เครื่องเก็บข้อมูลภาพ 3 มิติ

3D SCANNER

โดย

นายอะโณ จันทเณร

นางสาวสุกัญญา ศรีสุเทพ

นางสาววิจิตรา บุรพพงษานนท์

อาจารย์ที่ปรึกษา

อาจารย์ เทอดศักดิ์ ลิ่วหาทอง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

รายงานเรื่อง (ภาษาไทย)	เครื่องเก็บข้อมูลภาพ 3 มิติ
(ภาษาอังกฤษ)	3D SCANNER
จัดทำโดย	นายอะโณ จันทเณร
	นางสาวสุกัญญา ศรีสุเทพ
	นางสาววิจิตรา บุรพัพงษานนท์
อาจารย์ที่ปรึกษา	อ. เทอดศักดิ์ ลิ่วหาทอง

รายงานฉบับนี้ได้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ ..... อาจารย์ที่ปรึกษา

(นายเทอดศักดิ์ ลิ่วหาทอง)

วันที่ 20 / มี.ค. / 2541

กิตติกรรมประกาศ

ในระหว่างการดำเนินงานการทำโครงงานนี้ผู้จัดทำได้รับความอนุเคราะห์จากผู้มีพระคุณหลายท่านด้วยกันซึ่งทำให้โครงงานนี้สำเร็จลุล่วงไปด้วยดี โดยเฉพาะอย่างยิ่งอาจารย์ที่ปรึกษาซึ่งให้คำปรึกษาและความสนใจเป็นอย่างดี ขอบคุณภาคเครื่องกล ซึ่งได้ให้ความเอื้อเฟื้อสถานที่และเครื่องมือในการปฏิบัติการด้านฮาร์ดแวร์ของโครงงานนี้ใน Shop เครื่องกล และขอขอบคุณผู้มีพระคุณอีกหลายท่านที่มีส่วนเกี่ยวข้องจนทำให้โครงงานนี้สำเร็จลุล่วงไปด้วยดี

อะโณ จันทเณร
(นายอะโณ จันทเณร)

สุกัญญา ศรีสุเทพ
(นางสาวสุกัญญา ศรีสุเทพ)

วิจิตรา บุรพพงษานนท์
(นางสาววิจิตรา บุรพพงษานนท์)

ผู้จัดทำ

เครื่องเก็บข้อมูลภาพ 3 มิติ

นาย อะโณ จันทเณร
นางสาว สุกัญญา ศรีสุเทพ
นางสาว วิจิตรา บุรพพงษานนท์
อ. เทอดศักดิ์ ลิ่วหาทอง (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2540

บทคัดย่อ

เครื่องเก็บข้อมูลภาพ 3 มิติเป็นเครื่องมือที่สามารถสแกนลักษณะโครงสร้างภายนอกของวัตถุออกมาเป็นโครงสร้าง 3 มิติโดยจะทำการถ่ายภาพของวัตถุในแต่ละมุมจนครบ 1 รอบของวัตถุ โดยใช้กล้องวิดีโอและแสงเลเซอร์ในการถ่ายภาพจากนั้นนำภาพทั้งหมดไปประมวลผลภาพเพื่อให้ได้โครงสร้างข้อมูลภาพ 3 มิติแล้วแสดงออกมาที่หน้าจอมอนิเตอร์ ซึ่งโครงสร้างข้อมูลภาพ 3 มิติที่ได้สามารถนำไปประยุกต์ใช้กับงานด้านต่างๆ เช่น ด้านการแพทย์ หรือ งานด้านอุตสาหกรรม

3D SCANNER

Mr. Ano Juntanen
Miss Sukanya Srisuthep
Miss Wijittra Burapapongsanon
Mr.Thurdsak Leauhatong (Advisor)
1997

Abstract

3D SCANNER is an equipment which is able to scan structure of any objects and display to the monitor in 3D-animation figure by taking the picture of that objects in every degree until around it. This equipment requires a video camera and a laser for capturing the pictures after that all of the pictures are processed to 3D-animation figure by using "Image Processing" then they are abled to show on the monitor. That image are modified to use with others branches such as medical or industrial works.

สารบัญ

สารบัญรูป

หน้า

หน้า

หน้า

สารบัญตาราง

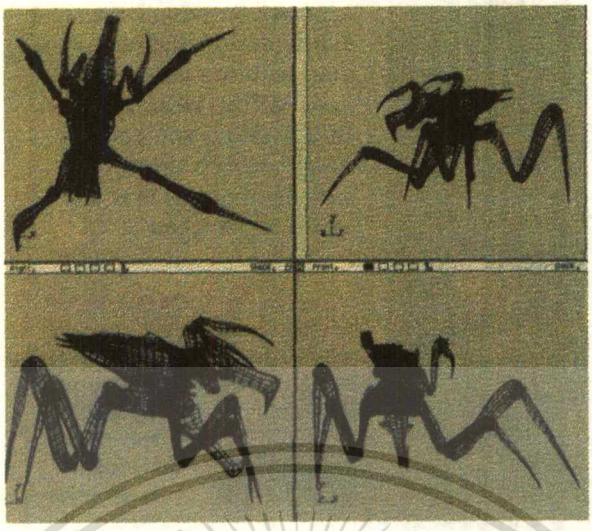
หน้า

บทที่ 1
บทนำ

เครื่องเก็บข้อมูลภาพ 3 มิติ เป็นโครงงานที่สร้างขึ้นมาโดยมีแนวความคิดจากเทคโนโลยีการสร้างภาพ 3 มิติ ที่ดูว่าจะเริ่มเข้ามาแพร่หลายในประเทศไทยมากขึ้น ภาพยนตร์จากต่างประเทศที่นำเข้ามาฉายในประเทศไทย ไม่ว่าจะเป็น Toy Story , Jurassic Park และอีกหลายเรื่อง ล้วนแล้วแต่ใช้เทคโนโลยีการสร้างภาพ 3 มิติทั้งสิ้น ตัวอย่างที่จะยกมาให้ดูต่อไปนี้ เป็นขั้นตอนการสร้างภาพ 3 มิติ ของสัตว์ประหลาดตัวหนึ่งซึ่งกว่าที่จะมาเป็นภาพ 3 มิติได้นั้น ต้องผ่านขั้นตอนที่ยุ่งยากซับซ้อนมาก

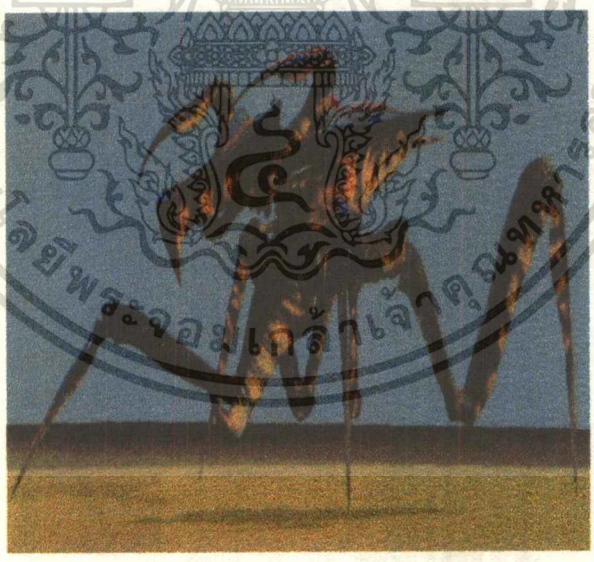
รูปที่ 1.1 แสดงรูปการใช้เครื่องสแกนเนอร์สแกนไปตามโครงสร้าง

ขั้นตอนแรก ก็คือ ต้องทำโครงสร้างคร่าว ๆ ของสัตว์ประหลาดตัวนั้นขึ้นมาก่อน
ขั้นตอนที่สอง ซึ่งเป็นขั้นตอนที่เกี่ยวข้องโดยตรงกับโครงงานนี้ คือ การใช้สแกนเนอร์สแกนไปตามโครงสร้างนั้น ซึ่งสแกนเนอร์ที่ใช้ ในรูปมีประสิทธิภาพค่อนข้างสูงมาก สามารถรู้ว่าจุดหักมุมมีขนาดกี่องศา แล้วนำข้อมูลที่สแกนได้เหล่านั้นมาประมวลผลเป็นภาพโครงสร้าง 3 มิติออกมาได้ดังรูปที่ 1.2



รูปที่ 1.2 แสดงภาพโครงสร้าง 3 มิติที่ได้จากการสแกน

ขั้นตอนต่อไปก็จะใช้เทคนิคทางด้านกราฟฟิกแต่งภาพโครงสร้างนั้นตามจินตนาการ แล้วนำภาพที่แต่งแล้วนั้นไปสร้างเป็นตัวสัตว์ประหลาดออกมา ตามรูปที่ 1.3



รูปที่ 1.3 แสดงหุ่นจำลองของสัตว์ประหลาดที่สร้างมาจากภาพกราฟฟิก 3 มิติ

จากตัวอย่างที่ยกมา คงจะเห็นประโยชน์ของเครื่องสแกนเนอร์ 3 มิติแล้ว แต่ประโยชน์ของมันไม่ได้มีเพียงเท่านี้ เทคโนโลยีทางการแพทย์ก็ได้นำเอาหลักการสแกนนี้ไปใช้ด้วย เช่น MRI (Magnetic Resonance Image) ใช้ในการสแกนหัวกะโหลก เพื่อหาตำแหน่งของการผ่าตัดที่จะ

กระทบกระเทือนต่อเนื้อเยื่อสมองน้อยที่สุด หรือแม้แต่เทคโนโลยีทางอุตสาหกรรม เช่น CNC ก็ใช้การสแกนวัตถุจริงออกมาเป็นภาพโครงสร้าง 3 มิติ แล้วนำไปผลิตเป็นชิ้นส่วนออกมา

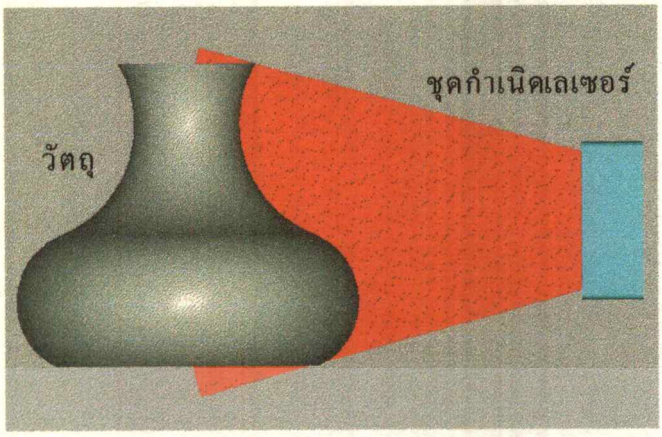
ที่กล่าวมาเป็นเพียงประโยชน์ส่วนหนึ่งเท่านั้น ยังมีเทคโนโลยีทางด้านอื่นอีกมากมายที่นำเอาหลักการนี้ไปใช้ และโครงงานนี้ก็จัดทำขึ้น เพื่อเป็นจุดเริ่มต้นของแนวความคิดเหล่านั้น ให้คนไทยได้รับรู้ และพัฒนาไปให้ถึงเทคโนโลยีตรงจุดนั้นให้สำเร็จ

บทที่ 2
โครงสร้างของระบบโดยรวม

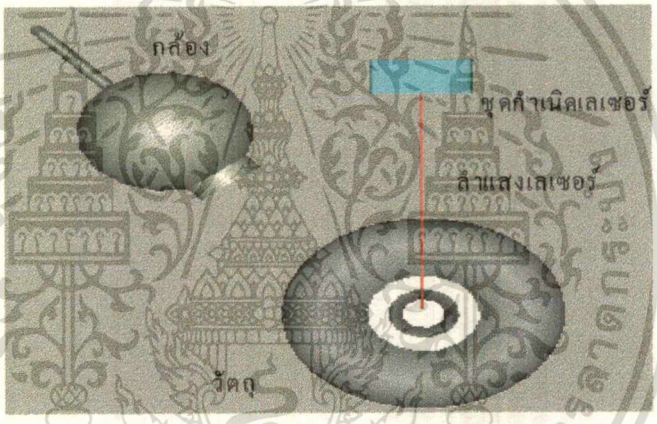
3D-SCANNER เป็นเครื่องมือที่ใช้ในการสแกนลักษณะโครงสร้างภายนอกของวัตถุที่เรามองเห็นกันอยู่ออกมาเป็นรูปโครงสร้าง 3 มิติแล้วแสดงออกมาทางหน้าจอมอนิเตอร์ โดยหลักการทำงานของ 3D-SCANNER คือ การถ่ายภาพของวัตถุในแต่ละมุมจนครบรอบวัตถุโดยใช้กล้องวิดีโอแล้วรวบรวมภาพที่ได้ครบ 1 รอบวัตถุนั้นเก็บไว้ในหน่วยความจำของ PC จากนั้นนำภาพที่เก็บทั้งหมดไปประมวลผลภาพโดยใช้วิธีการ Image Processing ตำแหน่งของกล้องวิดีโอจะถูกควบคุมโดยสเตปมอเตอร์ ซึ่งการควบคุมสเตปมอเตอร์ก็จะถูกควบคุมผ่านทางการ์ดพอร์ทขนานที่ต่อกับ ISA SLOT ของเครื่องคอมพิวเตอร์ด้วย โดยวงจรของการ์ดพอร์ทขนานที่ต่อควบคุมการทำงานของสเตปมอเตอร์นั้น ได้ถูกต่อแยกอิสระจากวงจรขับสเตปมอเตอร์โดยใช้ Opto-Isolator เพื่อไม่ให้เกิดการรบกวนกันและเพื่อป้องกันความเสียหายที่อาจจะเกิดขึ้นกับเครื่องคอมพิวเตอร์ได้

ส่วนประกอบของ 3D-SCANNER ที่สำคัญ ได้แก่ แท่นวางวัตถุสำหรับสแกน แขนยึดกล้องวิดีโอ แขนยึดเลเซอร์ การ์ดพอร์ทขนาน สเตปมอเตอร์และวงจรขับสเตปมอเตอร์ ซึ่งรายละเอียดและการออกแบบในแต่ละส่วนจะแสดงไว้ในบทต่อๆไป

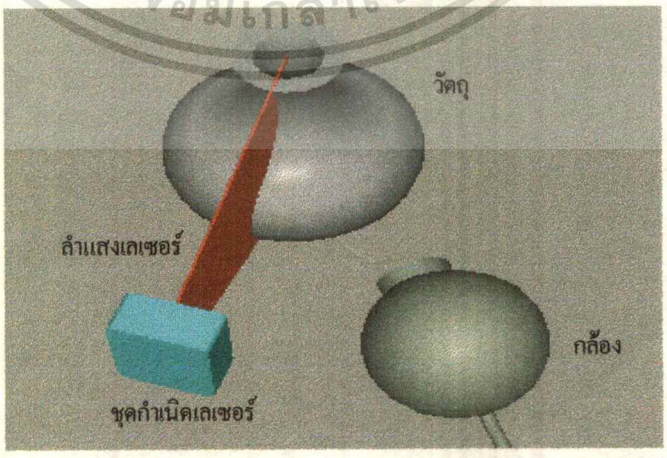
ในการนำภาพที่ได้ไปประมวลผลภาพนั้นเพื่อให้กระทำได้ง่ายจึงใช้แสง Laser ยิงไปตกกระทบที่ผิวของวัตถุในแนวแกนตั้งตามความสูงของวัตถุเพื่อทำให้เกิดขอบของภาพแล้วทำการหมุนวัตถุไปในมุมต่างๆจนครบรอบวัตถุ ซึ่งถ้าจัดตำแหน่งของกล้องในทิศทางตั้งฉากกับชุดกำเนิดเลเซอร์แล้วภาพที่ได้จากกล้องวิดีโอนั้นจะไม่เห็นขอบของภาพซึ่งเกิดจากแสง Laser ที่ยิงไป ดังนั้นในการจัดตำแหน่งของอุปกรณ์นั้น สิ่งสำคัญคือเราต้องจัดวางตำแหน่งของกล้องและชุดกำเนิด Laser ให้ทำมุมกันอย่างเหมาะสม ดังแสดงในรูป 2.1-2.4



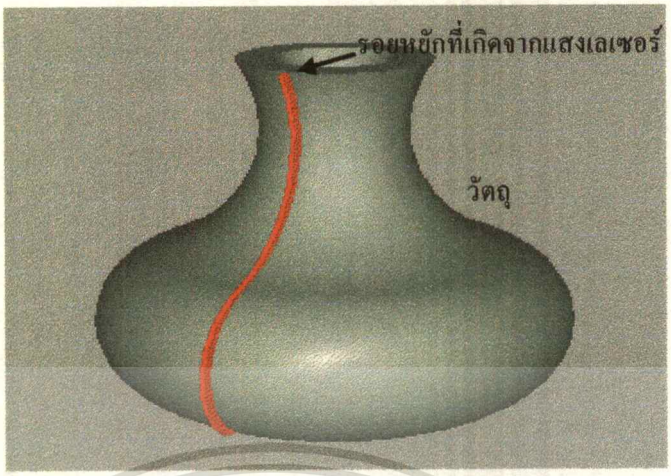
รูป 2.1 แสดงแสงเลเซอร์ที่ถูกส่งจากแหล่งกำเนิดไปสู่วัตถุมองจากด้านข้าง



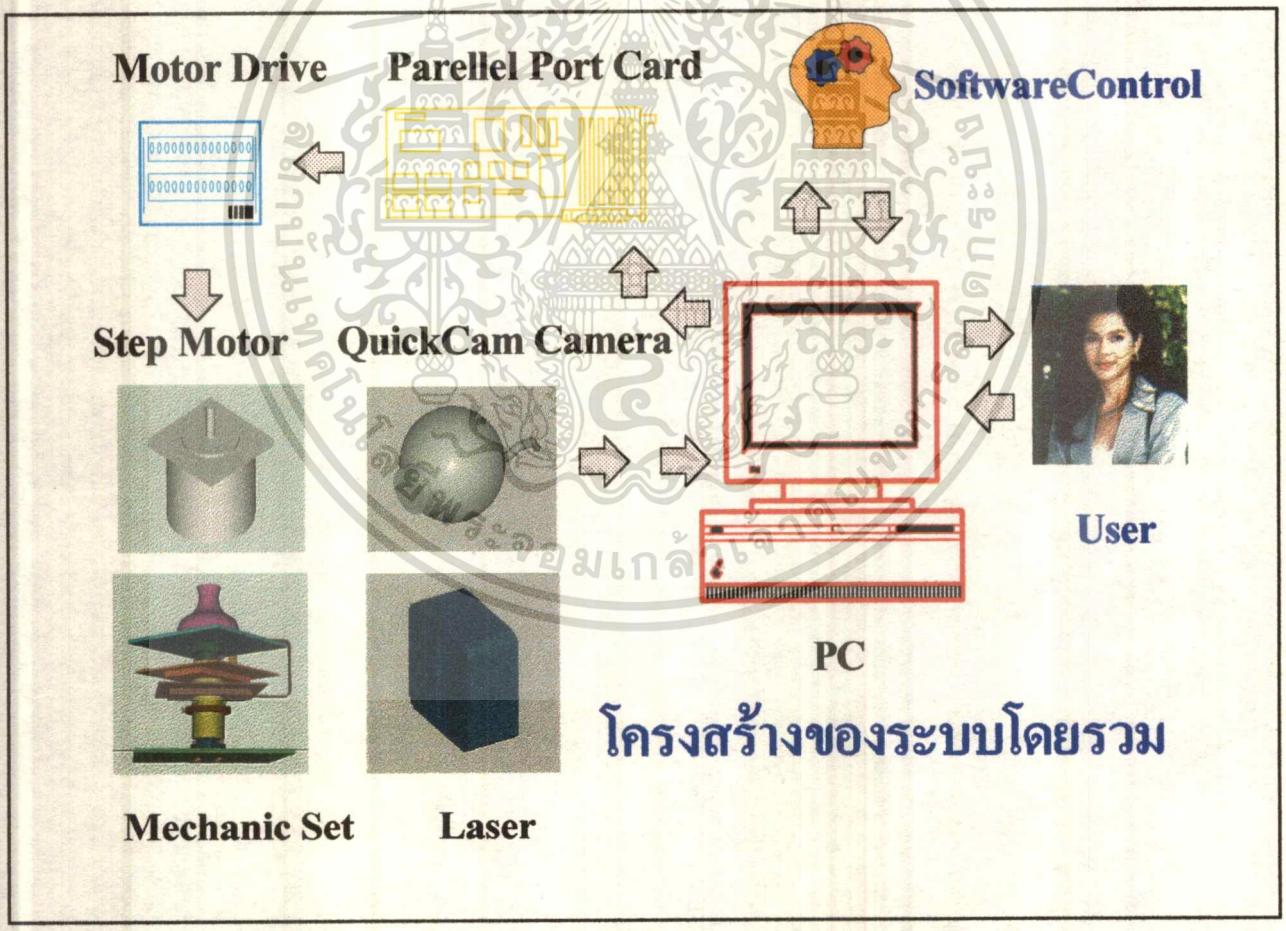
รูป 2.2 แสดงตำแหน่งของกล้อง ชุดกำเนิดเลเซอร์ วัตถุมองจากด้านบน



รูป 2.3 แสดงตำแหน่งของเลเซอร์และกล้องในการใช้งานจริง

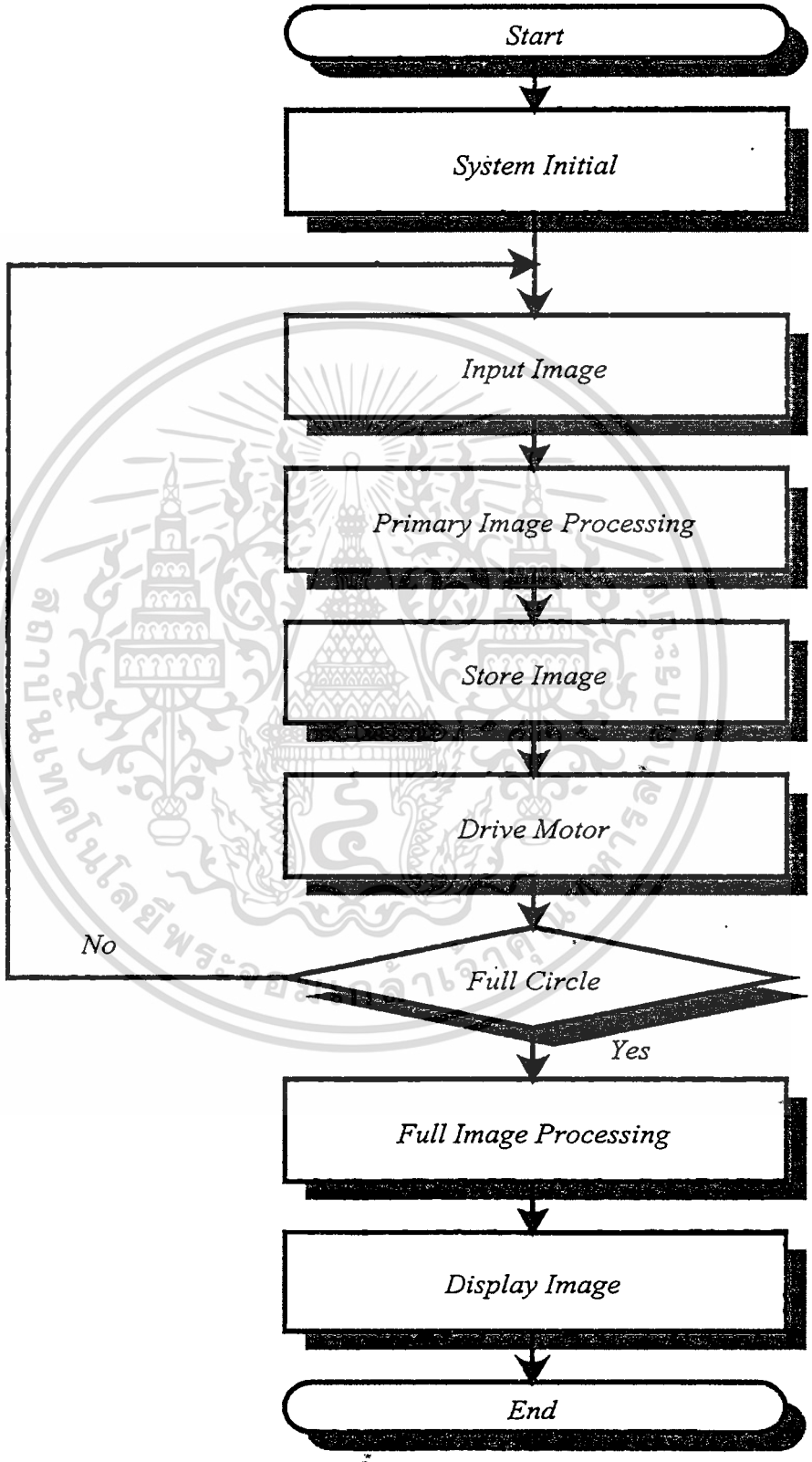


รูป 2.4 แสดงภาพที่กล้องรับมาเมื่อมีการยิงเลเซอร์



รูป 2.5 แสดงโครงสร้างของระบบโดยรวม

2.1 หลักการของระบบโดยรวม



รูปที่ 2.6 แสดงหลักการทำงานของระบบโดยรวม

1) System Initial

เป็นขั้นตอนทาง Software ที่ทำหน้าที่ติดต่อกับผู้ใช้งานโดยสามารถทำงานบน windows ได้ โดยจะให้ผู้ใช้กำหนด parameter ต่างๆของระบบดังนี้

- ความละเอียดของภาพ
- รูปแบบในการจัดเก็บภาพ
- การแก้ไขและเปลี่ยนแปลงภาพบางส่วน

ซึ่งข้อมูลจากส่วนนี้จะถูกเก็บไว้และนำไปใช้ในขั้นตอนต่อไป

2) Input Image

เป็นการรับภาพผ่านกล้อง quick cam ซึ่งถูกควบคุมด้วย software ภาพที่ได้จะเป็นภาพของวัตถุที่มุมใดๆซึ่งจะมองเห็นเฉพาะส่วนของวัตถุที่มี Laser มากระทบเท่านั้น แล้วนำภาพที่ได้ส่งต่อไปในส่วนถัดไป

3) Primary Image Process

เป็นการนำภาพที่ได้จาก Input Image ประมวลผลเพื่อเลือกเอาส่วนที่ใช้งานไว้ (ซึ่งเราต้องการเฉพาะส่วนที่เป็นเส้นของ Laser ไว้เท่านั้น) และกำจัดส่วนของภาพที่ไม่ต้องการออกไป เช่น noise หรือ รายละเอียดของภาพที่ไม่ต้องการซึ่งจะใช้ Image Processing แบบ noise rejection และ thresholding

4) Store Image

นำภาพที่ผ่านการทำ Primary Image Process ในทุกๆมุมของการหมุนกล้องมาเก็บรวบรวมเอาไว้ด้วยกัน เพื่อรอการนำไปสร้างเป็นภาพ 3 มิติ ในขั้นตอนต่อไป

5) Drive motor

จะสั่งงานโดย software ให้หมุน motor เพื่อหมุนกล้องไปตามองศากำหนดไว้ โดยจะถูกหมุนจนครบ 1 รอบ ซึ่งจะได้ image ในทุกองศาที่ต้องการ

6) Full Image Processing

เป็นของ software ที่จะนำข้อมูลภาพทั้งหมดที่ได้จากการถ่ายภาพวัตถุจนครบ 1 รอบแล้วมาสร้างเป็นภาพ 3 มิติ ซึ่งส่วนนี้ทั้งหมดจะทำในเทอมที่ 2 ดังนั้นวิธีและขั้นตอนในการเขียน software สำหรับเทอมนี้อยู่ในขั้นศึกษาและทดลอง

7) Display Image

เป็นส่วนจัดเก็บภาพ 3 มิติที่ได้จากขั้นตอนที่แล้วในรูปแบบต่างๆและแสดงภาพในแบบต่างๆ ออกมารวมทั้งการประยุกต์ใช้กับระบบกราฟฟิคอื่นๆ

บทที่ 3
ทฤษฎีพื้นฐานและหลักการทำงานของสเตปมอเตอร์

3.1 คุณลักษณะทั่วไป

สเตปมอเตอร์ คือ อุปกรณ์ที่เคลื่อนที่เป็นสเตปโดยการกระตุ้นด้วยวิธีการทางแม่เหล็กไฟฟ้า ซึ่งเปลี่ยนสัญญาณดิจิตอลอินพุทซึ่งเป็นพัลซ์ไปเป็นการเคลื่อนที่แบบอนาลอกที่เอาท์พุท สเตปมอเตอร์บางครั้งถูกเรียกว่า สเตปปิ้งมอเตอร์ หรือ สเตปเปอร์มอเตอร์ที่ได้สมญานามเช่นนี้ก็เพราะว่าสเตปมอเตอร์เป็นอุปกรณ์ซึ่งเคลื่อนที่เมื่อถูกกระตุ้นโดยโวลท์เตจหรือกระแส ซึ่งโดยมากจะเป็นไฟฟ้ากระแสตรง เอาท์พุทของสเตปมอเตอร์จะมีจำนวนเท่ากับจำนวนของคำสั่ง อินพุทซึ่งมีลักษณะเป็นพัลซ์ โดยเมื่อป้อนกระแสให้กับสเตปมอเตอร์แล้วสเตปมอเตอร์จะมีการเคลื่อนที่เพิ่มขึ้น 1 สเตป

สเตปมอเตอร์มีมาประมาณ 40 กว่าปีมาแล้ว สมัยก่อนการใช้งานสเตปมอเตอร์มีประสิทธิภาพต่ำสู้เอซีมอเตอร์และดีซีมอเตอร์ไม่ได้ แต่เมื่อไม่นานมานี้ได้มีการนำดิจิตอลคอมพิวเตอร์เข้ามาใช้จึงได้เปลี่ยนรูปแบบการควบคุมสเตปมอเตอร์ใหม่ โดยใช้การควบคุมด้วยไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์ซึ่งจะช่วยให้ใช้งานสเตปมอเตอร์ได้สะดวกขึ้นและมีประโยชน์ในการใช้สอยมากขึ้น ในปัจจุบันนี้สเตปมอเตอร์ได้ถูกนำมาใช้ในอุปกรณ์ที่ใช้ในการควบคุมเชิงเลข (NUMERICAL CONTROL) การควบคุมกระบวนการ (PROCESS CONTROL) และการควบคุมอุปกรณ์ทางเครื่องกล (MACHINE TOOL CONTROL) เป็นต้น

การควบคุมการทำงานของสเตปมอเตอร์โดยทั่วไปใช้วงจรขับ ซึ่งให้สเตปมอเตอร์ตอบสนองต่อสัญญาณพัลซ์ซึ่งทำให้มอเตอร์เคลื่อนที่เป็นจังหวะต่อเนื่องกันไป การเคลื่อนที่ลักษณะเช่นนี้ก่อให้เกิดการออสซิลเลทระหว่างสเตปของการเคลื่อนที่ทำให้สเตปมอเตอร์สั่นระหว่างใช้งาน การแก้ปัญหาดังกล่าวกระทำได้ด้วยวิธีการแด้มพ์ (DAMPING METHODS) ของสเตปมอเตอร์ ซึ่งวิธีการแก้ปัญหาตามลักษณะเฉพาะของมอเตอร์แต่ละตัว

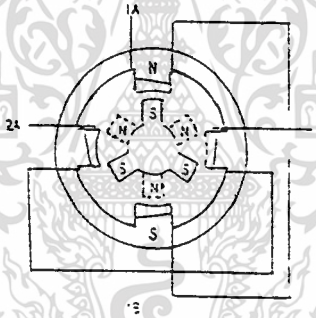
อีกแนวทางหนึ่งที่ใช้ในการแก้ไขปัญหาของการติดต่อระหว่างคอมพิวเตอร์กับมอเตอร์ ก็คือการใช้มอเตอร์ที่ง่ายต่อการควบคุมโดยคอมพิวเตอร์ สเตปมอเตอร์เป็นมอเตอร์ที่เหมาะสมสำหรับงานเช่นนี้มากเพราะ มันสามารถควบคุมให้หมุนได้โดยใช้สภาวะเพียง 2 สถานะ คือ ON และ OFF ซึ่งสัญญาณที่ว่านี้เครื่องคอมพิวเตอร์สามารถสร้างขึ้นอย่างง่ายดาย

สเตปมอเตอร์มีขดลวดอยู่หลายขด โดยที่แต่ละขดถูกควบคุมการจ่ายกระแสโดยสัญญาณ ON-OFF ทำให้สามารถควบคุมตำแหน่งในการเคลื่อนที่ได้ตราบใดที่ยังไม่เกิดการเกินกำลัง (OVERLOAD) ตราบนั้นมันยังคงรักษาตำแหน่งเดิม จนกระทั่งมีการกระตุ้นโดยสัญญาณควบคุมการจ่ายกระแสอีกครั้ง และเนื่องจากสเตปมอเตอร์สามารถควบคุมตำแหน่งได้โดยการส่งสัญญาณกระตุ้น

ทำให้ไม่จำเป็นต้องใช้ตัววัดตำแหน่ง (ENCODER) ในการทำงาน ข้อเสียที่สำคัญของ สเตปมอเตอร์ ก็คือ มีข้อจำกัดด้านความเร็วเมื่อเปรียบเทียบ DC SERVO MOTOR ที่มีขนาดเท่ากันและยังมีการสั่นสะเทือน (VIBRATION) สูงทำให้การเคลื่อนที่เป็นไปอย่างไม่ราบเรียบ

3.2หลักการทำงานของสเตปมอเตอร์

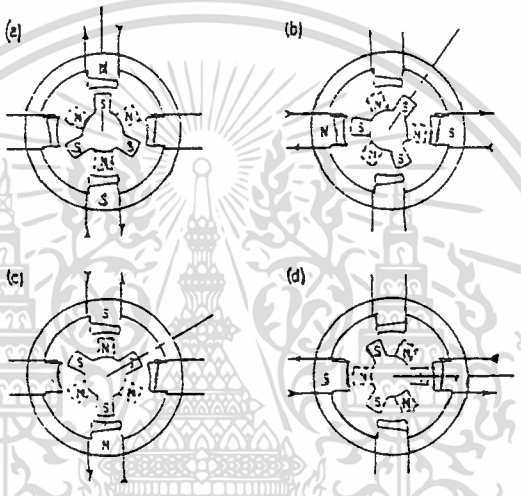
สเตปมอเตอร์สามารถแบ่งโครงสร้างทางกายภาพได้เป็น 2 ส่วน คือ สเตเตอร์ (STATOR) และ โรเตอร์ (ROTOR) ตัวสเตเตอร์เป็นส่วนที่อยู่กับที่ ประกอบด้วยขดลวดทองแดงซึ่งพันอยู่รอบแกนเหล็ก เพื่อสร้างสนามแม่เหล็กเมื่อมีการจ่ายกระแสผ่านขดลวด ส่วนโรเตอร์เป็นส่วนที่เคลื่อนที่ มีลักษณะเป็นแท่งเหล็กทรงกลม และที่ผิวรอบนอกมีลักษณะเป็นซีกฟันซึ่งทำจากแม่เหล็กถาวร ดังรูป 3.1



รูป 3.1 แสดงโครงสร้างของไฮบริดจ์สเตปมอเตอร์ที่มีจำนวนสเตปต่อรอบเท่ากับ 12

เมื่อยังไม่มีการจ่ายกระแสให้กับขดลวดของมอเตอร์ ซีกฟันอันใดอันหนึ่งของโรเตอร์จะอยู่ในตำแหน่งที่ตรงกันกับซีกฟันอันใดอันหนึ่งของสเตเตอร์ ทั้งนี้เป็นเพราะแม่เหล็กถาวรที่ตัวของโรเตอร์พยายามที่จะทำให้ค่าความต้านทานทางแม่เหล็กไฟฟ้า (RELUCTANCE) มีค่าน้อย ที่สุด ซึ่งณ .ที่จุดที่ซีกฟันของตัวโรเตอร์และสเตเตอร์ตรงกันนั้นมีค่าความต้านทานทางแม่เหล็กไฟฟ้าน้อยที่สุด ทำให้เกิดเส้นแรงแม่เหล็กไฟฟ้ามากที่สุด. และจากรูปที่ 3.1 เส้นแรงแม่เหล็กไฟฟ้าจะทำให้เกิดขั้วแม่เหล็กเหนือและใต้ขึ้นมา 2 คู่ ทั้งที่ตัวสเตเตอร์และตัวโรเตอร์ ดังรูป ค่าทอร์ค (TORQUE) ที่ทำให้ตัวโรเตอร์สามารถยึดอยู่ในตำแหน่งดังกล่าวนี้เรียกว่า ดีเท็นทอร์ค (DETENT TORQUE) (หมายความว่าการที่จะทำให้มอเตอร์เคลื่อนที่ในขณะที่ไม่ได้จ่ายกระแสให้กับขดลวดของมอเตอร์จะต้องออกแรงมากกว่าค่าของดีเท็นทอร์ค จึงจะทำให้โรเตอร์เริ่มเคลื่อนที่ได้) รูป 3.1 นั้นมี 12 ตำแหน่งที่สามารถเกิดดีเท็นท์ทอร์คได้

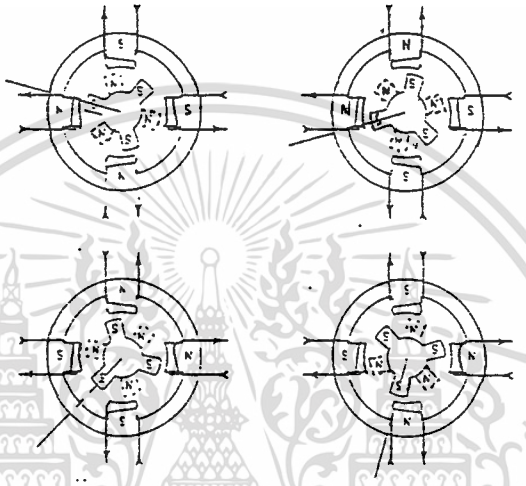
เมื่อจ่ายกระแสให้กับขดลวดที่อยู่ในสเตเตอร์คู่ใดคู่หนึ่ง ดังรูป 3.2(a) จะทำให้เกิดขั้วแม่เหล็กเหนือและใต้ที่ซีกฟันของตัวสเตเตอร์ ซึ่งจะดึงดูดซีกฟันของตัวโรเตอร์ที่มีขั้วแม่เหล็กที่มีศักย์ต่างกันที่อยู่ใกล้ที่สุดเข้าไว้ ตำแหน่งนี้เรียกว่า สเตเบิ้ลโพสซิชั่น (STABLE POSITION) ของโรเตอร์ จะมีจำนวนตำแหน่งเท่ากับจำนวนซีกฟันของโรเตอร์ และแรงที่จะทำให้โรเตอร์เปลี่ยนตำแหน่งไปจากตำแหน่งสเตเบิ้ลโพสซิชั่นได้นี้เรียกว่า โฮลดิ้ง ทอร์ค (HOLDING TORQUE)



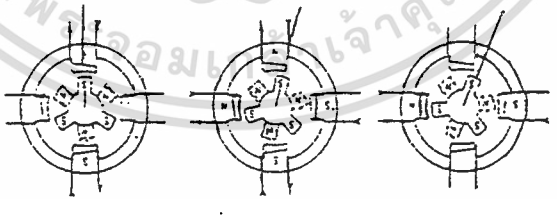
รูป 3.2 แสดงขั้นตอนการทำงานของสเตปมอเตอร์แบบ เต็มสเตปหนึ่งเฟส

เมื่อสับเปลี่ยนการจ่ายกระแสให้แก่ขดลวด จากขดหนึ่งไปอีกขดหนึ่งเนื่องจากขดลวด วางอยู่ในตำแหน่งที่ต่างกัน 90° ก็จะทำให้ตัวสเตเตอร์ดึงดูดซีกฟันของตัวโรเตอร์อีกซีกหนึ่งที่ใกล้ที่สุดเข้าไว้ ซึ่งจะทำให้โรเตอร์เคลื่อนที่ไป 1 สเตปหรือ 30° ดังรูปที่ 3.2(b) จากนั้นก็เปลี่ยนไปจ่ายกระแสให้กับขดลวดชุดแรกโดยในคราวนี้เปลี่ยนทิศทางการไหลของกระแสให้ตรงข้ามกับครั้งแรก ซึ่งจะทำให้ตัวโรเตอร์เคลื่อนที่ไปอีก 1 สเตป (เคลื่อนที่ไป 30°) ดังรูปที่ 3.2(c) หลังจากนั้นก็ไปจ่ายกระแสให้กับขดลวดชุดที่สองโดยกลับทิศทางของกระแสที่ป้อนให้อีกเช่นกัน ทำให้โรเตอร์หมุนไป 90° ดังรูป 3.2(d) และถ้าหากเราป้อนกระแสให้กับมอเตอร์เหมือนที่เราป้อนในครั้งแรกแล้ว ซีกฟันซีกถัดไปของตัวโรเตอร์จะอยู่ในตำแหน่งที่เหมือนกับในรูปที่ 3.2 (a) อีกครั้งหนึ่ง ถ้าหากเราต้องการเคลื่อนที่หนึ่งรอบ เราต้องทำการกระตุ้นให้มอเตอร์เคลื่อนที่ไปจนครบ 12 สเตป และถ้าต้องการให้โรเตอร์หมุนไปอีกทิศทางหนึ่ง ก็ทำการสลับลำดับในการจ่ายกระแส จากรูปที่ 3.2 (a) , (b) ,(c) และ (d) ไปเป็น (a) , (d), (c) และ (b) ตามลำดับ

ถ้าจ่ายกระแสให้แก่ขดลวดทั้งสองขดพร้อมๆ กัน ซีกฟันของโรเตอร์จะอยู่ ณ.ตำแหน่งระหว่างซีกฟันของสเตเตอร์ เพราะฉะนั้นการจ่ายกระแสให้แก่มอเตอร์แบบนี้จะให้ทอร์คมากกว่าแบบที่จ่ายกระแสในเวลาขณะใดขณะหนึ่งเพียงขดเดียว ดังรูปที่ 3.3 ซึ่งแสดงถึงการป้อนกระแสให้กับขดลวดแต่ละขดเพื่อให้สเตปมอเตอร์เคลื่อนที่



รูปที่ 3.3 แสดงขั้นตอนการทำงานของสเตปมอเตอร์แบบเต็มสเตปสองเฟส



รูปที่ 3.4 แสดงขั้นตอนการทำงานของสเตปมอเตอร์แบบครึ่งสเตป

แต่ถ้าในตอนแรกจ่ายกระแสให้กับขดลวดพร้อมๆกันสองขด การทำอย่างนี้สลับกันไปจะทำให้โรเตอร์เคลื่อนที่ไปสเตปละ 15° ดังรูปที่ 3.4 และในการขับเคลื่อนแบบนี้จะทำให้สเตปต่อรอบ (STEP/REV) เพิ่มขึ้นเป็นเท่าตัว การขับสเตปมอเตอร์แบบนี้เรียกว่า การขับแบบครึ่งสเตป (HALF STEPPING) ซึ่งนิยมใช้กันในงานอุตสาหกรรมเป็นอย่างมาก แม้ว่าในบางครั้งจะให้ทอร์คน้อยกว่า

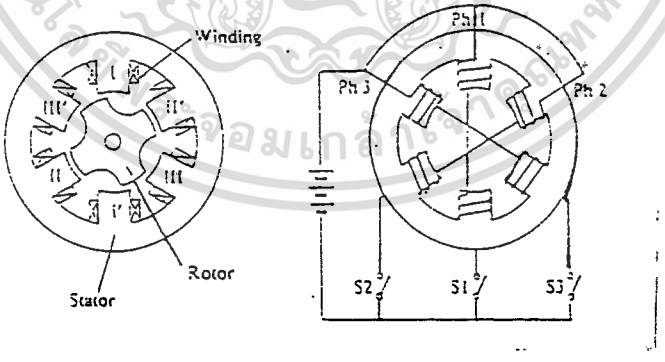
แต่มีข้อดีกว่าการขับแบบเต็มสเตป (FULL STEPPING) ก็คือ การเคลื่อนที่เป็นไปอย่างราบเรียบ (ที่ความเร็วไม่สูงนัก) อีกทั้งมีความแม่นยำสูงและการสั่นสะเทือนน้อยกว่า

3.3 ชนิดของสเตปมอเตอร์

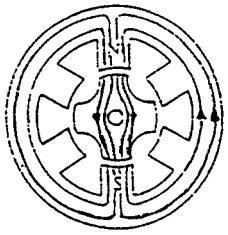
สเตปมอเตอร์สามารถแบ่งออกได้หลายชนิดตามลักษณะโครงสร้างและการใช้งาน ดังต่อไปนี้

1) สเตปมอเตอร์ชนิดปรับค่ารีลัคแตนซ์ได้ (VARIABLE RELUCTANCE STEPPING MOTOR)

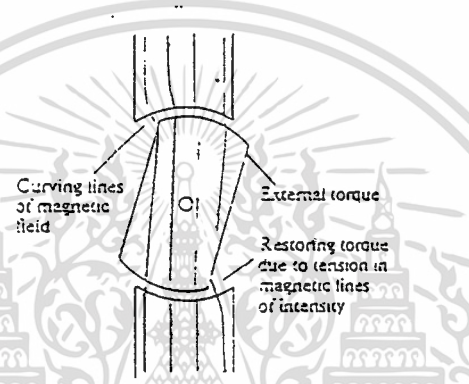
สเตปมอเตอร์ชนิดนี้สามารถปรับค่ารีลัคแตนซ์ได้ ซึ่งรูปที่ 3.5 แสดงภาพตัดขวางของ สเตปมอเตอร์แบบ 3 เฟส โดยที่สเตเตอร์มีฟันทั้งหมด 6 ซี่ ซี่ที่อยู่ตรงข้ามกันหรือทำมุม 180 ° ซึ่งกันและกันจะเป็นเฟสเดียวกัน ขดลวดที่พันอยู่ที่ฟันของสเตเตอร์ในแต่ละเฟสจะต่ออนุกรมหรือขนานก็ได้ จากรูปที่ 3.5 เป็นการต่อแบบอนุกรม ส่วนโรเตอร์นั้นมีฟัน 4 ซี่ ทั้งโรเตอร์และ สเตเตอร์ทำมาจากโลหะ ซิลิกอน ซึ่งมีสภาพซึมซับทางแม่เหล็กสูงและยอมให้สนามแม่เหล็กจำนวนมากไหลผ่านได้ ฟันของสเตเตอร์ในเฟสดียวกันจะมีขั้วต่างกันโดยซี่ฟัน I ,II , III เป็นขั้วเหนือและซี่ I′ ,II′ ,III′ เป็นขั้วใต้หลังจากถูกกระตุ้น



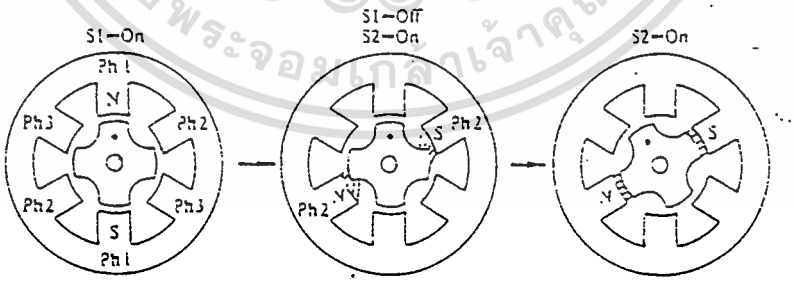
รูปที่ 3.5 แสดงภาพตัดขวางของสเตปมอเตอร์แบบ 3 เฟส



รูปที่ 3.6 แสดงตำแหน่งสมดุลย์ เมื่อเฟสใดเฟสหนึ่งของสเตปมอเตอร์ถูกกระตุ้น



รูปที่ 3.7 แสดงแรงภายนอกที่มีผลต่อเส้นแรงแม่เหล็ก



รูปที่ 3.8 แสดงขั้นตอนการเคลื่อนที่ของโรเตอร์เมื่อสเตปมอเตอร์ถูกกระตุ้น

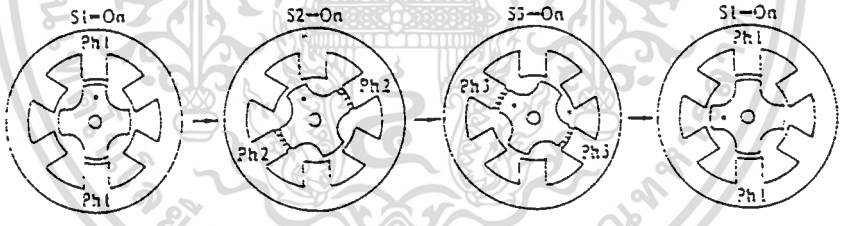
กระแสที่ไหลในแต่ละเฟสถูกควบคุมโดยสวิทช์ ปิด/ เปิด ถ้าเฟส I ถูกกระตุ้นจะมี กระแสไหลและเกิดฟลักซ์แม่เหล็กดังแสดงในรูปที่ 3.6 แกนโรเตอร์จะอยู่ตำแหน่งเดียวกับซี่ I และ I′ ทำให้ทั้งโรเตอร์และสเตเตอร์อยู่ในแนวเดียวกัน กรณีนี้จะทำให้ค่ารีลัคแตนซ์มีค่าน้อยที่สุดซึ่งเป็น

ตำแหน่งที่สมดุลย์ ถ้าโรเตอร์ถูกกระทำจากแรงภายนอกจะทำให้เปลี่ยนตำแหน่ง ดังรูปที่ 3.7 แรงบิดกระทำกับโรเตอร์ในทิศตามเข็มนาฬิกาทำให้ตำแหน่งเปลี่ยนไป มีผลทำให้เส้นแรงแม่เหล็กเคลื่อนที่จากซี่ของโรเตอร์และสเตเตอร์ เมื่อโรเตอร์และสเตเตอร์ไม่ได้อยู่ในแนวเดียวกันแล้วค่ารีลัคแตนซ์จะมีค่ามาก จากนั้นสเตปมอเตอร์จะทำตัวให้มีค่ารีลัคแตนซ์น้อยที่สุด การย้ายจากมุมที่เกิดการกระตุ้นแต่ละครั้งให้กลับไปยังตำแหน่งเดิมเรียกว่า สเตป

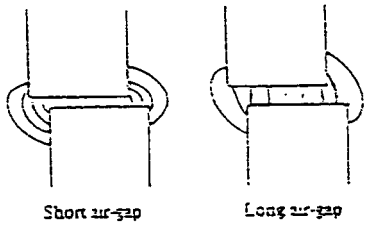
1.1 คุณสมบัติพื้นฐานของสเตปมอเตอร์แบบปรับค่ารีลัคแตนซ์

1. ช่องว่าง (AIR GAP) ต้องมีขนาดเล็กที่สุด

ช่องว่างระหว่างฟันของโรเตอร์และสเตเตอร์ต้องเล็กที่สุดเท่าที่จะเป็นได้เพื่อให้ทอร์ค ที่เกิดขึ้นมีค่ามากและมีความแม่นยำทางตำแหน่งมากขึ้น รูปที่ 3.10 แสดงการเปรียบเทียบระหว่างช่องว่างที่กว้างและแคบในขณะที่แหล่งกำเนิดสนามแม่เหล็กแหล่งเดียวกันหรือมีระดับความแรงเท่ากัน ช่องอากาศที่เล็กจะให้ทอร์คมากกว่าและทำให้เคลื่อนที่จากจุดสมดุลย์น้อยกว่าช่องว่างขนาดใหญ่เมื่อมีแรงบิดจากภายนอกมากระทำต่อโรเตอร์



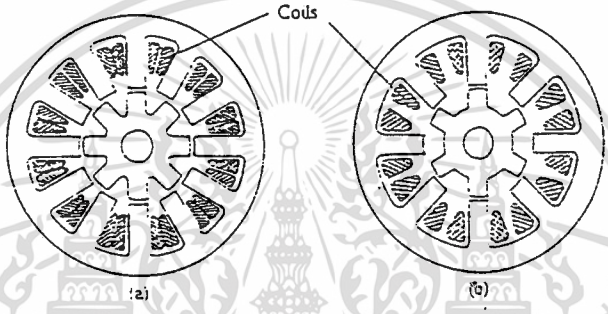
รูปที่ 3.9 แสดงขั้นตอนการเคลื่อนที่ของสเตปมอเตอร์



รูปที่ 3.10 แสดงการเปรียบเทียบเส้นแรงแม่เหล็กระหว่างช่องว่างที่กว้างและแคบ

2. มุมของการสเตปแคบ

คุณสมบัติอีกประการของสเตปคือจะต้องมีมุมของการสเตปเล็กที่สุดเท่าที่จะเป็นไปได้ มุมที่สดงในรูปที่ 3.6 ยังไม่ถือว่าเป็นมุมที่เล็ก แต่รูปที่ 3.11 (a) แสดงมอเตอร์ 3 เฟสซึ่งมีจำนวนฟันของโรเตอร์และสเตเตอร์เป็น 2 เท่าของรูปที่ 3.6 ส่วนรูปที่ 3.11(b)แสดงมอเตอร์ 4 เฟส มุมของการสเตปของทั้ง 2 โครงสร้างนี้เท่ากับ 15° นอกจากนี้ยังมีบางชนิดที่มีมุมของการสเตป 7.5° โดยฟันของโรเตอร์และสเตเตอร์มี 12 และ 16 ซี่ตามลำดับ



รูปที่ 3.11 (a) แสดงมอเตอร์ 3 เฟส และ รูปที่ 3.11 (b) แสดงมอเตอร์ 4 เฟส

ความสัมพันธ์ของมุมของการสเตป θ_s , มุมเฟส m , จำนวนซี่ฟันของโรเตอร์ N_r , จำนวนสเตป S แสดงดังสมการ

$$S = 360/\theta_s = mN_r$$

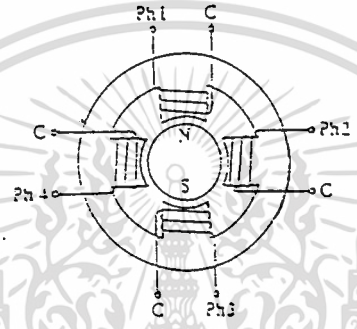
เพื่อที่จะลดขนาดของมุมที่สเตปลงต้องเพิ่มจำนวนซี่ฟันของโรเตอร์โดยที่โครงสร้างของแต่ละขั้วของเฟสใดๆ สเตเตอร์จะมีหลายซี่ฟัน แต่ก็ไม่ใช่องค์ประกอบโดยตรงที่จะกำหนดมุมของสเตปมอเตอร์

3. การสร้างสเตปมอเตอร์ให้มีโครงสร้างหลายสแตคเพื่อเพิ่มประสิทธิภาพ

โครงสร้างของสเตปมอเตอร์แบบนี้จะมี 1 เฟส โดยที่โรเตอร์และสเตเตอร์มีซี่ฟัน เหมือนกัน ซึ่งจะเป็นการเพิ่มประสิทธิภาพในด้านทอร์คต่อหน่วยปริมาตรของโรเตอร์

2) สเตปมอเตอร์แบบแม่เหล็กถาวร

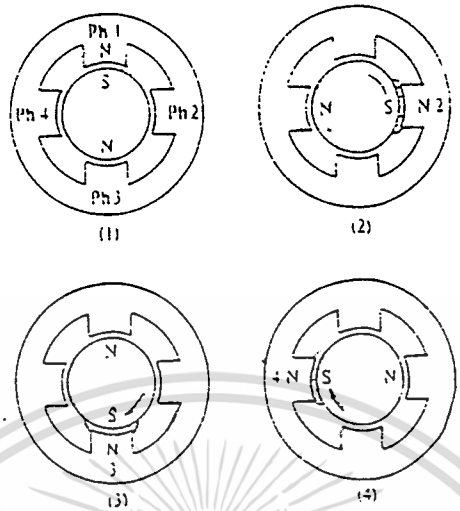
สเตปมอเตอร์ชนิดนี้ใช้แม่เหล็กถาวรเป็นโรเตอร์ และมีซี่ฟันของสเตเตอร์ล้อมรอบ ซี่ฟันของสเตเตอร์ถูกพันด้วยขดลวดสำหรับสร้างสนามแม่เหล็ก เมื่อต้องการให้สเตปมอเตอร์แบบแม่เหล็กถาวรมีขนาดมุมของสเตปเล็กลงจะต้องเพิ่มขั้วแม่เหล็กของโรเตอร์และจำนวนซี่ฟันของ สเตเตอร์ แต่ก็มีขีดจำกัดในการเพิ่มจำนวนขั้วแม่เหล็กของโรเตอร์ เนื่องจากการสร้างแม่เหล็กถาวรสร้างให้มีโครงสร้างโดยมีขั้วแม่เหล็กหลายขั้วทำได้ยาก



รูปที่ 3.12 แสดงโครงสร้างของสเตปมอเตอร์แบบแม่เหล็กถาวร

ตัวอย่างการทำงานของสเตปมอเตอร์แบบแม่เหล็กถาวร สมมติว่าสเตปมอเตอร์แบบ แม่เหล็กถาวรขนาด 4 เฟส มีโรเตอร์เป็นแม่เหล็กถาวรทรงกระบอกและสเตเตอร์มี 4 ซี่ฟันซึ่งรอบๆพันด้วยขดลวด มีรูปแบบพื้นฐานของการทำงานคือ เมื่อสร้างสัญญาณกระตุ้นตามลำดับเฟส โรเตอร์จะหมุนไปตามทิศทางของการกระตุ้น ดังแสดงในรูปที่ 3.13

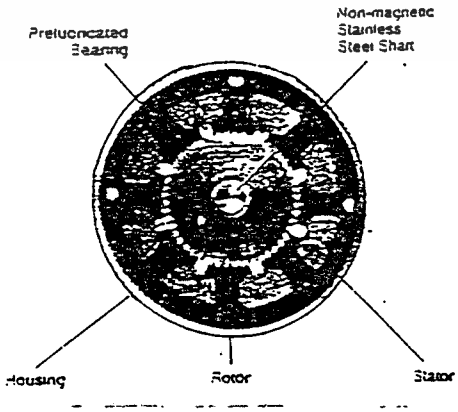
ข้อเสียของสเตปมอเตอร์แบบแม่เหล็กถาวรคือ มีขนาดมุมสเตปใหญ่ทำให้ความละเอียดของสเตปต่อรอบน้อยเนื่องจากโครงสร้างของโรเตอร์เป็นแม่เหล็กถาวร การสร้างแม่เหล็กถาวรให้มีขั้วหลายขั้วทำได้ยากทำให้ไม่สามารถสร้างสเตปขนาดเล็กได้ สเตปมอเตอร์แบบแม่เหล็กถาวรส่วนใหญ่จะมีโครงสร้างขนาดเล็ก ทำให้ค่าทอร์คที่ได้ต่อหน่วยต่อปริมาตรต่ำ ถ้าต้องการปรับปรุงประสิทธิภาพในเรื่องของทอร์ค แม่เหล็กถาวรที่ใช้ต้องทำจากสารแม่เหล็กที่มีสภาพ ความเป็นแม่เหล็กสูง



รูปที่ 3.13 แสดงการทำงานของสเตปมอเตอร์แบบแม่เหล็กถาวรขนาด 4 เฟส

3) สเตปมอเตอร์แบบไฮบริดจ์

สเตปมอเตอร์ชนิดนี้มีแกนโรเตอร์เป็นแม่เหล็กถาวร โดยมีการทำงานร่วมกันของมอเตอร์แบบแม่เหล็กถาวรและมอเตอร์แบบปรับค่ารีลัคแตนซ์ได้ ไฮบริดจ์สเตปมอเตอร์นี้มีโครงสร้างของสเตเตอร์คล้ายกับโครงสร้างของสเตปมอเตอร์แบบปรับค่ารีลัคแตนซ์ได้ แต่ต่างกันที่การต่อขดลวด โดยที่แต่ละเฟสของสเตปมอเตอร์แบบปรับค่ารีลัคแตนซ์ได้จะมีขดลวด 2 ขด แต่ละขดมีขั้วต่างกัน แต่ไฮบริดจ์สเตป มอเตอร์ขดลวดทั้งสองจะพันอยู่ที่ขั้วเดียวกัน เรียกว่า ไบโพล่าร์ (BIPOLAR) ซึ่งในการกระตุ้นแต่ละครั้งจะให้ขั้วที่แตกต่างกัน



รูปที่3.14 แสดงโครงสร้างของไฮบริดจ์สเตปมอเตอร์

3.1 คุณสมบัติที่สำคัญของไฮบริดจ์สเตปมอเตอร์

โครงสร้างของมอเตอร์จะมีแม่เหล็กถาวรอยู่ตรงกลางระหว่างเฟสทั้งสอง การเหนี่ยวนำสนามแม่เหล็กทำได้โดยใช้สนามแม่เหล็กซึ่งสร้างจากสเตเตอร์ซึ่งเป็นสนามแม่เหล็กแบบเฮทเทอโรโพลาร์ (HETEROPOLAR FIELD) ดังนั้นทอร์คเกิดจากการทำงานร่วมกันของสนามแม่เหล็ก 2 ชนิด คือ สนามจากแม่เหล็กถาวรและสนามแม่เหล็กเหนี่ยวนำที่เกิดจากการกระตุ้นของขดลวดแต่ละขด โครงสร้างของซี่ฟันของสเตเตอร์จะใหญ่กว่าซี่ฟันของโรเตอร์เล็กน้อยเพื่อเพิ่มความถูกต้องแม่นยำทางตำแหน่งของการเคลื่อนที่

หลักการทำงานของไฮบริดจ์สเตปมอเตอร์ที่แตกต่างจากสเตปมอเตอร์แบบแปรค่ารีลัคแตนซ์ได้คือแรงบิดที่เกิดขึ้นจากสนามแม่เหล็กจะไม่ขึ้นอยู่กับกระแสที่ไหลผ่านขดลวดเพียงอย่างเดียวแม่ขึ้นอยู่กับโครงสร้างของซี่ฟันด้วย ซึ่งซี่ฟันถูกออกแบบเพื่อให้ได้โครงสร้างขนาดเล็ก และใช้แม่เหล็กถาวรเป็นแกนกลางเพื่อลดผลของการออสซิเลททางแมคคานิคส์

ข้อดีของไฮบริดจ์สเตปมอเตอร์คือมีขนาดสเตปขนาดเล็กมีความละเอียดของสเตปต่อรอบสูง มีค่าทอร์คสูงกว่าสเตปมอเตอร์แบบแปรค่ารีลัคแตนซ์ได้ แต่สเตปมอเตอร์แบบแปรค่ารีลัคแตนซ์ได้มีแรงเฉื่อยทางแมคคานิคส์น้อยกว่าไฮบริดจ์สเตปมอเตอร์

นอกจากสเตปมอเตอร์ทั้ง 3 ชนิดที่กล่าวมาแล้วยังมีสเตปมอเตอร์ อื่นๆที่ไม่ได้กล่าวถึงอีกเช่น ลิเนียร์สเตปมอเตอร์ ซึ่งเป็นมอเตอร์ที่ได้รับการออกแบบให้มีการเคลื่อนที่แบบเป็นเชิงเส้น อิเล็คโตรดไฮดรอลิคสเตปมอเตอร์ซึ่งเป็นสเตปมอเตอร์กำลังสูงที่ใช้ในงานอุตสาหกรรม เป็นต้น

3.4 การ Control Step Motor

วิธีการควบคุมการทำงานของสเตปมอเตอร์โดยการกระตุ้นเฟสมี 3 วิธี คือ

1) การกระตุ้นแบบซิงเกิลเฟส (Single Phase Excitation)

เป็นการกระตุ้นแบบเฟสเดี่ยวตามสัญญาณพัลส์ที่ป้อนเข้ามาที่ชุดขับสเตปมอเตอร์ ดังนี้

PHASE	PULSE							
A	1	0	0	0	1	0	0	0
B	0	1	0	0	0	1	0	0
C	0	0	1	0	0	0	1	0
D	0	0	0	1	0	0	0	1

2) การกระตุ้นแบบสองเฟส (Two Phase Excitation)

เป็นการกระตุ้นแบบทีละสองเฟสคู่พร้อมกัน ดังนี้

PHASE	PULSE							
A	1	0	0	1	1	0	0	1
B	1	1	0	0	1	1	0	0
C	0	1	1	0	0	1	1	0
D	0	0	1	1	0	0	1	1

3) การกระตุ้นแบบครึ่งสเตป (Half Step Excitation)

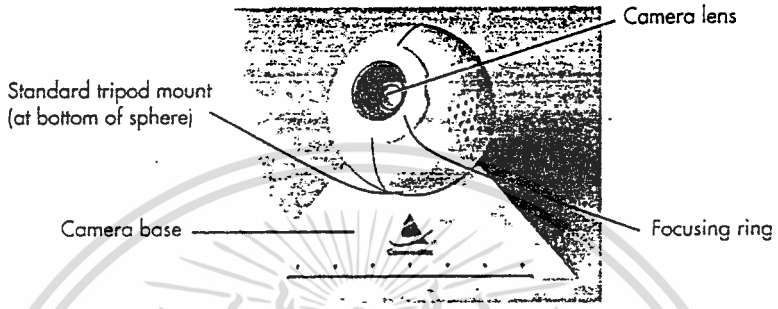
เป็นการรวมรูปแบบการหมุนของทั้งสองแบบไว้ในแบบเดียวกัน

PHASE	PULSE							
A	1	1	0	0	0	0	0	1
B	0	1	1	1	0	0	0	0
C	0	0	0	1	1	1	0	0
D	0	0	0	0	0	1	1	1

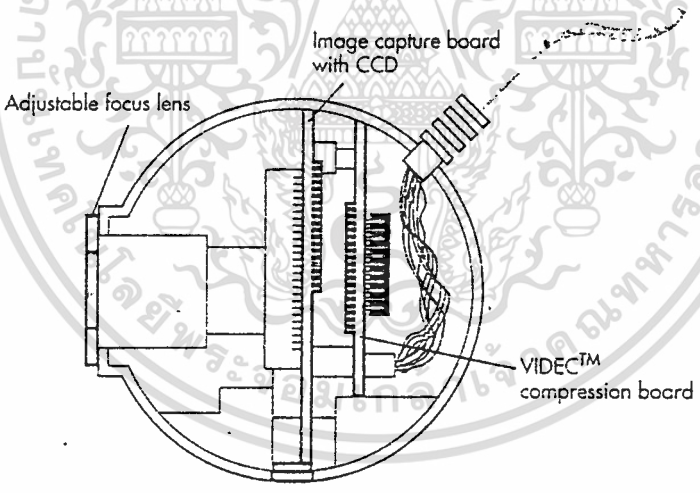
วิธีการกระตุ้นเฟสทั้ง 3 แบบนี้จะมีคุณสมบัติที่แตกต่างกัน ดังนั้นการเลือกวิธีการกระตุ้นจึงจำเป็นที่จะต้องพิจารณาเพื่อให้เหมาะสมกับงานนั้นๆ การกระตุ้นแบบเฟสเดียว จะเป็นแบบที่มีความเที่ยงตรงของตำแหน่งสูง แต่จะมีแรงบิด (Torque) น้อย ส่วนการกระตุ้นแบบ 2 เฟส มีความเที่ยงตรงของตำแหน่งน้อยกว่าแบบแรก แต่มีแรงบิดสูงกว่าและการกระตุ้นแบบครึ่งสเตปเป็นแบบที่มีความเที่ยงตรงของตำแหน่งน้อยมาก แต่จะมีแรงบิดสูงมากเช่นกัน

บทที่ 4

กล้อง Color Quick Cam



รูปที่ 4.1 แสดงลักษณะภายนอกของกล้อง Color Quick Cam



รูปที่ 4.2 แสดงลักษณะภายในของกล้อง Color Quick Cam

4.1 สิ่งที่จำเป็นต้องมีในการใช้งานกล้อง Color Quick Cam ได้แก่

- เครื่อง PC ตั้งแต่รุ่น 486 ขึ้นไป
- Microsoft Windows 3.1, Windows for Workgroups3.11, หรือ Windows 95
- หน่วยความจำ 8 MB สำหรับ Windows3.1 หรือ 12 MB สำหรับ Windows 95
- Video for Windows1.1e หรือ สูงกว่านั้น (รวมการติดตั้ง)

- พอร์ทขนาน 1 พอร์ท โดยพอร์ทเป็นแบบทิศทางเดียว(unidirectional) หรือ 2 ทิศทาง (bidirectional)
- หน่วยความจำในฮาร์ดดิสก์ 2 MB สำหรับซอฟแวร์ ของกล้อง Color Quick Cam
- หน่วยความจำในฮาร์ดดิสก์อย่างน้อย 5 MB สำหรับสร้างภาพเคลื่อนไหว
- หน่วยความจำในฮาร์ดดิสก์ 350 KB สำหรับภาพ 24 bit color(640×480พิกเซล) หรือประมาณ 80KB สำหรับภาพ 320 × 240 พิกเซลในJPEG
- ไมโครโฟนสำหรับการบันทึกเสียงในภาพแบบ movie
- Sound Card ซึ่ง compatible กับ Windows

4.2 คุณสมบัติพิเศษของกล้อง Color Quick Cam

- ให้ภาพที่มีความละเอียด 640 × 480 พิกเซล 24 bit color
- frame rate 30 fps (เปลี่ยนแปลงได้ตามขนาดของ windows และ PC)
- ได้รับไฟเลี้ยงจากคีย์บอร์ดด้วยกำลังไฟอย่างน้อย 2 วัตต์
- ปรับโฟกัสได้จาก 1 นิ้วถึงอินฟินิตี้ maximum len speed = f/1.6

4.3 ความสามารถของกล้อง Color Quick Cam

1) เก็บภาพเคลื่อนไหวใน PC

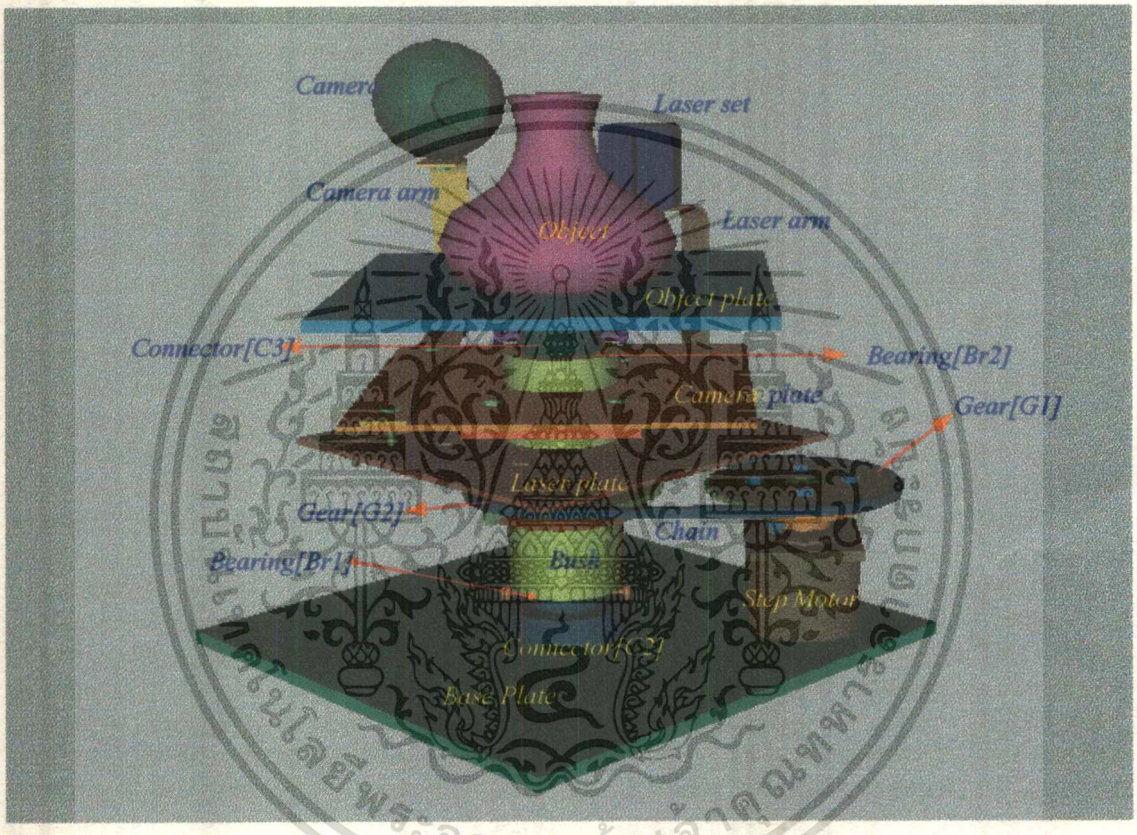
การใช้งานกล้อง Color Quick Cam นั้นเป็นวิธีที่ง่ายและประหยัดวิธีหนึ่งในการเก็บภาพเคลื่อนไหวไว้ใน PC ทำให้สามารถที่จะติดต่อสื่อสารกันโดยแสดงภาพเคลื่อนไหวได้และสามารถหลีกเลี่ยงปัญหาในขั้นตอนการติดตั้ง Hardware , การซื้อ video board ที่แพง หรือการศึกษาถึงวิธีการใช้งาน การใช้กล้อง Color Quick Cam นั้นจะช่วยให้เราสามารถที่จะ capture ภาพได้ง่ายและทำการดัดแปลงแก้ไขได้ด้วย

2) เก็บภาพนิ่งใน PC

กล้อง Color Quick Cam มี Software ที่ทำให้สามารถบันทึกภาพนิ่งให้อยู่ในรูปแบบ BMP, TIFF หรือ JPEG ซึ่งสามารถนำมาประมวลผลใน Word Processing , Page Layout , การ Present งาน หรือ โปรแกรม Graphics ซึ่งภาพนิ่งเหล่านั้นสามารถนำไปใช้ในการบอกตำแหน่ง , การใช้ในงานพิมพ์ Graphicsที่ต้องการคุณภาพสูง เช่น จดหมาย หรือ เอกสารสำคัญต่างๆ

บทที่ 5
ชุดทดลองต้นแบบ

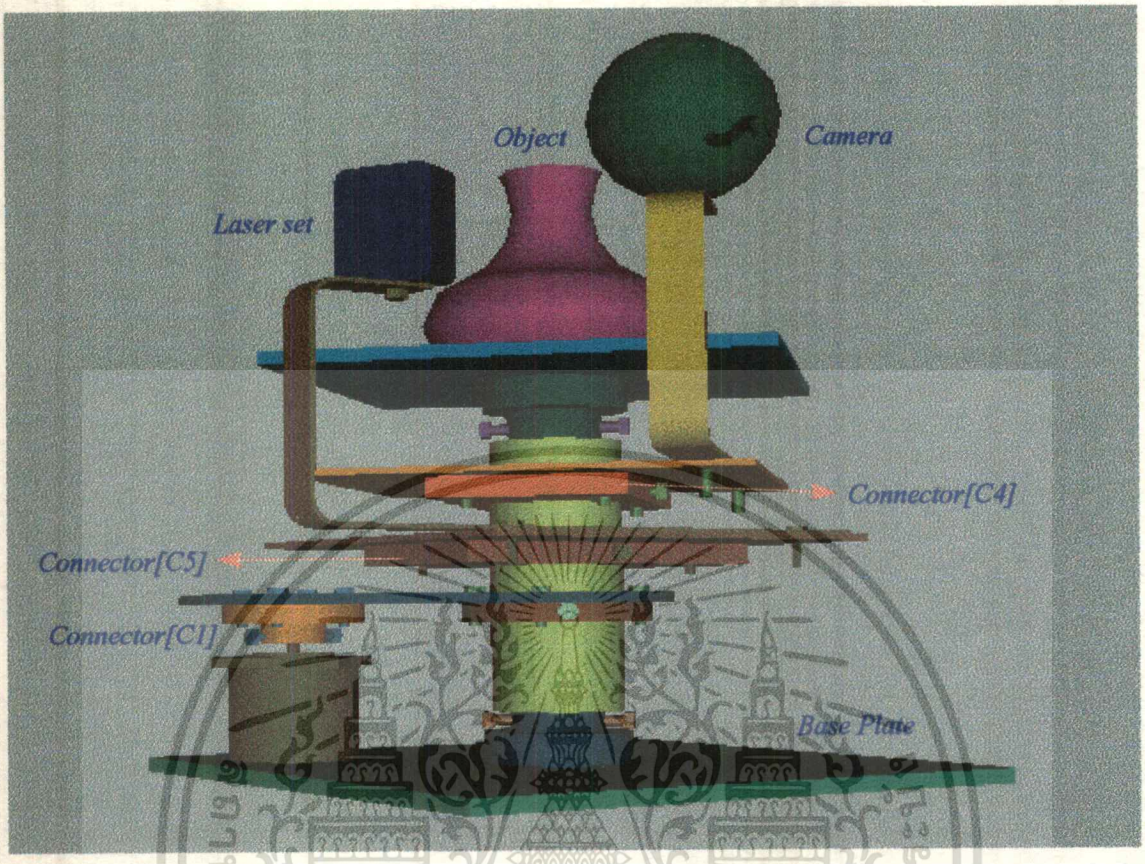
5.1 โครงสร้างและส่วนประกอบของเครื่อง 3D-SCANNER



รูปที่ 5.1 โครงสร้างและส่วนประกอบโดยรวมของชุดทดลองต้นแบบ(ด้านหน้า)

5.2 หลักการทำงาน

- ต้นกำลังหลักของระบบหมุนกล้องนี้จะเป็น Motor แบบ Stepping ซึ่งจะถูกสั่งงานจาก PC
- กำลังจาก Motor จะส่งผ่าน Gear [G1] ผ่านโซ่ (Chain)ไปสู่ Gear[G2] โดยมีอัตราทดเป็น 1:1
- ส่วนที่ไม่มีการเคลื่อนที่คือ แผ่นรองวัตถุ (Object plate) จะถูกยึดติดกับเพลากลาง (Shaft)โดยผ่าน C3
-เพลากลางจะถูกยึด ติดกับฐาน (Base Plate) ผ่าน C2 ซึ่งทำให้ส่วนนี้ไม่มีการหมุนเช่นกัน



รูปที่ 5.2 โครงสร้างและส่วนประกอบโดยรวมของชุดทดลองต้นแบบ(ด้านหลัง)

- แผ่นรองวัตถุ (Object plate) , Shaft , Base plate, C2 และ C3 นั้นจะทำหน้าที่รองรับน้ำหนักของวัตถุ นอกจากนี้ เพลากลาง(Shaft) ยังทำหน้าที่เป็นแกนกลางในการหมุนของกล้องอีกด้วย

- ปลอก (Bush) จะยึดติดกับเพลากลาง Shaft โดยใช้ Bearing [Br1 และ Br2] และสามารถหมุนได้อิสระจากเพลากลาง (Shaft)

- ปลอก[Bush] จะยึดติดกับ Gear [G2] ซึ่งจะหมุนตามการทำงานของ Motor

- ชุดขายึดกล้องจะเป็นแผ่นรอง [Camera plate] ยึดติดกับปลอก [Bush] โดยผ่านตัวยึด [C4] และแผ่นรองก็จะยึดติดกับแขนยึดกล้อง [Camera arm]

- ชุดขายึดเลเซอร์จะเป็นแผ่นรอง [Laser plate] ยึดติดกับปลอก [Bush] โดยผ่านตัวยึด [C5] และแผ่นรองก็จะยึดติดกับแขนยึดเลเซอร์ [Laser arm]

ดังนั้น ตัวกล้องและตัวเลเซอร์จะหมุนตามการทำงานของ Motor แต่วัตถุจะอยู่กับที่

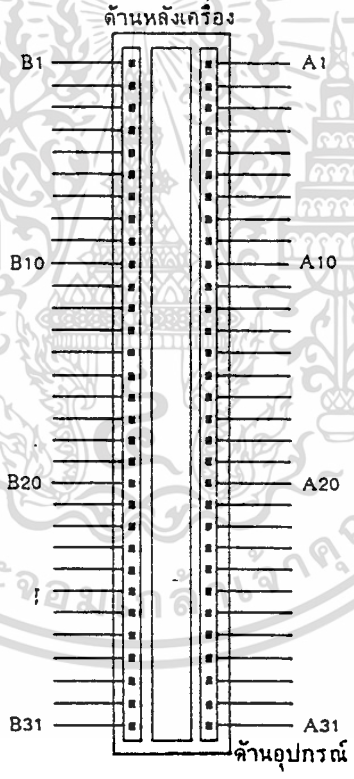
บทที่ 6

การ์ดควบคุมการหมุน Stepping Motor

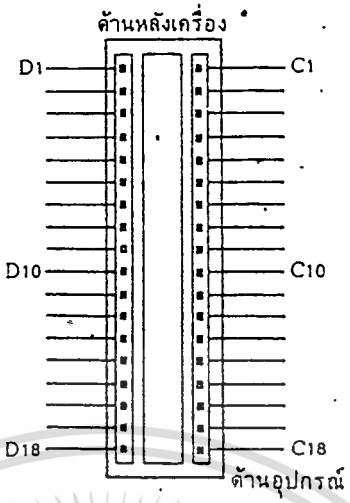
เนื่องจาก Parallel port ของ PC ถูกใช้ไปแล้วโดยกล้อง quick cam ดังนั้นการควบคุม Stepping Motor จึงจำเป็นต้องใช้ Address ของ port ใหม่ ดังนั้นจึงจำเป็นต้องสร้างการ์ดขึ้นมา

6.1 ISA slot

ที่เครื่องคอมพิวเตอร์นั้นจะมีช่องเสียบที่เรียกว่า Slot PC โดยการเอา card มาเสียบที่ Slot นี้ เพื่อเป็นการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์ภายนอก



รูปที่6.1 แสดงการนับขาของ slot แบบ 62 ขา



รูปที่ 6.2 แสดงการนับขาของ slot แบบ 36 ขา

ตารางที่ 6.1 แสดงชื่อของสัญญาณขาต่างๆของ slot

ขาอินพุท	ชื่อสัญญาณ	อินพุท/เอาท์พุท
A1	-I/O CH CK	I
A2	SD7	I/O
A3	SD6	I/O
A4	SD5	I/O
A5	SD4	I/O
A6	SD3	I/O
A7	SD2	I/O
A8	SD1	I/O
A9	SD0	I/O
A10	-I/O CH RDY	I
A11	AEN	O
A12	SA19	I/O
A13	SA18	I/O
A14	SA17	I/O
A15	SA16	I/O
A16	SA15	I/O
A17	SA14	I/O
A18	SA13	I/O
A19	SA12	I/O
A20	SA11	I/O
A21	SA10	I/O
A22	SA9	I/O
A23	SA8	I/O
A24	SA7	I/O
A25	SA6	I/O
A26	SA5	I/O

ตารางที่ 6.1 ต่อ

ขาอินพุท	ชื่อสัญญาณ	อินพุท/เอาท์พุท
A27	SA4	I/O
A28	SA3	I/O
A29	SA2	I/O
A30	SA1	I/O
A31	SA0	I/O
B1	GND	กราวน์
B3	+ 5 Vdc	แหล่งจ่ายไฟเลี้ยง
B4	IRQ9	I
B5	-5 Vdc	แหล่งจ่ายไฟเลี้ยง
B6	DRQ2	I
B7	-12 Vdc	แหล่งจ่ายไฟเลี้ยง
B8	IOWS	I
B9	+12 Vdc	แหล่งจ่ายไฟเลี้ยง
B10	GND	กราวน์
B11	-SMEMW	O
B12	-SMEMR	O
B13	-IOW	I/O
B14	-IOR	I/O
B15	-DACK3	O
B16	DRQ3	I
B17	-DACK1	O
B18	-Refresh	I/O
B19	CLK	O
B20	IRQ7	I
B21	IRQ6	I
B22	IRQ5	I

ตารางที่ 6.1-ต่อ

ขาอินพุท	ชื่อสัญญาณ	อินพุท/เอาท์พุท
B23	IRQ5	I/O
B24	IRQ4	I/O
B25	IRQ3	I/O
B26	-DACK2	O
B27	T/C	O
B28	BALE	O
B29	+ 5 Vdc	แหล่งจ่ายไฟเลี้ยง
B30	OSC	O
B31	GND	กราวน์
C1	SBHE	I/O
C2	LA23	I/O
C3	LA22	I/O
C4	LA21	I/O
C5	LA20	I/O
C6	LA19	I/O
C7	LA18	I/O
C8	LA17	I/O
C9	-MEMR	I/O
C10	-MEMW	I/O
C11	SD08	I/O
C12	SD09	I/O
C13	SD10	I/O
C14	SD11	I/O
C15	SD12	I/O
C16	SD13	I/O
C17	SD14	I/O

ตารางที่ 6.1 ต่อ

ขาอินพุท	ชื่อสัญญาณ	อินพุท/เอาท์พุท
C18	IRQ5	I/O
D1	IRQ4	I
D2	IRQ3	I
D3	IRQ10	I
D4	IRQ11	I
D5	IRQ12	I
D6	IRQ15	I
D7	IRQ14	I
D8	-DACK	O
D9	DRQ0	I
D10	-DACK5	O
D11	DRQ5	I
D12	-DACK6	O
D13	DRQ6	I
D14	-DACK7	O
D15	DRQ7	I
D16	+5 Vdc	แหล่งจ่ายไฟเลี้ยง
D17	-MASTER	I
D18	GND	กราวน์

สัญญาณที่ออกจาก Slot PC ที่ใช้กับ card มีดังนี้

A0-A11 : เป็นแอดเดรสของระบบที่ใช้ติดต่อกับหน่วยความจำและอุปกรณ์อินพุท/เอาท์พุท

D0-D7 : เป็นสัญญาณข้อมูลขนาด 8 บิทที่ใช้ติดต่อกับหน่วยความจำไมโครโปรเซสเซอร์

IOW : เป็นสัญญาณเขียนข้อมูลลงบนอุปกรณ์อินพุท/เอาท์พุท สัญญาณนี้ควบคุมจากไมโครโปรเซสเซอร์ แอดทีฟที่ 0

RES : สัญญาณนี้ใช้สำหรับรีเซ็ทระบบในขณะเปิดเครื่องหรือขณะที่แหล่งจ่ายไฟเลี้ยงขาด

AEN : อีนาเบิลแอดเดรส (เป็นเอาท์พุท)

1) ส่วนประกอบของการ์ด

- ส่วนเปรียบเทียบ Address
- ส่วนรับส่งข้อมูล
- ส่วน Drive Stepping Motor

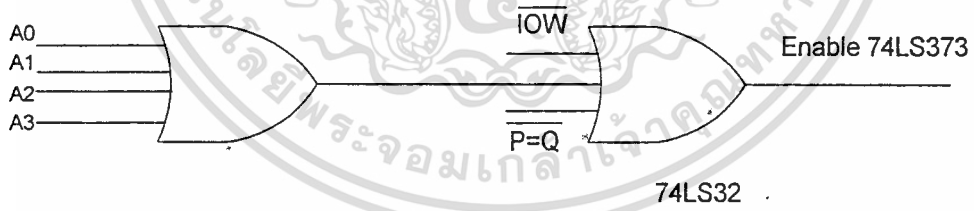
2) หลักการทำงาน

- ส่วนเปรียบเทียบ Address

จะใช้ IC 74LS688 เป็นตัวเปรียบเทียบค่า Address A4-An ของ PC กับค่า Address ที่ set ไว้ที่ DIP Switch ว่าตรงกันหรือเปล่า ถ้าตรงกันก็จะให้สัญญาณ Enable ออกไป ซึ่งตัว 74LS688 จะได้รับสัญญาณ Enable จากขา A_{EN} ของ PC โดยผ่าน Buffer 74LS245

- ส่วนรับส่งข้อมูล

จะใช้ IC 74F245ซึ่งเป็น Buffer รับข้อมูลขา DATA ของ PC แล้วส่งออกไปให้กับส่วน Drive Stepping Motor โดย 74F245 นี้จะถูก Enable โดย output ของ OR Gate ซึ่งการทำงานของ Gate ต่างๆแสดงดังรูป



รูปที่ 6.3 สัญญาณ Enable data buffer

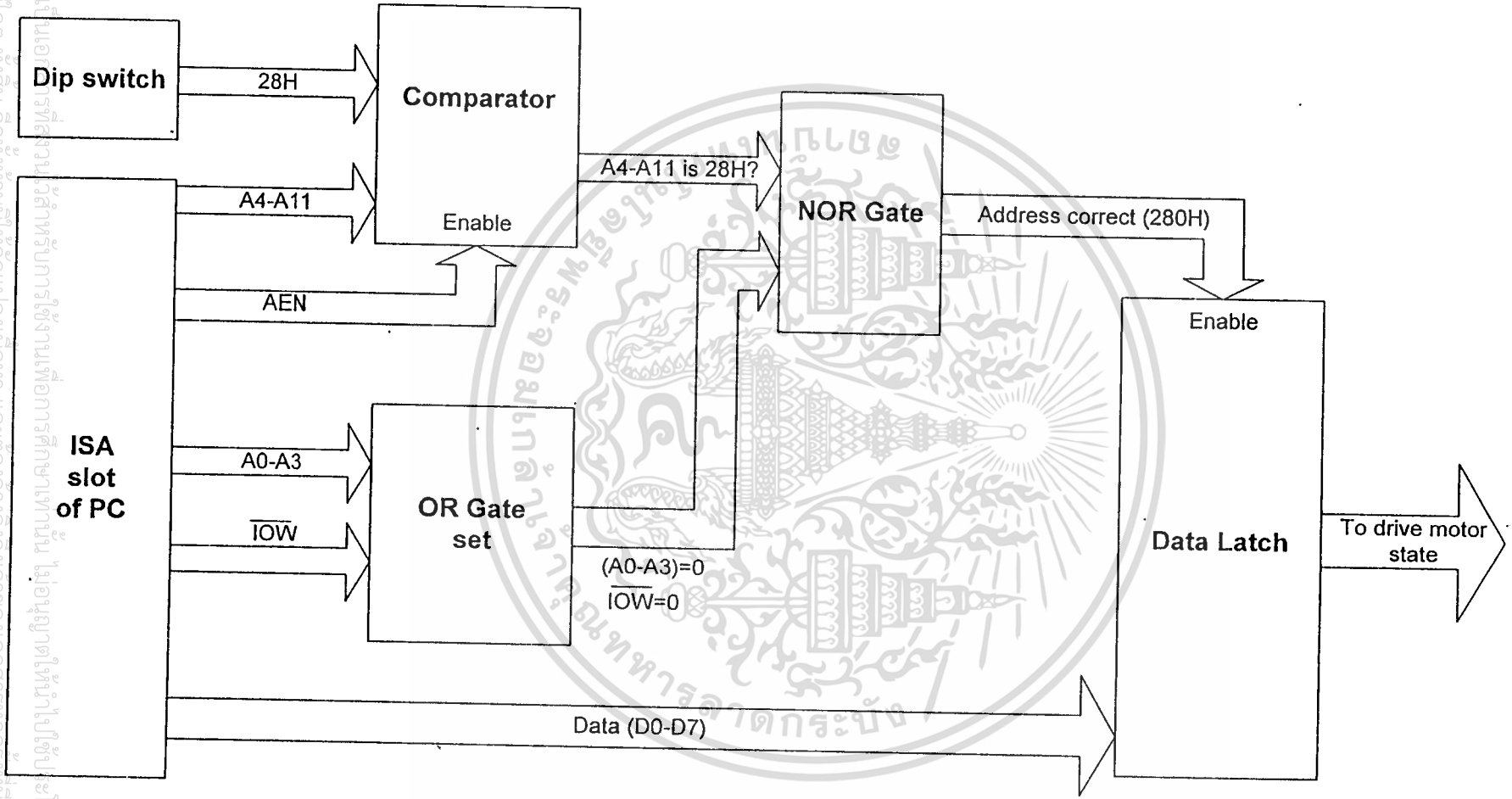
โดยที่

- A0-A3, $\overline{IOW}$ เป็นสัญญาณจาก PC โดยผ่าน Buffer 74LS245
- $\overline{P=Q}$ เป็นOutput ของส่วนเปรียบเทียบ Address มีค่าเป็น 0 เมื่อ A4-An ตรงกับค่า Address ที่ตั้งไว้ ค่า Enable จะเป็น 0 ก็ต่อเมื่อ ทั้ง3 กรณีต่อไปนี้เป็นจริง
- $\overline{IOW}$ active (เป็น0)
- $\overline{P=Q}$ active (เป็น0)
- A0-A3 active (เป็น1 ทั้งหมด)

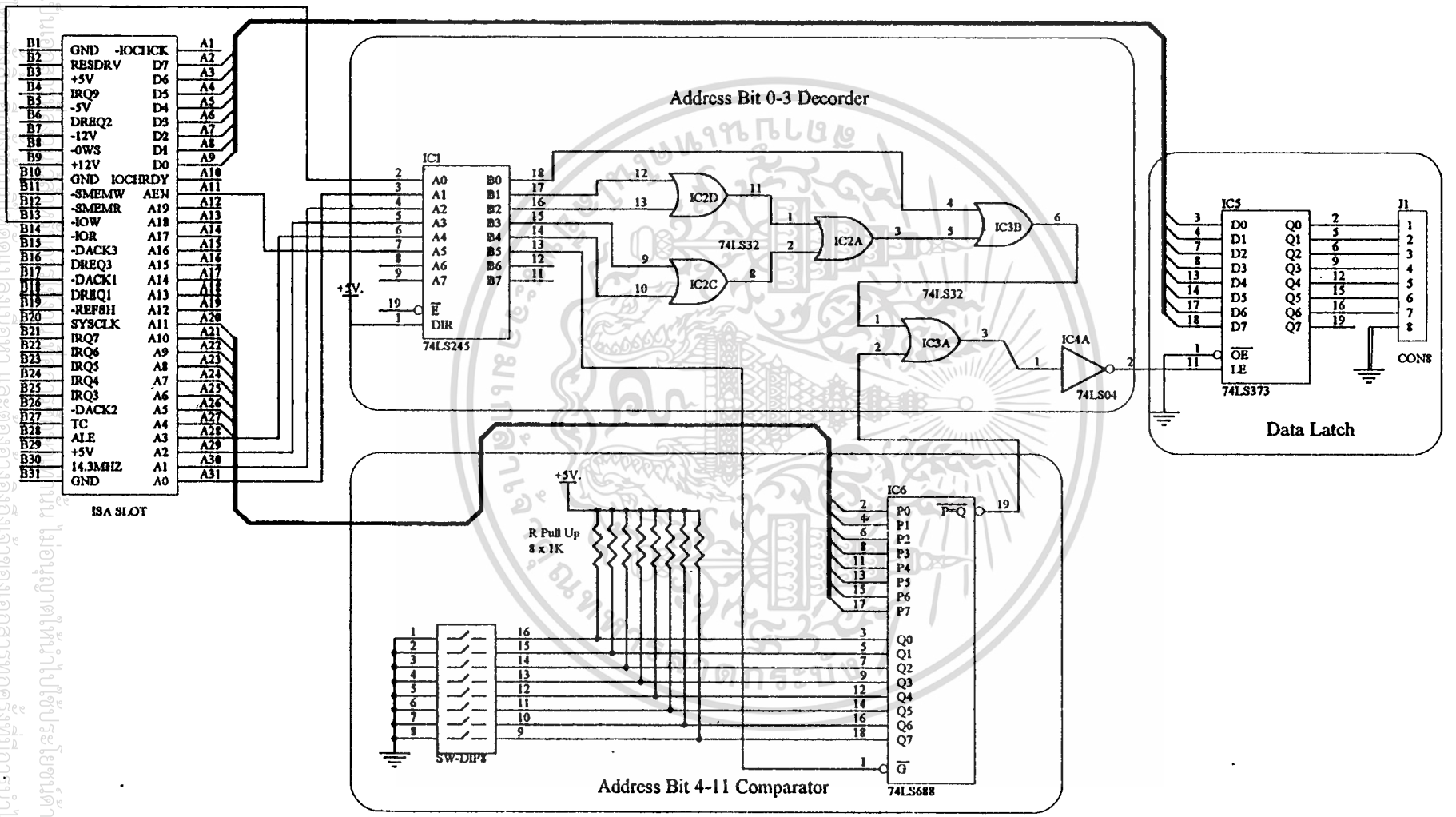
ค่า Direction ของ Buffer ทั้ง 2 ตัวคือ 74LS245 และ 74F245 ต่อกับ VCC เพราะต้องการส่งข้อมูลทางเดียวจาก A ไป B

ส่วน Drive Stepping Motor

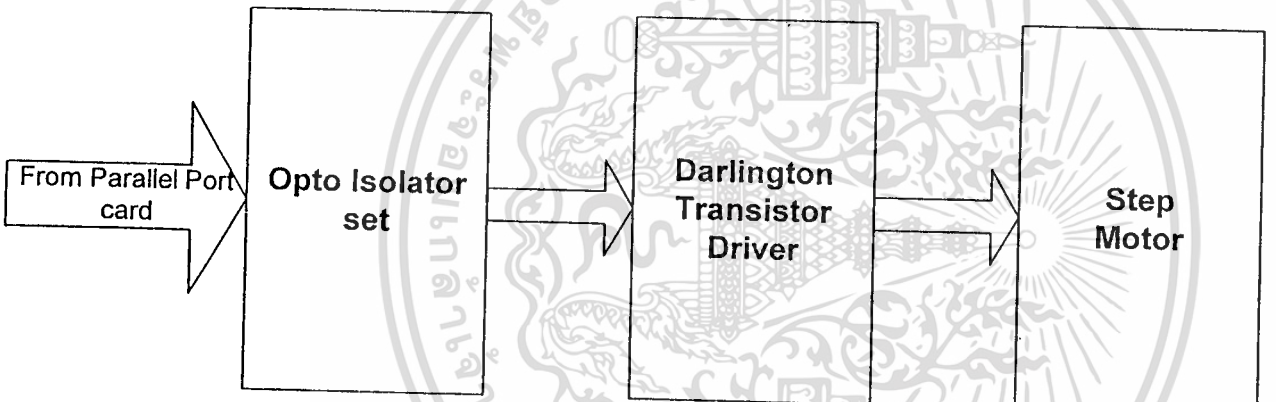
ข้อมูล 8 bit จาก PC จะถูกนำมาควบคุมการหมุนของ Stepping Motor โดย Bit 0-Bit3 ต่อเข้ากับ phase1-phase4 ของMotor ผ่าน Opto Isolator เพื่อถยกสัญญาณ 5 V จาก PC กับแรงดันที่ใช้ขับ Stepping Motor ซึ่งจะมีค่าประมาณ 28 V นอกจากนี้จะมี LED แสดงสถานะon/off ในแต่ละ phase อีกด้วย สัญญาณที่ผ่าน Opto Isolator จะป้อนให้กับชุด Darlington ของ Transistor เบอร์ 2222 และ Power Transistor เบอร์ 6123 เพื่อขยายกระแสให้เพียงพอที่จะนำไปขับ Motor ในแต่ละเฟส ให้มี Diode IN4001 เป็น Free Wheeling



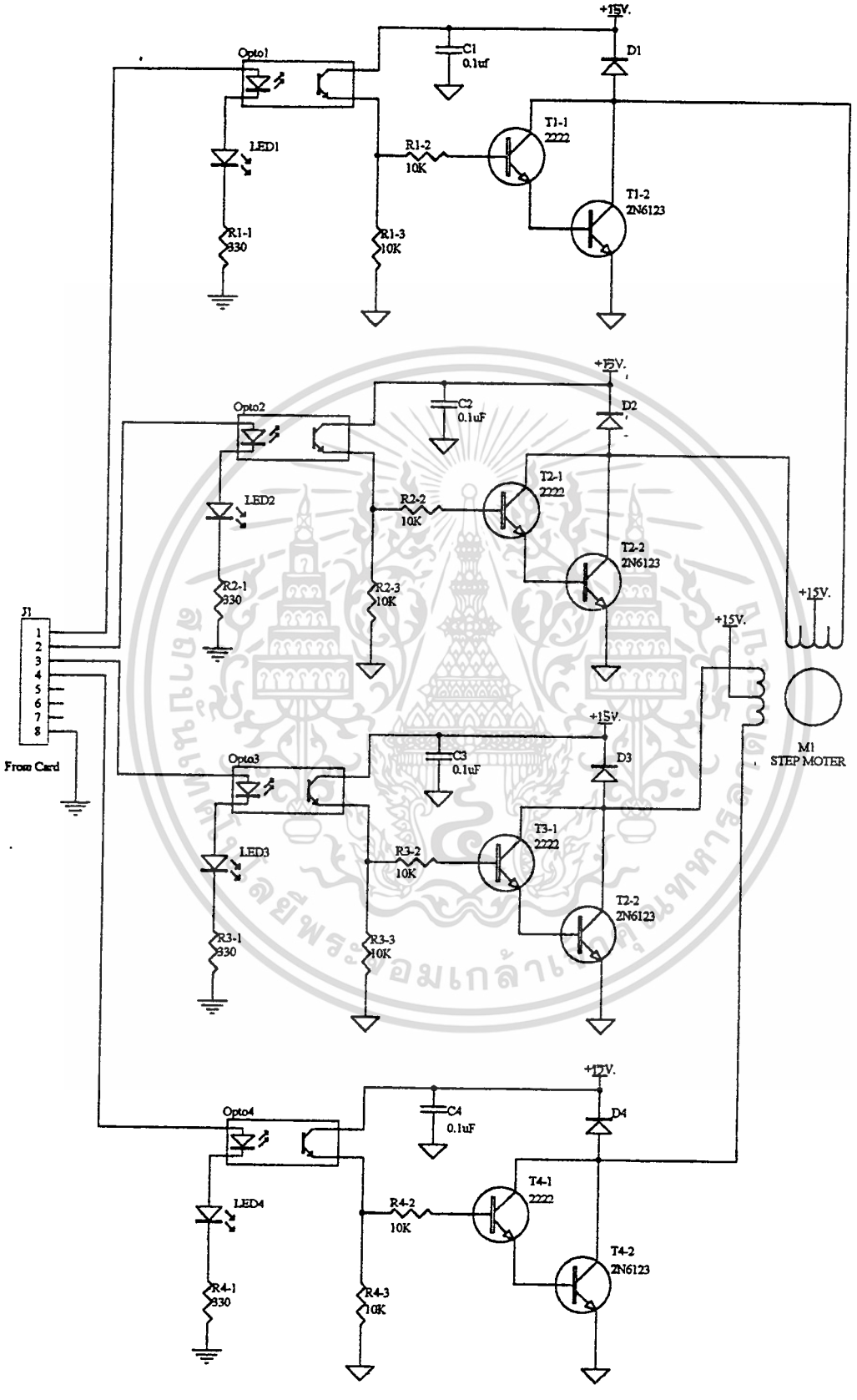
รูปที่ 6.4 รูปแสดง Block Diagram การทำงานของวงจรการ์ดพอร์ทขนาน



รูปที่ 6.5 แสดงวงจรการ์ดพอร์ทขนาน



รูปที่ 6.6 รูปแสดง Block Diagram การทำงานของวงจร Drive Step Motor



รูปที่ 6.7 แสดงวงจร Drive Step Motor

บทที่ 7
คณิตศาสตร์ในการสร้างโครงสร้าง 3 มิติ

7.1 Two – Dimentional Coordinate Geometry

ในงานด้าน Computer Graphics นั้น รูปร่างและขนาดของวัตถุที่แสดงเป็น 2 มิติจะต้องใช้ระบบ coordinate ซึ่งปกติจะใช้ x ,y ในระบบ cartesian coordinate ดังนั้นการ transform รูปแบบต่างๆในทางเรขาคณิต จึงได้ถูกนำไปใช้เป็น model สำหรับการ shift , resize หรือ reorient coordinate เหล่านั้น

พื้นฐานของ model 2 มิตินี้ คือ จุด ยกตัวอย่างเช่น เส้นก็คือจุด 2 จุดต่อกัน หรือ ระนาบจะประกอบไปด้วยจุดหลายๆจุดที่อยู่บนระนาบ ดังนั้นในระบบ 2 มิติจะถูกกำหนดขึ้นด้วยกลุ่มของ coordinate x , y หรือ จุดใดๆนั่นเอง

เราจะทำการสมมติรูป 3 เหลี่ยมในระนาบ 2 มิติซึ่งสามารถเขียนเป็น matrix ขนาด 3×2ได้ดังนี้

$$[P] = \begin{bmatrix} x1 & y1 \\ x2 & y2 \\ x3 & y3 \end{bmatrix} \qquad (7.1)$$

ซึ่ง x และ y นี้เป็นเวกเตอร์ตำแหน่งที่มีความสัมพันธ์กับระบบcoordinate แบบ cartesian ถ้าเราใช้ระบบ coordinate แบบ cartesian แล้วเราจะพบกับข้อจำกัดในการ transform ซึ่งในการ transform เราจะต้องเกี่ยวข้องกับ operation การคูณ ซึ่งมีข้อจำกัดในเรื่องขนาดของ matrix และเงื่อนไขของเวกเตอร์ ดังนั้นเราจึงหลีกเลี่ยงปัญหาเหล่านี้ด้วยการ transform โดยใช้ระบบ coordinate แบบ homogeneous เพื่อความเข้าใจระบบ coordinate แบบ homogeneous แล้วเราจะสมมติจุด p(x,y) ให้อยู่ในระนาบ 3 มิติ ระยะห่างของจุด p(x,y) กับจุด origin จะถูกแสดงโดยพารามิเตอร์ h ดังนั้นเราสามารถเขียนสมการได้คือ

$$P(x, y, z) = P(hx, hy, h) \qquad (7.2)$$

ซึ่งเราจะมองจุด p(x ,y) นี้ในลักษณะของระนาบ 3 มิติโดย h ก็จะขึ้นค่าของ z นั่นเอง เพราะฉะนั้นจุด2มิติใดๆก็สามารถแสดงเป็นจุดในระนาบ3มิติได้ซึ่งเราเรียกระนาบนี้ว่า

homogeneous แต่ยกเว้นในกรณีที่จุดนั้นเป็นจุด origin (ซึ่งค่า h=0) ตัวอย่างเช่นเมื่อมีจุดP(2,4)ใน cartesian coordinateจะทำให้เกิดจุดP(4,8,2) , P(6,12,3) , P(2,4,1) , P(m,n,h) ใน homogeneous coordinate ดังนั้นถ้าเรามีจุดต่างๆเหล่านี้จะทำให้เราสามารถหาจุดใน cartesian coordinate ได้คือ P(m/h , n/h , 1)

โดยที่ $x = m/h$

$y = n/h$ (7.3)

เมื่อใช้ homogeneous coordinate แล้วต่างๆในระนาบ 2 มิติ จะถูกแสดงในรูปของ matrix ขนาด [n × 3] ซึ่ง n = จำนวนของจุดของวัตถุ ดังนั้น

$$[P] = \begin{bmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \end{bmatrix} \quad (7.4)$$

7.1.1) การ Scaling 2 มิติ

การทำ Scaling คือ การทำให้วัตถุมีขนาดที่เพิ่มขึ้น การ Scaling ในทิศทาง x และ y ด้วยค่าคงที่นั้นจะเกิดการเปลี่ยนแปลงความยาวเกิดขึ้น ถ้ามีขนาดมากกว่า 1 จะเป็นการขยาย แต่ถ้าน้อยกว่า 1 จะเป็นการลดขนาด แต่ไม่ว่าจะเป็นกรณีใดจะต้องเป็นค่าบวก เพราะถ้าเป็นค่าลบแล้วจะเกิดการ reflection ขึ้น

การ Scaling จุด P(x,y) เป็น P(x*,y*) แสดงได้ดังสมการ (7.5)

$$x^* = x\,(Sx)$$

$$y^* = y\,(Sy) \quad (7.5)$$

หรือแสดงในรูป matrix ได้ดังสมการ (7.6)

$$[x^* \; y^* \; 1] = [x \; y \; 1] \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7.6)$$

ในการ Scaling นี้เป็นการ Scaling ที่ใช้จุด origin เป็นจุดอ้างอิง ซึ่งแต่ละจุดบนวัตถุนั้นจะเปลี่ยนขนาดหรือทิศทางก็ตามต้องขึ้นอยู่กับจุด origin เสมอ

7.1.2 การ Translation 2 มิติ

การ translation นั้นจะทำให้วัตถุเปลี่ยนตำแหน่งไปในทิศทางตามค่าที่กำหนดไป ซึ่งสามารถแสดงให้เห็นตามสมการ (7.7)

$$\begin{aligned} x^* &= x+Tx \\ y^* &= y+Ty \end{aligned} \qquad (7.7)$$

แสดงในรูป matrix ตามสมการ (7.8)

$$[x^* \;\; y^* \;\; 1] = [x \;\; y \;\; 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix} \qquad (7.8)$$

7.1.3 การ Rotation 2 มิติ

การ rotation นั้นจะใช้สำหรับการมองวัตถุในทิศทางต่างๆกัน ในการ rotation จะต้องใช้ origin เป็นจุดอ้างอิงเช่นกัน ตามค่าของมุม θ ที่กำหนดไป ในการ rotate จุด P(x,y) เป็น P(x*,y*) ได้ดังสมการ (7.9)

$$\begin{aligned} x &= r\cos\alpha \\ y &= r\sin\alpha \end{aligned} \qquad (7.9)$$

ซึ่งมุม α เป็นค่าพารามิเตอร์ที่แทนค่ามุมที่กวาดไปในทิศทางตามแกน x

$$\begin{aligned} x^* &= r\cos(\alpha+\theta) = r\cos\alpha\cos\theta - r\sin\alpha\sin\theta \\ y^* &= r\sin(\alpha+\theta) = r\sin\alpha\cos\theta + r\cos\alpha\sin\theta \end{aligned} \qquad (7.10)$$

เมื่อแทนค่า x , y จากสมการ (7.9) จะได้สมการดังนี้

$$\begin{aligned} x^* &= x\cos\theta - y\sin\theta \\ y^* &= x\sin\theta + y\cos\theta \end{aligned} \qquad (7.11)$$

หรือแสดงในรูป matrix ได้ดังสมการ (7.12)

$$[x^* \;\; y^* \;\; 1] = [x \;\; y \;\; 1]\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (7.12)$$

7.2 Three - Dimentional Coordinate Geometry

ในการ design งานทางด้านวิศวกรรมส่วนมากจะเกี่ยวข้องกับวัตถุที่เป็น 3 มิติ โดยปกติรูปทรงหรือรูปแบบจำนวนมากจะถูกใช้งานในการสร้างรูปทรง 3 มิติ ซึ่งสามารถแสดงออกมาบนหน้าจอ display เช่น ผิว poligon surface หรือ framework ของเส้น สำหรับการใช้งานด้านอื่นๆ เช่น การออกแบบเครื่องจักรกล หรือ body ของเครื่องบิน เป็นการออกแบบที่พิเศษซึ่งถูกใช้เพื่อพัฒนารูปแบบของพื้นผิว 3 มิติ จนกระทั่งได้เงื่อนไขที่พอใจ

เมื่อใดที่วิธีการเหล่านี้ถูกใช้งาน วัตถุ 3 มิติก็จะถูกสร้างขึ้นในระบบ coordinate 3 มิติ และจะทำการ map ไปในระบบ 2 มิติเพื่อที่จะทำการ display การสร้างภาพของวัตถุจำเป็นต้องใช้การ transform coordinate 3 มิติ ซึ่งแสดงได้โดย matrix 4×4 ดังนี้

$$[P] = \begin{bmatrix} 0 & 0 & 0 & 1 \\ x_B & 0 & 0 & 1 \\ 0 & y_C & 0 & 1 \\ 0 & 0 & z_D & 1 \end{bmatrix} \qquad (7.13)$$

7.2.1) การ Transformation 3 มิติ

การ transform ในระบบ 3 มิติ จะถูกใช้ด้วยวิธีการเดียวกับที่ใช้ในระบบ 2 มิติดังที่อธิบายในส่วนการ Transformation 2 มิติ ข้างต้น แต่จะเพิ่ม coordinate ของ z เข้ามาด้วย ซึ่งในระบบ 2 มิติจะประกอบไปด้วย coordinate x และ y เท่านั้น

ในระบบ homogeneous coordinate จุดต่างๆในระนาบ 3 มิติก็จะถูกแสดงเป็น matrix ขนาด 4×4 ดังนี้

$$\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ j & k & l & s \end{bmatrix} \qquad (7.14)$$

7.2.2) การ Scaling 3 มิติ

ในการทำ scaling transform จะทำได้โดยการใส่ค่าไปในแนวทแยงมุมของ matrix ที่ใช้ในการ transform ขนาด 4×4 ดังเช่นการ scaling จุด P(x,y ,z,1) ไปเป็น P(x*,y*,z*,1) ได้สมการดังนี้

$$[x^*, y^*, z^*, 1] = [x, y, z, 1]\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.15)$$

ด้วยวิธีการนี้เราจะทำการ scaling โดยใช้จุด origin เป็นจุดอ้างอิงและใช้หลักการเช่นเดียวกับการทำ scaling ใน 2 มิติ ถ้าตัวแปรในการทำ scaling คือ a , e , i มีค่าที่ไม่เท่ากันแล้วทำให้ภาพและขนาดของวัตถุที่ได้จะผิดเพี้ยนไป แต่ ถึงแม้จะเกิดการเปลี่ยนแปลงขนาดเกิดขึ้นแต่จุด origin ยังเหมือนเดิม

$$[x^*, y^*, z^*, 1] = [x, y, z, 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & s \end{bmatrix}$$

$$= [x, y, z, s] \quad (7.16)$$

จากวิธีการดังกล่าว เมื่อเราใช้การ scaling วิธีนี้แล้วจะสังเกตเห็นว่าใน column ที่ 4 ของ matrix ตรงจุดที่ใช้ในการ transform อาจจะไม่เท่ากับ 1 ก็ได้ ดังเช่นหลักการของ 2 มิติที่ได้กล่าวแล้วข้างต้น ซึ่ง matrix ที่ใช้นี้อาจถูกเปลี่ยนแปลงได้เพื่อให้ค่าของ x , y , z นี้เป็นค่า coordinate ที่เป็นมาตรฐาน ซึ่งก็คือระบบ cartesian coordinate ดังนั้นสมการ (7.16) จะเขียนได้ดังนี้

$$[x, y, z, s] = [x/s, y/s, z/s, 1] \quad (7.17)$$

เมื่อค่า s มีค่ามากกว่า 1 จะเป็นการลดขนาดของวัตถุลง แต่เมื่อต้องการที่ขยายขนาดของวัตถุต้องใช้ค่า 1/s ใน matrix ที่ทำ scaling ซึ่งก็จะได้ค่าของ coordinate สุดท้ายเป็น

$$[sx, sy, sz, 1] \quad (7.18)$$

7.2.3) การ Translation 3 มิติ

จากสมการ (7.18) Matrix ที่ใช้ในการ transform นี้จะทำการ translate หรือเคลื่อนย้ายจุด (x,y,z) เป็นจุดใหม่ (x*,y*,z*) โดยใช้ (j,k,l)

$$[x^*, y^*, z^*, 1] = [x, y, z, 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ j & k & l & 1 \end{bmatrix} \quad (7.19)$$

ค่าของ j , k , l นี้ จะแสดงความสัมพันธ์ของการเคลื่อนย้ายจุดไปตามทิศทางของ x , y , z ยกตัวอย่างเช่นเราจะทำการ translate รูปลูกบาศก์ขนาด 1 หน่วยซึ่งมีจุดยอดจุดหนึ่งอยู่ที่ตำแหน่ง origin ไปที่ตำแหน่งใหม่ในระนาบที่เราพิจารณา โดยใช้ matrix ที่ใช้สำหรับการทำ translation ดังสมการ (7.19) แต่ละจุดยอดจะถูก translate ไปตามค่าที่กำหนดใน matrix ดังนั้นค่า coordinate ที่ถูก transform ของลูกบาศก์นี้จะมีค่าดังสมการ (7.20)

$$[P^*]_{CUBE} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ J & K & L & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 & 1 \\ 3 & 2 & 0 & 1 \\ 3 & 2 & 1 & 1 \\ 3 & 2 & 1 & 1 \\ 2 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 \\ 3 & 3 & 0 & 1 \\ 2 & 3 & 0 & 1 \end{bmatrix} \quad (7.20)$$

7.2.4) การ Rotation 3 มิติ

การ Rotation ในระบบ 3มิติเป็นเรื่องที่สำคัญสำหรับการทำความเข้าใจรูปร่างของวัตถุ หรือการพิจารณางานที่ถูกออกแบบมาในมุมที่แตกต่างกัน ซึ่งเป็นสิ่งที่ซับซ้อนมากกว่าการ rotation ในระบบ2 มิติ เพราะในระบบ 3 มิติต้องระบุแกนในการทำ rotation ด้วยในขณะที่ระบบ 2 มิติใช้เพียงจุดในการทำ rotation เท่านั้น ในการทำ ratation ตามแกนใดๆก็ตาม เราสามารถที่จะคิดอย่างง่ายๆ โดยคิดแยกเป็นการหมุนตามแกนหลักทั้ง 3 แกนคือ แกน x , y และ z แล้วนำมาสร้างตามวิธีการเดียวกันกับที่ใช้ในการ rotation 2 มิติ ทำให้เราได้ matrix สำหรับการ rotation ตามแกน z ดังสมการ (7.21)

$$[T_R]_z^{\theta} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.21)$$

ซึ่งจะได้ผลลัพธ์ของการ mapping ดังสมการ (7.22)

$$\begin{aligned} x^* &= x\cos\theta - y\sin\theta \\ y^* &= x\sin\theta + y\cos\theta \\ z^* &= z \end{aligned} \quad (7.22)$$

ในวิธีการเดียวกันเมื่อเราทำการ rotate เป็นมุม θ ไปตามแกน y จะได้สมการ (7.23)

$$[T_R]_Y^{\theta} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.23)$$

ซึ่งจะได้ผลลัพธ์ของการ mapping ดังสมการ (7.24)

$$\begin{aligned} x^* &= x\cos\theta + z\sin\theta \\ y^* &= y \\ z^* &= -x\sin\theta + z\cos\theta \end{aligned} \quad (7.24)$$

เมื่อ rotate ไปตามแกน x จะได้ผลดังสมการ (7.25)

$$[T_R]_x^{\theta} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.25)$$

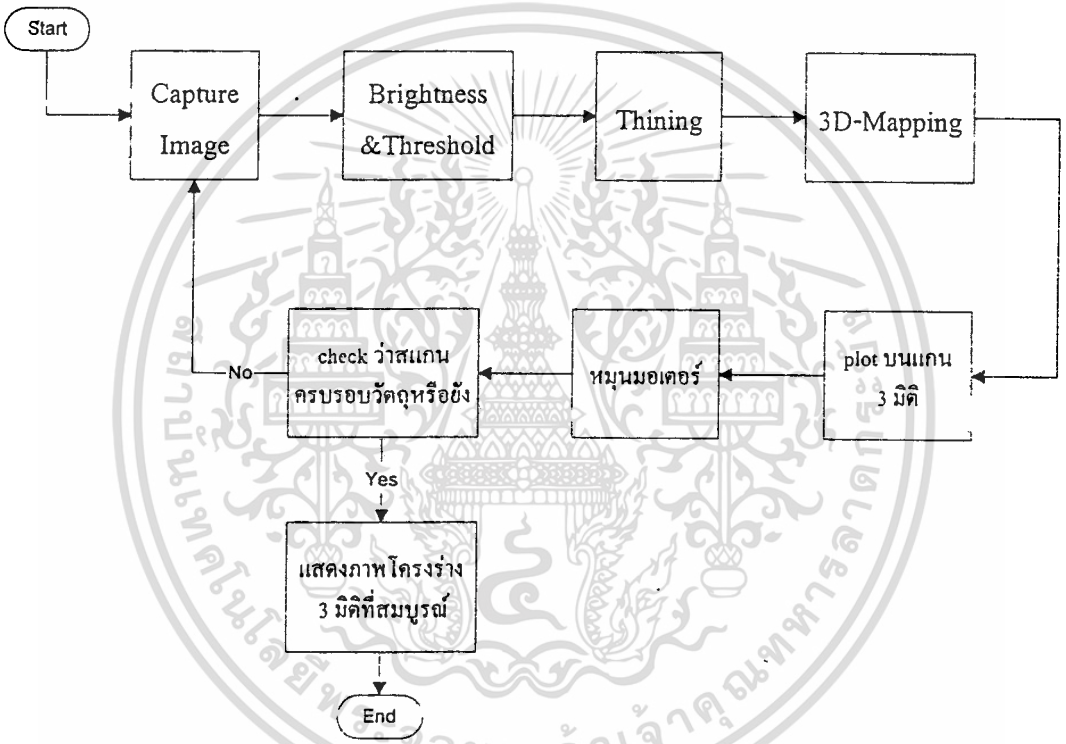
ซึ่งจะได้ผลลัพธ์ของการ mapping ดังสมการ (7.26)

$$
\begin{aligned}
x^* &= x \\
y^* &= y\cos\theta - z\sin\theta \\
z^* &= y\sin\theta + z\cos\theta
\end{aligned} \qquad (7.26)
$$

ข้อสังเกตที่สำคัญคือเครื่องหมายในเทอม sine ใน matrix $[T_R]_Y^{\theta}$ ซึ่งจะมีค่าตรงกันข้ามกับ matrix $[T_R]_Z^{\theta}$, $[T_R]_X^{\theta}$ เนื่องจากผลลัพธ์ของการใช้กฎมือขวา

บทที่ 8
โปรแกรมควบคุมการทำงานของ 3D-SCANNER

8.1 โครงสร้างการทำงานของ 3D-SCANNER



รูปที่ 8.1 Block diagram การทำงานของ 3D-SCANNER

การทำงานของ 3D-SCANNER ในแต่ละขั้นตอนหลักๆมีดังนี้

1) Capture Image

คือขั้นตอนการรับภาพจากกล้องวิดีโอมาเก็บไว้ใน memory เพื่อไปทำการประมวลผลในขั้นต่อไป

2) Brightness & Threshold

เป็นการนำภาพที่ได้จากขั้นตอนแรก ซึ่งเป็นภาพสีจริงมาแปลงให้เป็นภาพที่อยู่ในรูปแบบของความสว่าง แล้วทำการ Threshold คือ เลือกเอาเฉพาะส่วนที่มีความสว่างอยู่ในช่วงที่เราต้องการ

3) Thining

เป็นการนำภาพที่ได้จากการ Threshold มาทำให้ส่วนที่เลือกมาบางลง เพื่อทำให้ง่ายต่อการประมวลผลต่อไป

4) 3D-Mapping

เป็นการนำภาพจากการ Thining แล้วมาทำการเลือกเฉพาะจุดต่างๆที่ทำ Thining แล้วนำ coordinate ของจุดเหล่านั้นมาทำการแปลงจากรูปแบบ 2 มิติให้เป็น 3 มิติ

5) Plot บนแกน 3 มิติ

คือการนำเอาจุดที่ได้จากขั้นตอน 3D-Mapping มาทำการ plot บนแกนจำลอง 3 มิติ ที่จะแสดงบนหน้าจอมอนิเตอร์

6) Check ว่าครบรอบหรือยัง

ถ้ายังไม่ครบก็วนกลับไปทำซ้ำอีกครั้งโดยเริ่มที่การ Capture Image ถ้าครบแล้วก็จะเข้าสู่ขั้นตอนสุดท้ายของการทำงาน

7) แสดงภาพโครงร่าง 3 มิติ ที่สมบูรณ์

เป็นขั้นตอนสุดท้ายของการทำงาน โดยจะแสดงภาพที่ได้จากการ plot coordinate ทั้งหมดจากการสแกนจนครบรอบวัตถุ ดังนั้นภาพที่ได้ในขั้นตอนนี้จึงเป็นภาพที่สมบูรณ์ใกล้เคียงกับวัตถุจริง

8.2 โครงสร้างโปรแกรม

ในส่วนของโปรแกรมที่ควบคุมการทำงานของเครื่อง3D-SCANNERนี้จะแบ่งการควบคุมออกเป็น 2 ส่วนใหญ่ๆ คือ

1.ส่วนของโปรแกรมที่ควบคุมฮาร์ดแวร์ภายนอก คือ ควบคุมการหมุนของ Stepping Motor ซึ่งจะควบคุมผ่านทางการ์ดวงจรพอร์ทขนานที่ต่อเพิ่มเติมเข้ามา โดยมีรูปแบบและวงจรการทำงานตามที่ได้กล่าวมาแล้ว รูปแบบของคำสั่งจะเป็นการส่งค่าลอจิก 1/0 ออกไปทางพอร์ทเพื่อ on/off ทรานซิสเตอร์ที่ต่ออยู่กับแต่ละเฟสของStepping Motor เพื่อเหนี่ยวนำให้เกิดการหมุนตามที่ต้องการ.

2. ส่วนของโปรแกรมการ capture ภาพและประมวลผล เป็นโปรแกรมการรับภาพจากกล้อง quick cam มาแสดงผลที่จอ monitor โดยจะทำงานสัมพันธ์กับโปรแกรมควบคุมการหมุนของ Stepping Motor คือ ในแต่ละตำแหน่งที่จะทำการ capture ภาพ จะต้องรอให้กล้องอยู่นิ่งก่อน (ซึ่งการหมุนของขากล้องถูกควบคุมจาก Stepping Motor) เมื่อกล้องหยุดนิ่งแล้วก็จะทำการ capture ภาพ ณ ตำแหน่งนั้นเพื่อนำไปประมวลผลแล้ว display มาที่จอ monitor ทันที จากนั้นก็ควบคุมให้

มอเตอร์หมุนเพื่อเปลี่ยนตำแหน่งในการ capture ประมวลผลและ display ต่อไปเรื่อยๆจนวนครบรอบวัตถุ

ภาษาที่ใช้เขียนโปรแกรมนี้จะใช้ MFC VISUAL C++ ซึ่งเป็นภาษาที่ run บน windows ทำให้ไม่มีปัญหาเรื่องหน่วยความจำไม่เพียงพอ เนื่องจากการประมวลผลเกี่ยวกับรูปภาพเป็นกระบวนการที่ต้องใช้หน่วยความจำมากพอสมควร

โครงสร้างการทำงานของโปรแกรมควบคุมนี้ จะอยู่บนพื้นฐานของ Interrupt Service Routine 2 อย่างคือ

1. Interrupt Service Routine ของเวลาซึ่งเป็นลักษณะของการนับเวลาจนครบ เมื่อเวลาครบแล้วไม่ว่าโปรแกรมจะทำตรงไหนอยู่ก็ตามมันจะกระโดดไปทำงานที่โปรแกรมบริการอินเตอร์รัปต์ชนิดนี้ทันทีซึ่งในโปรแกรมนี้ ก็คือ ส่วนของฟังก์ชั่น OnTimer

2.Interrupt Service Routine ของการรับ Frameภาพใหม่เข้ามา ซึ่งคล้ายกับ Interrupt Service Routine ของเวลา แต่โปรแกรมจะกระโดดไปทำตรงส่วนบริการอินเตอร์รัปต์ก็ต่อเมื่อมี Frame ของภาพเข้ามาใหม่ โปรแกรมบริการอินเตอร์รัปต์ชนิดนี้ ก็คือ ฟังก์ชั่น OnFrameCallback

การรับ Frame ของภาพมานั้น นอกจากจะได้ข้อมูลภาพจริงๆมาแล้ว เรายังได้รายละเอียดเกี่ยวกับภาพด้วย ตัวอย่างเช่น

- ความสูงของภาพ ในหน่วย Pixel
- ความกว้างของภาพ ในหน่วย Pixel
- ความละเอียดของภาพในหน่วย Bit/Pixel

ซึ่งรายละเอียดต่างๆเหล่านี้ จะทำให้สามารถสร้าง Bitmap file ขึ้นมาใหม่ได้ แล้วนำข้อมูลภาพเหล่านั้นมาทำการประมวลผล ซึ่งในส่วนนี้จะเป็นขั้นตอนที่อยู่ใน Routine ของฟังก์ชั่น OnFrameCallback ทั้งสิ้น การประมวลผลภาพในแต่ละเฟรมจะถูกแบ่งออกได้ตามขั้นตอนดังนี้คือ

2.1) Display Capture Bitmap เป็นการแสดงภาพนิ่งที่ได้จากการถ่ายภาพวัตถุในแต่ละมุมโดยการสร้าง Bitmap file ขึ้นมาแล้ว Copy ข้อมูลภาพจากกล้องมาไว้ที่ file นั้น ภาพที่ Capture ได้จะแสดงบนหน้าจอ monitor ทางด้านมุมบนซ้าย ส่วนภาพที่รับมาจากกล้องนั้นจะแสดงตรงตำแหน่งกึ่งกลางของหน้าต่างแสดงผล หลักการทางโปรแกรมในการรับจากกล้องวิดีโอและ capture ภาพมาแสดงบนหน้าจอเป็นไปตาม flowchart ในรูปที่ 8.4 และ 8.5

2.2) Threshold Bitmap จากการออกแบบในส่วนของเครื่อง 3D-SCANNER นี้จะมีข้อจำกัดอยู่บางประการคือ จะต้องทำการสแกนวัตถุในที่มืด ซึ่งอันที่จริงแล้วไม่จำเป็นต้องทำเช่นนี้ก็ได้ เพราะวิธีการที่จะเลือกส่วนที่ต้องการจากภาพนั้นอาจใช้วิธีการทำ Edge dectection , Gradient , Laplacian และอื่นๆ อีกหลายวิธี โดยไม่ต้องทำในที่มืดก็ได้ แต่วิธีต่างๆ เหล่านั้นจะยุ่งยากมาก

เพราะต้องอาศัยการพิจารณาจาก pixel รอบๆ ด้วย เพื่อดูความแตกต่าง ดังนั้นจึงเลือกใช้วิธีการทำ threshold เพราะเป็นการตรวจสอบทีละ pixel เทียบกับค่าคงที่ที่เราต้องการ ซึ่งในที่นี้ก็คือค่าความสว่างที่เราต้องการนั่นเอง เพราะฉะนั้นการเขียนโปรแกรมจะง่ายขึ้นมาก ถ้าเราทำให้พื้นที่ทดลองมืด ซึ่งจะทำให้สามารถหาแนวสันเลเซอร์ได้ง่าย ไม่ซับซ้อนเพราะภาพจะมีเพียงส่วนของเส้นเลเซอร์เท่านั้นที่สว่างชัดเจน ดังนั้นจึงต้องมีข้อจำกัดตรงส่วนนี้ด้วย ขั้นตอนของการ Threshold นี้จะเป็นการ detect ภาพในส่วนที่เป็นแสงเลเซอร์เท่านั้น ซึ่งจะใช้การ detect โดยดูจากความสว่างของภาพในแต่ละ pixel ตามสูตร

$$\text{Brightness} = (R+G+B)/3 \qquad (8.1)$$

Brightness : ความสว่างของแต่ละ pixel

R : Red Component เป็นองค์ประกอบสีแดงในแต่ละ pixel

G : Green Component เป็นองค์ประกอบสีเขียวในแต่ละ pixel

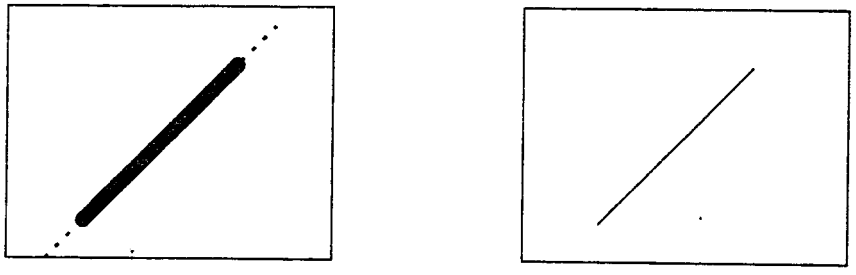
B : Blue Component เป็นองค์ประกอบสีน้ำเงินในแต่ละ pixel

หลักการทางโปรแกรมในการเปลี่ยนภาพให้อยู่ในรูปแบบของความสว่างเป็นไปตาม flow chart ในรูปที่ 8.6

หลังจากคำนวณความสว่างของแต่ละ pixel ออกมาแล้ว ก็เลือก detect เฉพาะส่วนที่มีความสว่างมากซึ่งจะเป็นในส่วนของเส้นเลเซอร์ที่ต้องการ ทำให้ได้ภาพที่เป็น Threshold Bitmap ออกมาแสดงที่หน้าจอ monitor ทางด้านมุมล่างซ้าย ซึ่งจะแสดงตรงส่วนที่ detect เลเซอร์ได้เป็นสีขาวและส่วนอื่นเป็นสีดำ

หลักการทางโปรแกรมในการ threshold เอาเฉพาะส่วนที่ต้องการเป็นไปตาม flowchart ในรูปที่ 8.7

2.3) Thining Bitmap เป็นขั้นตอนที่ทำหลังจาก Threshold Bitmap ออกมาได้แล้วซึ่งหลักการในขั้นตอนนี้ ก็คือ จะทำให้เส้นเลเซอร์ที่ detect ได้นั้นบางลง เพื่อให้ได้เส้นที่คมชัดและแน่นอน โดยตามโปรแกรมนี้จะ Thining ให้เหลือความบางของเส้นเพียง 1 pixel นั่นคือถ้าหากเจอแถบสีขาวของเลเซอร์ในแนวนอนก็จะเลือกเอาเฉพาะ pixel ที่อยู่ตรงกลาง หรือถ้าเจอเพียง 1 pixel ก็ให้เลือก pixel นั้นเลย ทำตามขั้นตอนนี้ไปจนครบแถวในแนวนอนก็จะทำให้ได้ Thining Bitmap ออกมาแสดงที่หน้าจอ monitor ทางด้านมุมบนขวา โดย pixel ที่ได้จากการ Thining แล้วจะเป็นสีขาว และส่วนอื่นเป็นสีดำ ส่วนรูปที่ 8.2 จะแสดงให้เห็นถึง algorithm ของการทำ thining ได้อย่างชัดเจน



ก) ข)

รูปที่ 8.2 แสดง algorithm ของการทำ Thining

ก) ก่อนการทำ Thining ข) หลังการทำ Thining

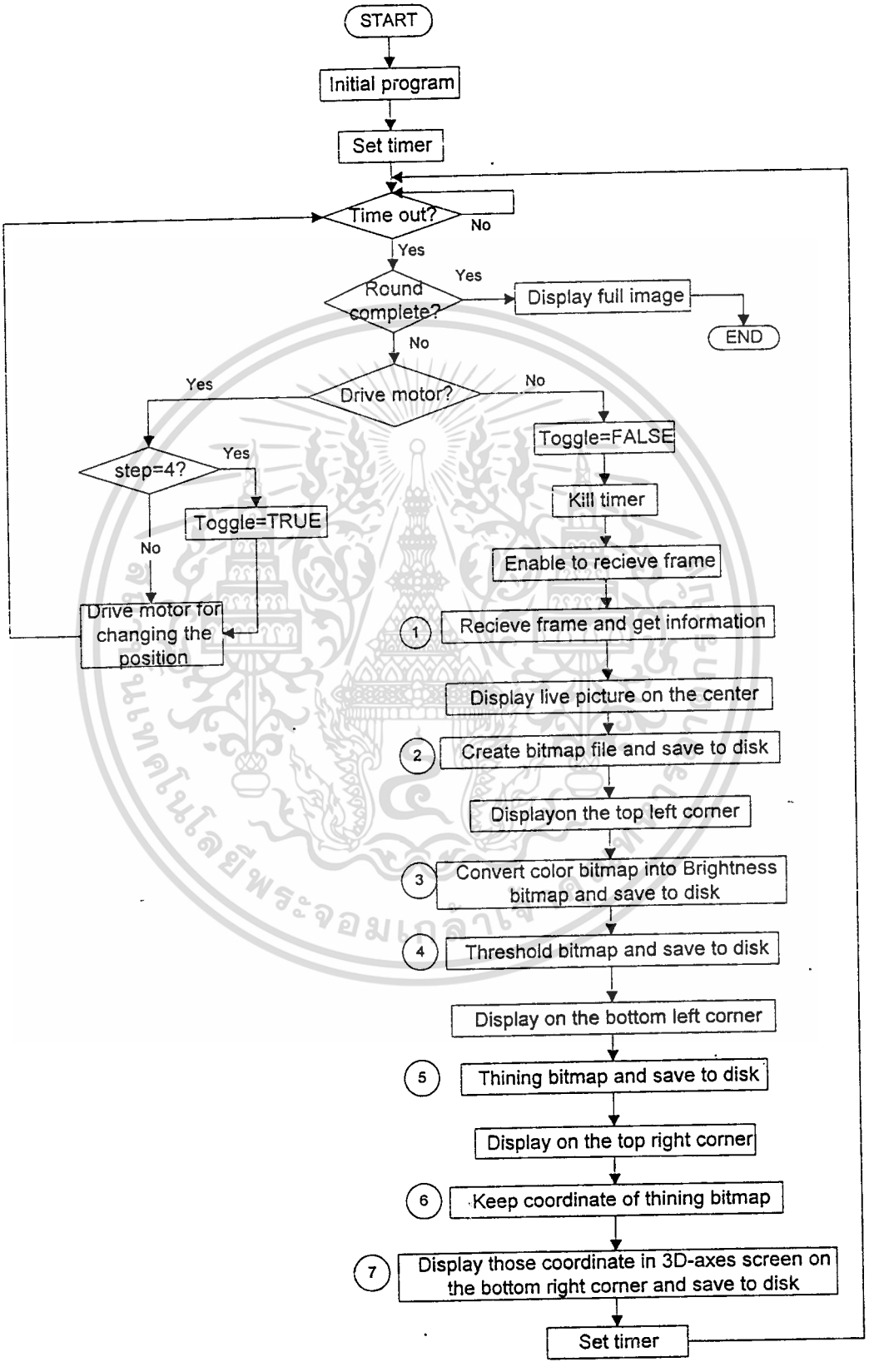
ส่วนหลักการทางโปรแกรมในการทำ thining เป็นไปตาม flow chart ในรูปที่ 8.8

2.4) Mapping Coordinate เป็นส่วนของการนำจุดหรือ pixel ที่ได้จากการ Thining แล้วไปทำการคำนวณในเชิง graphics 3 มิติ เนื่องจากจุดที่ได้จากภาพนั้นเป็นการรับมาจากกล้องที่ได้ตั้งในลักษณะได้ฉากกับแนวลำแสงของเลเซอร์ และรวมถึงระยะจากวัตถุมายังส่วนที่รับภาพคือกล้องนั่นเองดังนั้นจึงต้องมีการ Mapping ให้เป็น Coordinate ที่ถูกต้องในการมองเห็นจริง หลักการในการเก็บ coordinate ของข้อมูลภาพจาก thining bitmap เป็นไปตาม flow chart ในรูปที่ 8.8 ส่วนการ Mapping coordinate จาก 2 มิติ ไปเป็น 3 มิติ นั้นจะเป็นขั้นตอนก่อนที่จะ display ภาพบนแกนจำลอง 3 มิติ ซึ่งหลักการทางคณิตศาสตร์ในการ Mapping ได้แสดงไว้ในหัวข้อ 8.3

2.5) โครงร่างบนแกนจำลอง 3 มิติ เป็นการนำจุดที่ได้จากภาพที่ทำการ Mapping coordinate แล้วมา plot บนแกนจำลอง 3 มิติ แล้วแสดงผลบนจอ monitor ทางด้านมุมล่างขวา หลักการ plot ภาพ 3 มิติ เป็นไปตาม flow chart ในรูปที่ 8.10

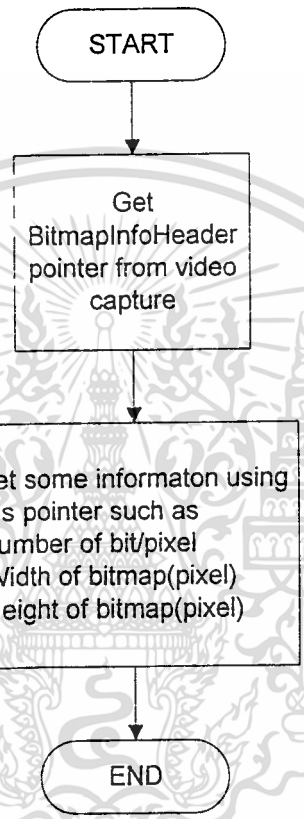
ทั้ง 5 ขั้นตอนนี้จะทำทุกๆครั้งที่มีการรับ Frame ภาพใหม่เข้ามา ดังนั้นจึงสามารถแสดงให้เห็นถึงการประมวลผลภาพทุกๆภาพได้บนหน้าจอ monitor หลังจากนั้นโปรแกรมก็จะถูกตั้งเวลาแล้วให้มีการเรียกโปรแกรม Interrupt ของเวลาอีกครั้งสลับกับ Interrupt ของการรับ Frame ภาพ จนกระทั่งจนเก็บภาพจนครบรอบวัตถุแล้ว ก็จะนำผลรวมของการประมวลผลทั้งหมดมาแสดงอีกครั้งในลักษณะโครงร่างของวัตถุบนแกนจำลอง 3 มิติ ซึ่งหลักการของการสร้างโครงร่าง 3 มิติได้มีการอธิบายต่อไปหลังจากผ่านขั้นตอนนี้แล้วก็จะเสร็จขั้นตอนของโปรแกรมทั้งหมด

การทำงานของโปรแกรมควบคุมฮาร์ดแวร์ภายนอกและโปรแกรมการ capture ภาพและประมวลผลจะทำงานอย่างสัมพันธ์กันตามโครงสร้างโปรแกรม และ Flow Chart ดังรูปที่ 8.3



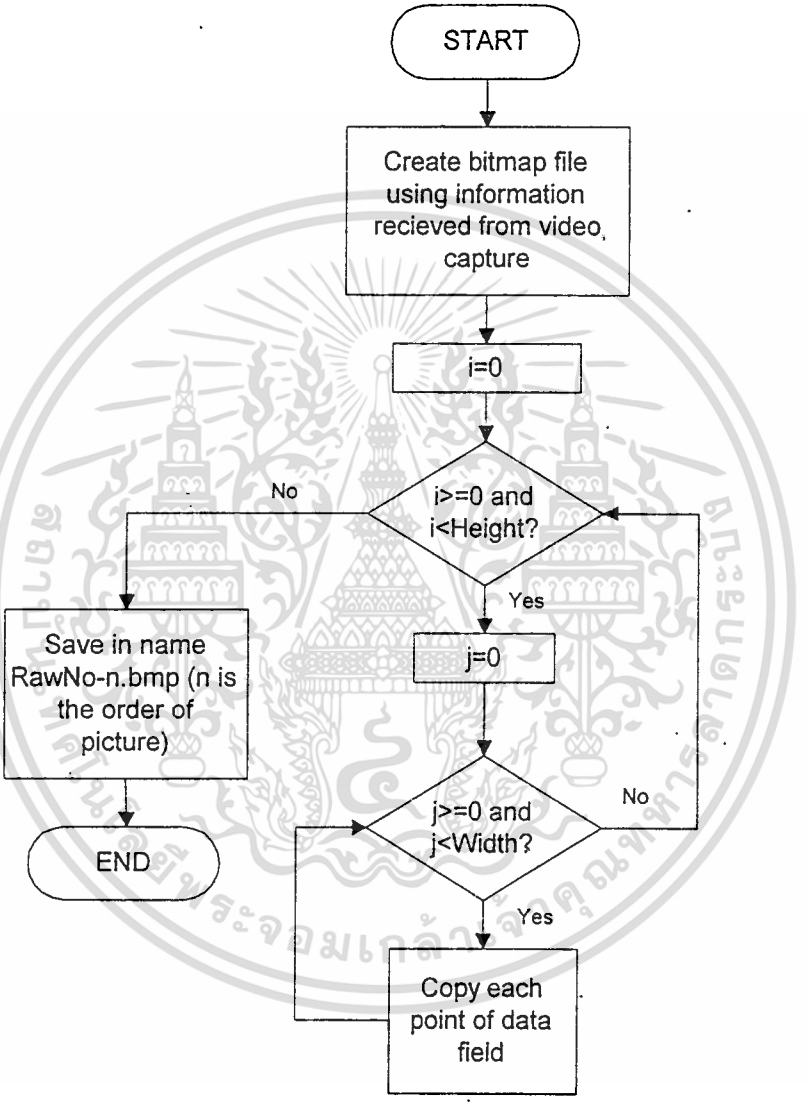
รูปที่ 8.3 Flow chart แสดงโครงสร้างโดยรวมของโปรแกรม

1 Receive Frame and Get Information



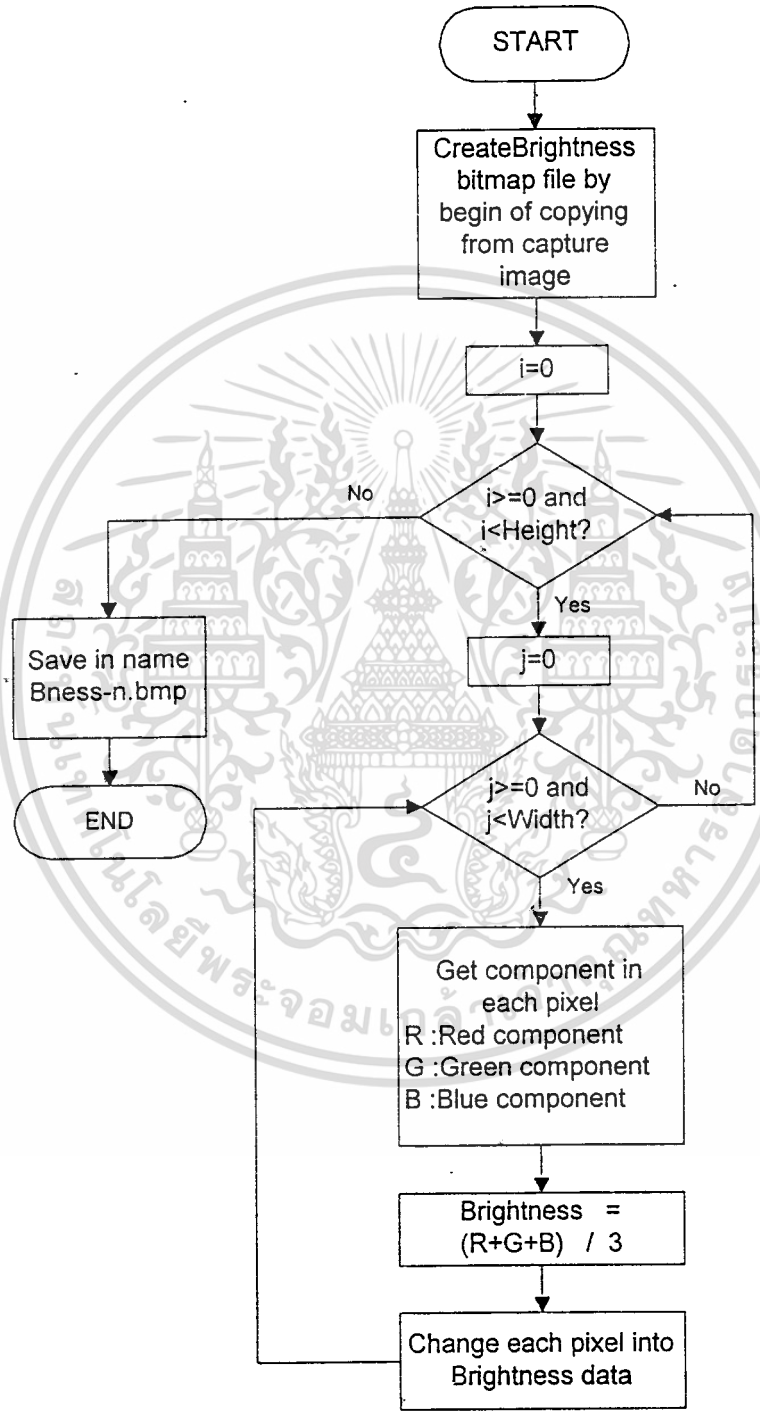
รูปที่ 8.4 Flow chart แสดงการรับภาพและข้อมูลภาพจากกล้องวิดีโอ

2 Create Bitmap File and Save to Disk



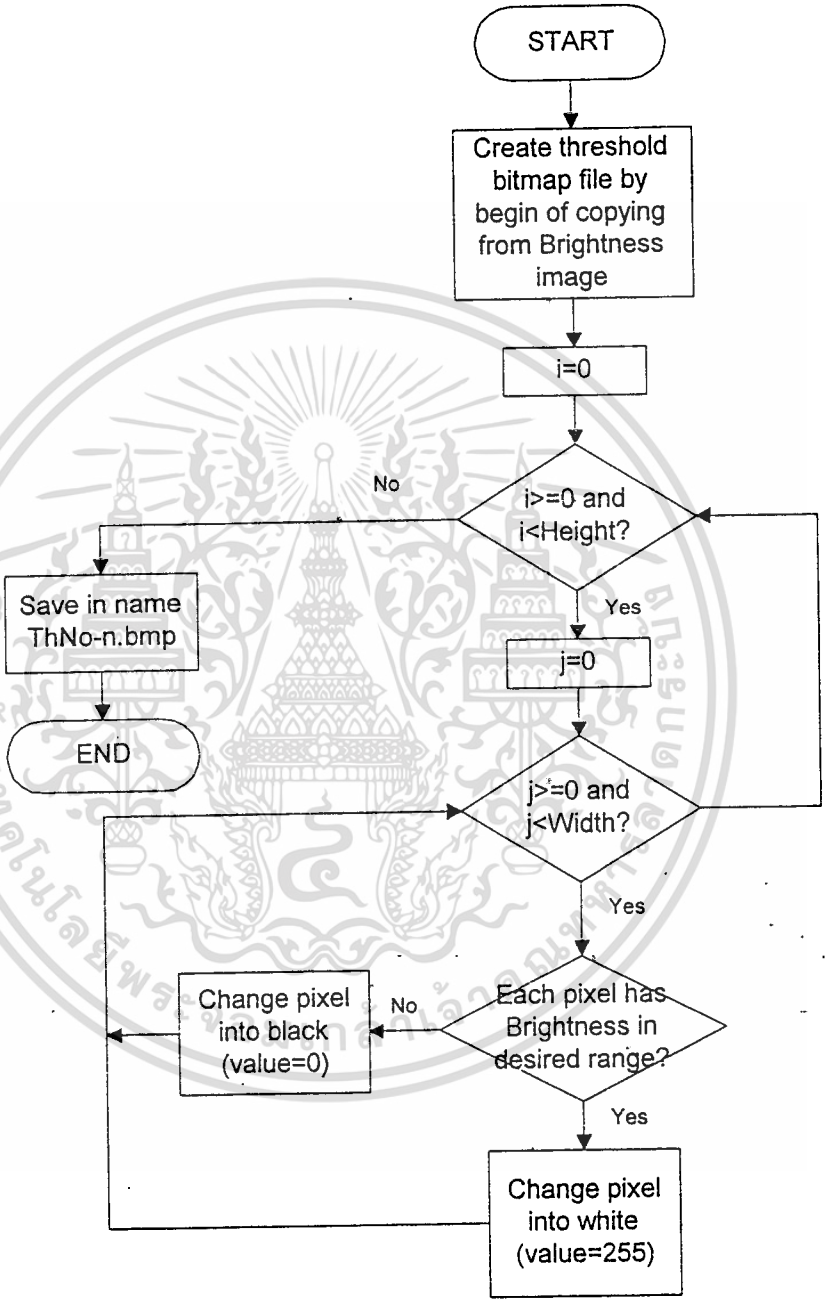
รูปที่ 8.5 Flow chart แสดงการสร้าง bitmap file จากภาพที่ Capture มาแล้วแสดงผล

3 Convert Color Bitmap into Brightness Bitmap and Save to Disk



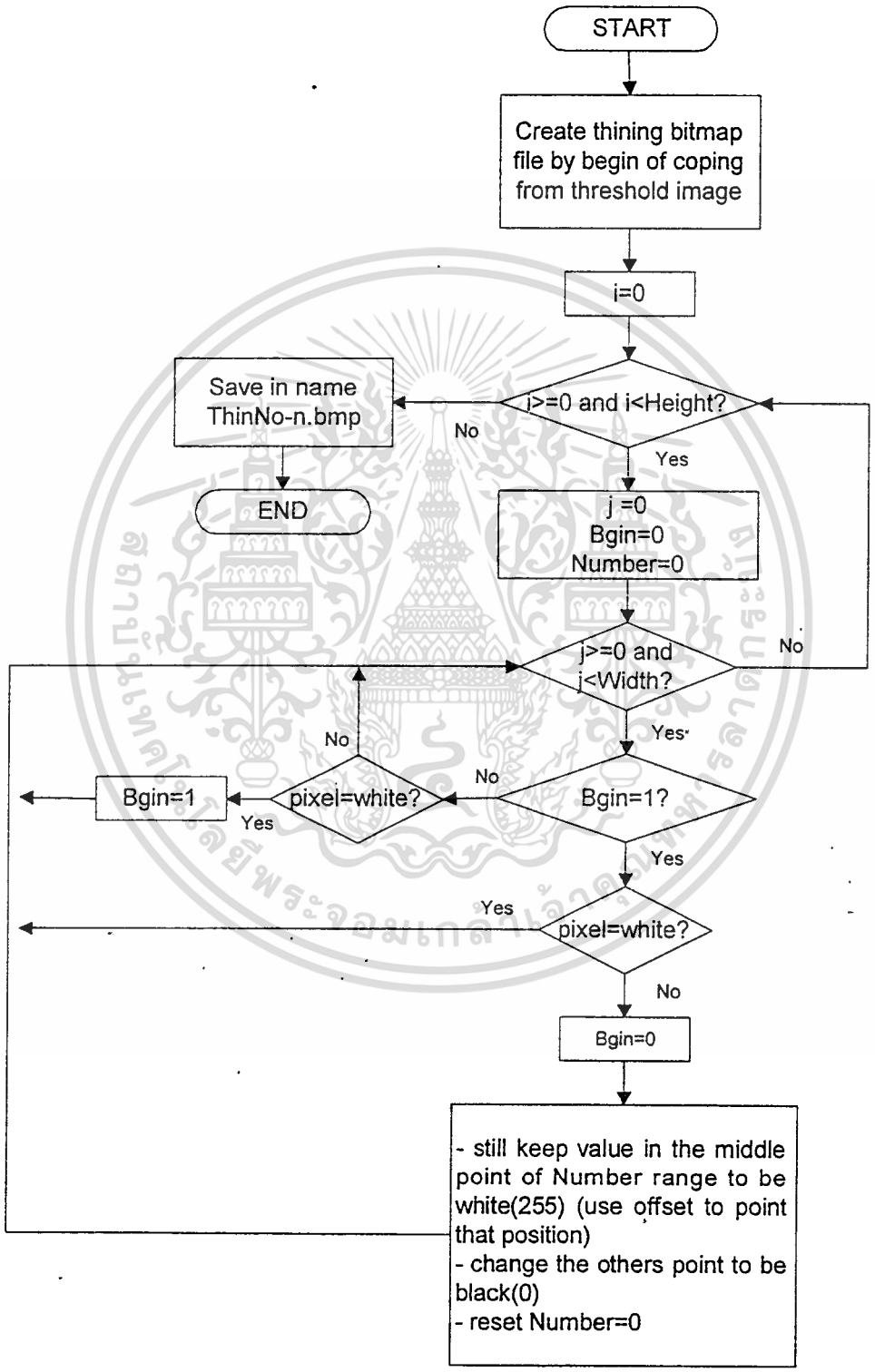
รูปที่ 8.6 Flowchart แสดงการ convert ภาพจาก color bmp เป็น brightness bmp

4 Threshold Bitmap and Save to Disk



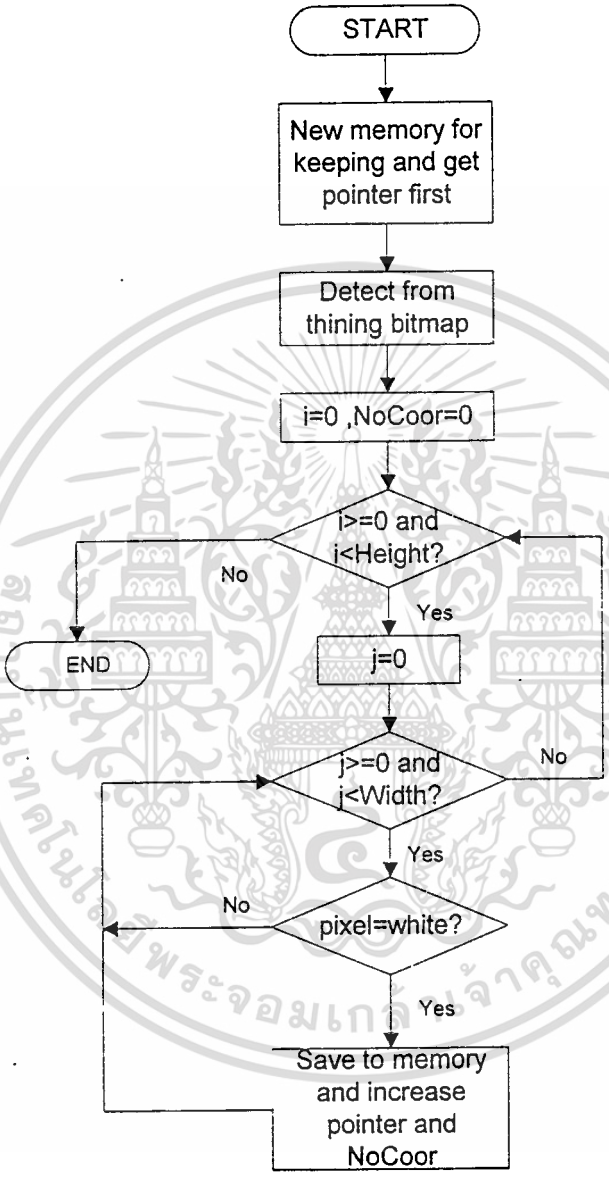
รูปที่ 8.7 Flow chart แสดงขั้นตอนการทำ threshold

5 Thining Bitmap and Save to Disk



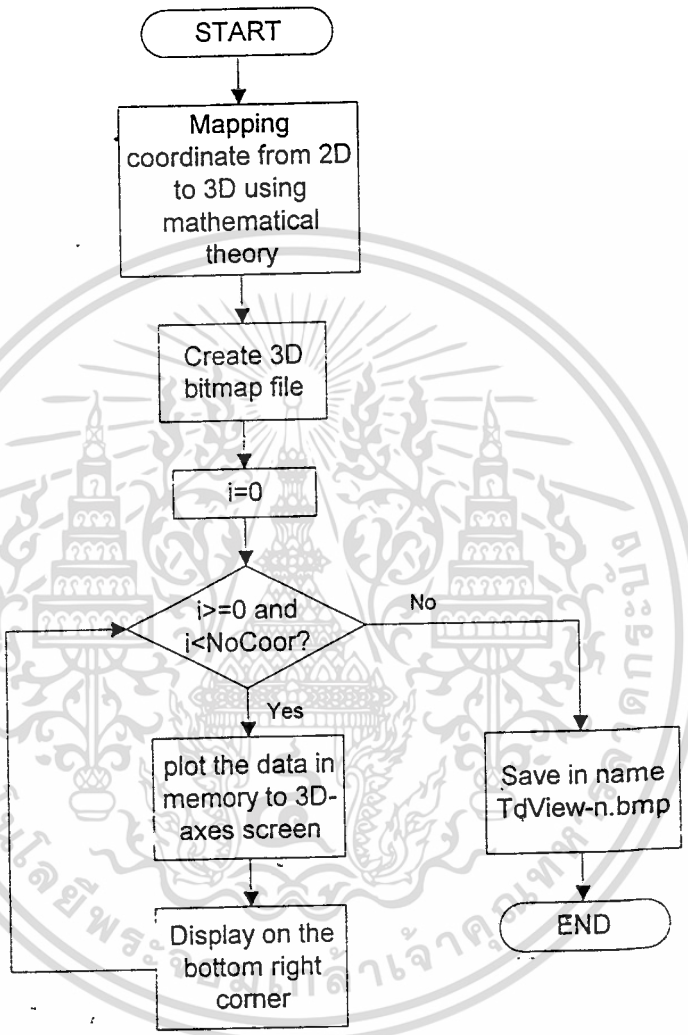
รูปที่ 8.8 Flow chart **แสดงขั้นตอนการทำ** thining

6 Keep Coordinate of Thining Bitmap



รูปที่ 8.9 Flow chart แสดงขั้นตอนการเก็บ coordinate ของข้อมูลภาพจาก thining bmp

7 Display those coordinate 3D-axes screen on the bottom right comer and Save to disk



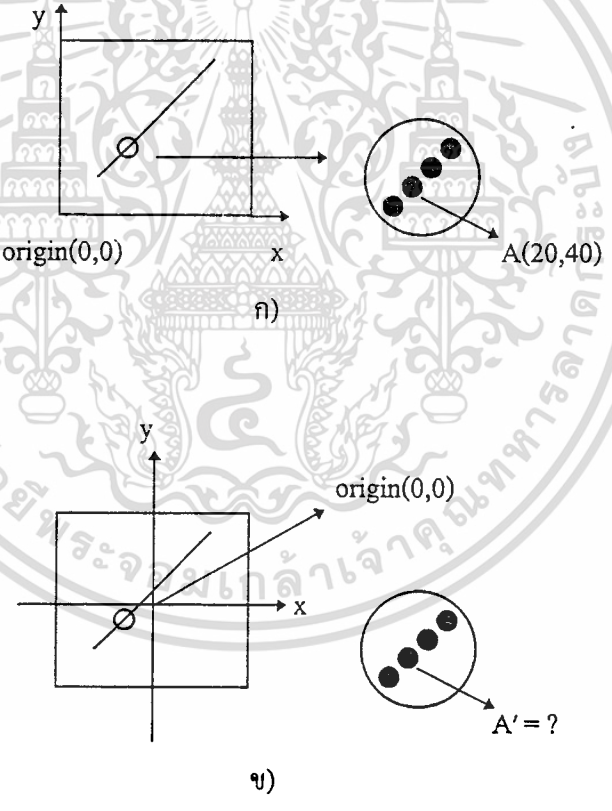
รูปที่ 8.10 Flow chart แสดงการ plot ภาพบนแกน 3 มิติ

8.3 หลักการของการสร้างโครงร่าง 3 มิติ

8.3.1) การหาตำแหน่งของวัตถุในระนาบตั้งฉาก

โดยอาศัยหลักการทางคณิตศาสตร์ในการหาจุดตัดระหว่างระนาบของเลเซอร์กับสมการเส้นตรงซึ่งได้จากภาพที่รับจากกล้องกับจุดสังเกต (แทนจุดโฟกัสของกล้อง) ซึ่งจุดตัดนั้น ก็คือ จุดบนวัตถุที่เราต้องการนั่นเอง

1) จากขั้นตอนการทำ Thining ทำให้ได้ภาพ 2 มิติในแต่ละมุมของวัตถุซึ่งภาพที่ได้เหล่านี้มีจุด origin (0,0) ที่มุมล่างซ้ายของภาพ ดังนั้นจึงทำการ translate จุด origin ไปอยู่ที่จุดกึ่งกลางของภาพ เพื่อใช้หลักการทางคณิตศาสตร์คำนวณ ซึ่งแสดงได้ดังรูป 8.11



รูปที่ 8.11 แสดงการ translate จุด origin ของภาพ 2 มิติ

ก) ภาพ 2 มิติที่ได้จากกล้อง ข) ภาพ 2 มิติที่ translate จุด origin แล้ว

2) การ translate จุด origin ในขั้นตอนที่ 1 จะใช้หลักการ translation 2 มิติ ซึ่งได้อธิบายในหัวข้อ 7.1.2 โดยใช้ matrix ดังสมการ (8.2)

$$[X^* \quad Y^* \quad 1] = [X \quad Y \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -W/2 & -H/2 & 1 \end{bmatrix} \qquad (8.2)$$

เมื่อ (x,y) เป็นจุดบนภาพที่รับจากกล้องซึ่งมี Origin อยู่ที่มุมล่างซ้ายของภาพ

เมื่อ (x*,y*) เป็นจุดที่เกิดจากการ translate (x,y) หลังจากการย้าย Origin

W เป็นความกว้างของภาพตามแนวแกน x

H เป็นความสูงของภาพตามแนวแกน y

ซึ่งภาพที่รับมาจากกล้องมี W=160 pixels และ H=120 pixels ดังนั้นเมื่อ translate origin ตามสมการ (8.2) แล้ว จะได้ Coordinate ใหม่ของจุด A คือ A'

3) กำหนด Z_C และ Z_E บนแกน Z (เพื่อสะดวกในการคำนวณ) โดยให้

Z_C เป็นระยะทางจากวัตถุมายังภาพ

Z_E เป็นระยะทางจากภาพมายังจุดสังเกต

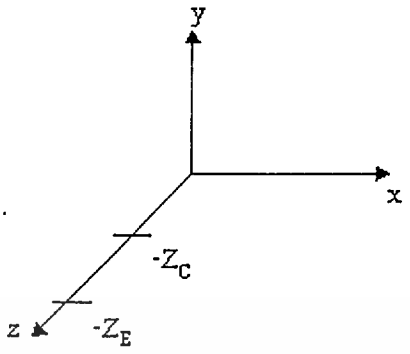
ทำให้เราได้ตำแหน่ง origin ของภาพที่รับมาซึ่งในขณะนี้อยู่ที่จุดกึ่งกลางภาพแล้ว คือ

$$P_{OC} = (0, 0, -Z_C)$$

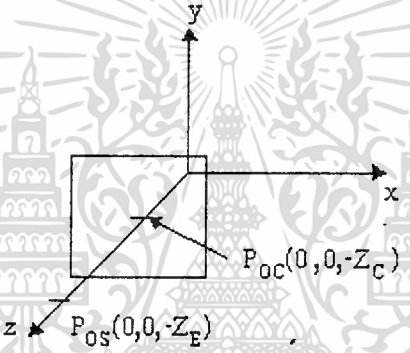
และได้ตำแหน่งของจุดสังเกต คือ

$$P_{OS} = (0, 0, -Z_E)$$

ดังรูปที่ 8.12



ก)



ข)

รูปที่ 8.12 รูปแสดงตำแหน่งของ Z_E, Z_C และระนาบของภาพบนแกน 3 มิติ

ก) แสดงตำแหน่งของ Z_E, Z_C ข) ระนาบของภาพ

4) หาสมการเส้นตรงระหว่างจุดสังเกตกับจุดบนภาพใดๆ Pi

กำหนดให้ Pi = $(Xi, Yi, -Z_C)$ เป็นพิกัดของจุด A' บนระนาบ

การหาสมการเส้นตรงระหว่างจุด 2 จุดคือ P1 (X1,Y1,Z1) และ P2 (X2,Y2,Z2) ในระบบ 3 มิติ จะได้สมการเส้นตรง คือ

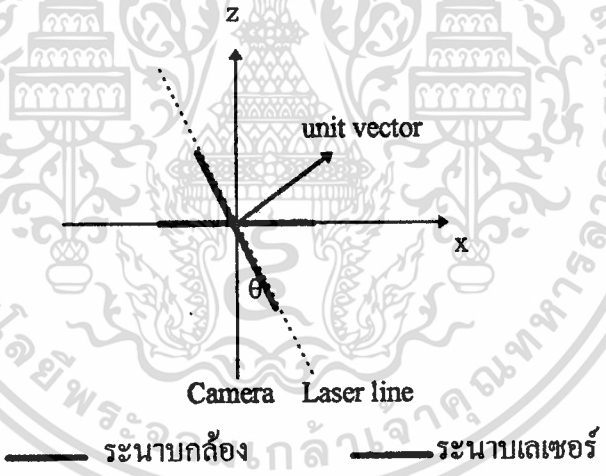
$$Li = \begin{bmatrix} X = X1 + t(X2-X1) \\ Y = Y1 + t(Y2-Y1) \\ Z = Z1 + t(Z2-Z1) \end{bmatrix}$$

จากรูปที่ 8.12 จุด P1 ก็คือจุด $P_{OS}(0, 0, -Z_E)$ และจุด P2 ก็คือจุด $P_{OC}(0, 0, -Z_C)$ ดังนั้น

$$Li = \begin{bmatrix} X = 0 + t\,Xi \\ Y = 0 + t\,Yi \\ Z = -Z_E + t\,(-Z_C + Z_E) \end{bmatrix}$$

$$= \begin{bmatrix} X = tXi \\ Y = tYi \\ Z = -Z_E + (Z_E - Z_C)t \end{bmatrix} \qquad (8.3)$$

5) ให้เวกเตอร์ [A,B,C] เป็นเวกเตอร์หนึ่งหน่วย (unit vector) ที่ตั้งฉากกับระนาบของเลเซอร์ $[V_L]$ ซึ่งเป็นระนาบที่ทำมุม θ กับระนาบของฉากรับภาพจากกล้อง $[P_L]$ โดยหมุนรอบแกน y ซึ่งถ้ามองจากด้านบนจะเห็นการวางตัวของระนาบดังรูปที่ 8.13



รูปที่ 8.13 แสดงการวางตัวของระนาบกล้องและระนาบเลเซอร์

ดังนั้น unit vector ที่ตั้งฉากกับระนาบเลเซอร์ที่ได้คือ $(\sin\theta\ ,\ 0\ ,\cos\theta)$ จากสมการระนาบใดๆ

$$AX + BY + CZ = D \qquad (8.4)$$

โดยที่ (A , B , C) คือ unit vector ที่ตั้งฉากกับระนาบ ดังนั้นจะได้ว่า

$$A = \sin\theta$$

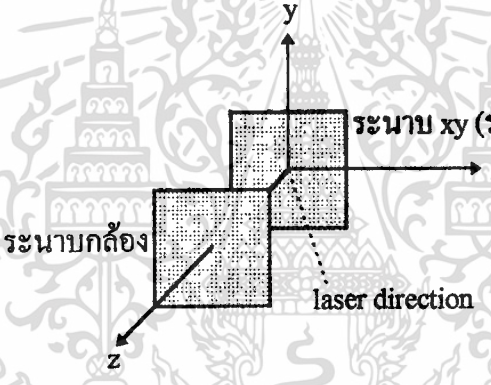
$$B = 0 \quad \therefore$$

$$C = \cos\theta$$

ดังนั้น unit vector $V_L = [\sin\theta, 0, \cos\theta]$

และระนาบของเลเซอร์ $[P_L]$ คือ $X\sin\theta + Z\cos\theta = 0$ (8.5)

6) หาระนาบของเลเซอร์ $[P_L]$ เนื่องจากภาพที่รับมาได้ในแต่ละมุมของวัตถุเป็นภาพที่กล้องได้ถ่ายในลักษณะที่เอียงทำมุม θ กับชุดกำเนิดเลเซอร์ ซึ่งตำแหน่งของเส้นเลเซอร์ในภาพที่ได้มาจากการ Capture ภาพนั้นยังไม่เป็นตำแหน่งที่ถูกต้อง จึงต้องหาตำแหน่งบนระนาบของเลเซอร์ที่ถูกต้องโดยวิธีการดังนี้



รูปที่ 8.14 แสดงระนาบ xy ที่จุด origin

6.1) กำหนดให้แสงเลเซอร์ตัดผ่านจุด origin (0,0,0) ในแกน 3 มิติ

6.2) กำหนดระนาบ xy ที่ origin (0,0,0) ดังรูปที่ 8.14 จะได้สมการระนาบที่ Z=0 และเวกเตอร์หนึ่งหน่วยตั้งฉากกับระนาบ (unit vector) คือ (0,0,1)

6.3) หามุมระหว่างระนาบของกล้องกับระนาบของเลเซอร์ที่ได้จากการทดลองจากการวัด

6.4) ทำการหมุนระนาบ xy ไปเป็นมุม θ โดยใช้สมการทางคณิตศาสตร์ที่กล่าวถึงในบทที่7

7) หาจุดของภาพจริงในระบบ 3 มิติ โดยแทนสมการเชิงเส้น (8.3) ลงในสมการระนาบของเลเซอร์ (8.5) เพื่อหาค่า t จะได้

$$tXi\sin\theta + [-Z_E\cos\theta + (Z_E - Z_C)t\cos\theta] = 0$$

$$t[(Z_E - Z_C)\cos\theta + Xi\sin\theta] = Z_E\cos\theta$$

จะได้ $t = Z_E \cos\theta / (Z_E - Z_C)\cos\theta + Xi \sin\theta$ (8.6)

นำค่า t ที่ได้จากสมการ (8.6) แทนในสมการ (8.3) จะได้จุดของภาพจริงที่จะนำมาแสดงในแต่ละองศาของการหมุนกล้องถ่ายภาพ

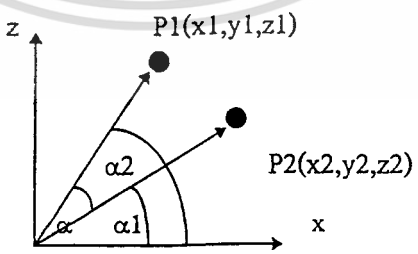
จากขั้นตอน 1-6 เราจะได้ระนาบเลเซอร์ระนาบแรกเป็นระนาบอ้างอิงซึ่งได้จากการรับภาพจากกล้องในขณะที่กล้อง ยังไม่ได้หมุนไปตามมุมต่างๆ และหลังจากขั้นตอน 7 ทำให้ได้จุดต่างๆ ในระนาบเลเซอร์ระนาบแรก ซึ่งจุดเหล่านี้เป็นจุดที่แสดงตำแหน่งของเส้นเลเซอร์ที่เรา detect ได้อย่างถูกต้อง ทั้ง 7 ขั้นตอนนี้จะใช้เป็นอัลกอริทึมในการหาระนาบเลเซอร์ในทุกๆครั้งที่กล้องรับภาพมา แต่เมื่อกล้องหมุนไปเป็นมุมต่างๆเพื่อทำการรับภาพรอบวัตถุนั้น ระนาบเลเซอร์ของภาพที่รับมาจึงต้องหมุนไปตามมุมที่กล้องหมุนด้วยเพื่อเป็นระนาบจริง มิฉะนั้นภาพที่ display ออกมาที่มุมต่างๆ จะเกิดการซ้อนทับกันที่ตำแหน่งของระนาบอ้างอิง ดังนั้นขั้นตอนต่อไปจึงต้องหาตำแหน่งของวัตถุในระนาบจริงที่มุมใดๆ

8.3.2) การหาตำแหน่งของวัตถุในระนาบจริง

โดยการ rotate จุดบนระนาบตั้งฉากซึ่งผ่านกระบวนการทั้ง 7 ขั้นตอนในข้อ 8.3.1) ไปตามองศาการหมุนของกล้องในแต่ละสเตป

- ในการทดลองเราใช้การหมุนทั้งหมด 50 สเตป จึงครบรอบ(360°) ดังนั้นจะได้แต่ละสเตปเท่ากับ 360/50 = 7.2 องศา

-ให้จุด P1 เป็นจุดที่อยู่บนระนาบ y = y1 และต้องการจะหมุนรอบแกน y ตามเข็มนาฬิกาไปเป็นมุม α เราจะได้จุดใหม่ คือ P2



รูปที่ 8.15 แสดงการ rotate จุดบนระนาบรอบแกน y

- เนื่องจาก จุด P1 และ P2 อยู่บนระนาบ y1 เหมือนกันจึงเปรียบเสมือนเป็นการ rotation ในระนาบ 2 มิติ

- ให้ $\alpha 1$ เป็นมุมที่ P1 ทำกับแกน x

$$x1 = r\cos\alpha 1$$

$$z1 = r\sin\alpha 1 \qquad (8.7)$$

- ต้องการหมุนรอบแกน y ไปเป็นมุม เพราะฉะนั้นจะได้ P2 ทำมุม $\alpha 2 = \alpha 1 - \alpha$ องศา

$$x2 = r\cos\alpha 2 = r\cos(\alpha 1 - \alpha)$$

$$z2 = r\sin\alpha 2 = r\sin(\alpha 1 - \alpha)$$

กระจายพจน์ $\cos(\alpha 1 - \alpha)$ และ $\sin(\alpha 1 - \alpha)$ ได้

$$x2 = r\cos\alpha 1\cos\alpha 2 + r\sin\alpha 1\sin\alpha 2$$

$$z2 = r\sin\alpha 1\cos\alpha 2 - r\cos\alpha 1\sin\alpha 2 \qquad (8.8)$$

จากสมการ (8.7) จะได้

$$x2 = x1\cos\alpha 2 + z1\sin\alpha 2$$

$$z2 = z1\cos\alpha 2 - x1\sin\alpha 2 \qquad (8.9)$$

- นำค่า x1 , z1 ของแต่ละจุดที่ได้จากระนาบตั้งฉากมาแทนในสมการ (8.9) จะได้จุดบนวัตถุจริงออกมา

- เป็นค่า α ตามสเตปการหมุนโดยเริ่มที่ 0 , 7.2 ,14.4 ,...,352.8

$$\alpha_n = 7.2\,(n - 1) \qquad (8.10)$$

- นำจุดบนวัตถุจริงของแต่ละสเตปมา plot รวมกันก็จะได้โครงร่าง 3 มิติออกมา

บทที่ 9

ผลการทดลอง

9.1 การ Set อุปกรณ์

1) จัดให้มุมระหว่างกล้องกับเลเซอร์มีขนาดประมาณ 25 องศา
2) ต่อ Connector จากการ์ดพอร์ทขนานที่เสียบอยู่กับ ISA Slot เข้ากับชุดควบคุมการหมุนของสเตปมอเตอร์
3) จ่ายไฟเลี้ยงให้กับชุดเลเซอร์และชุดควบคุมการหมุนสเตปมอเตอร์
4) จำลองพื้นที่มืดในการทดลอง โดยในที่นี้จะใช้กล่องขนาดใหญ่ครอบชุดmechanicทั้งหมด

9.2 วิธีทดลอง

1) นำวัตถุที่จะสแกนมาวางบนแท่นสแกน
2) ปิดฝาครอบกล่องที่นำมาจำลองเป็นพื้นที่มืดในการทดลอง
3) ทำการRun โปรแกรม 3D-Scanner
4) สังเกตผลการทดลองที่ได้ในแต่ละสเตปการหมุนบนหน้าจอมอนิเตอร์

9.3 ผลการทดลอง

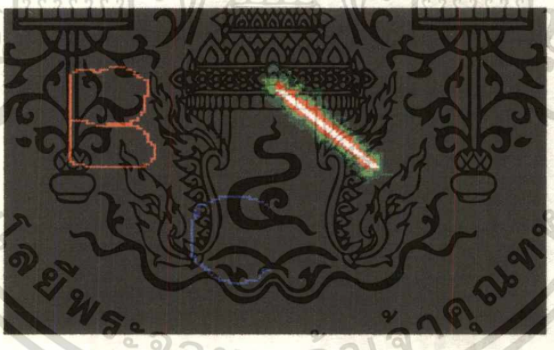
สำหรับผลการทดลองในภาคเรียนที่ 2 นี้ จะเป็นผลที่เสร็จสมบูรณ์ทั้งหมดของโครงงานนี้ ซึ่งมีเป้าหมาย คือ นำภาพที่ได้จากกล้องมาประมวลผลภาพแล้วนำไป plot ให้ออกมาเป็นรูปแบบของภาพ 3 มิติบนหน้าจอมอนิเตอร์ โดยในส่วนของการประมวลผลนั้นก็ได้แก่การ detect แนวเส้นเลเซอร์จากภาพ (การทำ threshold) จากนั้นก็ทำ thining แนวเส้นเลเซอร์นั้นๆ แล้วนำมา plot เพื่อแสดงบนหน้าจอมอนิเตอร์ ทำเช่นนี้ทุกสเตปของการหมุนจนครบรอบวัตถุก็จะได้ภาพที่สมบูรณ์ของวัตถุนั้น

การหมุนของสเตปมอเตอร์จะขึ้นอยู่กับลอจิกที่ส่งออกมาที่การ์ดพอร์ทขนาน ซึ่งจะทำให้มอเตอร์หมุนไปในมุมต่างๆของวัตถุออกมาซึ่งสเตปมอเตอร์ที่ใช้จะหมุนด้วยมุม 1.8 องศา/สเตป แต่ในโครงงานนี้จะทำการสแกนโดยหมุนไปครั้งละ 7.2 องศาแล้วทำการ capture ภาพ เนื่องจากใน

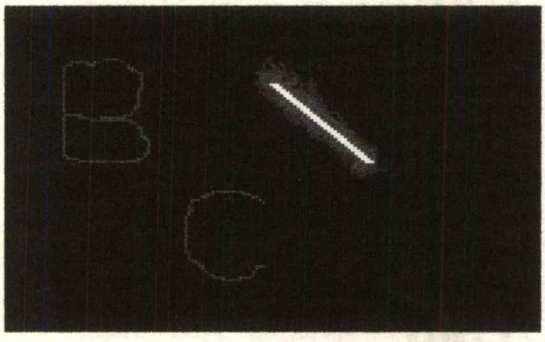
การประมวลผลภาพนั้นใช้เวลาค่อนข้างนานจึงทำการสแกนเพียง 7.2 องศาต่อ 1 ครั้งหรือ 50 ครั้งต่อ 1 รอบวัตถุนั่นเอง

ภาพต่อไปนี้เป็นภาพตัวอย่างที่ได้จากการทดลอง Run โปรแกรม ซึ่งจะแสดงการประมวลผลภาพทุกขั้นตอนในแต่ละสเตปของการหมุนกล้อง

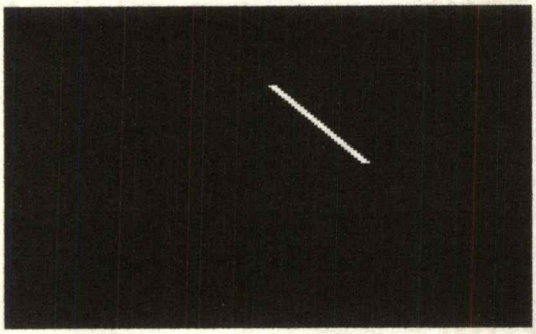
รูปที่ 9.1 แสดงโครงสร้างของ 3D-SCANNER



รูปที่ 9.2 แสดงภาพที่สร้างขึ้นเองเพื่อนำมาทดสอบกับโปรแกรมในทุกๆสเตป

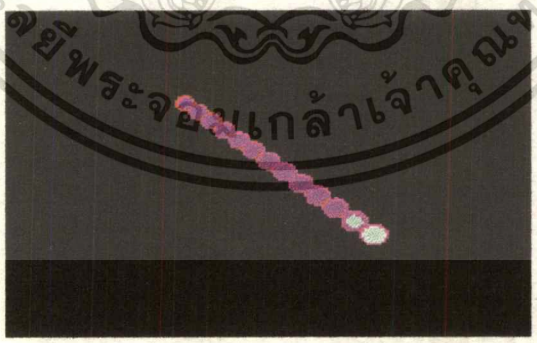


รูปที่ 9.3 แสดงภาพที่ผ่านการแปลงเป็นภาพ Brightness แล้ว

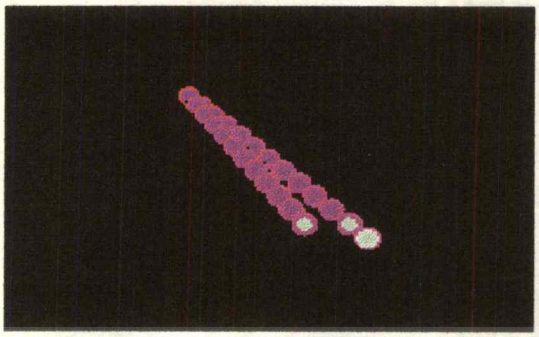


รูปที่ 9.4 แสดงภาพที่ได้จากการทำ Threshold

รูปที่ 9.5 แสดงภาพที่ได้จากการทำ Thining



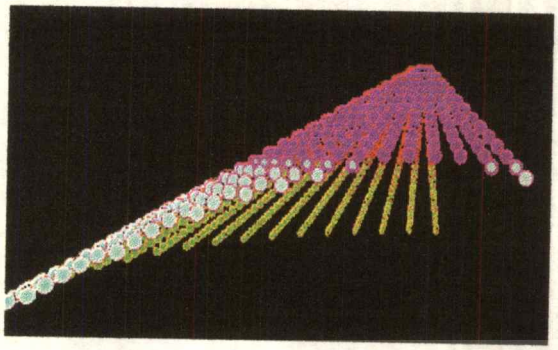
รูปที่ 9.6 แสดงภาพโครงร่าง 3 มิติที่ได้จากการประมวลผลภาพที่สร้างขึ้นในสเตปที่ 1



รูปที่ 9.7 แสดงภาพโครงร่าง 3 มิติที่ได้จากการประมวลผลภาพที่สร้างขึ้นในสเตปที่ 2

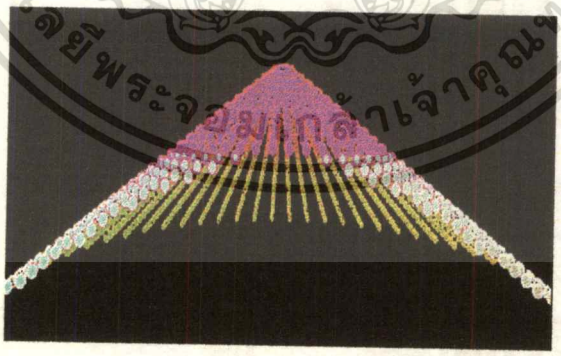
รูปที่ 9.8 แสดงภาพโครงร่าง 3 มิติที่ได้จากการประมวลผลภาพที่สร้างขึ้นในสเตปที่ 5

รูปที่ 9.9 แสดงภาพโครงร่าง 3 มิติที่ได้จากการประมวลผลภาพที่สร้างขึ้นในสเตปที่ 15

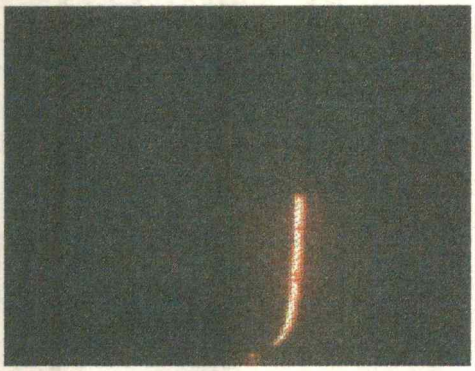


รูปที่ 9.10 แสดงภาพโครงร่าง 3 มิติที่ได้จากการประมวลผลภาพที่สร้างขึ้นในสเตปที่ 30

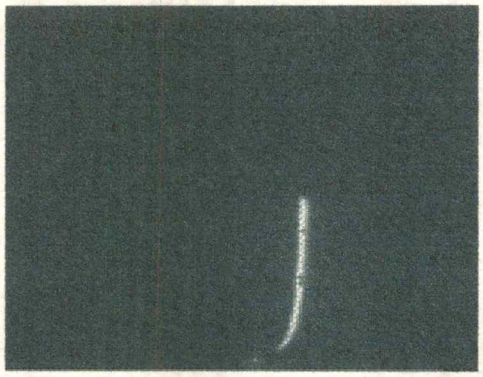
รูปที่ 9.11 แสดงภาพโครงร่าง 3 มิติที่ได้จากการประมวลผลภาพที่สร้างขึ้นในสเตปที่ 40



รูปที่ 9.12 แสดงภาพโครงร่าง 3 มิติที่ได้จากการประมวลผลภาพที่สร้างขึ้นในสเตปที่ 50



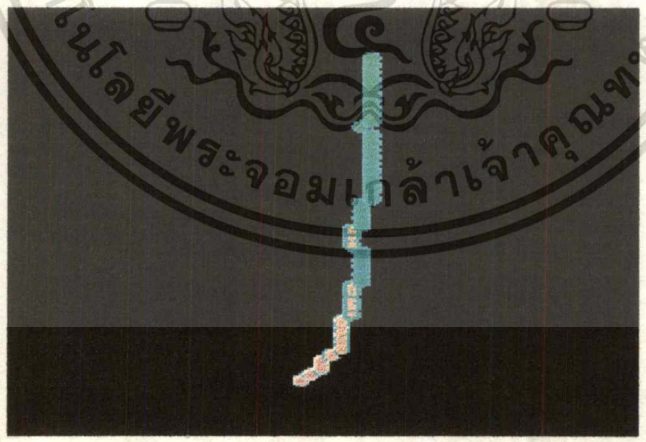
(a)



(b)

(c)

(d)



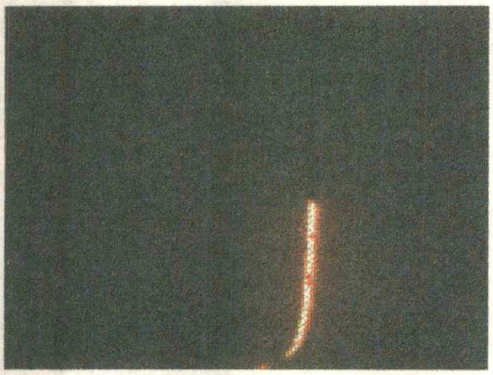
(e)

รูปที่ 9.13 แสดงภาพที่ได้จากการประมวลผลของภาพที่รับจากกล้องสเตปที่ 1

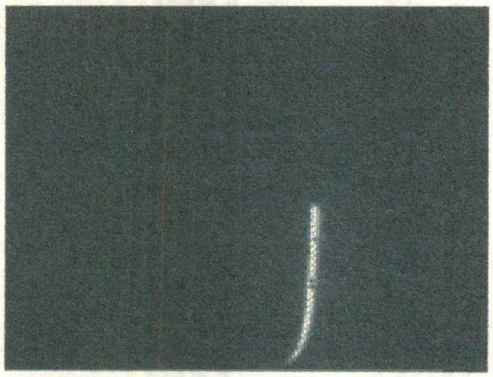
(a) ภาพที่ได้จากการ Capture (b) ภาพที่ได้จากการทำ Brightness

(c) ภาพที่ได้จากการทำ Threshold (d) ภาพที่ได้จากการทำ Thining

(e) ภาพโครงร่าง 3 มิติที่สร้างได้



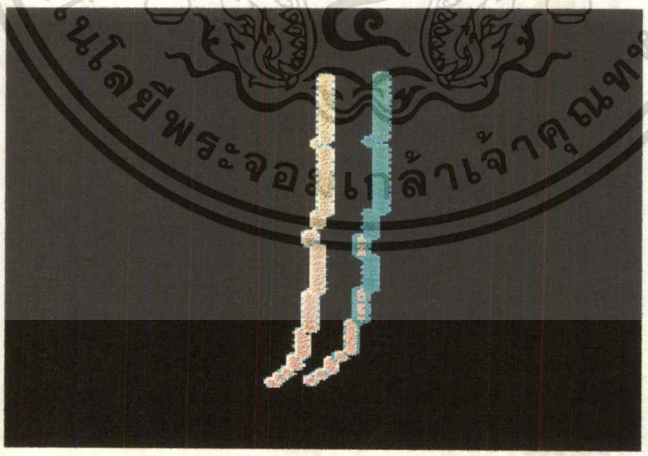
(a)



(b)

(c)

(d)



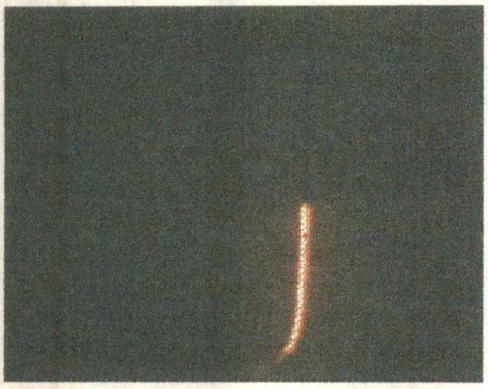
(e)

รูปที่ 9.14 แสดงภาพที่ได้จากการประมวลผลของภาพที่รับจากกล้องสเตปที่ 2

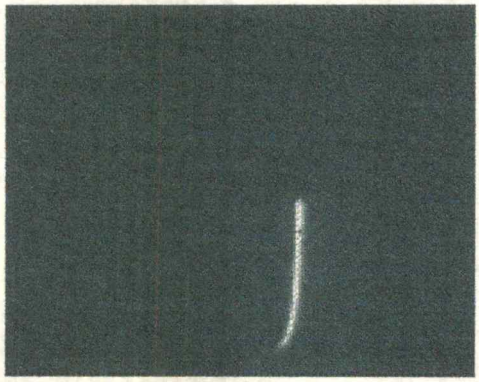
(a) ภาพที่ได้จากการ Capture (b) ภาพที่ได้จากการทำ Brightness

(c) ภาพที่ได้จากการทำ Threshold (d) ภาพที่ได้จากการทำ Thining

(e) ภาพโครงร่าง 3 มิติที่สร้างได้



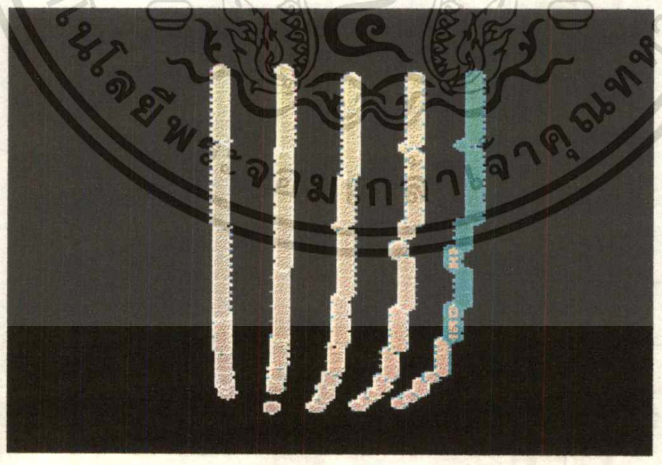
(a)



(b)

(c)

(d)



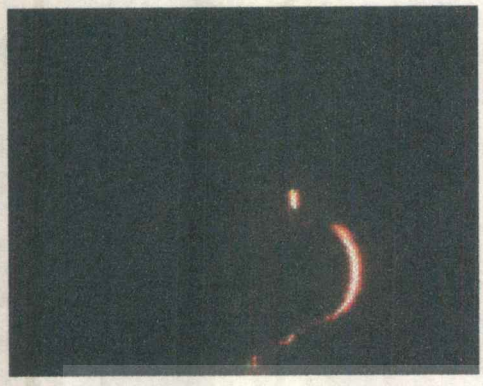
(e)

รูปที่ 9.15 แสดงภาพที่ได้จากการประมวลผลของภาพที่รับจากกล้องสเตปที่ 5

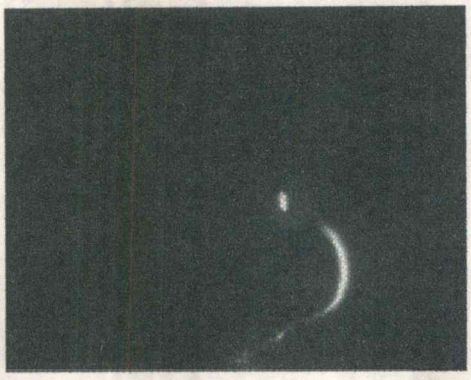
(a) ภาพที่ได้จากการ Capture (b) ภาพที่ได้จากการทำ Brightness

(c) ภาพที่ได้จากการทำ Threshold (d) ภาพที่ได้จากการทำ Thining

(e) ภาพโครงร่าง 3 มิติที่สร้างได้



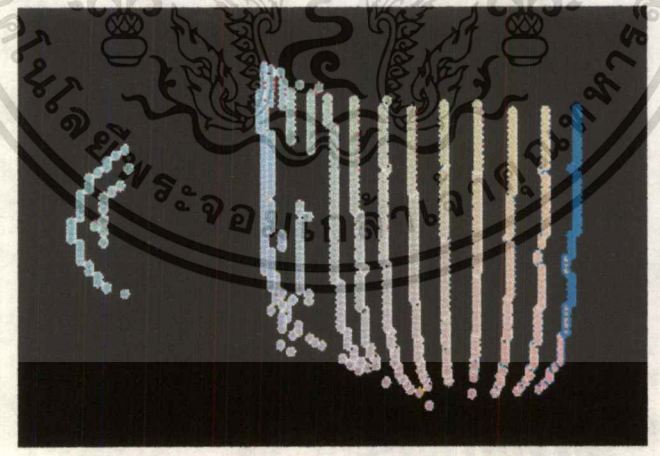
(a)



(b)

(c)

(d)



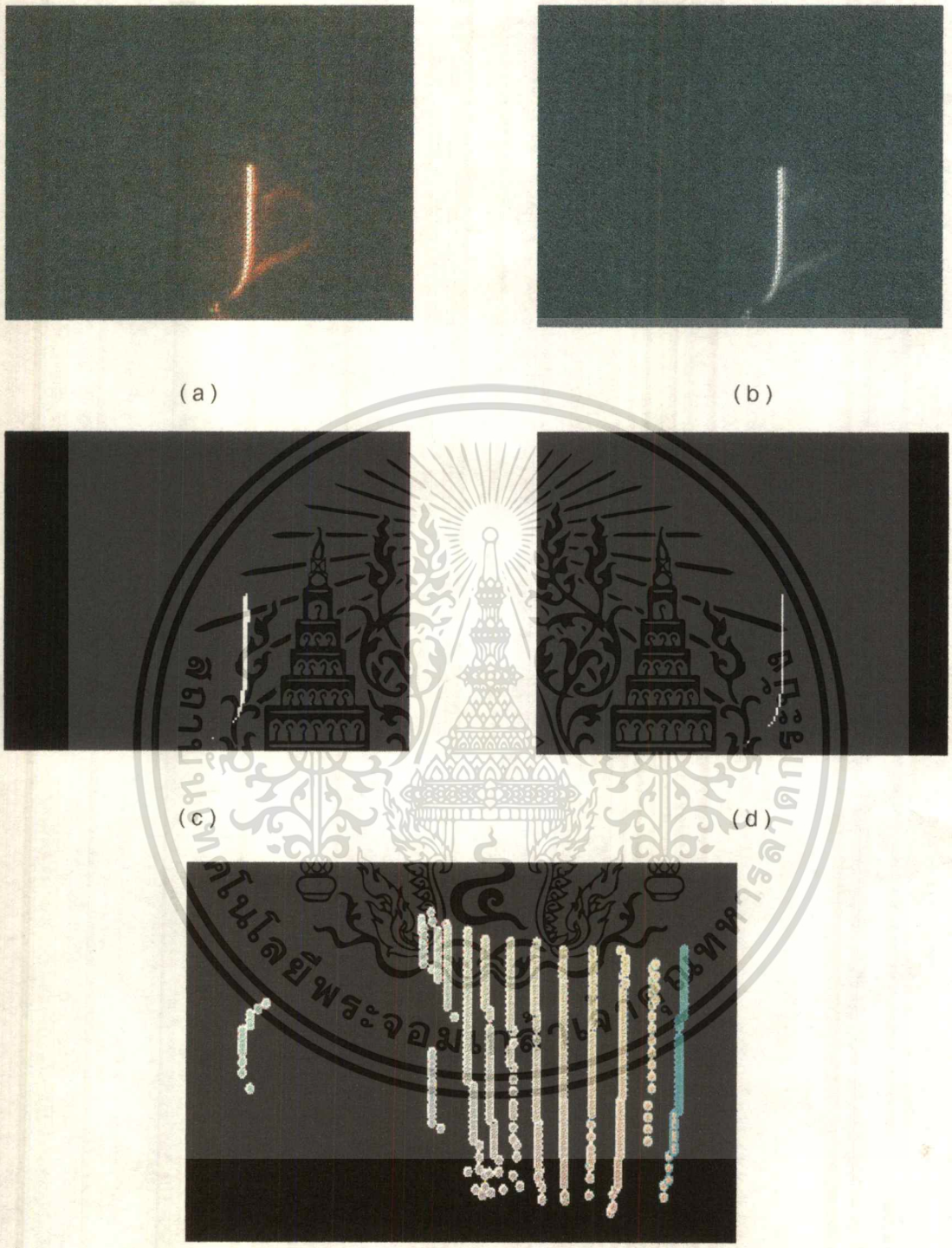
(e)

รูปที่ 9.16 แสดงภาพที่ได้จากการประมวลผลของภาพที่รับจากกล้องสเตปที่ 13

(a) ภาพที่ได้จากการ Capture (b) ภาพที่ได้จากการทำ Brightness

(c) ภาพที่ได้จากการทำ Threshold (d) ภาพที่ได้จากการทำ Thining

(e) ภาพโครงร่าง 3 มิติที่สร้างได้



(a) (b)

(c) (d)

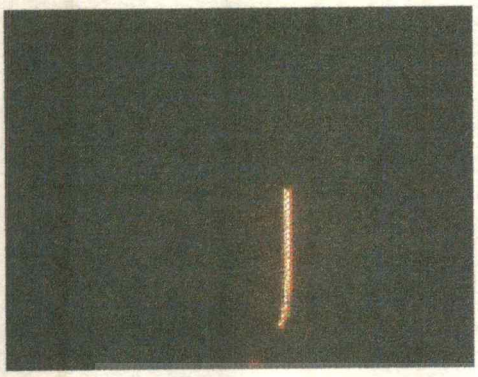
(e)

รูปที่ 9.17 แสดงภาพที่ได้จากการประมวลผลของภาพที่รับจากกล้องสเตปที่ 15

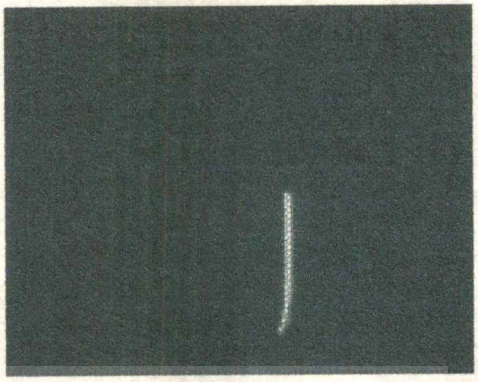
(a) ภาพที่ได้จากการ Capture (b) ภาพที่ได้จากการทำ Brightness

(c) ภาพที่ได้จากการทำ Threshold (d) ภาพที่ได้จากการทำ Thining

(e) ภาพโครงร่าง 3 มิติที่สร้างได้



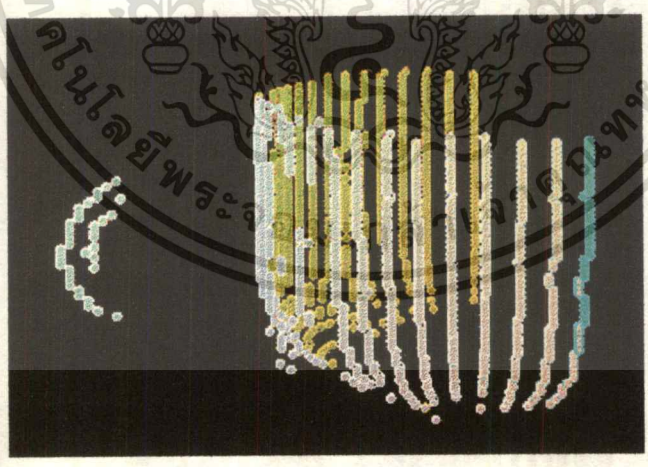
(a)



(b)

(c)

(d)



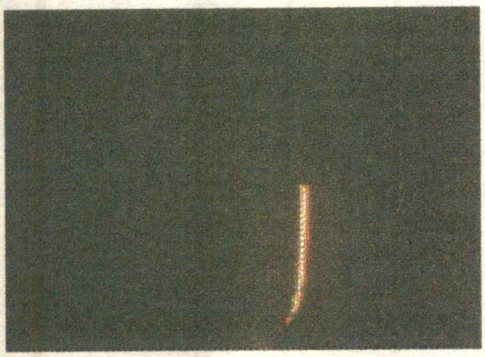
(e)

รูปที่ 9.18 แสดงภาพที่ได้จากการประมวลผลของภาพที่รับจากกล้องสเตปที่ 30

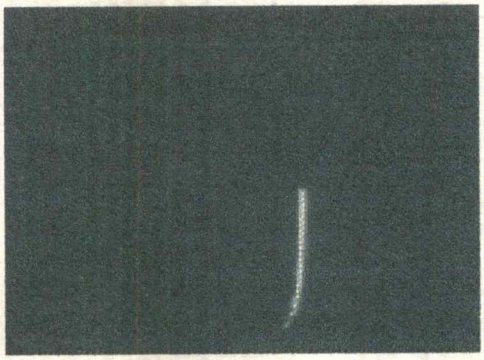
(a) ภาพที่ได้จากการ Capture (b) ภาพที่ได้จากการทำ Brightness

(c) ภาพที่ได้จากการทำ Threshold (d) ภาพที่ได้จากการทำ Thining

(e) ภาพโครงร่าง 3 มิติที่สร้างได้



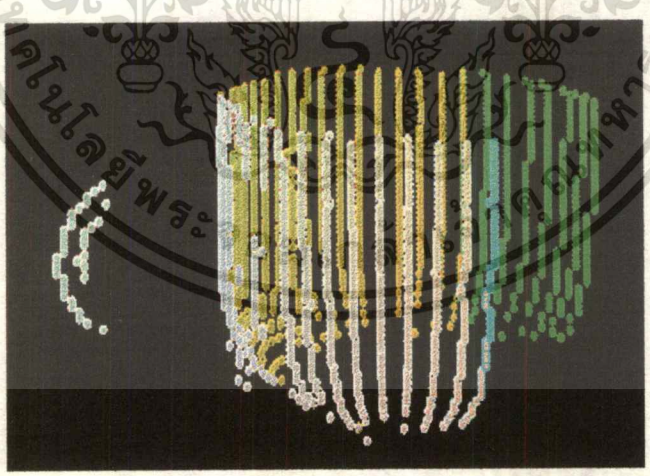
(a)



(b)

(c)

(d)



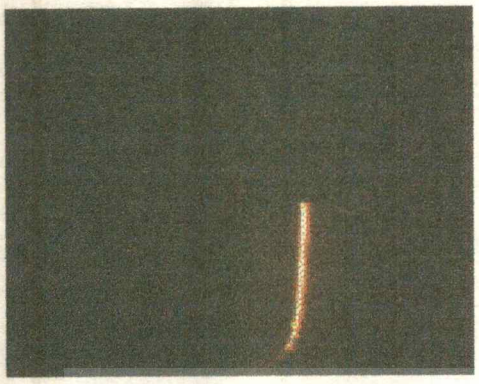
(e)

รูปที่ 9.19 แสดงภาพที่ได้จากการประมวลผลของภาพที่รับจากกล้องสเตปที่ 40

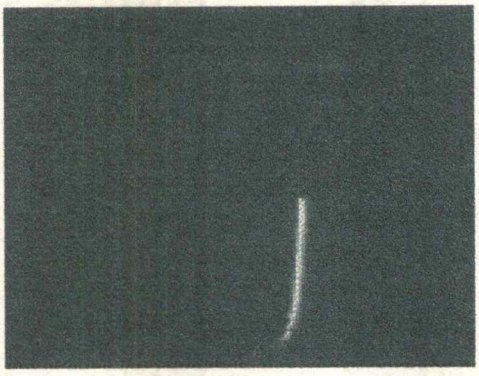
(a) ภาพที่ได้จากการ Capture (b) ภาพที่ได้จากการทำ Brightness

(c) ภาพที่ได้จากการทำ Threshold (d) ภาพที่ได้จากการทำ Thining

(e) ภาพโครงร่าง 3 มิติที่สร้างได้



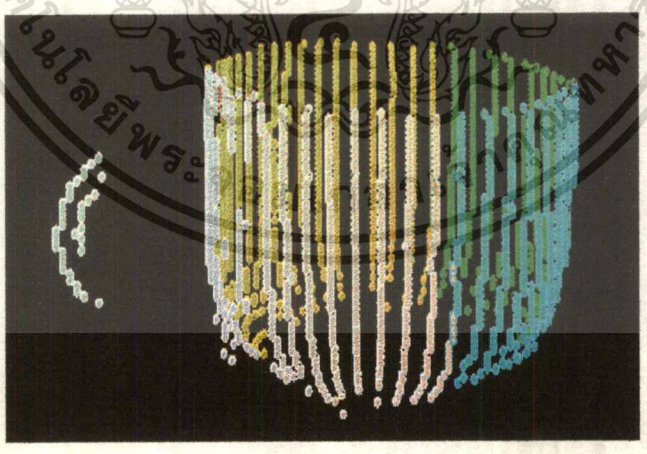
(a)



(b)

(c)

(d)



(e)

รูปที่ 9.20 แสดงภาพที่ได้จากการประมวลผลของภาพที่รับจากกล้องสเตปที่ 50

(a) ภาพที่ได้จากการ Capture (b) ภาพที่ได้จากการทำ Brightness

(c) ภาพที่ได้จากการทำ Threshold (d) ภาพที่ได้จากการทำ Thining

(e) ภาพโครงร่าง 3 มิติที่สร้างได้

บทที่10
สรุปผลและวิจารณ์ผลการทดลอง

10.1 สรุปผลและวิจารณ์ผลการทดลอง

จากการใช้วัตถุที่นำมาสแกน คือ ถ้วยกาแฟ เมื่อนำมาทำตามขั้นตอนต่างๆในวิธีการทดลองแล้วผลการทดลองสุดท้ายจะออกมาเป็นรูปร่างของถ้วยกาแฟซึ่งคล้ายกับวัตถุจริงมาก ดังรูปที่ 10.1

ก) ข)

รูปที่ 10.1 แสดงการเปรียบเทียบภาพวัตถุจริงกับภาพจากการสแกน
ก) ภาพวัตถุจริงที่รับมาจากกล้อง ข) ภาพจากการสแกน

ซึ่งผลที่ได้ในภาพจากการสแกนจะพบว่ายังมีบางเส้นที่ควรจะเป็นเส้นในแนวตรง แต่มันไม่เป็นเช่นนั้นเนื่องจากเราไม่สามารถควบคุมความกว้างของแนวลำแสงเลเซอร์ที่ยิงออกมาได้ จึงเป็นผลให้แนวเส้นเลเซอร์ที่ได้ออกมาจากการ Threshold และ Thining นั้นไม่เป็นเส้นตรงทีเดียว อาจจะมี error ในตำแหน่งต่างๆของ pixel บ้าง แต่ภาพโดยรวมที่ออกมาก็ยังคงความเป็นวัตถุจริงนั้นอยู่ซึ่งหากนำไปพัฒนาต่อไปไม่ว่าจะเป็นการทำ wireframe , rendering และอื่นๆ ก็อาจจะทำให้ภาพสแกนนั้นดูเสมือนวัตถุจริงมากยิ่งขึ้น

ผลการทดลองที่ได้เป็นไปตามขั้นตอนที่ต้องการ คือ ได้ภาพที่แต่ละมุมของวัตถุออกมาจากการควบคุมโดยโปรแกรมผ่านทางการ์ดพอร์ทขนานและวงจรขับสเตปมอเตอร์ทำให้สเตปมอเตอร์หมุนกล้องวิดีโอเพื่อ capture ภาพในแต่ละมุมจนครบรอบวัตถุ การออกแบบให้ใช้เลเซอร์ยิง

วัตถุจะทำให้การหาขอบภาพง่ายยิ่งขึ้นแต่เนื่องจากแผงยิงเลเซอร์นั้นมีราคาค่อนข้างสูงใน 3D-SCANNER เครื่องนี้จึงใช้ laser pointer มาดัดแปลงให้ได้เป็นแผงของลำแสงออกมา

10.2 สิ่งที่ดำเนินการในภาคเรียนที่ 1

1) ส่วนของ hardware

- ชุด Machanic สำหรับทางวัตถุที่จะสแกน
- ชุดกล้องวิดีโอ
- การ์ดควบคุม step motor

2) ส่วนของ software

- โปรแกรมควบคุมการหมุนของสเตปมอเตอร์
- โปรแกรมรับภาพจากกล้องและแสดงผล

10.3 สิ่งที่ดำเนินการในภาคเรียนที่ 2

1) ส่วนของ hardware

- ปรับปรุงชุด Machanic ให้มีโครงสร้างที่สมบูรณ์
- สร้างชุด Laser

2) ส่วนของ software

- โปรแกรม Threshold เพื่อ detect แนวเส้นเลเซอร์จากภาพ
- โปรแกรม Thining ให้เส้นเลเซอร์บางเท่ากัน
- โปรแกรม Keep Coordinate เพื่อเก็บพิกัดของเส้นเลเซอร์
- โปรแกรม 3DBubbleRender เพื่อนำ Coordinate ที่ได้มา plot แล้วแสดงผลบนหน้าจอมอนิเตอร์

10.4 ข้อจำกัดของโครงงาน

1) ความสูงของวัตถุ

เนื่องจากชุด Machanic นี้ต้องมีโครงสร้างที่มั่นคง เพื่อแก้ปัญหาการสั่นของชุด Machanic และปัญหาต่างๆที่จะเกิดตามมา อีกทั้งข้อจำกัดของกล้องในการรับภาพด้วย เพราะถ้าหากต้องการรับภาพของวัตถุที่มีความสูงมาก จะต้องให้ตำแหน่งของกล้องอยู่ห่างจากวัตถุพอสมควร ซึ่งต้องต่อแขนของกล้องออกไปอีกเป็นผลให้โครงสร้างมีขนาดใหญ่มาก ไม่สะดวกต่อการทดลอง ดังนั้นรูปแบบของโครงสร้างที่ใช้ในโครงงานนี้จึงมีข้อจำกัดในด้านความสูงของวัตถุด้วย

2) ความมืดในพื้นที่ของการทดลอง

เนื่องจากในโครงงานนี้ ต้องการ detect แนวเส้นเลเซอร์ที่ตกกระทบบนวัตถุ ซึ่งจะทำได้ง่ายและเร็วหากมีแสงสว่างบริเวณที่ทดลองน้อยหรือไม่มีเลย เพราะจะทำให้กล้องสามารถจับภาพแนวเส้นเลเซอร์นั้นได้อย่างชัดเจนและประมวลผลได้ง่าย ดังนั้นในโครงงานนี้จึงได้จำลองพื้นที่ของการทดลองให้มืดโดยการนำกล่องขนาดใหญ่มาครอบไว้

3) สีของวัตถุที่นำมาสแกน

เนื่องจากการทดลองนี้อยู่ในพื้นที่มืด ดังนั้นถ้าใช้วัตถุที่มีสีดำหรือสีที่ดูดกลืนแสงได้ดีก็จะทำให้ลำแสงเลเซอร์ที่ไปตกกระทบถูกดูดกลืนไปด้วย เป็นผลให้ไม่สามารถ detect แนวเส้นเลเซอร์จากภาพได้ ดังนั้นวัตถุที่จะนำมาสแกนควรจะเป็นสีขาวหรือสีอื่นๆ ที่ดูดกลืนแสงได้น้อยๆ

4) ความโค้ง-เว้าของวัตถุ

ถ้าวัตถุมีความโค้ง-เว้ามากเกินกว่ามุมของกล้องกับเลเซอร์ ก็อาจจะเป็นผลให้แนวเส้นเลเซอร์ถูกบังทำให้กล้องไม่สามารถรับภาพตรงส่วนที่ถูกบังได้ ดังนั้นจึงอาจทำให้แนวเส้นเลเซอร์ที่ได้จากการทำ Threshold และ Thining นั้นขาดหายไปเป็นผลให้เกิดความไม่สมบูรณ์ของรูปโครงร่าง 3 มิติได้

ภาคผนวก

โปรแกรมควบคุมการทำงานของเครื่องเก็บข้อมูลภาพ 3 มิติ

```
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
// sampleView.cpp : impiementation of the CSampleView class
////////////////////////////////////////////////////////

#include "stdafx.h"
#include "sample.h"
#include "VidCap.h"
//#include <math.h>
#include "3DClass.h"
#include "Camera.h"
#include "cdib.h"
#include "cdibtrue.h"
#include "cdibpsuedo.h"
#include "3DBubbleRender.h"
#include "sampleDoc.h"
#include "sampleView.h"
#include <stdio.h>
#include <conio.h>
#include "Conop.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

unsigned short port=0x280;
int icount=0;
int i;
int value=0x09;
bool Toggle=TRUE;
```

```
bool Round=FALSE;
bool Frame=TRUE;
CDC* hDC;
double result;
LPBITMAPINFO PicInfo;
LPVIDEOHDR lpVhr;
DWORD InfoHDRSize;
int Width;
int Height;
UINT BitPerPixel;
UINT Plane;
Conop Opob;
CFrameData FrameData[50];
////////////////////////////
//LPBITMAPINFOHEADER bmih;
////////////////////////////////////////////////////////////
// CSampleView
IMPLEMENT_DYNCREATE(CSampleView, CView)
BEGIN_MESSAGE_MAP(CSampleView, CView)
        //{{AFX_MSG_MAP(CSampleView)
        ON_WM_TIMER()
        ON_WM_LBUTTONDOWN()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////////////////////////////////////////
// CSampleView construction/destruction
CSampleView::CSampleView()
{
}
CSampleView::~CSampleView()
{
}
BOOL CSampleView::PreCreateWindow(CREATESTRUCT& cs)
{
```

```
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs
        return CView::PreCreateWindow(cs);
}
////////////////////////////////////////////////////////////////////////
// CSampleView drawing
void CSampleView::OnDraw(CDC* pDC)
{
        if (icount==50)
        {
        LPBITMAPFILEHEADER bmfh3 = m_p3DView->GetView()->ReceiveBMPFileHeader();
        LPBITMAPINFOHEADER bmih3 = m_p3DView->GetView()->ReceiveBMPInfoHeader();
        LPBYTE Dbmp = m_p3DView->GetView()->ReceivePic();
        int ColorTableEntries3 = 0;
        CString FileName3;
        FileName3.Format("c:\\Tdview\\Tdv-%d.bmp",icount);
        CFile WFile3(FileName3,CFile::modeCreate | CFile::modeWrite );
        int nSizeHdr3 = sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD) * ColorTableEntries3;
        WFile3.Write((LPVOID) bmfh3, sizeof(BITMAPFILEHEADER));
        WFile3.Write((LPVOID) bmih3,nSizeHdr3);
        WFile3.Write((LPVOID) Dbmp,bmih3->biSizeImage);
////////////////////////////////////////////////////////
        KillTimer(2);
        VidCap.DriverDisconnect();
        m_p3DView->GetView()->Display(pDC,CPoint(0,0));
        }
        else
        {
                if (icount != 0)
                {
                        BMP->Display(pDC,CPoint(0,50));
                        Pic1->Display(pDC,CPoint(0,200));
                        Pic2->Display(pDC,CPoint(400,50));
                        //m_p3DView->SetZoom(4.0f);
```

```
                    m_p3DView->GetView()->Stretch(pDC.CPoint(400,200),CSize(160,160));
                    //save to disk///
        LPBITMAPFILEHEADER bmfh3 = m_p3DView->GetView()->ReceiveBMPFileHeader();
        LPBITMAPINFOHEADER bmih3 = m_p3DView->GetView()->ReceiveBMPInfoHeader();
        LPBYTE Dbmp = m_p3DView->GetView()->ReceivePic();
        int ColorTableEntries3 = 0;

        CString FileName3;
        FileName3.Format("c:\\Tdview\\Tdv-%d.bmp",icount);
        CFile WFile3(FileName3,CFile::modeCreate | CFile::modeWrite );
        int nSizeHdr3 = sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD) * ColorTableEntries3;

        WFile3.Write((LPVOID) bmfh3, sizeof(BITMAPFILEHEADER));

        WFile3.Write((LPVOID) bmih3,nSizeHdr3);

        WFile3.Write((LPVOID) Dbmp,bmih3->biSizeImage);

                }
                char String[80];
                wsprintf(String,"Capture = %d",icount);
                pDC->TextOut(1,1,String);
        }
}
////////////////////////////////////////////////////////////////////////
// CSampleView diagnostics
#ifdef _DEBUG
void CSampleView::AssertValid() const
{
        CView::AssertValid();
}
////////////////////////////////////////////////////////////////////////
void CSampleView::Dump(CDumpContext& dc) const
{
```

```
        CView::Dump(dc);
}
CSampleDoc* CSampleView::GetDocument() // non-debug version is inline
{
        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CSampleDoc)));
        return (CSampleDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////
// CSampleView message handlers
void CSampleView::OnInitialUpdate()
{
        CView::OnInitialUpdate();
        ////////////////////
        VidCap.SetView((CView*)this);
        VidCap.Create(WS_CHILD | WS_VISIBLE,CRect(200,130,360,250),this);
        VidCap.DriverConnect(0);
        VidCap.AutoSize();
        VidCap.Preview(TRUE);
        //VidCap.CenterWindow();
        VidCap.SetCallbackOnFrame(FALSE);
        LPBITMAPINFO PicInfo;
        PicInfo = VidCap.GetVideoFormat();
        long Size;
        Width = PicInfo->bmiHeader.biWidth;
        Height = PicInfo->bmiHeader.biHeight;
        WORD BitPerPixel = PicInfo->bmiHeader.biBitCount;
        switch(BitPerPixel)
        {
        case 24:
                BMP = (CDib*) new CDibTrue(Width,Height);
                Size = sizeof(BITMAPINFOHEADER);
                break;
        case 8:
```

```
                BMP = (CDib*) new CDibPsuedo(Width,Height);
                Size = sizeof(BITMAPINFOHEADER) + 1024;
                break;
        }
        LPBITMAPINFO bmi = BMP->ReceiveBMI();
        Pic1 = (CDib*) new CDibPsuedo(Width,Height,GRAY);
        Pic2 = (CDib*) new CDibPsuedo(Width,Height,GRAY);
        Lght = (CDib*) new CDibPsuedo(Width,Height,GRAY);
        ////////////////////
        C3DVector EPoint(-28.0f,0.0f,-500.0f);
        C3DVector EDir(0.0f,0.0f,200.0f);
        C3DVector XAxeScreen(1.0f,0.0f,0.0f);
        C3DVector YAxeScreen(0.0f,1.0f,0.0f);
        CCamera Cam(EPoint,EDir,XAxeScreen,YAxeScreen,8.0f);
        m_p3DView = new C3DBubbleRender(640,640,Cam,FIXRADIUS,1.0f);
/*      LPBYTE IData = m_p3DView->GetView()->ReceivePic();
        LPBITMAPINFOHEADER lbmih = m_p3DView->GetView()->ReceiveBMPInfoHeader();
        int num = lbmih->biSizeImage;
        for (int t=1;t<num;t++)
        {
                *IData = 100;
                IData++;
        }
        */
        m_p3DView->ChangeToNormalDisplay();
        SetTimer(2,100,NULL);
}
void CSampleView::UpdateScreen(void)
{
        delete BMP;
        delete Pic1;
        delete Pic2;
        ////////////////////////////////////////////////////
        CString t1;
```

```
        t1.Format("Finished...Please strike any key!!!");
        AfxMessageBox(t1);
        Invalidate();
        //InvalidateRect(CRect(1,1,100,20),TRUE);
}
void MyVidClass::OnFrameCallback(LPVIDEOHDR lpVidHdr)
{
        SetCallbackOnFrame(FALSE);
        Preview(FALSE);
        KillTimer(2);
        icount++;
        CDib* BMP =  ((CSampleView*) View)->GetBitmap();          //return CDib* Pic
        CDib* Pic1 = ((CSampleView*) View)->GetBitmap1();         //return CDib* Pic1
        CDib* Pic2 = ((CSampleView*) View)->GetBitmap2();         //return CDib* Pic2
        CDib* Lght = ((CSampleView*) View)->GetBitmapLght();
        unsigned char* Des = BMP->ReceivePic();
        unsigned char* Src = lpVidHdr->lpData;
        unsigned char* SrcOld = Des;
        long Size;
        if(BMP->It24Bit())
                Size = BMP->ReceiveBMPInfoHeader()->biSizeImage;
        else
                Size = BMP->ReceiveWidth()*BMP->ReceiveHeight();
/*      ////////////////////////////////////////////////
        //Read from disk//////////////////////////
        BITMAPFILEHEADER filehd;
        LPBITMAPINFOHEADER infohd;
        CFile file;
        CString fName;
        fName.Format("c:\\Tdview\\Testpic.bmp");
        file.Open(fName,CFile::modeRead);
        file.Read((LPVOID) &filehd,sizeof(BITMAPFILEHEADER));
        if (filehd.bfType != 0x4d42)
        {
```

```
        AfxMessageBox("This is not BMP file");
        }
        int nSize = filehd.bfOffBits - sizeof(BITMAPFILEHEADER);
        infohd = (LPBITMAPINFOHEADER) new char[nSize];
    file.Read(infohd,nSize);
        if(infohd->biSize != sizeof(BITMAPINFOHEADER))
        {
                AfxMessageBox("Not valid Windows BITMAP");
        }
        int Size = 57600;//infohd->biSizeImage;
        if(Size == 0)
        {
                DWORD dwBytes = ((DWORD) infohd->biWidth * infohd->biBitCount) / 32;
                if(((DWORD) infohd->biWidth * infohd->biBitCount) % 32)
                {
                        dwBytes++;
                }
                dwBytes *= 4;
//              lSize = dwBytes * infohd->biHeight; // no compression
        }
                LPBYTE Src = new unsigned char[57600];
                file.Read(Src,Size);
                file.Close();
*/
        ////////Threshold Section//////////////////
        unsigned char* Threshold;
        unsigned char* Original;
        unsigned char* LprtOld;
        Threshold = new unsigned char[Width*Height];
//      LightData = new unsigned char[Width*Height];
        unsigned char* Lprt = Lght->ReceivePic();
        LprtOld = Lprt;
        Original = Threshold;
        int i;
```

```
        for(i=0;i<Size;i++)                                    //copy Image
        {
                *Des = *Src;
                Des++;
                Src++;
        }
        Des  = SrcOld;
////////////////////////////////////////////////
        ///Save to disk///
        BITMAPFILEHEADER bmfh;
   LPBITMAPINFOHEADER bmih = new BITMAPINFOHEADER;
        bmih->biSize=40;
        bmih->biWidth=160;
        bmih->biHeight=120;
        bmih->biPlanes=1;
        bmih->biBitCount=24;
        bmih->biCompression=0;
        bmih->biSizeImage=57600;
        bmih->biXPelsPerMeter=0;
        bmih->biYPelsPerMeter=0;
        bmih->biClrUsed=0;
        bmih->biClrImportant=0;
        int ColorTableEntries = 0;

        CString FileName;
        FileName.Format("c:\\Rawpic\\RawNo-%d.bmp",icount);
        CFile WFile(FileName,CFile::modeCreate | CFile::modeWrite );
        bmfh.bfType = 0x4d42;
        int nSizeHdr = sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD) * ColorTableEntries;
        bmfh.bfSize = 0;
        bmfh.bfReserved1 = bmfh.bfReserved2 = 0;
        bmfh.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) +
                        sizeof(RGBQUAD) * ColorTableEntries;
```

```
        WFile.Write((LPVOID) &bmfh, sizeof(BITMAPFILEHEADER));

        WFile.Write((LPVOID) bmih,nSizeHdr);

        WFile.Write((LPVOID) Des,Size);
    delete bmih;
        Des = SrcOld;
////////////////////////////////////////////////
        BYTE Red;
        BYTE Green;
        BYTE Blue;
        unsigned char Lightness;
        int j;
        for (j=0;j<(Size/3);j++)          //Lightness convert
        {
            Red = *Des;
            Des++;
            Green = *Des;
            Des++;
            Blue = *Des;
            Des++;
            Lightness = (Red+Green+Blue)/3;
            *Lprt = Lightness;
            Lprt++;

            if ((Lightness>=0)&&(Lightness<200))
            {
                *Threshold = 0;
            }
            else
                *Threshold = 255;
            Threshold++;
        }
```

```
Threshold = Original;
Des = SrcOld;
Lprt = LprtOld;
unsigned char* Des1 = Pic1->ReceivePic();
unsigned char* Src1 = Threshold;
unsigned char* Des2 = Pic2->ReceivePic();
unsigned char* OldDes1 = Des1;
unsigned char* OldDes2 = Des2;
long n;
for(n=0;n<(Size/3);n++)`                    //copy threshold Image
{
        *Des1 = *Src1;
        *Des2 = *Src1;
        Des1++;
        Des2++;
        Src1++;
}
Des1 = OldDes1;
Des2 = OldDes2;
delete Threshold;
////////////////////////////////////
///Save to disk Lightness///
LPBITMAPFILEHEADER Lghtfh = Lght->ReceiveBMPFileHeader();
LPBITMAPINFOHEADER Lghtih = Lght->ReceiveBMPInfoHeader();
int ColorTable = 256;

CString LghtFileName;
LghtFileName.Format("c:\\Light\\Lght-%d.bmp",icount);
CFile LghtFile(LghtFileName,CFile::modeCreate | CFile::modeWrite );

int SizeHdr = sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD) * ColorTable;
LghtFile.Write((LPVOID) Lghtfh, sizeof(BITMAPFILEHEADER));
```

```
            LghtFile.Write((LPVOID) Lghtih,SizeHdr);

            LghtFile.Write((LPVOID) Lprt,19200);
////////////////////////////////////////////////////////////
            ///Save to disk Treshold///
            LPBITMAPFILEHEADER bmfh1 = Pic1->ReceiveBMPFileHeader();
    LPBITMAPINFOHEADER bmih1 = Pic1->ReceiveBMPInfoHeader();
            int ColorTableEntries1 = 256;

            CString FileName1;
            FileName1.Format("c:\\Thold\\ThNo-%d.bmp",icount);
            CFile WFile1(FileName1,CFile::modeCreate | CFile::modeWrite );
//          bmfh.bfType = 0x4d42;
            int nSizeHdr1 = sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD) * ColorTableEntries1;
//          bmfh.bfSize = 0;
//          bmfh.bfReserved1 = bmfh.bfReserved2 = 0;
//          bmfh.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) +
//                           sizeof(RGBQUAD) * ColorTableEntries;

            WFile1.Write((LPVOID) bmfh1, sizeof(BITMAPFILEHEADER));

            WFile1.Write((LPVOID) bmih1.nSizeHdr1);

            WFile1.Write((LPVOID) Des1,19200);
            Des1 = OldDes1;
            ////////////////////////////////////
            ///////////Thinning Section//////////
            long x,y,z;
            long WhitePoint=0;
            bool White;
            int LaserPoint = 0;
            unsigned char* StartPoint;
            long StartCoor;
```

```
int MidDes2;
for (y=0;y<Height;y++)
{
        White = FALSE;
        for(x=0;x<Width;x++)
        {
                if(*Des2==255)
                {
                        White = TRUE;
                        LaserPoint++;
                        if(LaserPoint==1)
                        {
                                StartPoint=Des2;
                                StartCoor = x;
                        }
                }
                else
                        White = FALSE;

                if(((LaserPoint>1)&&(White==FALSE))||((LaserPoint>1)&&(x==Width-1)))
                {
                        MidDes2 = LaserPoint/2;
                        for(z=0;z<LaserPoint;z++)
                                if(z!=MidDes2)
                                        *(StartPoint+z) = 0;
                        WhitePoint++;
                        LaserPoint = 0;
                }

                if((LaserPoint==1)&&(White==FALSE))
                {
                        WhitePoint++;
                        LaserPoint = 0;
```

```
                    }
                    Des2++;
                }
            }
            Des2 = OldDes2;
            ////////////////////////////////////////
            ///Save to disk///
            LPBITMAPFILEHEADER bmfh2 = Pic2->ReceiveBMPFileHeader();
            LPBITMAPINFOHEADER bmih2 = Pic2->ReceiveBMPInfoHeader();
            int ColorTableEntries2 = 256;

            CString FileName2;
            FileName2.Format("c:\\Thin\\ThinNo-%d.bmp",icount);
            CFile WFile2(FileName2,CFile::modeCreate | CFile::modeWrite );
//          bmfh.bfType = 0x4d42;
            int nSizeHdr2 = sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD) * ColorTableEntries2;
//          bmfh.bfSize = 0;
//          bmfh.bfReserved1 = bmfh.bfReserved2 = 0;
//          bmfh.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) +
//                           sizeof(RGBQUAD) * ColorTableEntries;

            WFile2.Write((LPVOID) bmfh2, sizeof(BITMAPFILEHEADER));

            WFile2.Write((LPVOID) bmih2,nSizeHdr2);

            WFile2.Write((LPVOID) Des2,19200);//Size
            Des2 = OldDes2;

            ////////////////////////////////////////
            ////////////Store in Array////////////
            LPBYTE Thsorce=Des2;
            BYTE ThImage[160][120];
            BYTE MapImage[160][120];
            int H,W;
```

```
for (H=0;H<Height;H++)
{
        for (W=0;W<Width;W++)
        {
                ThImage[W][H] = *Thsorce;
                MapImage[W][H]=0;
                Thsorce++;
        }
}
Thsorce=Des2;
////////////////////////////////////////
int Zc=40;
int Ze=300;
double pi = 3.14159265359;
double CLAngle=(pi/180.0)*(90-25);
double StepAngle = (pi/180.0)*(360.0/50.0)*(icount-1);
C3DVector* Coor = new C3DVector[WhitePoint];
FrameData[icount-1].CoorMap(Coor);
int NoWhP=0;
/////////////Keep Coordinate/////////////
for (H=0;H<Height;H++)
{
        for (W=0;W<Width;W++)
        {
                if (ThImage[W][H] ==255)
                {
                        int TrX1,TrY1;
                        double TA,Vx,Vy,Vz,RtX1,RtY1,RtZ1;
                        ///////////Translate origin to center of image////////
                        TrX1=W-Width/2;
                        TrY1=H-Height/2;
                        /*CString s;
                        s.Format("W=%d,H=%d,TrX1=%d,TrX2=%d",W,H,TrX1,TrY1);
                        AfxMessageBox(s);*/
```

```
                    ////////////Find constant of line funtion//////////////
                TA = (Ze*cos(CLAngle))/(((Ze-Zc)*cos(CLAngle))+(TrX1*sin(CLAngle)));
                    Vx = TA*TrX1;
                    Vy = TA*TrY1;
                    Vz = ((TA-1)*Ze)-(TA*Zc);
                    ////////////Rotate about y axis follow motor step///////
                    RtX1 = Vx*cos(StepAngle)+Vz*sin(StepAngle);
                    RtY1 = Vy;
                    RtZ1 = Vz*cos(StepAngle)-Vx*sin(StepAngle);
                    ////////////Store coordinate/////////////////////////////
                    /*CString s1;

        s1.Format("W=%d,H=%d,TrX1=%d,TrX2=%d,TA=%f,Vx=%f,Vy=%f,Vz=%f,RtX1=%f,RtY1=
%f,RtZ1=%f",W,H,TrX1,TrY1,TA,Vx,Vy,Vz,RtX1,RtY1,RtZ1);
                    AfxMessageBox(s1);*/
                    ////////////////////////////////////////////////////
                    double crx=RtX1-20;
                    double cry=RtY1-5;
                    double crz=RtZ1;
                    ////////////////////////////////////////////////////
                ((CSampleView*)View)->m_p3DView->ComputeRender(crx,cry,crz);
                    FrameData[icount-1].IncCoor(Coor,NoWhP);
                    //FrameData[icount-1].PutCoor(RtX1,RtY1,RtZ1,Coor,NoWhP);
                    NoWhP++;
                }
            }
        }
        ////////////////////////////////////////////
        //////////3DViewTest//////////////
/*      double Xvalue,Yvalue,Zvalue;
        for(x=0;x<WhitePoint;x++)
        {
            Xvalue = (FrameData[icount-1]).GetVectorX();
            Yvalue = (FrameData[icount-1]).GetVectorY();
```

```
                Zvalue = (FrameData[icount-1]).GetVectorZ();
                ////////////////////////////////////////
                CString s2;
                s2.Format("Xvalue=%f,Yvalue=%f,Zvalue=%f",Xvalue,Yvalue,Zvalue);
                AfxMessageBox(s2);
                ////////////////////////////////////////
                ((CSampleView*)View)->m_p3DView->ComputeRender(Xvalue,Yvalue,Zvalue);
                FrameData[icount-1].IncCoor(Coor,x);
        }*/
        FrameData[icount-1].CoorMap(Coor);
        //((CSampleView*) View)->InvalidateRect(NULL,FALSE);
        Preview(TRUE);
////////////////////////////////////////////////
        if (icount==50)
        {
                ((CSampleView*) View)->InvalidateRect(NULL,FALSE);
                ((CSampleView*) View)->UpdateScreen();
                KillTimer(2);
        }
        else
        {
                ((CSampleView*) View)->InvalidateRect(NULL,FALSE);
                SetTimer(2,50,NULL);
        }
}
void CSampleView::OnTimer(UINT nIDEvent)
{
        if(icount==50)
        {
                KillTimer(2);
                VidCap.SetCallbackOnFrame(FALSE);
                VidCap.Preview(TRUE);
                GetDC()->TextOut(10,20,"End Task..... :) ");
                return;
```

```
}
else
{
if(Toggle==FALSE)
        {
                if(Round==TRUE)
                {
                        Toggle=TRUE;
                        Round=FALSE;
                }
                else
                {
                        GetDC()->TextOut(10,20,"Change Capture Position");
                        _outp(port,value);
                        //value<<=1;
                        switch(value)
                        {
                        case(0x01):
                                value=0x03;
                                break;
                        case(0x03):
                                value=0x02;
                                break;
                        case(0x02):
                                value=0x06;
                                break;
                        case(0x06):
                                value=0x04;
                                break;
                        case(0x04):
                                value=0x0c;
                                break;
                        case(0x0c):
                                value=0x08;
```

```
                    break;
                case(0x08):
                    value=0x09;
                    break;
                case(0x09):
                    value=0x01;
                    Round=TRUE;
                }
            }
        }
    else
        {
        Toggle = FALSE;
        KillTimer(2);
        GetDC()->TextOut(10,20,"CAPTURE!!!!");
        VidCap.SetCallbackOnFrame(TRUE);
        SetTimer(2,50,NULL);
        }
    }
    CView::OnTimer(nIDEvent);
}
```

```
////////////////////////////////////////////////////////////////////
// sampleView.h : interface of the CSampleView class
////////////////////////////////////////////////////////////////////
class CFrameData
{
private:
        C3DVector* Coordinate;
public:
        void CoorMap(C3DVector* Coorprt){Coordinate = Coorprt;}
        void PutCoor(double x,double y,double z,C3DVector* Org,int Index)
        {
                Coordinate=Org+Index;
                Coordinate->Passvalue(x,y,z);
        }
        void IncCoor(C3DVector* Org,int Index)
        {
                Index++;
                Coordinate=Org+Index;
        }
        double GetVectorX(void)
        {
                return(Coordinate->GetX());
        }
        double GetVectorY(void)
        {
                return(Coordinate->GetY());
        }
        double GetVectorZ(void)
        {
                return(Coordinate->GetZ());
        }
};
```

```
class MyVidClass : public CVidCap
{
private:
        CView* View;

public:
        void SetView(CView* V) { View = V;}
        virtual void OnFrameCallback(LPVIDEOHDR lpVidHdr);
};

class CSampleView : public CView
{
private:
        CDib* BMP;
        CDib* Lght;
        CDib* Pic1;
        CDib* Pic2;
        CPoint* pt;
        CFrameData* Data;
        double dx1,dy1,dz1,dx2,dy2,dz2;
        CCamera cam;
protected: // create from serialization only
        CSampleView();
        DECLARE_DYNCREATE(CSampleView)

// Attributes
public:
        C3DBubbleRender* m_p3DView;
        C3DBubbleRender* m_p3DView1;

        MyVidClass VidCap;
        CSampleDoc* GetDocument();
        CDib* GetBitmap(void) { return BMP;}
        CDib* GetBitmap1(void) { return Pic1;}
```

```
        CDib* GetBitmap2(void) { return Pic2;}
        CDib* GetBitmapLght(void) {return Lght;}
        void UpdateScreen(void);
        void SetBubbleRender(double x,double y,double z,double zoom);
        long Length(long X1,long Y1,long Z1)
        {
                return ((long) sqrt(X1*X1+Y1*Y1+Z1*Z1));
        }
        void TrueCoor(int Frame,double degree,long X1,long Y1,long Z1,double A)
        {
                dx1 = (A*X1/Length(X1,Y1,Z1));
                dy1 = (A*Y1/Length(X1,Y1,Z1));
                dz1 = ((A*Z1/Length(X1,Y1,Z1))+Z1);

                dx2 = dx1*cos((Frame-1)*degree)+dz1*sin((Frame-1)*degree);
                dy2 = dy1;
                dz2 = -dx1*sin((Frame-1)*degree)+dz1*cos((Frame-1)*degree);
        }

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CSampleView)
        public:
        virtual void OnDraw(CDC* pDC);  // overridden to draw this view
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        virtual void OnInitialUpdate();
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CSampleView();
```

```
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
        //{{AFX_MSG(CSampleView)
        afx_msg void OnTimer(UINT nIDEvent);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in sampleView.cpp
inline CSampleDoc* CSampleView::GetDocument()
   { return (CSampleDoc*)m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////
// sample.cpp : Def
//3ines the class behaviors for the application.
////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "sample.h"

//#include "3DBubbleRender.h"
//#include <math.h>
#include "VidCap.h"
#include "3DClass.h"
#include "MainFrm.h"
#include "sampleDoc.h"
#include "cdib.h"
#include "cdibtrue.h"
#include "Camera.h"
#include "3DBubbleRender.h"
#include "sampleView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////
// CSampleApp

BEGIN_MESSAGE_MAP(CSampleApp, CWinApp)
        //{{AFX_MSG_MAP(CSampleApp)
        ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
```

```
        //}}AFX_MSG_MAP
        // Standard file based document commands
        ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
        ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSampleApp construction

CSampleApp::CSampleApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CSampleApp object

CSampleApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CSampleApp initialization

BOOL CSampleApp::InitInstance()
{
        // Standard initialization
        // If you are not using these features and wish to reduce the size
        //  of your final executable, you should remove from the following
        //  the specific initialization routines you do not need.

#ifdef _AFXDLL
        Enable3dControls();                     // Call this when using MFC in a shared DLL
#else
        Enable3dControlsStatic(); // Call this when linking to MFC statically
```

```
#endif

        LoadStdProfileSettings();  // Load standard INI file options (including MRU)

        // Register the application's document templates.  Document templates
        //  serve as the connection between documents, frame windows and views.

        CSingleDocTemplate* pDocTemplate;
        pDocTemplate = new CSingleDocTemplate(
                IDR_MAINFRAME,
                RUNTIME_CLASS(CSampleDoc),
                RUNTIME_CLASS(CMainFrame),       // main SDI frame window
                RUNTIME_CLASS(CSampleView));
        AddDocTemplate(pDocTemplate);

        // Parse command line for standard shell commands, DDE, file open
        CCommandLineInfo cmdInfo;
        ParseCommandLine(cmdInfo);

        // Dispatch commands specified on the command line
        if (!ProcessShellCommand(cmdInfo))
                return FALSE;

        return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();
```

```
// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
                // No message handlers
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
```

```
                // No message handlers
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CSampleApp::OnAppAbout()
{
        CAboutDlg aboutDlg;
        aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// CSampleApp commands
```

```
////////////////////////////////////////////////////////
// Conop.cpp
////////////////////////////////////////////////////////

#include "StdAfx.h"
#include "Conop.h"
////////////////////////////////////
Conop::Conop()
{
        bmih = NULL;
//      Image = NULL;
}
////////////////////////////////////////
Conop::~Conop()
{
        delete bmih;
        delete Image;
}
////////////////////////////////////////
void Conop::Initial(void)
{
        delete bmih;
        delete Image;
}
////////////////////////////////////////
BOOL Conop::bmpRead(void)
{
        CFileDialog dlg(TRUE,"bmp","*.bmp");
        if (dlg.DoModal() != IDOK)return(0);
        file.Open(dlg.GetPathName(),CFile::modeRead);
//      CString FileName;
//      FileName.Format("c:\\Myprojects\\No-%d.bmp",icount+1);
//      file.Open(FileName,CFile::modeRead);
        file.Read((LPVOID) &bmfh,sizeof(BITMAPFILEHEADER));
        if (bmfh.bfType != 0x4d42)
```

```
    {
      AfxMessageBox("This is not BMP file");
                return(0);
        }
        return(1);
}
BOOL Conop::SizeImage(void)
{
   int nSize = bmfh.bfOffBits - sizeof(BITMAPFILEHEADER);
        bmih = (LPBITMAPINFOHEADER) new char[nSize];
   file.Read(bmih,nSize);
        if(bmih->biSize != sizeof(BITMAPINFOHEADER))
        {
                AfxMessageBox("Not valid Windows BITMAP");
        }
        sizex=bmih->biWidth;
        sizey=bmih->biHeight;
        int ISize = bmih->biSizeImage;
        if(ISize == 0)
        {
                DWORD dwBytes = ((DWORD) bmih->biWidth * bmih->biBitCount) / 32;
                if(((DWORD) bmih->biWidth * bmih->biBitCount) % 32)
                {
                        dwBytes++;
                }
                dwBytes *= 4;
                ISize = dwBytes * bmih->biHeight; // no compression
        }
                iSize = ISize;
                Image=(LPBYTE) new char[ISize];
                file.Read(Image,ISize);
                file.Close();
                return(1);
}
//////////////////////////////////////////////
```

```
BOOL Conop::CreatbmpPalete(void)
{
	LPBYTE ColorTable = (LPBYTE)bmih + sizeof(BITMAPINFOHEADER);
	if((bmih == NULL) || (bmih->biClrUsed == 0)) {
		switch(bmih->biBitCount) {
			case 1:
				ColorTableEntries = 2;
				break;
			case 4:
				ColorTableEntries = 16;
				break;
			case 8:
				ColorTableEntries = 256;
				break;
			case 16:
			case 24:
			case 32:
				ColorTableEntries = 0;
				break;
			default:
				return(FALSE);
		}
	}
	else {
		ColorTableEntries = bmih->biClrUsed;
	}
	ASSERT((ColorTableEntries >= 0) && (ColorTableEntries <= 256));
	////////////////////////////////////////////////////
	LPLOGPALETTE pLogPal = (LPLOGPALETTE) new char[2 * sizeof(WORD) +
		ColorTableEntries * sizeof(PALETTEENTRY)];
	pLogPal->palVersion = 0x300;
	pLogPal->palNumEntries = ColorTableEntries;
	LPRGBQUAD pDibQuad = (LPRGBQUAD) ColorTable;
	for(int i = 0; i < ColorTableEntries; i++) {
		pLogPal->palPalEntry[i].peRed = pDibQuad->rgbRed;
```

```
                pLogPal->palPalEntry[i].peGreen = pDibQuad->rgbGreen;
                pLogPal->palPalEntry[i].peBlue = pDibQuad->rgbBlue;
                pLogPal->palPalEntry[i].peFlags = 0;
                pDibQuad++;
        }
        Palette = ::CreatePalette(pLogPal);
        delete pLogPal;
        return(TRUE);
}
////////////////////////////////////
BOOL Conop::bmpWrite(int aSize)
{
        CFileDialog dlg(FALSE,"bmp","*.bmp");
        if (dlg.DoModal() != IDOK)return(0);
        CFile WFile( dlg.GetPathName(), CFile::modeCreate | CFile::modeWrite );
        bmfh.bfType = 0x4d42;
        int nSizeHdr = sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD) * ColorTableEntries;
        bmfh.bfSize = 0;
//      bmfh.bfSize = sizeof(BITMAPFILEHEADER) + nSizeHdr + SizeImage;
        bmfh.bfReserved1 = bmfh.bfReserved2 = 0;
        bmfh.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) +
                        sizeof(RGBQUAD) * ColorTableEntries;

        WFile.Write((LPVOID) &bmfh, sizeof(BITMAPFILEHEADER));

        WFile.Write((LPVOID) bmih,  nSizeHdr);

        WFile.Write((LPVOID) Image,aSize);
        return(1);
}
////////////////////////////////////
CPoint Conop::Origin(int x,int y)
{
        point=(x,y);
        return(point);
```

```
}
//////////////////////////////////////
int Conop::GetSize(void)
{
        return(iSize);
}
//////////////////////////////////////
int Conop::Getsx(void)
{
        return(sizex);
}
//////////////////////////////////////
int Conop::Getsy(void)
{
        return(sizey);
}
//////////////////////////////////////
LPBYTE Conop::SentImage(void)
{
        return(Image);
}
//////////////////////////////////////
HPALETTE Conop::GetPalette(void)
{
        return(Palette);
}
//////////////////////////////////////
BOOL Conop::Draw(CDC* pdc,LPBYTE ImagePT,int sx,int sy)
{
        pdc->SetStretchBltMode(COLORONCOLOR);
        ::StretchDIBits(pdc->GetSafeHdc(),point.x,point.y,
                sx,sy,0,0,sx,sy,ImagePT,(LPBITMAPINFO) bmih,
                DIB_RGB_COLORS,SRCCOPY);
        return(1);
}
```

```
////////////////////////////////////////////////
// Conop.h
////////////////////////////////////////////////
#include "stdafx.h"

class Conop
{
private:
        CFile file;
        BITMAPFILEHEADER bmfh;
        LPBITMAPINFOHEADER bmih;
        LPBYTE Image;
        HPALETTE Palette;
        int ColorTableEntries;
        CPoint point;
        int sizex,sizey;
        int iSize;
public:
        Conop();
        ~Conop();
        void Initial(void);
        BOOL bmpRead(void);
        BOOL SizeImage(void);
        BOOL CreatbmpPalete(void);
        BOOL bmpWrite(int aSize);
        CPoint Origin(int x,int y);
        int GetSize(void);
        int Getsx(void);
        int Getsy(void);
 HPALETTE GetPalette(void);
        LPBYTE Sentimage(void);
        BOOL Draw(CDC* pdc,LPBYTE ImagePT,int sx,int sy);
        LPBITMAPINFOHEADER GetBMIH(void){return bmih;}
};
```

```
////////////////////////////////////////////////////////////
// Cdib.cpp
////////////////////////////////////////////////////////////

#include "stdafx.h"
//#include "BinaryTree.h"
#include "cdib.h"
#include <windowsx.h>  // for GlobalAllocPtr

IMPLEMENT_SERIAL(CDib, CObject, 0)

////////////////////////////////////////////////////////////
CDib::CDib()
{
   m_dwLength = 0L;
   m_nBits    = 0;
   m_lpBuf    = NULL;
}

BOOL CDib::MakeDib(long x,long y,int Bits)
{
        DWORD Length;

        long LUTsize;
        long Width;
        switch(Bits)
        {
                case 24:
                        LUTsize = 0;
                        Width = x*3L;
                        break;
                case 8:
                        LUTsize = sizeof(RGBQUAD)*256L;
                        Width = x;
```

```
                    break;
                default: return FALSE;
            }
            if(Width&3) Width = (Width&0xfffffc)+4;

            Length = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) + LUTsize + (Width
* y);
            m_nBits = Bits;
            m_dwLength = Length;
            if(!AllocateMemory()) return FALSE;

            m_lpBMFH->bfType = 0x4d42;
            m_lpBMFH->bfSize = Length;
            m_lpBMFH->bfReserved1 = 0;
            m_lpBMFH->bfReserved2 = 0;
            m_lpBMFH->bfOffBits = sizeof(BITMAPFILEHEADER) + LUTsize
+sizeof(BITMAPINFOHEADER);

            m_lpBMIH->biSize = sizeof(BITMAPINFOHEADER);
            m_lpBMIH->biWidth = x;
            m_lpBMIH->biHeight = y;
            m_lpBMIH->biPlanes = 1;
            m_lpBMIH->biBitCount = Bits;
            m_lpBMIH->biCompression = BI_RGB;
            if(Bits==24) m_lpBMIH->biSizeImage = Width*y;
            else m_lpBMIH->biSizeImage = 0;
            m_lpBMIH->biXPelsPerMeter = 0;
            m_lpBMIH->biYPelsPerMeter = 0;
            m_lpBMIH->biClrUsed = 0;
            m_lpBMIH->biClrImportant = 0;

            m_lpData = m_lpBuf+m_lpBMFH->bfOffBits;
            m_lpPic = (unsigned char*) m_lpData;
            return TRUE;
```

```
}

//////////////////////////////////////////////////////////////////////
CDib::~CDib()
{
   if (m_lpBuf) {
      GlobalFreePtr(m_lpBuf);  // free the DIB memory
   }
}

//////////////////////////////////////////////////////////////////////
void CDib::Serialize(CArchive& ar)
{
   WORD wBM;

   CObject::Serialize(ar);
   if (ar.IsStoring()) {
      ar.GetFile()->WriteHuge(m_lpBuf, m_dwLength);
   }
   else {
                  if(m_dwLength!=0)
                        DeleteData();
      ASSERT(m_dwLength == 0L); // DIB must be empty
      ar >> wBM;
      if (wBM == (WORD) 0x4d42) {
         ar.Flush();
         ar.GetFile()->SeekToBegin();
         m_dwLength = ar.GetFile()->GetLength();
                        if(m_dwLength > 0L)
                        {
            if (!AllocateMemory()) {
            return;
         }
                                    ar.Flush();
```

```
            ar.GetFile()->ReadHuge(m_lpBuf, m_dwLength);
                m_lpData = (unsigned char*) m_lpBMFH + m_lpBMFH->bfOffBits;
                    m_lpPic = (unsigned char*) m_lpBMFH + m_lpBMFH->bfOffBits;
            m_nBits = m_lpBMIH->biBitCount;
          }
        }
        else {
          AfxMessageBox("Invalid DIB archive");
          m_dwLength = 0L;
          m_nBits = 0;
        }
      }
    }

    ////////////////////////////////////////////////////////////
    BOOL CDib::Display(CDC* pDC, CPoint origin)
    {
      // direct to device--bypass the GDI bitmap
      if (!m_lpBuf) {
        return FALSE; // nothing to display
      }
      if (!::SetDIBitsToDevice(pDC->GetSafeHdc(), origin.x, origin.y,
        (WORD) m_lpBMIH->biWidth, (WORD) m_lpBMIH->biHeight, 0, 0, 0,
        (WORD) m_lpBMIH->biHeight, m_lpData, m_lpBMI,
        DIB_RGB_COLORS)) {
        return FALSE;
      }
      return TRUE;
    }

    ////////////////////////////////////////////////////////////
    BOOL CDib::Stretch(CDC* pDC, CPoint origin, CSize size)
    {
      // direct to device--bypass the GDI bitmap
```

```
    if (!m_lpBuf) {
        return FALSE; // nothing to display
    }
    if (!::StretchDIBits(pDC->GetSafeHdc(), origin.x, origin.y,
        size.cx, size.cy, 0, 0, (WORD) m_lpBMIH->biWidth,
        (WORD) m_lpBMIH->biHeight, m_lpData, m_lpBMI,
        DIB_RGB_COLORS, SRCCOPY)) {
        return FALSE;
    }
    return TRUE;
}

////////////////////////////////////////////////////////////
int CDib::GetColorBits()
{
    return m_nBits;
}

////////////////////////////////////////////////////////////
DWORD CDib::GetLength()
{
    return m_dwLength;
}

////////////////////////////////////////////////////////////
BOOL CDib::AllocateMemory(BOOL bRealloc) // bRealloc default = FALSE
{
        if (bRealloc) {
                m_lpBuf = (unsigned char*) GlobalReAllocPtr(m_lpBuf,
                        m_dwLength, GHND);
        }
        else {
                m_lpBuf = (unsigned char*) GlobalAllocPtr(GHND, m_dwLength);
        }
```

```
        if (!m_lpBuf) {
                AfxMessageBox("Unable to allocate DIB memory");
                m_dwLength = 0L;
                m_nBits = 0;
                return FALSE;
        }
        m_lpBMFH = (LPBITMAPFILEHEADER) m_lpBuf;
        m_lpBMIH = (LPBITMAPINFOHEADER) (m_lpBuf + sizeof(BITMAPFILEHEADER));
        m_lpBMI = (LPBITMAPINFO) m_lpBMIH;
        return TRUE;
}

void CDib::ClearPic(void)
{
        long dwSize = m_dwLength-m_lpBMFH->bfOffBits;
        long i;
        for(i=0;i<dwSize;i++)
                m_lpPic[i]=0;
}

void CDib::WhiteBackground(void)
{
        long dwSize = m_dwLength-m_lpBMFH->bfOffBits;
        long i;
        for(i=0;i<dwSize;i++)
                m_lpPic[i]= 255;
}

void CDib::SetGrayBackground(unsigned char Level)
{
        long dwSize = m_dwLength-m_lpBMFH->bfOffBits;
        FillMemory(m_lpPic,dwSize,Level);
/*      long i;
        for(i=0;i<dwSize;i++)
```

```
                    m_lpPic[i]= Level;*/
    }

    void CDib::DeleteData(void)
    {
            GlobalFreePtr(m_lpBuf);
            m_lpBuf = NULL;
            m_dwLength = 0;
    }

    void CDib::CopyData(CDib* Src)
    {
            if(m_dwLength == 0)
            {
                    m_dwLength = Src->m_dwLength;
                    AllocateMemory();
            }
            else
            {
                    m_dwLength = Src->m_dwLength;
                    AllocateMemory(TRUE);
            }
            CopyMemory(m_lpBuf,Src->m_lpBuf,m_dwLength);
    }

    /*void CDib::GetColorBinaryTree(BinaryTree* BT)
    {
            int Width,Height,iX,iY;
            unsigned char* Pic1 = m_lpPic;
            unsigned char* Pic2;
            Width = m_lpBMIH->biWidth;
            Height = m_lpBMIH->biHeight;
            long Step;
            if(m_nBits==24)
```

```
        {
            Step = ((long) Width)*3L;
            if(Step&3) Step = (Step&0xfffffc)+4;
            for(iY=0;iY<Height;iY++)
            {
                Pic2 = Pic1;
                for(iX=0;iX<Width;iX++)
                {
                    BT->BuildTree(RGB(Pic2[2],Pic2[1],Pic2[0]));
                    Pic2+=3;
                }
                Pic1+=Step;
            }
        }
        if(m_nBits==8)
        {
            Step = (long) Width;
            if(Step&3) Step = (Step&0xfffffc)+4;
            for(iY=0;iY<Height;iY++)
            {
                Pic2 = Pic1;
                for(iX=0;iX<Width;iX++)
                {
                    BT->BuildTree(RGB(m_lpBMI->bmiColors[*Pic2].rgbRed,
                        m_lpBMI->bmiColors[*Pic2].rgbGreen,
                        m_lpBMI->bmiColors[*Pic2].rgbBlue));
                    Pic2++;
                }
                Pic1+=Step;
            }
        }
    }*/
```

```
////////////////////////////////////////
// cdib.h
////////////////////////////////////////

//#include "binarytree.h"
class CDib : public CObject
{
   DECLARE_SERIAL(CDib)
protected:
        unsigned char* m_lpBuf;
   DWORD     m_dwLength; // total buffer length, including file header
   int       m_nBits;   // number of color bits per pixel
   //pointers for internal use
   LPBITMAPFILEHEADER m_lpBMFH;
   LPBITMAPINFOHEADER m_lpBMIH;
   LPBITMAPINFO      m_lpBMI;
   unsigned char* m_lpData;
        unsigned char* m_lpPic;
        MakeDib(long x,long y,int Bits);
public:
   CDib();
        CDib(long x,long y,int Bits);
   ~CDib();
   virtual void Serialize(CArchive &ar);
   void SetLength(DWORD Length) { m_dwLength = Length; }
   BOOL Display(CDC*, CPoint origin);
   BOOL Stretch(CDC*, CPoint origin, CSize size);
   int GetColorBits();  // bits per pixel
   DWORD GetLength();
        BOOL It24Bit(void){
        if(m_nBits==24) return TRUE;
        else return FALSE;
   }
   //        void GetColorBinaryTree(BinaryTree* BT);
```

```
        long ReceiveWidth(void) { return m_lpBMIH->biWidth;}
        long ReceiveHeight(void) { return m_lpBMIH->biHeight;}
        unsigned char* ReceivePic(void) { return m_lpPic;}
    LPBITMAPINFOHEADER ReceiveBMPInfoHeader(void) { return m_lpBMIH;}
    LPBITMAPFILEHEADER ReceiveBMPFileHeader(void) { return m_lpBMFH;}
        LPBITMAPINFO ReceiveBMI(void) { return m_lpBMI;}
    unsigned char* ReceiveBuf(void) { return (unsigned char*) m_lpBuf;}
        void ClearPic(void);
        void WhiteBackground(void);
        void SetGrayBackground(unsigned char Level);
        void DeleteData(void);
        void CopyData(CDib* Src);
private:
  BOOL AllocateMemory(BOOL bRealloc = FALSE);
};
```

```
////////////////////////////////////////////////////
// 3Dclass.cpp
////////////////////////////////////////////////////
#include "stdafx.h"
#include "3DClass.h"

C3DOperator::C3DOperator()
{
        dA11 = 1.0f; dA12 = 0.0f; dA13 = 0.0f; dA14 = 0.0f;
        dA21 = 0.0f; dA22 = 1.0f; dA23 = 0.0f; dA24 = 0.0f;
        dA31 = 0.0f; dA32 = 0.0f; dA33 = 1.0f; dA34 = 0.0f;
        dA41 = 0.0f; dA42 = 0.0f; dA43 = 0.0f; dA44 = 1.0f;
}
C3DOperator::~C3DOperator()
{
}
void C3DOperator::RotateXYPlaneRad(double dRad)
{
        double dCos = cos(dRad);
        double dSin = sin(dRad);
        double Temp = dA11;
        dA11 = (dCos*dA11) - (dSin*dA12);
        dA12 = (dSin*Temp) + (dCos*dA12);

        Temp = dA21;
        dA21 = (dCos*dA21) - (dSin*dA22);
        dA22 = (dSin*Temp) + (dCos*dA22);

        Temp = dA31;
        dA31 = (dCos*dA31) - (dSin*dA32);
        dA32 = (dSin*Temp) + (dCos*dA32);

        Temp = dA41;
        dA41 = (dCos*dA41) - (dSin*dA42);
```

```
        dA42 = (dSin*Temp) + (dCos*dA42);
}
void C3DOperator::RotateXYPlaneDeg(double dDeg)
{
        double dRad;
        dRad = dDeg*3.1415926535f/180.0f;
        double dCos = cos(dRad);
        double dSin = sin(dRad);
        double Temp = dA11;
        dA11 = (dCos*dA11) - (dSin*dA12);
        dA12 = (dSin*Temp) + (dCos*dA12);

        Temp = dA21;
        dA21 = (dCos*dA21) - (dSin*dA22);
        dA22 = (dSin*Temp) + (dCos*dA22);

        Temp = dA31;
        dA31 = (dCos*dA31) - (dSin*dA32);
        dA32 = (dSin*Temp) + (dCos*dA32);

        Temp = dA41;
        dA41 = (dCos*dA41) - (dSin*dA42);
        dA42 = (dSin*Temp) + (dCos*dA42);
}

void C3DOperator::RotateXZPlaneRad(double dRad)
{
        double dCos = cos(dRad);
        double dSin = sin(dRad);
        double Temp = dA11;
        dA11 = (dCos*dA11) + (dSin*dA13);
        dA13 = (dCos*dA13) - (dSin*Temp);

        Temp = dA21;
```

```
        dA21 = (dCos*dA21) + (dSin*dA23);
        dA23 = (dCos*dA23) - (dSin*Temp);

        Temp = dA31;
        dA31 = (dCos*dA31) + (dSin*dA33);
        dA33 = (dCos*dA33) - (dSin*Temp);

        Temp = dA41;
        dA41 = (dCos*dA41) + (dSin*dA43);
        dA43 = (dCos*dA43) - (dSin*Temp);
}

void C3DOperator::RotateXZPlaneDeg(double dDeg)
{
        double dRad;
        dRad = dDeg*3.1415926535f/180.0f;
        double dCos = cos(dRad);
        double dSin = sin(dRad);
        double Temp = dA11;
        dA11 = (dCos*dA11) + (dSin*dA13);
        dA13 = (dCos*dA13) - (dSin*Temp);

        Temp = dA21;
        dA21 = (dCos*dA21) + (dSin*dA23);
        dA23 = (dCos*dA23) - (dSin*Temp);

        Temp = dA31;
        dA31 = (dCos*dA31) + (dSin*dA33);
        dA33 = (dCos*dA33) - (dSin*Temp);

        Temp = dA41;
        dA41 = (dCos*dA41) + (dSin*dA43);
        dA43 = (dCos*dA43) - (dSin*Temp);
}
```

```
void C3DOperator::RotateYZPlaneRad(double dRad)
{
        double dCos = cos(dRad);
        double dSin = sin(dRad);
        double Temp = dA12;
        dA12 = (dCos*dA12) - (dSin*dA13);
        dA13 = (dSin*Temp) + (dCos*dA13);

        Temp = dA22;
        dA22 = (dCos*dA22) - (dSin*dA23);
        dA23 = (dSin*Temp) + (dCos*dA22);

        Temp = dA32;
        dA32 = (dCos*dA32) - (dSin*dA33);
        dA33 = (dSin*Temp) + (dCos*dA33);

        Temp = dA42;
        dA42 = (dCos*dA42) - (dSin*dA43);
        dA43 = (dSin*Temp) + (dCos*dA43);
}

void C3DOperator::RotateYZPlaneDeg(double dDeg)
{
        double dRad = dDeg*3.1415926535f/180.0f;
        double dCos = cos(dRad);
        double dSin = sin(dRad);
        double Temp = dA12;
        dA12 = (dCos*dA12) - (dSin*dA13);
        dA13 = (dSin*Temp) + (dCos*dA13);

        Temp = dA22;
        dA22 = (dCos*dA22) - (dSin*dA23);
        dA23 = (dSin*Temp) + (dCos*dA22);
```

```
        Temp = dA32;
        dA32 = (dCos*dA32) - (dSin*dA33);
        dA33 = (dSin*Temp) + (dCos*dA33);

        Temp = dA42;
        dA42 = (dCos*dA42) - (dSin*dA43);
        dA43 = (dSin*Temp) + (dCos*dA43);
}

void C3DOperator::Translator(double dX,double dY,double dZ)
{
        dA11 = dA11 + (dX * dA14);
        dA12 = dA12 + (dY * dA14);
        dA13 = dA13 + (dZ * dA14);

        dA21 = dA21 + (dX * dA24);
        dA22 = dA22 + (dY * dA24);
        dA23 = dA23 + (dZ * dA24);

        dA31 = dA31 + (dX * dA34);
        dA32 = dA32 + (dY * dA34);
        dA33 = dA33 + (dZ * dA34);

        dA41 = dA41 + (dX * dA44);
        dA42 = dA42 + (dY * dA44);
        dA43 = dA43 + (dZ * dA44);
}

void C3DOperator::Scale(double dXScale,double dYScale,double dZScale)
{
        dA11 = dA11 * dXScale;
        dA12 = dA12 * dYScale;
        dA13 = dA13 * dZScale;
```

```
        dA21 = dA21 * dXScale;
        dA22 = dA22 * dYScale;
        dA23 = dA23 * dZScale;

        dA31 = dA31 * dXScale;
        dA32 = dA32 * dYScale;
        dA33 = dA33 * dZScale;

        dA41 = dA41 * dXScale;
        dA42 = dA42 * dYScale;
        dA43 = dA43 * dZScale;
}

C3DVector::C3DVector()
{
}
C3DVector::~C3DVector()
{
}
//void C3DVector::Passvalue(double x,double y,double z)
void C3DVector::Transform(C3DOperator Operator)
{
        dX = (dX*Operator.dA11)+(dY*Operator.dA21)+(dZ*Operator.dA31)+Operator.dA41;
        dY = (dX*Operator.dA12)+(dY*Operator.dA22)+(dZ*Operator.dA32)+Operator.dA42;
        dZ = (dX*Operator.dA13)+(dY*Operator.dA23)+(dZ*Operator.dA33)+Operator.dA43;
}
```

```
////////////////////////////////////////////////
// 3Dclass.h
////////////////////////////////////////////////
#include <math.h>
class C3DOperator
{
        friend class C3DVector;
private:
        double dA11, dA21, dA31, dA41;
        double dA12, dA22, dA32, dA42;
        double dA13, dA23, dA33, dA43;
        double dA14, dA24, dA34, dA44;
public:
        C3DOperator();
        ~C3DOperator();
        void RotateXYPlaneRad(double dRad);
        void RotateXYPlaneDeg(double dDeg);
        void RotateXZPlaneRad(double dRad);
        void RotateXZPlaneDeg(double dDeg);
        void RotateYZPlaneRad(double dRad);
        void RotateYZPlaneDeg(double dDeg);
        void Translator(double dX,double dY,double dZ);
        void Scale(double dXScale,double dYScale,double dZScale);
};

class C3DVector
{
private:
        double dX, dY, dZ;
public:
        C3DVector();
        //void Passvalue(double x,double y, double z);
        C3DVector(double x,double y,double z);
        ~C3DVector();
```

```
        C3DVector operator+(const C3DVector Value) const;
        void operator=(const C3DVector Value);
        C3DVector operator-(const C3DVector Value) const;
        C3DVector operator*(const double Mul) const; // scale vector
        C3DVector operator/(const double Devide) const; // devide vector
        double InnerProduct(const C3DVector Value) const;
        C3DVector CrossProduct(const C3DVector Value) const;
        void Normal(void); // return normal vector
        void ChangeToNormal(void); // change to normal vector
        double Length(void); // return length of vector
        double Length_2(void); // return length square of vector
        void XYAngleYZAngle(double& XY,double& YZ);
        void Transform(C3DOperator Operator);
        double GetX(void) { return dX;}
        double GetY(void) { return dY;}
        double GetZ(void) { return dZ;}
        void Passvalue(double x,double y,double z)
        {
            dX = x;
            dY = y;
            dZ = z;
        }
        bool test(void){return(1);}
};
inline C3DVector::C3DVector(double x,double y, double z) :
dX(x), dY(y), dZ(z)
{
}
inline C3DVector C3DVector::operator+(const C3DVector Value) const
{
        return C3DVector(dX+Value.dX,dY+Value.dY,dZ+Value.dZ);
}
inline void C3DVector::operator=(const C3DVector Value)
{
```

```
        dX = Value.dX;
        dY = Value.dY;
        dZ = Value.dZ;
}
inline C3DVector C3DVector::operator-(const C3DVector Value) const
{
        return C3DVector(dX-Value.dX,dY-Value.dY,dZ-Value.dZ);
}
inline C3DVector C3DVector::operator*(const double Mul) const
{
        return C3DVector(dX*Mul,dY*Mul,dZ*Mul);
}
inline C3DVector C3DVector::operator/(const double Devide) const
{
        return C3DVector(dX/Devide,dY/Devide,dZ/Devide);
}
inline C3DVector C3DVector::CrossProduct(const C3DVector Value) const
{
        double X = (dY*Value.dZ)-(dZ*Value.dY);
        double Y = (dZ*Value.dX)-(dX*Value.dZ);
        double Z = (dX*Value.dY)-(dY*Value.dX);
        return C3DVector(X,Y,Z);
}
inline double C3DVector::InnerProduct(const C3DVector Value) const
{
        return ((dX*Value.dX)+(dY*Value.dY)+(dZ*Value.dZ));
}
inline void C3DVector::Normal(void)
{
        double Length = sqrt((dX*dX)+(dY*dY)+(dZ*dZ));
        dX /= Length;
        dY /= Length;
        dZ /= Length;
}
```

```
inline void C3DVector::ChangeToNormal(void) // change to normal vector
{
        double Length = sqrt((dX*dX)+(dY*dY)+(dZ*dZ));
        dX /= Length;
        dY /= Length;
        dZ /= Length;
}
inline double C3DVector::Length(void) // return length of vector
{
        return sqrt((dX*dX)+(dY*dY)+(dZ*dZ));
}
inline double C3DVector::Length_2(void)
{
        return (dX*dX) + (dY*dY) + (dZ*dZ);
}
inline void C3DVector::XYAngleYZAngle(double& XY,double& YZ)
{
        YZ = atan2(dZ,dY);
        XY = atan2(sqrt((dZ*dZ)+(dY*dY)),dX);
}
```

โครงสร้างโดยรวมของ DIB ใน BMP File

โครงสร้างโดยรวมของ DIB ใน BMP File ประกอบด้วยBITMAPFILEHEADER , BITMAPINFOHEADER และ BITMAP DATA ซึ่งรายละเอียดของส่วนต่างๆแสดงเป็น block ได้ดังรูป

BITMAPFILEHEADER	bfType = "BM"
(BMP files only)	bfOffBits
BITMAPINFOHEADER	bisize (of this structure)
	biwidth (in pixels)
	biHeight (in pixels)
	biPlanes = 1
	biBitCount (1,4,8,16,24 or 32)
	biCompression (0 for none)
	bisizeImage (only if compressions is used)
	biClrUsed (nonzero for short color tables)
Color Table	2 entries for mono DIBS
	16 or fewer entries for 4-bpp DIBS
	256 or fewer entries for 4-bpp DIBS
	Each entry is 32 bits
BITMAP DATA	Pixels ordered by column within row
	Rows padded to 4 byte boundaries

BITMAPFILEHEADER เป็นส่วนที่เก็บข้อมูลรูปแบบหลักของ bitmap เริ่มต้นทั้งหมดประกอบด้วย

- bfType = Bitmap File Type ถูกกำหนดให้ = "BM" ซึ่งเป็นรหัส ASCII , HEX (42,4D) มีขนาด 2 byte

- bfOffBits = Byte offset to bitmap data แสดงขนาดของ BITMAPFILEHEADER ทั้งหมดของภาพซึ่งเราสามารถที่จะใช้ในการคำนวณค่า BITMAPINFOHEADER และ Color table ได้ ในส่วนนี้ยังมีประกอบด้วย size of the file ซึ่งแสดงขนาดของ file ด้วยแต่เราจะไม่คำนึงถึงส่วนนี้ เพราะเราไม่สามารถทราบได้ว่าขนาดนี้ถูกวัดเป็นชนิด byte , word หรือ double word

BITMAPINFOHEADER โครงสร้างในส่วนนี้จะประกอบไปด้วยขนาดของภาพ bitmap, จำนวน bits per pixel , ข้อมูลเกี่ยวกับการบีบอัดสำหรับภาพขนาด 4-bpp และ 8-bpp และจำนวน

ของ color table ถ้าภาพถูกบีบอัดแล้วในส่วนของ header จะประกอบไปด้วยขนาดของ pixel array ซึ่งเราสามารถคำนวณขนาดจาก dimensions และ จำนวน bit per pixel

BITMAP DATA ในส่วนนี้จะเก็บข้อมูลของภาพที่ประกอบด้วย pixel ที่ถูกจัดเก็บไว้เป็นแถวเริ่มจากซ้ายไปขวา ซึ่งแถวจะเริ่มต้นจากแถวล่างสุดขึ้นข้างบน (bottom to top) จุดเริ่มต้นจะอยู่ที่มุมล่างซ้ายของภาพ บิตหลายๆบิตของแต่ละ pixel จะถูกอัดรวมเป็น byte ในแต่ละแถวจะมีได้ไม่เกิน 4 byte

ในบางครั้งเราอาจจะไม่พบส่วนของ BITMAPFILEHEADER ก็ได้ในกรณีที่เราได้ภาพมาจาก clipboard แต่เราก็สามารถดูส่วนของ color table ได้ในส่วนของ BITMAPINFOHEADER ในกรณีที่เราใช้ฟังก์ชั่นที่จัดการเกี่ยวกับภาพ เช่น CreateDIBSection เราต้องจองพื้นที่ของส่วน BITMAPINFOHEADER และ color table แล้วให้ Window เป็นตัวจัดการจองพื้นที่ของภาพที่ส่วนใดก็ได้

เอกสารอ้างอิง

1 Charles A. Mirho Andre Terrisse , " Communication Programming for Windows 95 " , Microsoft Press , 1996

2 "Color Quick Cam User Guide" ,Connectix Corporation ,1996

3 อนิรุต ลิ่วหาทอง , " การเขียนโปรแกรมบนวินโดวส์ด้วย Micrrosoft C++" ,ซีเอ็ดยูเคชั่น จำกัด (มหาชน) ,2539

4 Scott Stanfield with Ralph Arveson, "VISUAL C++4 How-To" , waite Group Press ™ A Division of Sams Publishing Corte Madera , CA , 1996

5 Brain W. Kernighan and Dennis M. Ritchie , "The C Programming Language" , Bell Laboratories Murray Hill , New Jersey

6 Vera B. Anand , "Computer Graphics and Geometric Modeling for Engineers" , John Wiley & Sons,Inc. , 1993

7 David J.Kruglinsik,"Inside Visual C+" ,Microsoft Press, fourth edition,1997.

8 David F.Rogers & J.Alan Adams ,"Mathematical Elements for Computer Grapics" , McGraw – Hill Publishing Company , second edition ,1989

9 Steve Rimmerm" WINDOWS BITMAPPED GRAPHICS", WINDCREST/McGraw – Hill,1993

10 C.Wayne Brown and Barry J.Shepherd, "Graphics File Formats reference and guide", Manning Publication Co. ,1995

11 F.S. HILL ,JR.,"COMPUTER GRAPHICS" , Macmillan Publishish Company , 1990

12 Keith Rule, "3D GRAPHICS FILE FORMATS" ADDISON – WESLEY DEVELOPERS PRESS, 1996

13 James D. Murray and William Vanryper," Encyclopedia of GRAPHICS FILE FORMATS", O' Reilly&associates,Inc. ,second edition ,1996

14 Jeff Prosise, "Programming Window95 with MFC " , Microsoft Press,1996

Web Site ที่เข้าไปหาข้อมูลทางอินเตอร์เนต

1 http://www.connectix.com

2 http://www.kolban.com

3 http://www.microsoft.com

4 ftp://ftp.microsoft.com

5 http://ourworld.compuserve.com