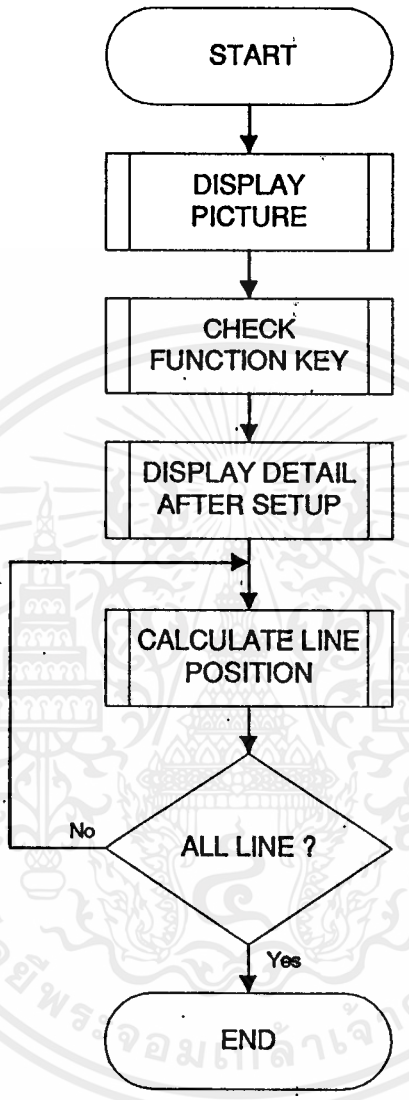
ภาคผนวก ค

แสดงแผนผังการทำงานของโปรแกรม

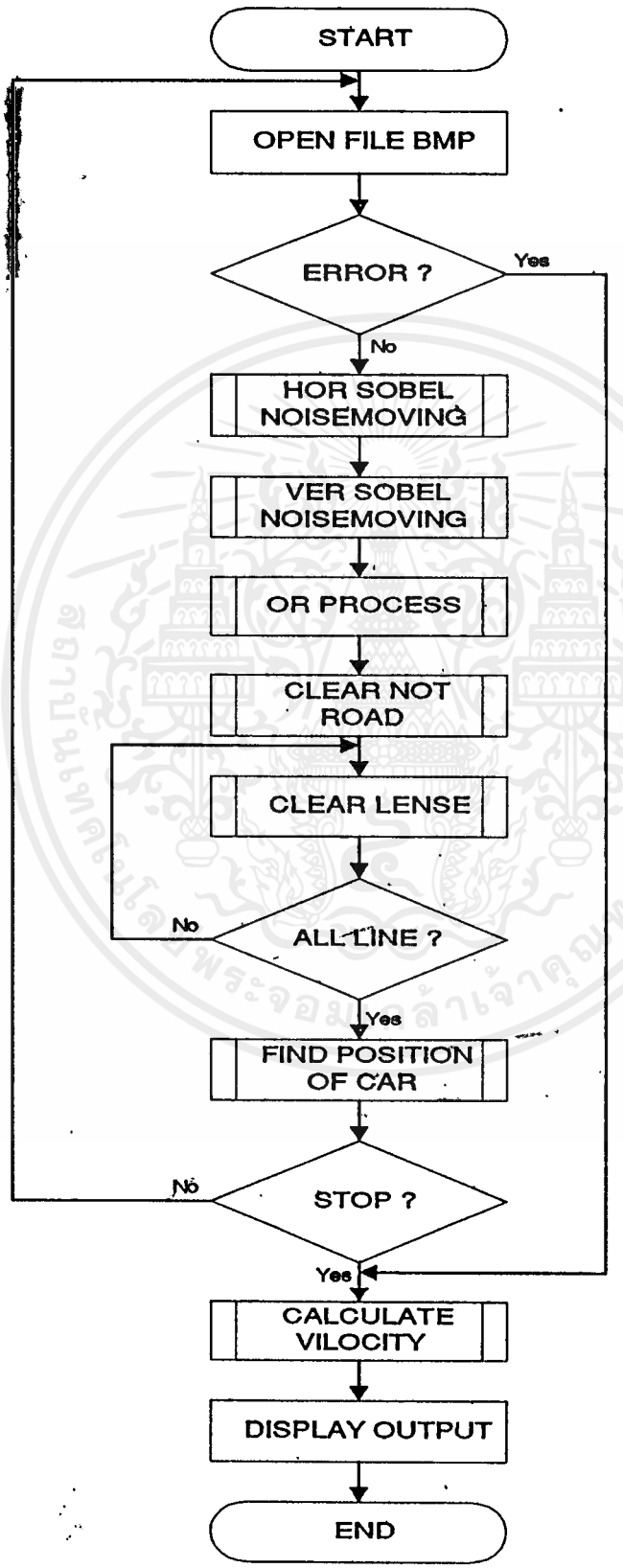
Flowchart Main Process



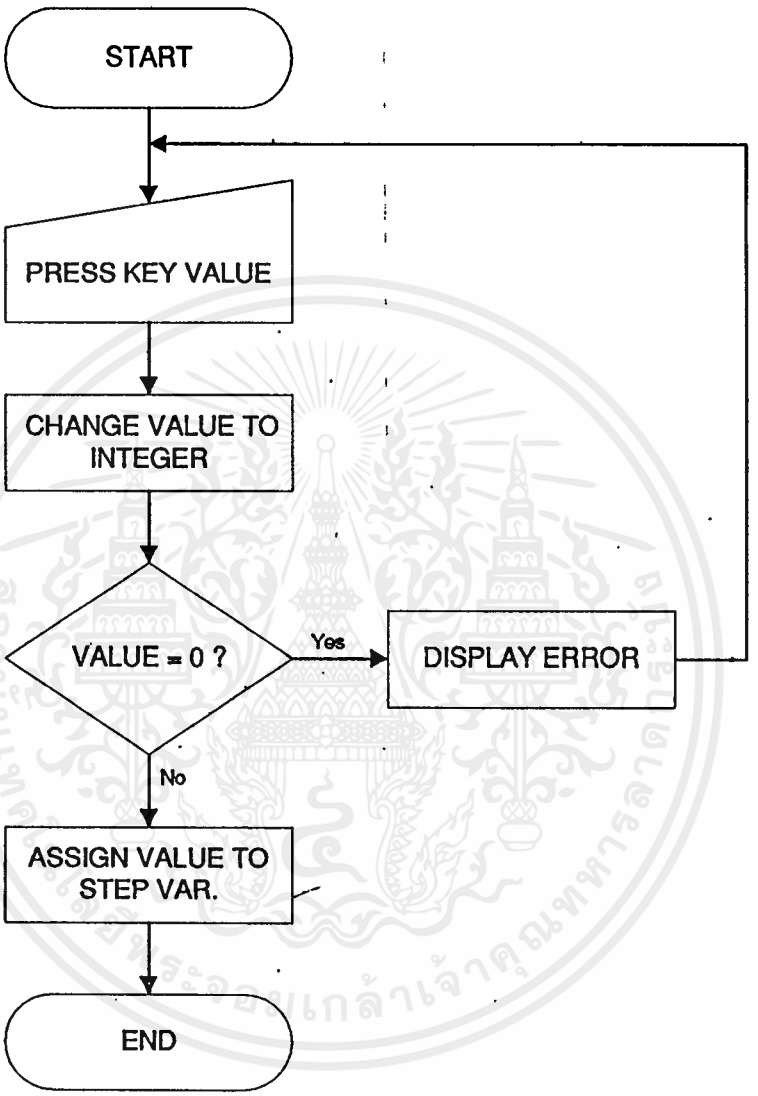
Flowchart Setup Process



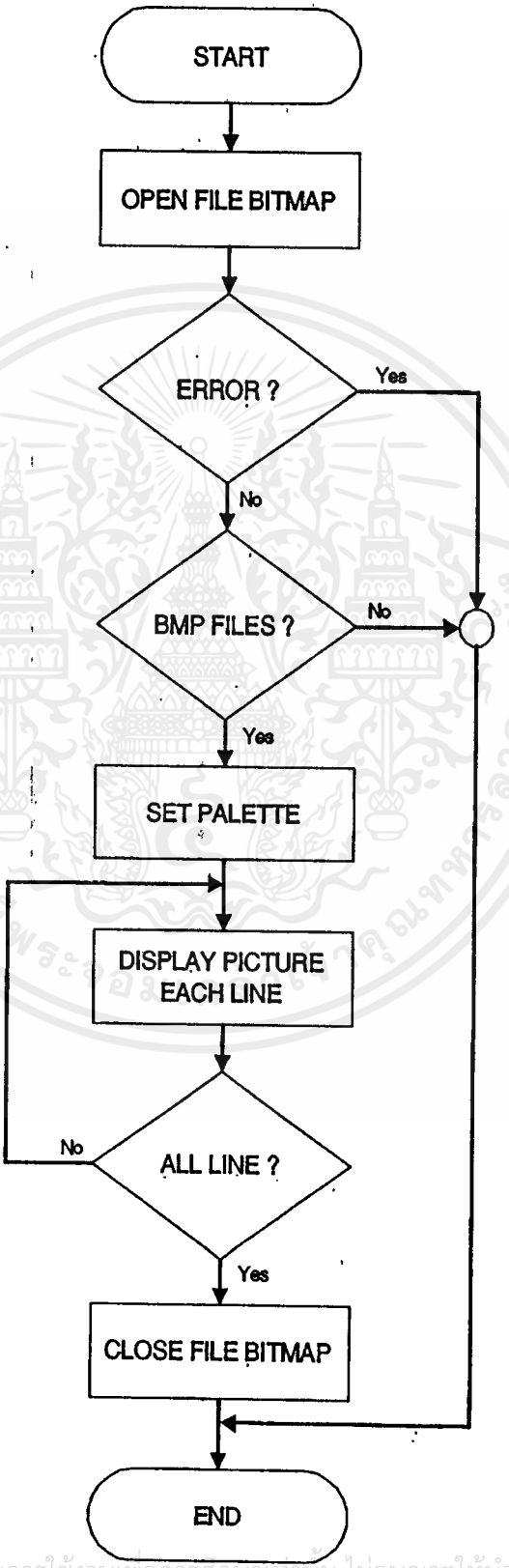
Flowchart Run Process



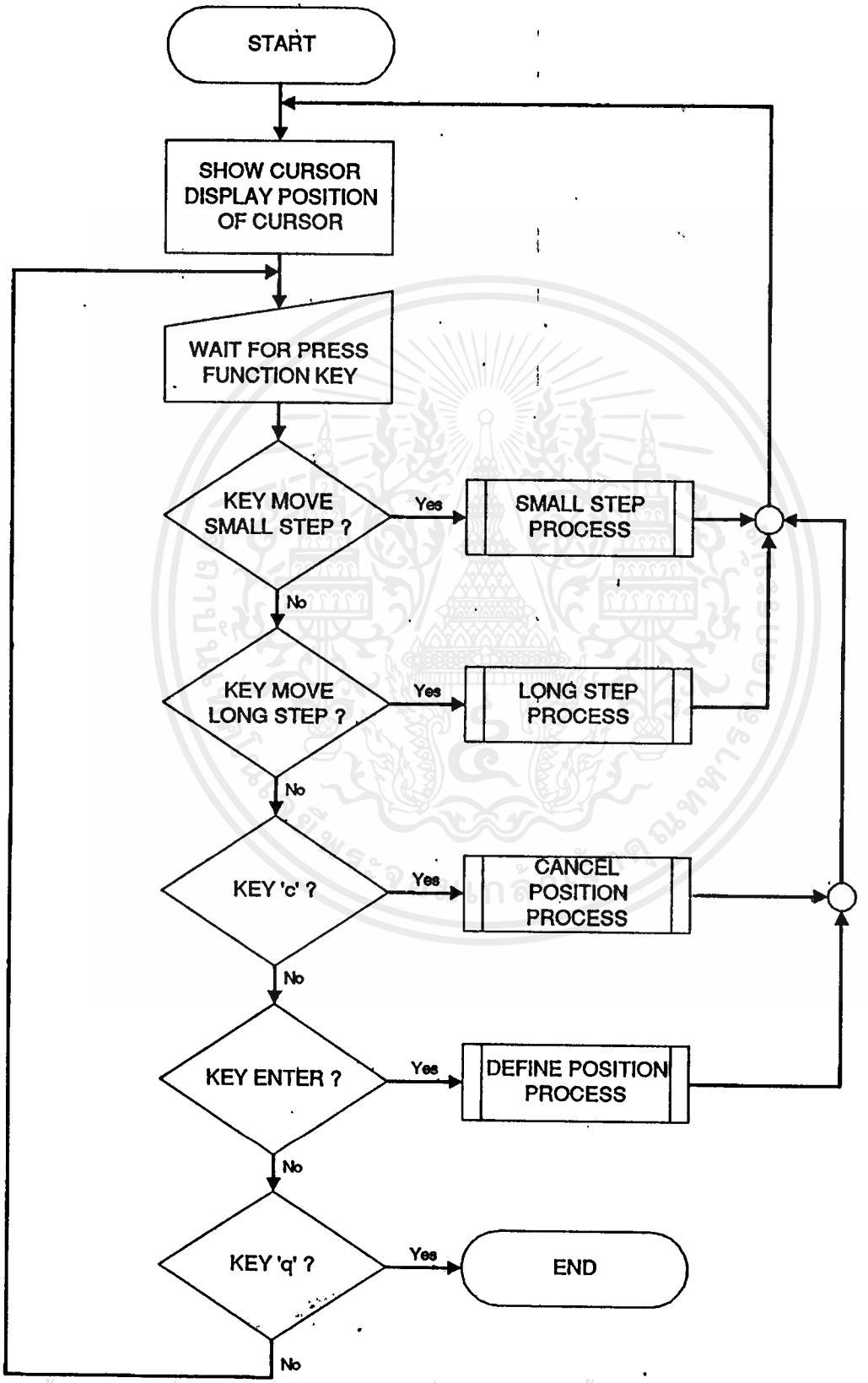
Flowchart Set Speed Process



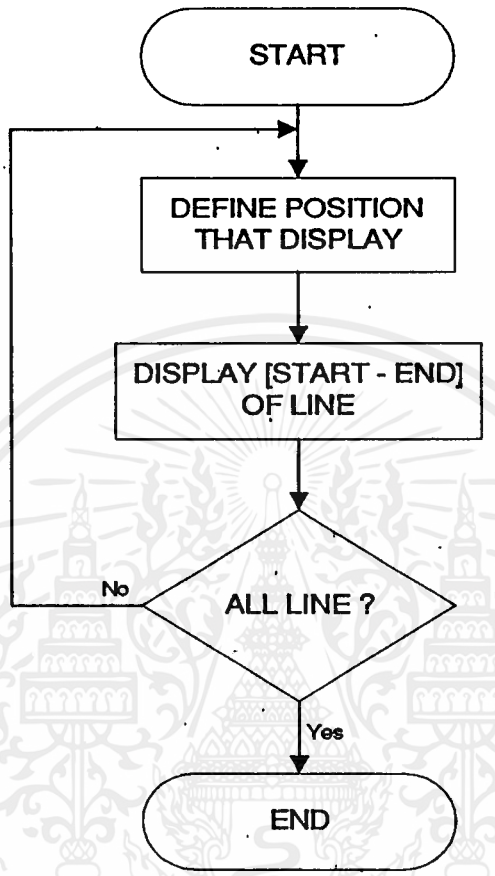
Flowchart Display Picture



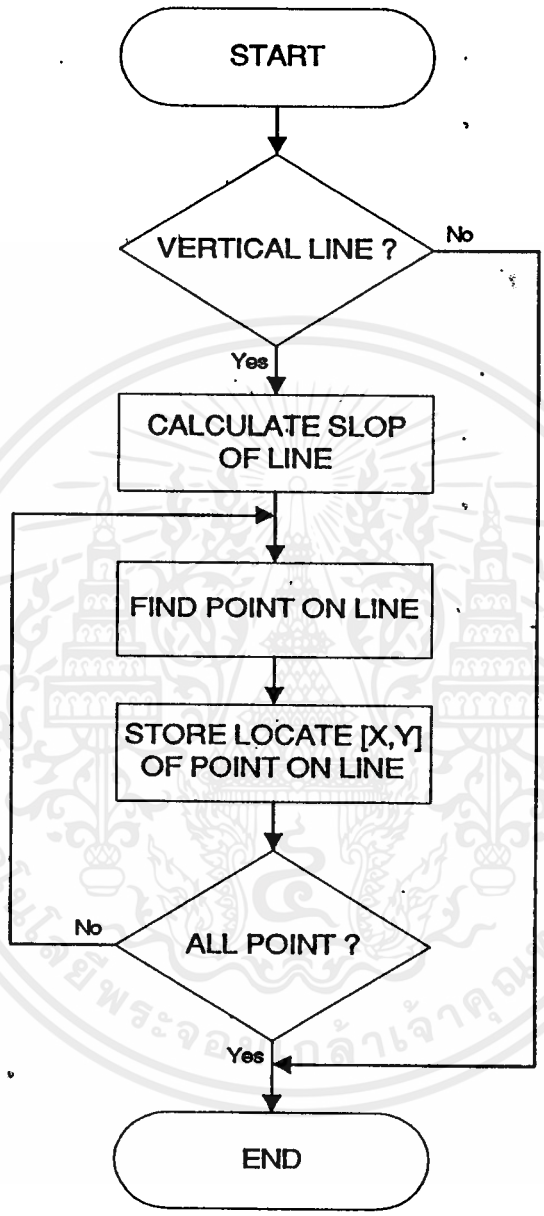
Flowchart Check Function Key



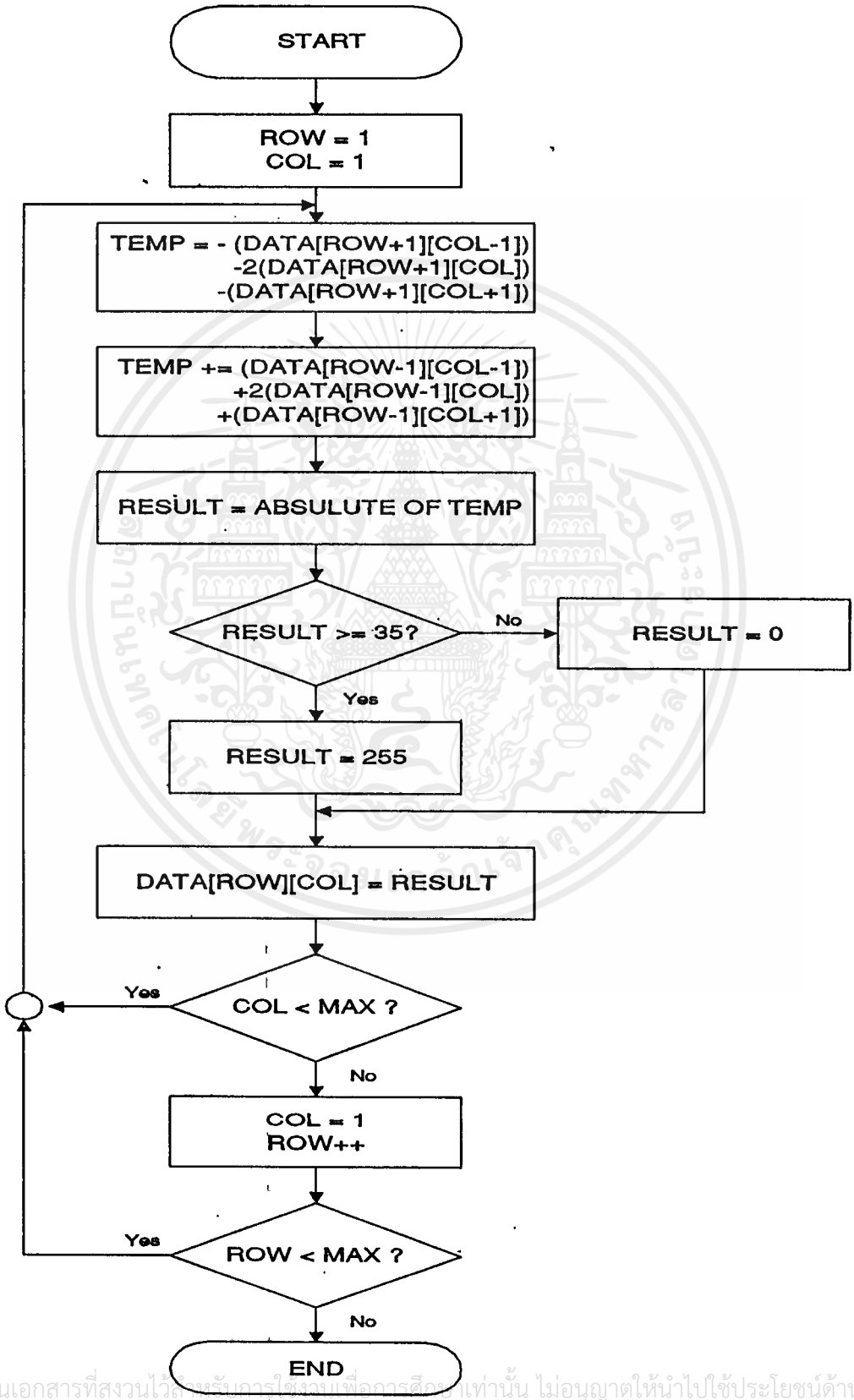
Flowchart Display Detail After Setup



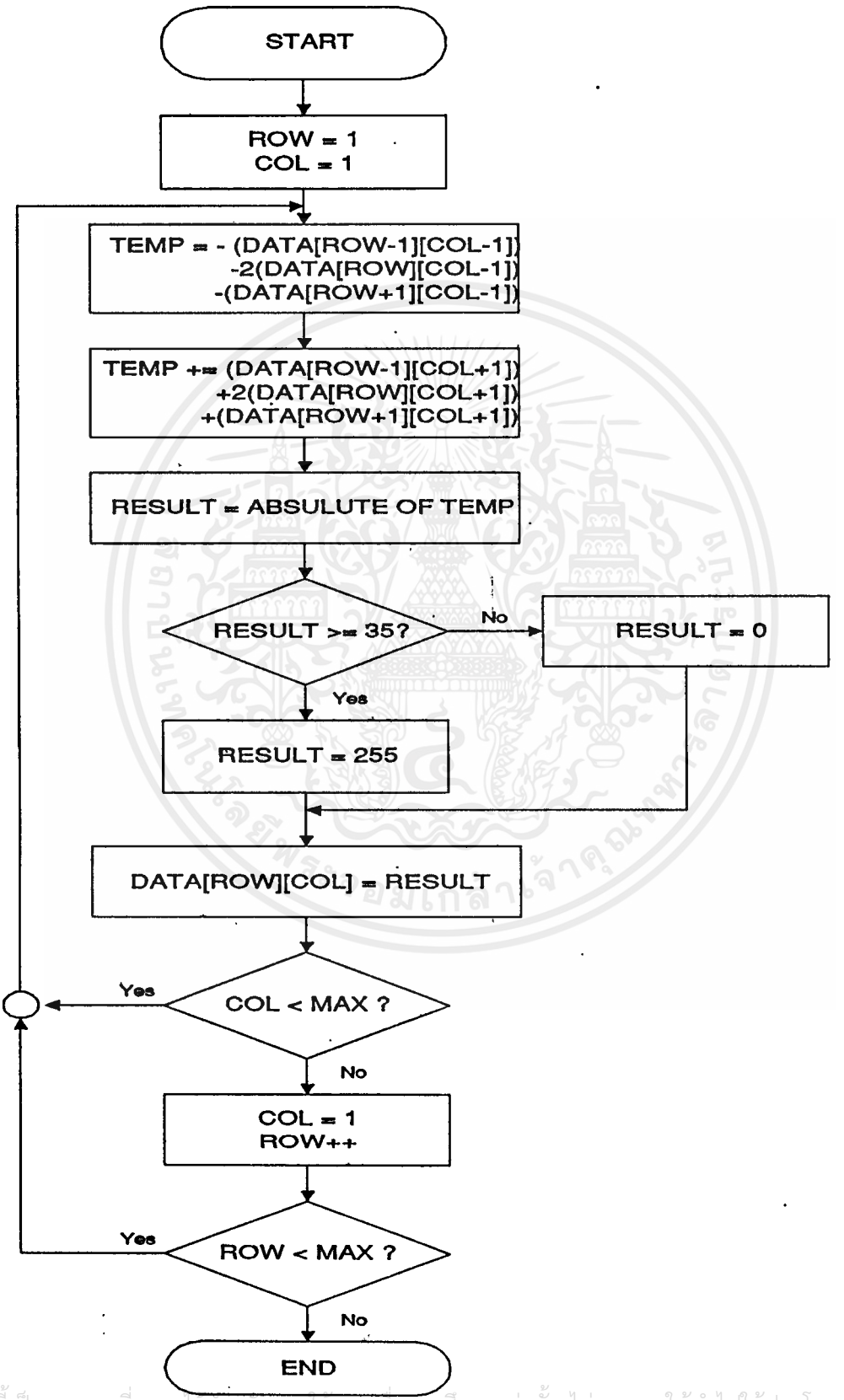
Flowchart Calculate Line Position



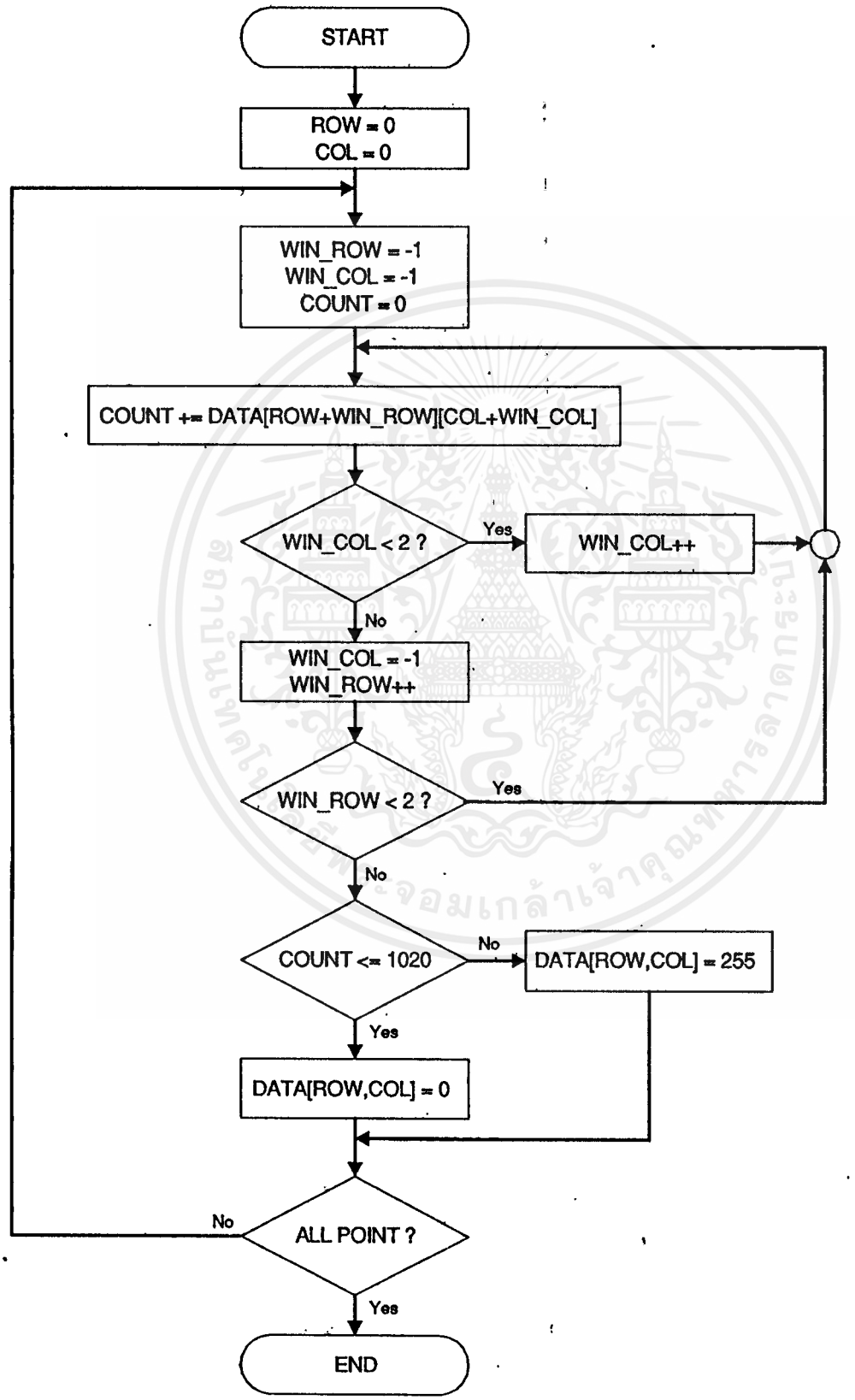
Flowchart Horizontal Sobel



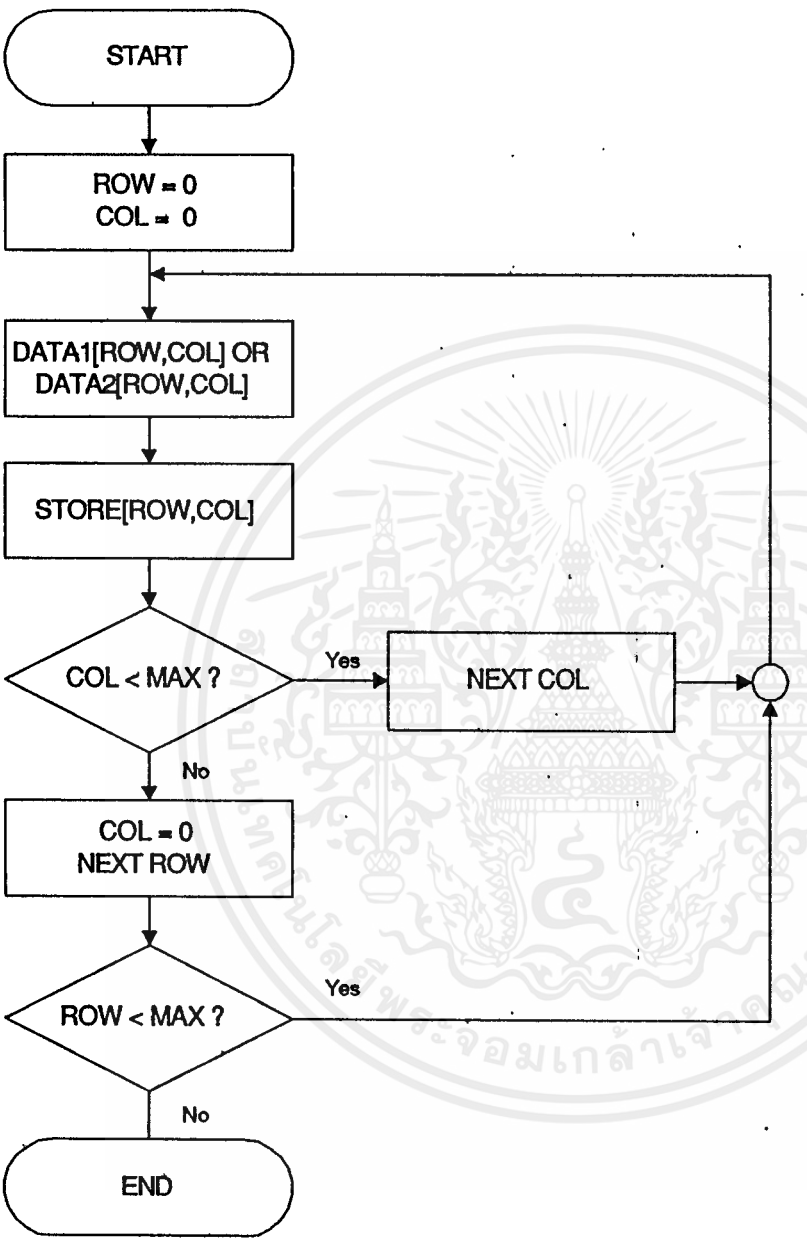
Flowchart Vertical Sobel



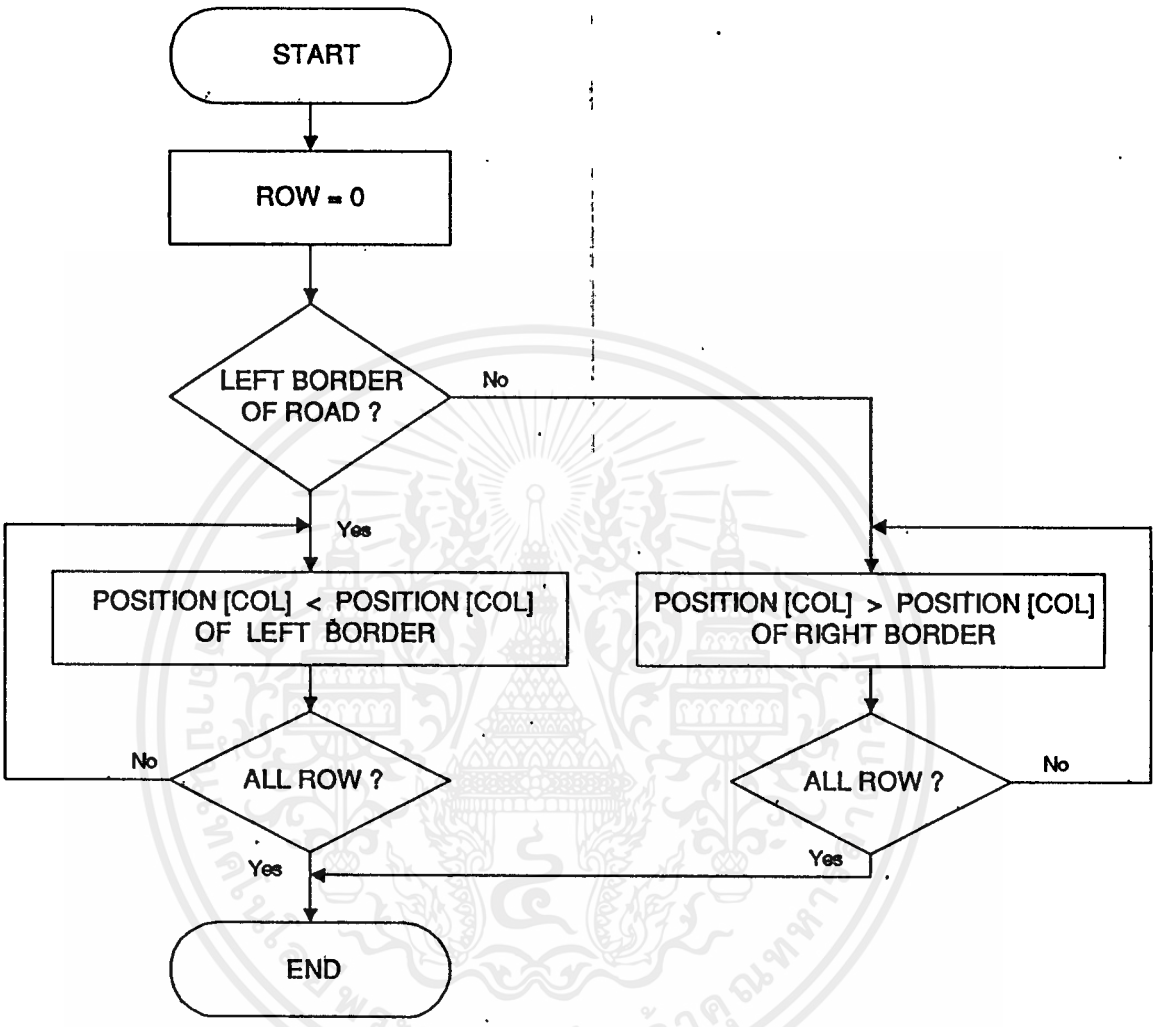
Flowchart Noisemoving



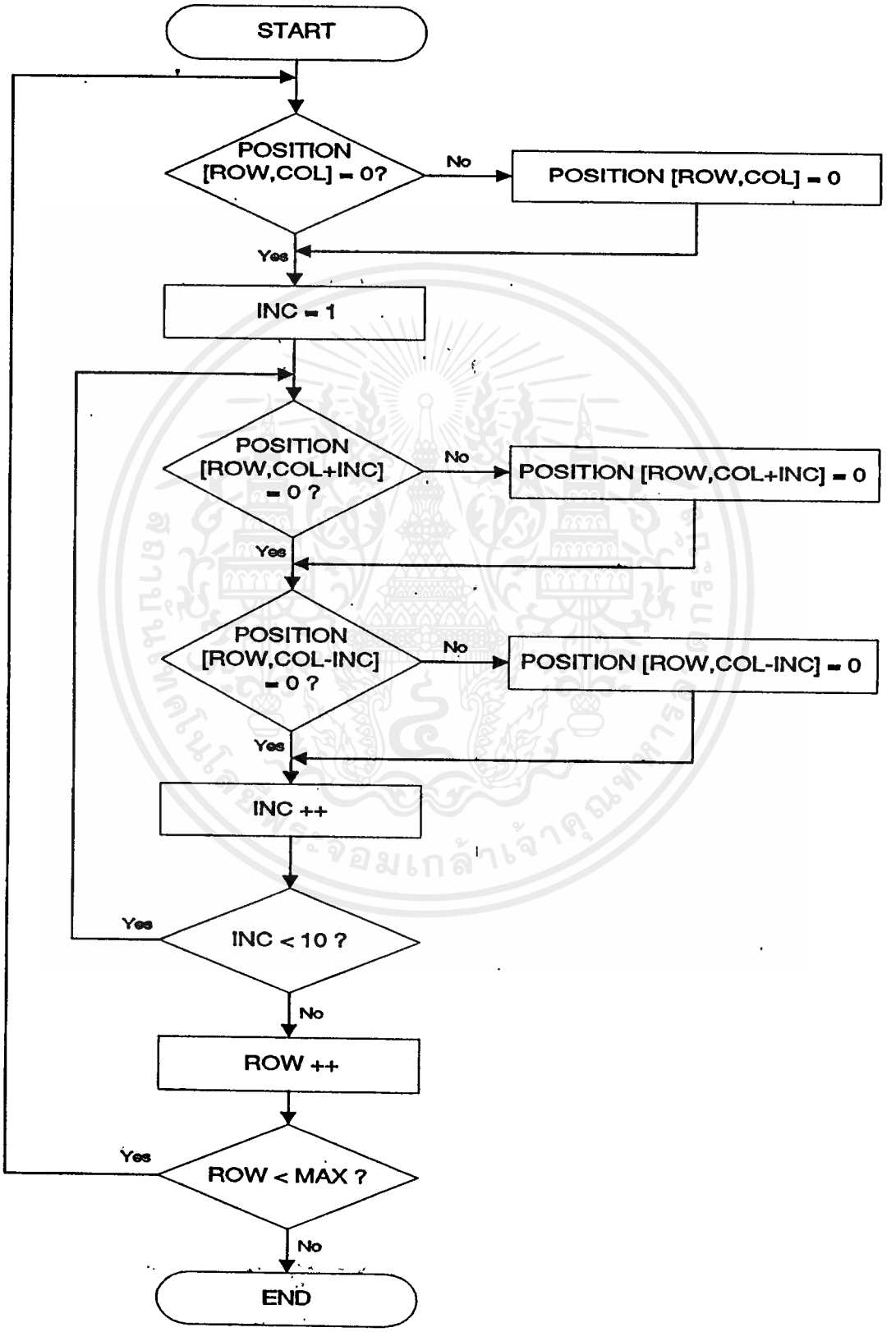
Flowchart Orprocess



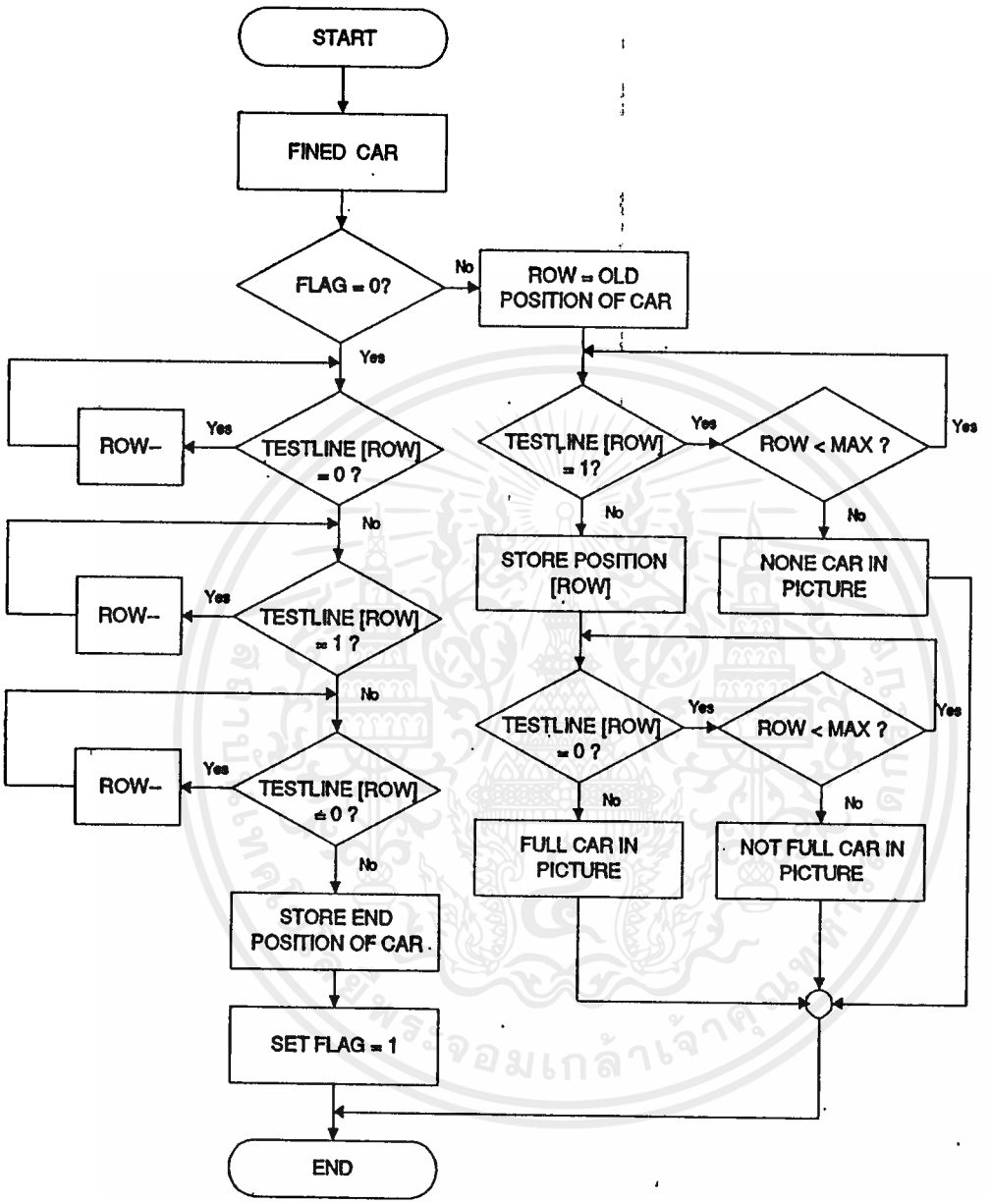
Flowchart Clear Not Road Process



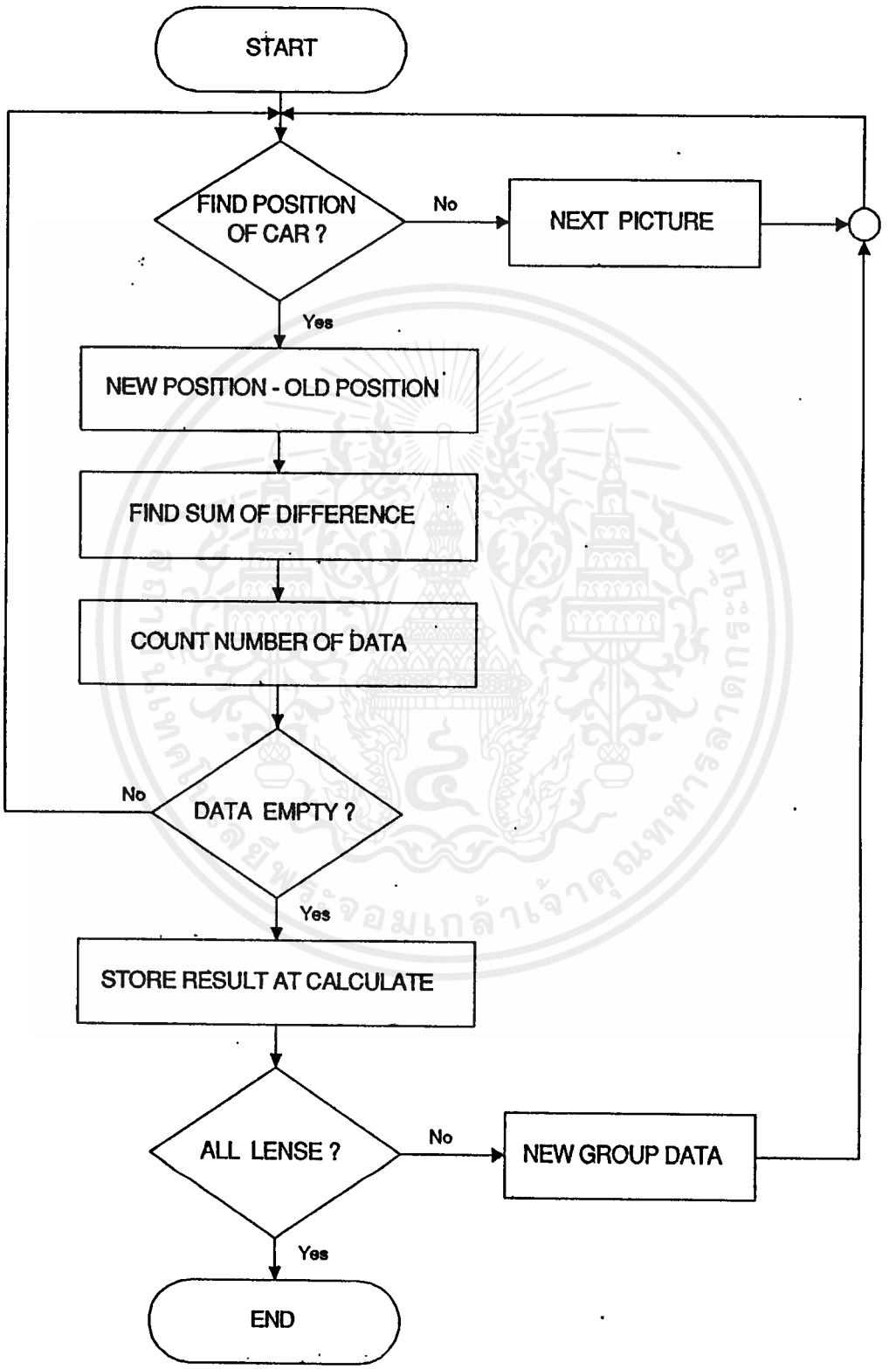
Flowchart Clear Lense



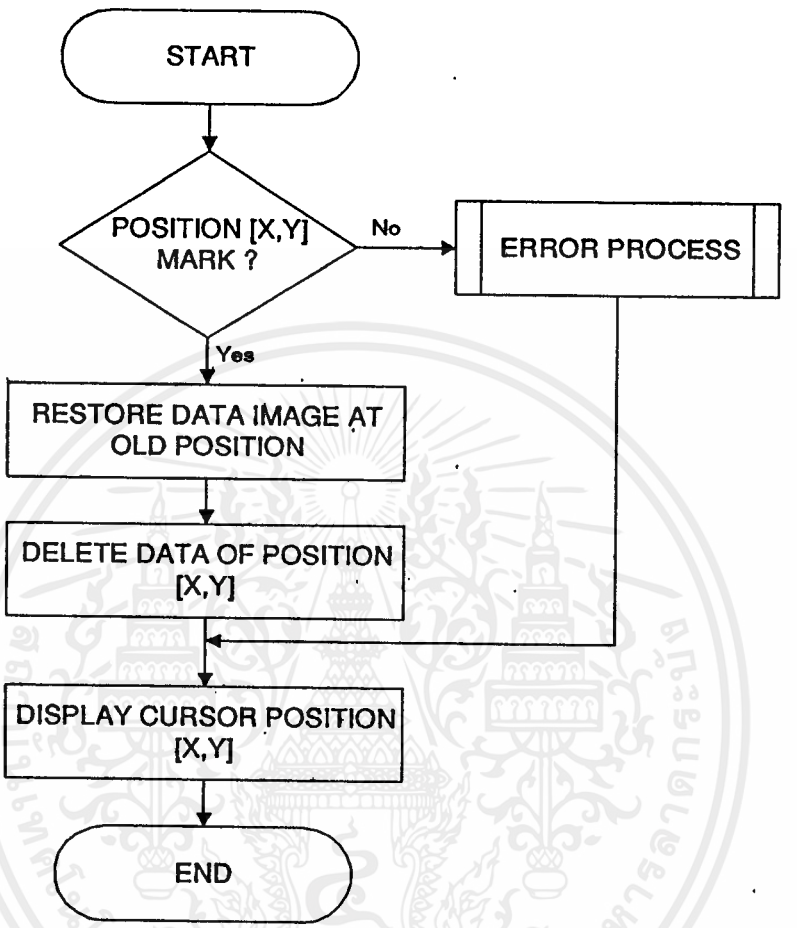
Flowchart Find Car Position Process



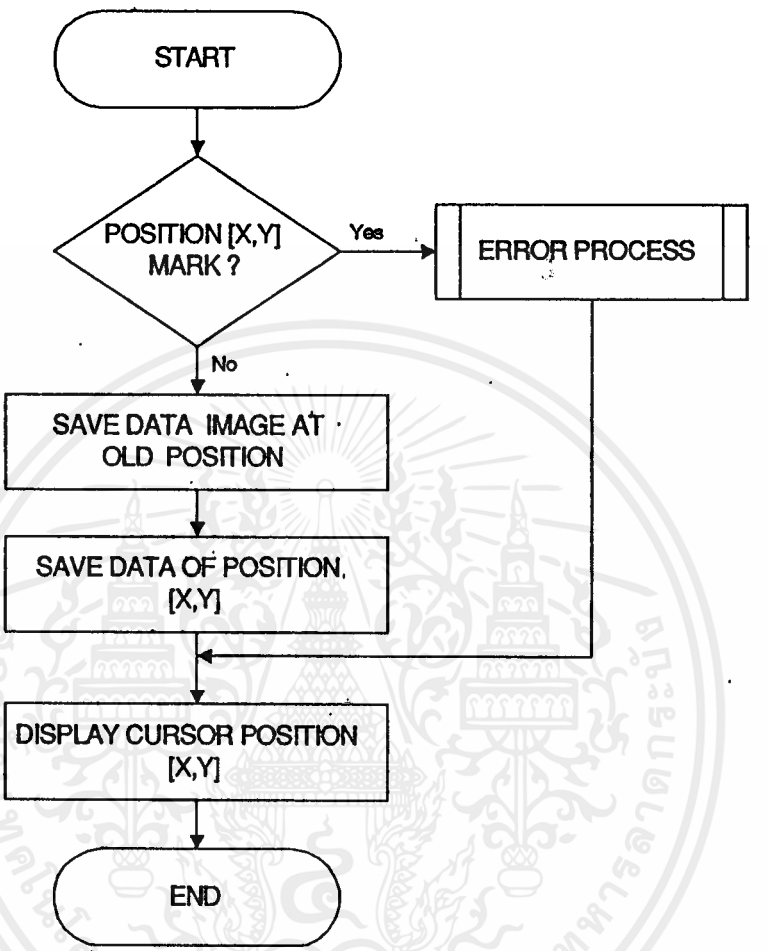
Flowchart Calculate Speed



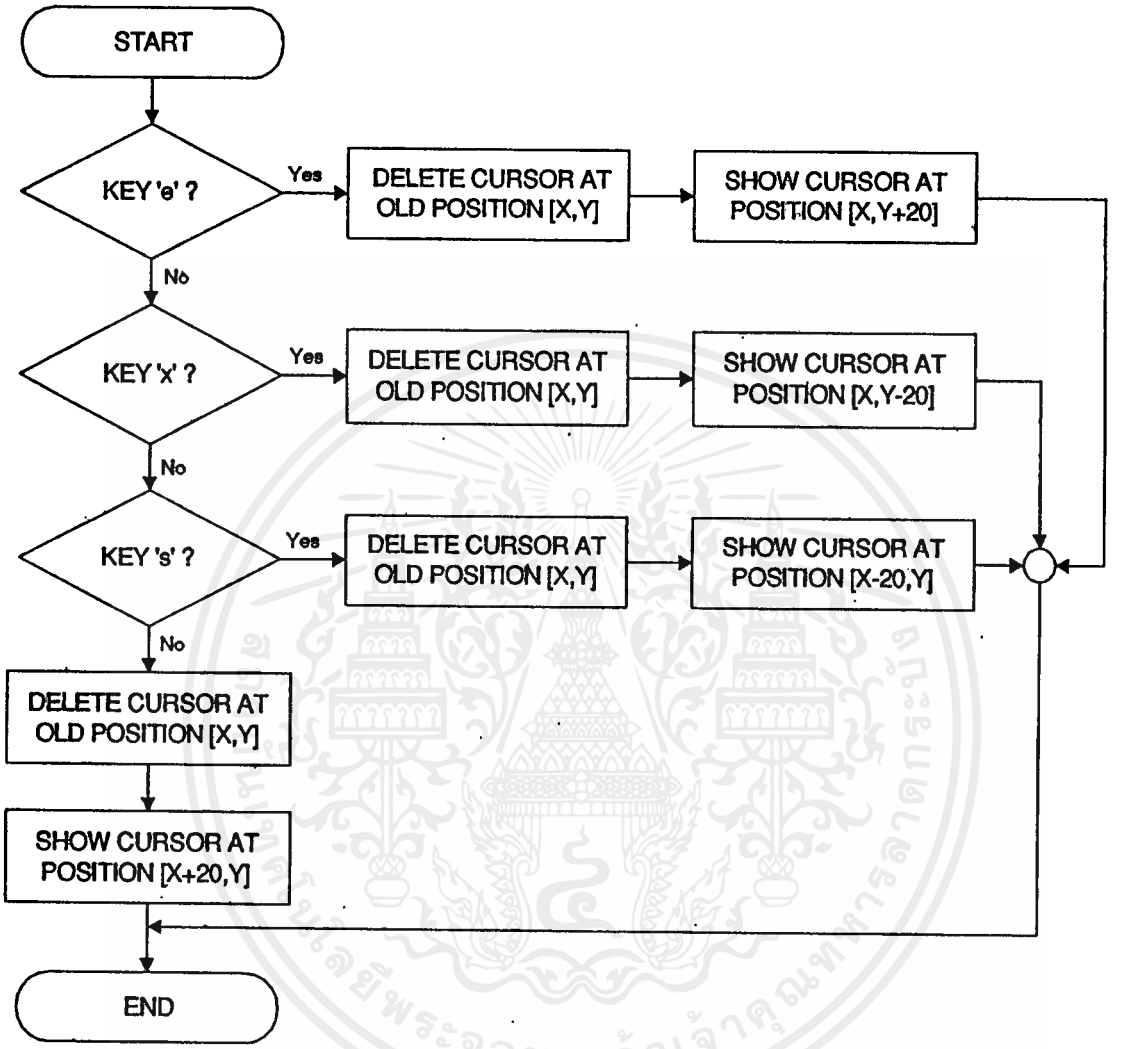
Flowchart Cancel Position Process



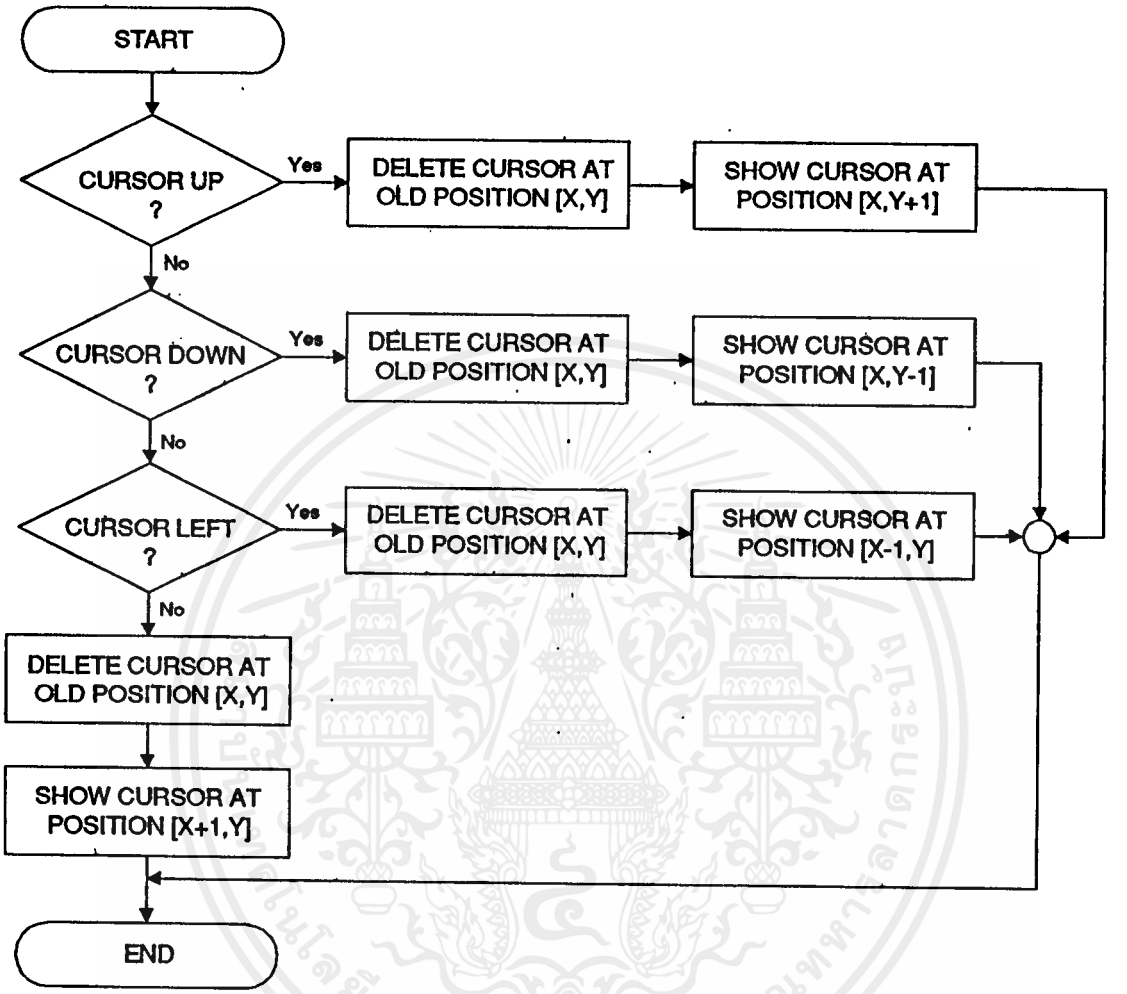
Flowchart Define Position Process



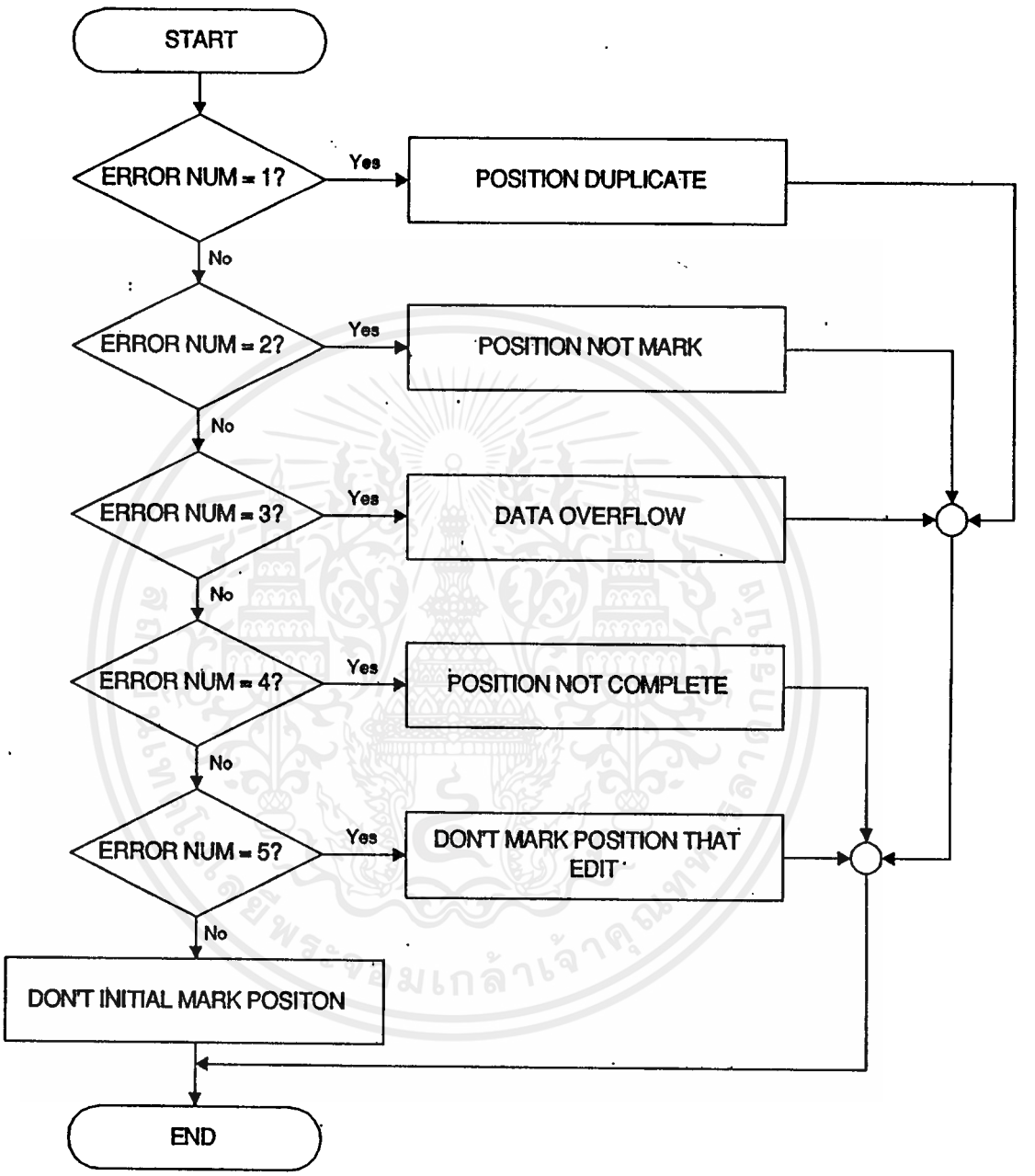
Flowchart Long Step Process



Flowchart Small Step Process



Flowchart Error Process



ภาคผนวก ง

แสดงโปรแกรมการทำงานของระบบ

```
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <alloc.h>
#include <dos.h>
#include "svga256.h"

#define  FileSetup      "C:\\PROJECT\\IMAGE\\1.BMP"
#define  PathImage      "C:\\PROJECT\\IMAGE\\"
#define  BMP            ".BMP"
#define  NORM           0       // normal cursor
#define  MARK           1       // mark   cursor
#define  B_LEFT         5       // left border of road
#define  B_RIGHT        6       // right border of road
#define  LEFT           10
#define  RIGHT          20
#define  UP             30
#define  DOWN           40
#define  MAXX           300     // maximun of wide
#define  MAXY           300     // maximun of high
#define  MAX_FILE       495     // maximun of picture at process

unsigned char far *pdata1[300];
unsigned char far *pdata2[300];
unsigned char far *ptemp[300];
```

```
int flag[4] = {-1,-1,-1,-1};
int linebuff[5][MAXY];
int savepathedg[2];
int carpos[500][4];
int currpoint[2];
int step = 1;

struct {
        unsigned int   password;
        unsigned long  file_size;
        unsigned int   reserve[2];
        unsigned long  headsize;
        unsigned long  infosize;
        unsigned long  xsize;
        unsigned long  ysize;
        unsigned int   bitplan;
        unsigned int   bitcount;
        unsigned long  bicompression ;
        unsigned long  bisizeImage;
        unsigned long  biXPelsPerMeter;
        unsigned long  biYPelsPerMeter;
        unsigned long  biClrUsed;
        unsigned long  biclrImportant;
} header;

void ChkMarkPos(int xinit,int yinit,int x,int y,int *buff,int *savecur);
void MarkPoint(int xinit,int yinit,int x,int y,int color);
void GenlinePos(int x2,int y2,int x1,int y1,int *buff);
void SetVgaPalette256(DacPalette256 *PalBuf);
int  BackWard(int lense,int oldpos,int *carpos);
void SortPos(int *buff,int *pos,int terminate);
int  EdgFollow(int row,int column,int len);
void ChkKey(int xinit,int yinit,int *buff);
void ClrNotRoad(int items,int direction);
```

```
void ShowCursor(int x,int y,int status);
void CursorDownMovement(int *y);
void ForWard(int lense,int *carpos);
void CursorRightMovement(int *x);
void VerNoiseMoving(int x,int y);
void HorNoiseMoving(int x,int y);
void CursorLeftMovement(int *x);
void CursorUpMovement(int *y);
int  FindCarpos(int file,int items);
int  huge DetectVGA256(void);
void ShowFileProNum(int file);
int  ShowLineMark(int *buff);
int  TestLine(int num,int row);
void DisplayBmp(int x,int y);
void PrintPosOut(int x,int y);
void GetError(int errortype);
void InitLenFlag(int items);
void CalVilocity(int items);
void ClearLense(int *buff);
void AssignPos(int *buff);
void InitOutbuff(int *out);
void ShowWaitMsg(void);
void ClearLinebuff(void);
void FindObject(int len);
void ShowPicture(void);
int  far* Getmem(void);
void ShowVilo(int file);
int  GgetInt(int x,int y);
void ShowMenu(void);
void Setup(int *items);
int  LoadBmp(int file);
void InitCarpos(void);
void OrProcess(void);
void ClrPdata1(void);
```

```
void FreeMem(void);
void HorSobel(void);
void InitGraph(void);
void SetPalette(void);
void VerSobel(void);
void Run(int items);
int  SetSpeed(void);

/*************************************************************************/
void main(void)
{
  char ch,buff[3],done = 'n';
  int  items,disflag = 1;

  installuserdriver("Svga256",DetectVGA256);
  registerfarbgidriver(Svga256_fdriver);
  InitGraph();
  SetPalette();
  do {
    if(disflag == 1) {
        disflag = 0;
        ShowMenu();
    }
    ch = getch();
    if(ch>'0' && ch<'5') {
        sprintf(buff,"%c",ch);
        setcolor(255);
        outtextxy(397,250,buff);
        getch();
        cleardevice();
        switch(ch) {
          case '1' :                    // mark position of Road
            Setup(&items);
            break;
```

```
            case '2' :                  // run
              Run(items);
              break;
            case '3' :                  // set speed
              step = SetSpeed();
              break;
            case '4' :                  // exit
              done = 'Y';
          }
          disflag = 1;
        }
      } while(done != 'Y');
      closegraph();
    }
    /*************************************************************************/
    int huge DetectVGA256(void)
    {
      return 2;       //      0     320x200x256 Standard VGA
                      //      1     640x400x256 Svga/VESA
                      //      2     640x480x256 Svga/VESA
                      //      3     800x600x256 Svga/VESA
    }
    /*************************************************************************/
    void SetVgaPalette256(DacPalette256 *PalBuf)
    {
      struct REGPACK reg;

      reg.r_ax = 0x1012;
      reg.r_bx = 0;
      reg.r_cx = 256;
      reg.r_es = FP_SEG(PalBuf);
      reg.r_dx = FP_OFF(PalBuf);
      intr(0x10,&reg);
    }
```

```
/*************************************************************************/
void InitGraph(void)
{
   int GraphDriver=DETECT;
   int GraphMode;
   int ErrorCode;

   initgraph(&GraphDriver,&GraphMode,"");
   if((ErrorCode = graphresult()) != grOk) {
      printf("Graphics System Error: %s!\n",grapherrormsg(ErrorCode));
      exit(1);
   }
}
/*************************************************************************/
void SetPalette(void)
{
   DacPalette256   pal_col_bmp;
   register int    i,j;

   for(i=0; i<256; i+=4)
      for(j=0; j<4; j++) {
            pal_col_bmp[i+j][0] = i/4;          // R
            pal_col_bmp[i+j][1] = i/4;          // G
            pal_col_bmp[i+j][2] = i/4;          // B
      }
   SetVgaPalette256(&pal_col_bmp);
}
/*************************************************************************/
void ShowMenu(void)
{

   cleardevice();  setcolor(255);
   outtextxy(250,40,"MAIN MENU");
   outtextxy(200,90,"1. SETUP");
```

```
  outtextxy(200,120,"2. RUN");
  outtextxy(200,150,"3. SET SPEED");
  outtextxy(200,180,"4. EXIT");
  outtextxy(170,250,"Press number do you want : ");
}
/*************************************************************************/
void Setup(int *items)
{
  int index,out[21];

  ClearLinebuff();
  InitOutbuff(&out[0]);
  DisplayBmp(31,31);
  ChkKey(31,31,&out[0]);
  AssignPos(&out[0]);
  index = ShowLineMark(&out[0]);
  *items = index/4;
  for(index=0; index<*items; index++)
    GenlinePos(out[4*index+2],out[4*index+3],out[4*index],
               out[4*index+1],&linebuff[index][0]);
  setcolor(255);
  outtextxy(180,400,"Please any key to continues.");
  getch();
}
/*************************************************************************/
void Run(int items)
{
  int file,terminate=0;

  InitCarpos();
  InitLenFlag(items-1);
  ShowWaitMsg();
  for(file=1; file<=MAX_FILE; file+=step) {
    if(LoadBmp(file) == 0) {
```

```
            file--;
            break;
        }
        ShowFileProNum(file);
        HorSobel();
        HorNoiseMoving(MAXX,MAXY);
        VerSobel();
        VerNoiseMoving(MAXX,MAXY);
        OrProcess();
        ClrNotRoad(items,B_LEFT);
        ClrNotRoad(items,B_RIGHT);
        switch(items) {
            case 5:
                ClearLense(&linebuff[3][0]);
            case 4:
                ClearLense(&linebuff[2][0]);
            case 3:
                ClearLense(&linebuff[1][0]);
        }
        ClrPdata1();
        terminate = FindCarpos(file,items);
        FreeMem();
        if(terminate == 1)
            break;
    }
    CalVilocity(items);
    ShowPicture();
    ShowVilo(file);
}
/*************************************************************************/
int SetSpeed(void)
{
    int  value,flag;
```

```
    do {
      flag = 1;
      cleardevice();
      outtextxy(150,200,"Enter frequency of Frame Image : ");
      value = GgetInt(410,200);
      if(value == 0) {
            outtextxy(120,230,"Invalid Frame Image !. Please any key...");
            getch();
            flag = 0;
      }
    } while(flag == 0);

    return (value);
}
/*************************************************************************/
void ChkKey(int xinit,int yinit,int *buff)
{
    int  markflag=1,editflag=0,delflag=0,exitflag;
    int  x=-2,y=-3,index=0,display=0,direction=0;
    int  xpos,ypos,temp,brk,tempindex;
    int  far *cur,*mark[10];
    char ch,locate[10];

    setcolor(255);
    rectangle(495,392,577,413);
    setfillstyle(SOLID_FILL,255);
    bar(498,395,574,410);
    PrintPosOut(x,y);
    cur = Getmem();
    getimage(xinit+x,yinit+y,xinit+x+5,yinit+y+7,cur);
    ShowCursor(xinit+x,yinit+y,NORM);
    do {
      exitflag = 0;
      xpos = xinit+x;
```

```
ypos = yinit+y;
switch(ch=toupper(getch())) {
    case '\r':
      if(index < 20) {
        for(temp=0; temp<20 && buff[temp]!=-1; temp+=2)
            if((buff[temp]==x+2)&&(buff[temp+1]==299-(y+3))) {
              GetError(1);
              markflag = 0;        // Data in buff duplicate
              break;
            }
        if(markflag) {
            buff[index++] = x+2;
            buff[index++] = 299-(y+3);
            putimage(xpos,ypos,cur,COPY_PUT);
            brk = index/2-1;
            mark[brk] = Getmem();
            getimage(xpos,ypos,xpos+5,ypos+7,mark[index/2-1]);
            MarkPoint(xinit,yinit,x,y,0);
            getimage(xpos,ypos,xpos+5,ypos+7,cur);
            ShowCursor(xpos,ypos,MARK);
            if(editflag) {
              index = tempindex;
              editflag = 0;
            }
            else
              buff[index] = -1;  // -1 used mark to end of data
        }
        else
            markflag = 1;
      }
      else
        GetError(3);            // Data overflow in  buff
      break;
    case '\0':                  // Cursor Movement Small Step
```

```
switch(getch()) {
  case 72 :              // Cursor Up
       if(y > -3) {
         putimage(xpos,ypos,cur,COPY_PUT);
         ChkMarkPos(xinit,yinit,x,--y,buff,cur);
         direction = 1;
         display = 1;
       }
       break;
  case 75 :              // Cursor Left
       if(x > -2) {
         putimage(xpos,ypos,cur,COPY_PUT);
         ChkMarkPos(xinit,yinit,--x,y,buff,cur);
         direction = 0;
         display = 1;
       }
       break;
  case 77 :              // Cursor Right
       if(x < xinit+266) {
         putimage(xpos,ypos,cur,COPY_PUT);
         ChkMarkPos(xinit,yinit,++x,y,buff,cur);
         direction = 0;
         display = 1;
       }
       break;
  case 80 :              // Cursor Down
       if(y < yinit+265) {
         putimage(xpos,ypos,cur,COPY_PUT);
         ChkMarkPos(xinit,yinit,x,++y,buff,cur);
         direction = 1;
         display = 1;
       }
}
break;
```

```
case 'E' :                    // Cursor Up More Step
  if(y > -3) {
    putimage(xpos,ypos,cur,COPY_PUT);
    CursorUpMovement(&y);
    ChkMarkPos(xinit,yinit,x,y,buff,cur);
    direction = 1;
    display = 1;
  }
  break;
case 'S' :                    // Cursor Left More Step
  if(x > -2) {
    putimage(xpos,ypos,cur,COPY_PUT);
    CursorLeftMovement(&x);
    ChkMarkPos(xinit,yinit,x,y,buff,cur);
    direction = 0;
    display = 1;
  }
  break;
case 'D' :                    // Cursor Right More Step
  if(x < xinit+266) {
    putimage(xpos,ypos,cur,COPY_PUT);
    CursorRightMovement(&x);
    ChkMarkPos(xinit,yinit,x,y,buff,cur);
    direction = 0;
    display = 1;
  }
  break;
case 'X' :                    // Cursor Down More Step
  if(y < yinit+265) {
    putimage(xpos,ypos,cur,COPY_PUT);
    CursorDownMovement(&y);
    ChkMarkPos(xinit,yinit,x,y,buff,cur);
    direction = 1;
    display = 1;
```

```
        }
        break;
    case 'C':
        if(index > 0) {
            for(temp=0; temp<20 && buff[temp]!=-1; temp+=2)
                if((buff[temp]==x+2) && (buff[temp+1]==299-(y+3))) {
                    if(buff[temp+2] == -1 && editflag == 0) {
                        putimage(xpos,ypos,mark[index/2-1],COPY_PUT);
                        buff[index]   = 0;
                        buff[--index] = 0;
                        buff[--index] = -1;
                        getimage(xpos,ypos,xpos+5,ypos+7,cur);
                        ShowCursor(xpos,ypos,NORM);
                    }
                    else {
                        if(!editflag) {
                            tempindex = index;
                            index = temp;
                            buff[index++] = 999;
                            buff[index++] = 999;
                            putimage(xpos,ypos,mark[index/2-1],COPY_PUT);
                            index-=2;
                            editflag = 1;
                            getimage(xpos,ypos,xpos+5,ypos+7,cur);
                            ShowCursor(xpos,ypos,NORM);
                        }
                        else {
                            GetError(5);
                            putimage(xpos,ypos,cur,COPY_PUT);
                            getimage(xpos,ypos,xpos+5,ypos+7,cur);
                            ShowCursor(xpos,ypos,MARK);
                        }
                    }
                    delflag = 1;
```

```
                break;
            }
        }
        if(!delflag)
            GetError(2);            // delete position not mark
        else
            delflag = 0;
        break;
    case 'Q':                       // exit section of mark position
        if(buff[0] == -1) {
            GetError(6);            // not init mark
            ch = 'q';
        }
        else {
            for(temp=0; temp<20 && buff[temp]!=-1; temp+=2)
                if((buff[temp]==999) && (buff[temp+1]==999)) {
                    exitflag = 1;
                    GetError(5);
                    ch = 'q';
                    break;
                }
            if(exitflag == 0) {
                if(buff[temp] == -1 && temp%4 != 0 ) {
                    GetError(4);
                    ch = 'q';
                }
                else {
                    farfree(cur);
                    for(index=0; index<=brk; index++)
                        farfree(mark[index]);
                }
            }
        }
    }
```

```
    if(display) {
        setfillstyle(SOLID_FILL,255);
        if(direction)              // check (x or y) positions update
          bar(538,395,564,410);    //      [direction = 1 is y]
        else                       //      [direction = 0 is x]
          bar(507,395,530,410);
        PrintPosOut(x,y);
        display = 0;               // set default of display flag
    }
  } while(ch != 'Q');
}
/*************************************************************************/

void ChkMarkPos(int xinit,int yinit,int x;int y,int *buff,int *savecur)
{
  int temp,flag=1;

  for(temp=0; temp<20 && buff[temp]!=-1; temp+=2)
    if((buff[temp]==x+2) && (buff[temp+1]==299-(y+3))) {
        getimage(xinit+x,yinit+y,xinit+x+5,yinit+y+7,savecur);
        ShowCursor(xinit+x,yinit+y,MARK);
        flag = 0;
        break;
    }
  if(flag) {
    getimage(xinit+x,yinit+y,xinit+x+5,yinit+y+7,savecur);
    ShowCursor(xinit+x,yinit+y,NORM);
  }
}
/*************************************************************************/
void AssignPos(int *buff)
{
  int pos[6] = {0,0,0,0,0,0};
  int loop,test,terminate;
```

```
int temp[21],save[2];

for(loop=0; loop<20 && buff[loop]!=-1; loop+=4)
  if(buff[loop] <= buff[loop+2])
      pos[loop/4] = loop;          // find low position eatch line
  else
      pos[loop/4] = loop+2;
pos[loop/4] = -1;
terminate = (loop/4)-1;
SortPos(buff,&pos[0],terminate);          // sort position of line
for(loop=0; loop<=terminate; loop++) {  // adjust new position to temp
  for(test=0; test<4; test++)
      temp[(loop*4)+test] = buff[(pos[loop]/4)*4+test];
  if(temp[(loop*4)+1] < temp[(loop*4)+3]) {
      save[0] = temp[loop*4];
      save[1] = temp[(loop*4)+1];
      temp[loop*4] = temp[(loop*4)+2];
      temp[(loop*4)+1] = temp[(loop*4)+3];
      temp[(loop*4)+2] = save[0];
      temp[(loop*4)+3] = save[1];
  }
}
temp[(--loop*4)+test] = -1;
for(loop=0; loop<21 && temp[loop]!=-1; loop++)
  buff[loop] = temp[loop];
}
/*************************************************************************/
int FindCarpos(int file,int items)
{
  int len,row,terminate=0;

  if(file != 1 && step != 1)
    file = file/step+1;
  for(len=1; len<items; len++) {
```

```
    FindObject(len);
    if(flag[len-1] == 0) {
        ForWard(len,&carpos[file-1][len-1]);
        if(carpos[file-1][len-1]!=999 && carpos[file-1][len-1]!=-99)
          flag[len-1] = 1;
    }
    else {
        row = carpos[file-2][len-1];
        if(BackWard(len,row,&carpos[file-1][len-1]) == -1)
          terminate = 1;
    }
  }

  return (terminate);
}
/*************************************************************************/
void ForWard(int lense,int *carpos)
{
  int row,temp;

  row = MAXY-1;
  while((temp=TestLine(lense,--row)) == 0);
  if(temp == -1)
    carpos[0] = 999;
  else {
    while((temp=TestLine(lense,--row)) == 1);
    if(temp == -1)
        carpos[0] = -99;
    else {
        while((temp=TestLine(lense,--row)) == 0);
        if(temp == -1)
          carpos[0] = -99;
        else
          carpos[0] = row+1;
```

```
    }
  }
}
/*************************************************************************/
int BackWard(int lense,int oldpos,int *carpos)
{
  int row,temp,flag;

  row = oldpos;
  while((temp=TestLine(lense,row++)) == 1);
  if(temp == -1) {
    carpos[0] = -999;
    flag = -1;
  }
  else {
    carpos[0] = row-1;
    while((temp=TestLine(lense,row++)) == 0);
    if(temp == -1)
        flag = 1;
    else
        flag = 0;
  }

  return (flag);
}
/*************************************************************************/
void FindObject(int len)
{
  register int row,col;

  for(row=0; row<MAXY; row++) {
    for(col=linebuff[len-1][row]; col<linebuff[len][row]; col++) {
        if(ptemp[row][col] == 255) {
          currpoint[0] = col;
```

```
                currpoint[1] = row;
                savepathedg[0] = col;
                savepathedg[1] = row;
                row = EdgFollow(row,col,len);
                break;
              }
            }
          }
        }
        /*************************************************************************/
        int EdgFollow(int row,int column,int len)
        {
          int x_min,x_max,y_min,y_max;
          int arrow;
          int r,c;

          x_min = x_max = savepathedg[0] ;
          y_min = y_max = savepathedg[1] ;
          arrow = RIGHT;
          do {
            switch(arrow) {
                case LEFT:
                  if((ptemp[row][column] == 255) && (arrow == LEFT)) {
                    row++;
                    arrow = DOWN;
                    if(row > header.ysize-1) {
                        row = header.ysize-1;     // test vertical line long
                        column--;
                        arrow = UP;
                    }
                  }
                  else {
                    row--;
                    arrow = UP;
```

```
        }
        break;
      case RIGHT:
        if((ptemp[row][column] == 255) && (arrow == RIGHT)) {
          row--;
          arrow = UP;
          if(row < 0) {
              row = 0;
              column++;
              arrow = DOWN;
          }
        }
        else {
          row++;
          arrow = DOWN;
        }
        break;

      case UP:
        if((ptemp[row][column] == 255) && (arrow == UP)) {
          column--;
          arrow = LEFT;
        }
        else {
          column++;
          arrow = RIGHT;
        }
        break;
      case DOWN:
        if((ptemp[row][column] == 255) && (arrow == DOWN)) {
          column++;
          arrow = RIGHT;
        }
        else {
```

```
                column--;
                arrow = LEFT;
              }
              break;
        }
        if(column > x_max)  x_max = column;
        if(column < x_min)  x_min = column;
        if(row > y_max)     y_max = row;
        if(row < y_min)     y_min = row;
        savepathedg[0] = column;
        savepathedg[1] = row;
      } while ((currpoint[0] != savepathedg[0]) ||
               (currpoint[1] != savepathedg[1]));

      if(y_max-y_min > 10 && x_max-x_min >= 25) {
        for(r=y_min+2; r<=y_max; r++)
            for(c=linebuff[len-1][r]+5; c<linebuff[len][r]-5; c++)
              pdata1[r][c] = 255;
      }
      return (y_max);
    }
    /*****************************************************************/
    void MarkPoint(int xinit,int yinit,int x,int y,int color)
    {

      putpixel(xinit+x+2,yinit+y+3,color);
      switch(x+2) {
        case 0 :
            putpixel(xinit+x+3,yinit+y+3,color);
            switch(y+3) {
              case 0 :
                putpixel(xinit+x+3,yinit+y+4,color);
                putpixel(xinit+x+2,yinit+y+4,color);
                break;
```

```
            case 299 :
              putpixel(xinit+x+3,yinit+y+2,color);
              putpixel(xinit+x+2,yinit+y+2,color);
              break;
            default :
              putpixel(xinit+x+3,yinit+y+2,color);
              putpixel(xinit+x+2,yinit+y+2,color);
              putpixel(xinit+x+3,yinit+y+4,color);
              putpixel(xinit+x+2,yinit+y+4,color);
          }
          break;
        case 299 :
          putpixel(xinit+x+1,yinit+y+3,color);
          switch(y+3) {
            case 0 :
              putpixel(xinit+x+2,yinit+y+4,color);
              putpixel(xinit+x+1,yinit+y+4,color);
              break;
            case 299 :
              putpixel(xinit+x+2,yinit+y+2,color);
              putpixel(xinit+x+1,yinit+y+2,color);
              break;
            default :
              putpixel(xinit+x+2,yinit+y+2,color);
              putpixel(xinit+x+1,yinit+y+2,color);
              putpixel(xinit+x+2,yinit+y+4,color);
              putpixel(xinit+x+1,yinit+y+4,color);
          }
          break;
        default :
          putpixel(xinit+x+3,yinit+y+3,color);
          putpixel(xinit+x+1,yinit+y+3,color);
          switch(y+3) {
            case 0 :
```

```
                putpixel(xinit+x+3,yinit+y+4,color);
                putpixel(xinit+x+2,yinit+y+4,color);
                putpixel(xinit+x+1,yinit+y+4,color);
                break;
            case 299 :
                putpixel(xinit+x+3,yinit+y+2,color);
                putpixel(xinit+x+2,yinit+y+2,color);
                putpixel(xinit+x+1,yinit+y+2,color);
                break;
            default :
                putpixel(xinit+x+3,yinit+y+2,color);
                putpixel(xinit+x+2,yinit+y+2,color);
                putpixel(xinit+x+1,yinit+y+2,color);
                putpixel(xinit+x+3,yinit+y+4,color);
                putpixel(xinit+x+2,yinit+y+4,color);
                putpixel(xinit+x+1,yinit+y+4,color);
        }
    }
}
/*************************************************************************/

void DisplayBmp(int x,int y)
{
  int i,x_c,x_p,y_c,y_p,linesize;
  DacPalette256   pal_col_bmp;
  unsigned char buff[1024];
  register int row,column;
  FILE *fp;

  if((fp = fopen(FileSetup,"rb")) == NULL) {
    closegraph();
    printf("Cannot Open files");
    exit(1);
  }
```

```
    fread(&header,sizeof(header),1,fp);
    if(header.password == 0x4d42) {          // BM
      if(header.bitcount == 8) {
          fread(buff,header.headsize-sizeof(header),1,fp);
          for(i=0;i<(header.headsize-sizeof(header));i+=4) {
            pal_col_bmp[i/4][0] = buff[i+2] >> 2;      // R
            pal_col_bmp[i/4][1] = buff[i+1] >> 2;      // G
            pal_col_bmp[i/4][2] = buff[i]   >> 2;      // B
          }
          SetVgaPalette256(&pal_col_bmp);
          fseek(fp,header.headsize,SEEK_SET);
          linesize = (header.file_size-header.headsize)/header.ysize;
          for(y_c=0,y_p=y+header.ysize-1;y_c<header.ysize;y_c++,y_p--) {
            fread(buff,linesize,1,fp);
            for(x_c=0,x_p=x;x_c<header.xsize;x_c++,x_p++)
              putpixel(x_p,y_p,buff[x_c]);
          }
          setcolor(255);
          rectangle(28,27,333,334);
      }
      else {
          closegraph();
          printf("\nBitmap file not 256 color!.");
          exit(1);
      }
    }
    else {
      closegraph();
      printf("\nThis file not Bitmap file!.");
      exit(1);
    }
    fclose(fp);
  }
  /*************************************************************************/
```

```
int LoadBmp(int file)                    // load 8 bit .BMP file format only
{                                        // to ptemp buffer picture
  register int row,column;
  char name[128];
  int flag = 1;
  FILE *fp;

  sprintf(name,"%s%d%s",PathImage,file,BMP); //Set C:\PROJECT\IMAGE\*.BMP
  if((fp = fopen(name,"rb")) == NULL)
    flag = 0;
  else {
    fread(&header,sizeof(header),1,fp);
    if(header.password == 0x4d42) {              // BM
        if(header.bitcount == 8) {
          fseek(fp,header.headsize,SEEK_SET);
          for(row=header.ysize-1; row>=0; row--) {
              pdata1[row]=(unsigned char far *)farmalloc(header.xsize);
              pdata2[row]=(unsigned char far *)farmalloc(header.xsize);
              ptemp[row] =(unsigned char far *)farmalloc(header.xsize);
              for(column=0; column<header.xsize; column++) {
                pdata1[row][column] = 0;          // initial values
                pdata2[row][column] = 0;          // initial values
                ptemp[row][column]  = getc(fp);   // load to memory
              }
          }
        }
        else {
          closegraph();
          printf("\nBitmap file not 16 or 256 color !");
          exit(1);
        }
    }
    else {
        closegraph();
```

```
            printf("\nThis file not Bitmap file!");
            exit(1);
        }
        fclose(fp);
    }

    return (flag);
}
/*************************************************************************/
int GgetInt(int x,int y)
{
    char ch[2] = {'0','\0'};
    int index=0,val=0;
    char *value;

    do {
        ch[0] = toupper(getch());
        if(ch[0] == '\b') {
            if(index > 0) {
                setfillstyle(SOLID_FILL,0);
                bar(x+8*(--index),y-3,x+(8*index),y+7);
                moveto(x+(8*index),y);
            }
        }
        else {
            value[index++] = ch[0];
            if(ch[0] != '\r')
                outtextxy(x+(8*(index-1)),y,ch);
        }
        while(index == 6 && ch[0] != '\r') {
            outtextxy(x+(8*(index-1)),y,ch);
            printf("\a");
            setfillstyle(SOLID_FILL,0);
            bar(x+8*(--index),y-3,x+(8*index),y+7);
```

```
            moveto(x+(8*index),y);
            ch[0] = toupper(getch());
            if(ch[0] == '\b') {
              setfillstyle(SOLID_FILL,0);
              bar(x+8*(--index),y-3,x+(8*index),y+7);
              moveto(x+(8*index),y);
            }
            else
              ++index;
        }
    } while(ch[0] != '\r');
    value[--index] = '\0';
    val = atoi(value);

    return (val);
}
/*************************************************************************/
void CalVilocity(int items)
{
  int len,carcount=0;
  int index=0,count=0,sum=0;

  for(len=1; len<items; len++) {
    while(carpos[index][len-1] == 999 || carpos[index][len-1] == -99) {
          if(carpos[index][len-1] == 999 && carcount < 20/step)
            carcount++;
          index++;
    }
    while(carpos[index][len-1]!=-999 && carpos[index+1][len-1]!=-999 &&
          carpos[index][len-1]!=-1   && carpos[index+1][len-1]!=-1) {
          sum = sum+(carpos[index+1][len-1]-carpos[index][len-1]);
          count++;
          index++;
    }
```

```
    if(carcount == 20/step)
        carpos[499][len-1] = 0;
    else
        carpos[499][len-1] = sum/count;
    carcount = 0;
    index = 0;
    count = 0;
    sum = 0;
  }
}
/*****************************************************************************/
void GenlinePos(int x2,int y2,int x1,int y1,int *buff)
{
  float m,b,tempx,tempy;
  register int row;
  int terminate;

  if(y2 != y1) {                    // specific vertical line
    terminate = MAXY-1-y2;
    if(x2 != x1) {                  // x1 != x2
        tempy = y2-y1;
        tempx = x2-x1;
        m = tempy/tempx;
        b = y1-(m*x1 );
        for(row=0; row<=terminate; row++)
          buff[terminate-row] = (int)(row+y2-b)/m;
    }
    else                            // x1 == x2
        for(row=0; row<=terminate; row++)
          buff[terminate-row] = x1;
    if(terminate < MAXY-1 && x2 == MAXX-1)
        while(terminate != MAXY)
          buff[terminate++] = MAXX-1;
  }
```

```
}
/*************************************************************************/
void SortPos(int *buff,int *pos,int terminate)
{
  int test,temp,loop;              // sort position of line left > right

  for(test=0; test<terminate; test++)
    for(loop=0; loop<terminate-test; loop++)
        if(buff[pos[loop]] <= buff[pos[loop+1]]) {
          if((buff[pos[loop]] == buff[pos[loop+1]]) &&
            (buff[pos[loop]+1] < buff[pos[loop+1]+1])) {
            temp       = pos[loop];
            pos[loop]  = pos[loop+1];
            pos[loop+1] = temp;
          }
        }
        else {
          temp       = pos[loop];
          pos[loop]  = pos[loop+1];
          pos[loop+1] = temp;
        }
}
/*************************************************************************/
void ShowVilo(int file)
{
  int index = 0;
  char buff[40];
  setcolor(255);
  while(carpos[499][index] != -1 && index < 4) {
    sprintf(buff,"lcn %d car is avg speed  %d",
            index+1,carpos[499][index]);
    outtextxy(330,90+(40*index),buff);
    ++index;
  }
```

```
    sprintf(buff,"process picture until file %3d",file);
    outtextxy(20,310,buff);
    if(step != 1)
      file = file/step+1;
    sprintf(buff,"process picture total %3d  file",file);
    outtextxy(330,310,buff);
    outtextxy(190,400,"Please any key to continues.");
    getch();
  }
  /*************************************************************************/
  void VerNoiseMoving(int x,int y)
  {
    register int row,col,win_row,win_col;
    int mcount;

    for(row=1; row<y-1; row++)
      for(col=1; col<x-1; col++) {
          mcount = 0;
          for(win_row=-1; win_row<2; win_row++)
            for(win_col=-1; win_col<2; win_col++)
              mcount += (int)pdata1[row+win_row][col+win_col];
          if(mcount <= 1020)
            pdata1[row][col] = 0;
          else
            pdata1[row][col] = 255;
      }
  }
  /*************************************************************************/
  void HorNoiseMoving(int x,int y)
  {
    register int row,col,win_row,win_col;
    int mcount;

    for(row=1; row<y-1; row++)
```

```
        for(col=1; col<x-1; col++) {
            mcount = 0;
            for(win_row=-1; win_row<2; win_row++)
              for(win_col=-1; win_col<2; win_col++)
                mcount += (int)pdata2[row+win_row][col+win_col];
            if(mcount <= 1020)
              pdata2[row][col] = 0;
            else
              pdata2[row][col] = 255;
        }
    }
    /*************************************************************************/
    void GetError(int errortype)
    {
      char errordetail[7][35] = {
            "Please any key to continuous.",     // 0 default all key
            "This position Duplicate!.",         // 1 "\r" enter key
            "This position Not Mark!.",          // 2 "C" delete key
            "Don't Mark data Overflow!.",        // 3 "\r" data full
            "Mark position Not Complete!.",      // 4 "Q" ! complete
            "Don't Mark position that edit!.",   // 5 "\r" ! mark
            "Don't Initial Mark position!."      // 6 "Q" ! init
      };

      setcolor(255);
      outtextxy(110,360,errordetail[errortype]);
      outtextxy(110,410,errordetail[0]);
      getch();
      setfillstyle(SOLID_FILL,0);
      bar(110,360,360,425);
    }
    /*************************************************************************/
    void VerSobel(void)                              // output to pdata1
    {
```

```
    register int row,col;
    int temp,result;

    for(row=1; row<MAXY-1; row++)
      for(col=1; col<MAXX-1; col++) {
          temp = -(int)ptemp[row-1][col-1]-2*(int)ptemp[row][col-1]
                 -(int)ptemp[row+1][col-1];
          temp += (int)ptemp[row-1][col+1]+2*(int)ptemp[row][col+1]
                 +(int)ptemp[row+1][col+1];
          result = abs(temp);
          if(result >= 35)
            result = 255;
          else
            result = 0;
          pdata1[row][col] = (unsigned char)result;
      }
}
/*************************************************************************/
void HorSobel(void)                          // output to pdata2
{
  register int row,col;
  int temp,result;

  for(row=1; row<MAXY-1; row++)
    for(col=1; col<MAXX-1; col++) {
          temp = -(int)ptemp[row+1][col-1]-2*(int)ptemp[row+1][col]
                 -(int)ptemp[row+1][col+1];
          temp += (int)ptemp[row-1][col-1]+2*(int)ptemp[row-1][col]
                 +(int)ptemp[row-1][col+1];
          result = abs(temp);
          if(result >= 35)
            result = 255;
          else
            result = 0;
```

```
            pdata2[row][col] = (unsigned char)result;
        }
    }
    /*************************************************************************/
    int TestLine(int num,int row)
    {
      register int col;
      int flag = 1;

      if(row>0 && row<MAXY-1) {
        for(col=linebuff[num-1][row]+1; col<linebuff[num][row]; col++)
            if(pdata1[row][col] == 255) {
              flag = 0;
              break;
            }
      }
      else
        flag = -1;
      return (flag);
    }
    /*************************************************************************/
    void ShowCursor(int x,int y,int status)
    {
      register int  row,col;
      unsigned char cursor [7][5] = {{'0','0','1','0','0'},
                                     {'0','0','1','0','0'},
                                     {'0','0','1','0','0'},
                                     {'1','1','1','1','1'},
                                     {'0','0','1','0','0'},
                                     {'0','0','1','0','0'},
                                     {'0','0','1','0','0'}};
      if(status == MARK) {
        cursor[2][1] = '1';
        cursor[2][3] = '1';
```

```
        cursor[4][1] = '1';
        cursor[4][3] = '1';
      }
      for(row=0; row<7 ;row++)
        for(col=0; col<5 ;col++)
            if(cursor[row][col] == '1')
              putpixel(x+col,y+row,255);
}
/*************************************************************************/
int ShowLineMark(int *buff)          // show position of line that mark
{
   char showbuff[30];
   int  index;

   setcolor(255);
   sprintf(showbuff,"[ X1, Y1] - [ X2, Y2]");
   outtextxy(435,30,showbuff);
   index = 0;
   while(buff[index] != -1) {
     sprintf(showbuff,"[%3d,%3d] - [%3d,%3d]",buff[index],
     buff[index+1],buff[index+2],buff[index+3]);
     outtextxy(435,90+(10*index),showbuff);
     index+=4;
   }

   return (index);
}
/*************************************************************************/
void CursorDownMovement(int *y)
{
   int temp;

   if((temp = (*y+3)%20) == 0)
     if(*y >= 277)
```

```
            *y += 19 ;
        else
            *y += 20;
    else
        if(*y >= 277)
            *y += 19-temp;
        else
            *y += 20-temp;
}
/*************************************************************************/
void CursorLeftMovement(int *x)
{
    int temp;

    if((temp = (*x+3)%20) == 0)
        if(*x <= 17)
            *x -= 19;
        else
            *x -= 20;
    else
        if(*x <= 17)
            *x -= temp-1;
        else
            *x -= temp;
}
/*************************************************************************/
void ShowPicture()
{
    int temp,index,row;
    char buff[5];

    cleardevice();
    setcolor(255);
    rectangle(80,80,231,231);
```

```
    for(index=0; index<5 &&linebuff[index][0] != 0; index++) {
        for(row=0; row<MAXY; row+=2)
            putpixel(81+linebuff[index][row]/2,81+row/2,255);
        if(index != 0 && index < 5) {
            sprintf(buff,"%d",index);
            temp = linebuff[index-1][139];
            outtextxy(81+temp/2+(linebuff[index][139]-temp)/4,150,buff);
        }
    }
}
/*************************************************************************/
void ClearLense(int *buff)
{
    int row,inc;

    for(row=0; row<MAXY; row++) {
        if(ptemp[row][buff[row]] != 0)
            ptemp[row][buff[row]] = 0;
        for(inc=1; inc<10; inc++) {
            if(buff[row]-inc > 0 && ptemp[row][buff[row]-inc] != 0)
                ptemp[row][buff[row]-inc] = 0;
            if(buff[row]+inc < MAXX && ptemp[row][buff[row]+inc] != 0)
                ptemp[row][buff[row]+inc] = 0;
        }
    }
}
/*************************************************************************/
void ClrNotRoad(int items,int direction)
{
    register int row,col;

    if(direction == B_LEFT)
        for(row=0; row<MAXY; row++)
            for(col=1; col<=(linebuff[0][row]+5); col++)
```

```
                ptemp[row][col] = 0;
        else
            for(row=0; row<MAXY; row++)
                for(col=(linebuff[items-1][row]-5); col<MAXX-1; col++)
                    ptemp[row][col] = 0;
    }
    /*************************************************************************/
    int far* Getmem(void)                    // get mem used save screen
    {
      int far* ptr;

      if((ptr = (int far*)farmalloc(100)) == NULL) {
        closegraph();
        printf("\nError:not enough heap space in save screen!.\n");
        exit(1);
      }

      return (ptr);
    }
    /*************************************************************************/
    void ShowFileProNum(int file)
    {
      char buff[40];

      setfillstyle(SOLID_FILL,0);
      bar(309,239,333,247);
      sprintf(buff,"File picture %3d is processed",file);
      setcolor(255);
      outtextxy(206,240,buff);
    }
    /*************************************************************************/
    void ShowWaitMsg(void)
    {
      char buff[50];
```

```
    cleardevice();
    sprintf(buff,"Please Wait program is being process picture");
    setcolor(255);
    outtextxy(150,190,buff);
  }
  /*************************************************************************/
  void FreeMem(void)
  {
    register int row;

    for(row=0; row<header.ysize; row++) {
      farfree(pdata1[row]);
      farfree(pdata2[row]);
      farfree(ptemp[row]);
    }
  }
  /*************************************************************************/
  void CursorUpMovement(int *y)
  {
    int temp;

    if((temp = (*y+3)%20) == 0)
      *y -= 20;
    else
      *y -= temp;
  }
  /*************************************************************************/
  void CursorRightMovement(int *x)
  {
    int temp;

    if((temp = (*x+3)%20) == 0)
      *x += 20;
```

```
    else
        *x += 20-temp;
}
/*************************************************************************/
void PrintPosOut(int x,int y)
{
  char locate[10];

  sprintf(locate,"[%3d,%3d]",x+2,299-(y+3));
  setcolor(0);
  outtextxy(500,400,locate);
}
/*************************************************************************/
void OrProcess(void)
{
  register int row,col;

  for(row=0; row<MAXY; row++)
    for(col=0; col<MAXX; col++)
      ptemp[row][col] = (int)pdata1[row][col] | (int)pdata2[row][col];
}
/*************************************************************************/
void ClrPdata1(void)
{
  register int row,col;

  for(row=0; row<MAXY; row++)
    for(col=0; col<MAXX; col++)
      pdata1[row][col] = 0;
}
/*************************************************************************/
void InitCarpos(void)                    // initial values to carpos
{
  register int row,col;
```

```
    for(row=0; row<500; row++)
      for(col=0; col<4; col++)
        carpos[row][col] = -1;
  }
  /*************************************************************************/
  void ClearLinebuff(void)
  {
    register int row,index;

    for(index=0; index<5;index++)
      for(row=0; row<MAXY; row++)
        linebuff[index][row] = 0;
  }
  /*************************************************************************/
  void InitOutbuff(int *out)
  {
    int index;
    out[0] = -1;
    for(index=1; index<21; index++)
      out[index] = 0;
  }
  /*************************************************************************/
  void InitLenFlag(int items)                    // initial flag of len
  {
    int row;
    for(row=0; row<items; row++)
      flag[row] = 0;
  }
```

กิตติกรรมประกาศ

ขอกราบขอบพระคุณ คุณพ่อ และ คุณแม่ ของข้าพเจ้าที่ให้การสนับสนุนด้านการศึกษา และดูแลเอาใจใส่ข้าพเจ้ามาโดยตลอด

ขอขอบพระคุณ อาจารย์บุญวัฒน์ อัตชู อาจารย์ที่ปรึกษาของข้าพเจ้าที่ได้ให้ความอนุเคราะห์ ทั้งทางด้านความรู้ คำปรึกษา และแนะนำแนวทางการทำโครงงานวิทยานิพนธ์นี้สำเร็จลุล่วงด้วยดี

ขอขอบพระคุณเหล่าคณาจารย์และเจ้าหน้าที่ภาควิชาวิศวกรรมศาสตร์คอมพิวเตอร์ ในความช่วยเหลือทุกๆ ด้าน

ขอบคุณ พี่ เพื่อน และน้อง ทุกๆ คนที่ช่วยการทำงานครั้งนี้สำเร็จลุล่วงไปได้

เอกสารอ้างอิง

1. ADRIAN LOW, "INTRODUCTORY COMPUTER VISION AND IMAGE PROCESSING", McGRAW-HILL Book Company, 244 p., 1992.

2. RAFAEL C. GONZALEZ, RICHARD E. WOODS, "Digital Image Processing", Addison Wesley Publishing Copany, 716 p., 1992.

3. สมศักดิ์ วลัยรัชต์, "การวิเคราะห์และระบุส่วนประกอบของหน้าเอกสารและการรู้จำตัวอักษร", วิทยานิพนธ์มหาบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2539

4. สาธิต ทองจีน, "รู้จัก Bitmapped Graphic ตอน PCX และ BMP", วารสารคอมพิวเตอร์รีวิว, ฉบับที่ 106, 2536, หน้า 221-230.