ระบบหาพิกัด 3 มิติ

3D COORDINATED EXTRACTION SYSTEM

โดย

นางสาว กมลชนก ศรุติไพศาล

นางสาว ขวัญฤทัย ศรีจงใจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ระบบหาพิกัด 3 มิติ

3D COORDINATED EXTRACTION SYSTEM

โดย

นางสาว กมลชนก ศรุติไพศาล รหัส 46010006

นางสาว ขวัญฤทัย ศรีจงใจ รหัส 46010076

อาจารย์ที่ปรึกษา

รศ.ดร. ชูชาติ ปิณฑวิรุจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ภาควิชา	อิเล็กทรอนิกส์
คณะ	วิศวกรรมศาสตร์
สถาบัน	เทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง (ภาษาไทย)	ระบบหาพิกัด 3 มิติ
เรื่อง (ภาษาอังกฤษ)	3D COORDINATED EXTRACTION SYSTEM
จัดทำโดย	นางสาว กมลชนก ศรุติไพศาล รหัส 46010006
	นางสาว ขวัญฤทัย ศรีจงใจ รหัส 46010076
อาจารย์ที่ปรึกษา	รศ.ดร. ชูชาติ ปิณฑวิรุจน์

ปริญญานิพนธ์ฉบับนี้ได้ผ่านการตรวจสอบจากอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ.....อาจารย์ที่ปรึกษา

(รศ.ดร. ชูชาติ ปิณฑวิรุจน์)

วันที่ 8 / ต.ค. / 56

ระบบหาพิกัด 3 มิติ

นางสาว กมลชนก ศรุติไพศาล รหัส 46010006
นางสาว ขวัญฤทัย ศรีจงใจ รหัส 46010076
รศ. ดร. ชูชาติ ปิณฑวิรุจน์ (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2549

บทคัดย่อ

ปัจจุบันการสร้างภาพเคลื่อนไหวที่แสดงให้เห็นในรูปของภาพแอนิเมชัน 3 มิตินั้นได้ถูกนำมาประยุกต์ใช้ในหลายๆด้านรวมถึงทางด้านการแพทย์ซึ่งได้นำมาใช้ในการวิเคราะห์การเคลื่อนไหวมนุษย์โดยในการวิเคราะห์จะได้มาซึ่งภาพเคลื่อนไหวที่สมจริงจำเป็นจะต้องมีโปรแกรมในการวิเคราะห์ภาพที่เรียกว่า ระบบหาพิกัด 3 มิติ

ระบบหาพิกัด 3 มิติคือการบันทึกภาพการเคลื่อนไหวของมนุษย์ (หรือการเคลื่อนไหวของวัตถุอื่นๆ) ที่ภาพของวัตถุที่ถูกติดด้วยมาร์คเกอร์ ต้องถูกทำการวิเคราะห์ให้ได้เป็นพิกัด 3 มิติ ของมาร์คเกอร์ระบบพิกัด 3 มิติ ประกอบด้วยกล้องหลาย ๆ กล้องวางที่ตำแหน่งใดก็ได้ กล้องจะทำการบันทึกภาพของวัตถุที่ติดมาร์คเกอร์ และทำการวิเคราะห์เพื่อให้ได้พิกัด 3 มิติของมาร์คเกอร์โดยอาศัยทฤษฎีการปรับเทียบกล้อง และ การหาพิกัด 3 มิติของจุดในฉาก ค่าพิกัดภาพ 3 มิติที่ได้นั้นเพื่อทดสอบว่าค่านั้นถูกต้องจึงถูกนำมาแสดงผลให้อยู่ในรูปของภาพแอนิเมชั่น 3 มิติ

3D COORDINATE EXTRACTION SYSTEM

Miss. Kamonchanok Sarutipaisal ID. 46010006

Miss. Khwanruthai Srijongjai ID. 46010076

Assoc. Prof. Dr. Chuchart Pintavirooj (Advisor)

Education year 2006

Abstract

Currently , computer technology developes very fast , especially in 3D animation and 3D coordinate extraction. One of the application of 3D animation in medicine include gait analysis. This thesis concerns about 3D extraction system.

3D Coordinate Extraction System consists of three or more cameras taking image of object in arbitrary positions. Image of marker attached on the object will be analyzed to derive 3D coordinate of markers. 3D coordinate extraction are searched based on using theory of Camera Calibration , Geometric Transformation in 3D and 3D Extraction. For testing that 3D coordinate extraction was created correctly. We will apply it by using the program construct 3D model.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดีนั้น ทางผู้จัดทำขอขอบคุณ อาจารย์ชูชาติ ปิณฑวิรุจน์ซึ่งเป็นอาจารย์ที่ปรึกษา และพี่นักศึกษาปริญญาโท ที่ได้ให้แนวคิด คำปรึกษาและความช่วยเหลือในระหว่างการดำเนินงาน ขอขอบคุณอาจารย์ในภาควิชาทุกท่านที่ได้ให้คำแนะนำและแนวทางต่างๆที่เป็นประโยชน์ต่อโครงงานนี้ให้สำเร็จลุล่วงไปได้ด้วยดี ขอขอบคุณเพื่อนๆทุกคนที่ได้ให้ความช่วยเหลือในด้านต่างๆ และขอขอบคุณบิดา มารดา รวมทั้งผู้มีพระคุณทุกท่าน

สารบัญ

สารบัญ(ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

ในปัจจุบันเนื่องจากความก้าวหน้าทางเทคโนโลยีทำให้เราได้เห็นถึงการเปลี่ยนแปลงหลายๆอย่าง ซึ่งรวมไปถึงการพัฒนาทางด้านคอมพิวเตอร์ในเรื่องที่เกี่ยวกับงานด้าน 3 มิติ ซึ่งได้ถูกนำมาใช้ในการสร้างภาพกราฟิก แอนิเมชั่นต่างๆ ที่มีให้เห็นกันอย่างแพร่หลาย โดยที่ภาพวัตถุเคลื่อนไหวที่มีให้เห็นตามภาพยนตร์หรือเกมส์นั้นต้องผ่านกระบวนการทำหลายขั้นตอน ซึ่งพื้นฐานในการทำจะมาจากการวิเคราะห์ภาพระบบ 3 มิติ ซึ่งจะนำไปใช้ในการวิเคราะห์การเคลื่อนไหวของสิ่งต่างๆ โดยที่ขึ้นอยู่กับการเขียนโปรแกรมและองค์ประกอบต่างๆเพื่อให้ได้มาภาพซึ่งภาพที่มีความสมจริง และสมบูรณ์มากที่สุด โดยที่ระบบพิกัด 3 มิติได้นำมาใช้ในด้านอื่นๆอีก เช่น ทางด้านการกีฬาเป็นการวิเคราะห์การเคลื่อนไหวระดับสูงในวิทยาศาสตร์การกีฬา และการใช้อุปกรณ์ทางกีฬาบางประเภท ทางด้านกลศาสตร์ชีวภาพจะเป็นการวิเคราะห์สำหรับงานวิจัยกลศาสตร์ชีวภาพด้านการกีฬาและด้านการแพทย์ ทางด้านศัลยกรรมกระดูกจะเป็นการวิเคราะห์คุณภาพโดยการวิเคราะห์การเคลื่อนไหวแบบ 2D และ 3D ทางด้านการบำบัดผู้ป่วยโดยใช้การวิเคราะห์การเคลื่อนไหวจากข้อมูลภาพวีดีโอที่สามารถเปรียบเทียบ วัด และ บันทึกข้อมูลการเคลื่อนไหวได้ในหลายๆวิธี ทางด้านอุตสาหกรรมใช้สังเกตการณ์เคลื่อนไหวที่สมบูรณ์ของเครื่องจักรและมนุษย์

จากโครงงานนี้ได้แสดงถึงกระบวนการต่างๆที่ใช้ในการสร้างภาพการเคลื่อนไหว 3 มิติ โดยเริ่มจากการหาพิกัดภาพ 3 มิติโดยจะเปลี่ยนจากพิกัดที่เป็น 2 มิติ จะอาศัยทฤษฎี เรขาคณิตสำหรับการเห็นใน 3 มิติ การปรับเทียบกล้อง และการหาพิกัด 3 มิติของจุดในฉาก เพื่อเปลี่ยนเป็นพิกัดแบบ 3 มิติ เมื่อได้ค่าพิกัดภาพที่เป็น 3 มิติแล้ว ก็จะนำค่าพิกัด 3 มิติที่ได้ มาสร้างเป็นภาพเคลื่อนไหวแบบ 3 มิติ

1.1 รายละเอียดโดยย่อของโครงงาน

จากโครงงานจะทำการหาพิกัดของภาพ 3 มิติเพื่อที่จะนำพิกัดภาพที่ได้มาสร้างเป็นภาพเคลื่อนไหวแบบ 3 มิติ โดยการแคปเจอร์ภาพที่เป็นท่าทางการเคลื่อนไหวของคนจากกล้อง 3 ตัวที่วางในตำแหน่งต่างกัน ซึ่งภาพที่แคปเจอร์ได้จะมีพิกัดเพียง 2 มิติ ดังนั้นภาพที่ได้จะต้องนำไปประมวลผลเพื่อให้ได้พิกัดเป็น 3 มิติโดยใช้โปรแกรมที่ได้ออกแบบ มาใช้ในปรับเทียบกล้องและช่วยคำนวณเพื่อหาค่าพิกัด โดยอาศัยทฤษฎีทางคณิตศาสตร์มาช่วยในการคำนวณ ซึ่งในการหาพิกัดของภาพนั้นจะใช้การคลิกที่จุดมาร์คเกอร์ในแต่ละกล้องเพียงครั้งเดียว ทั้ง 3 กล้อง ซึ่ง

โปรแกรมจะทำการ autolandmark เพื่อตรวจจับจุดสีที่ต่างกันของจุดมาร์คเกอร์แต่ละจุดในเฟรมถัดไป

1.2 วัตถุประสงค์ของโครงงาน

- เพื่อศึกษาการใช้งานและนำไปประยุกต์ใช้ซึ่งจากโครงงานนี้จะใช้โปรแกรม C++ Builder Development Environment

- เพื่อศึกษาถึงทฤษฎีที่ช่วยในการวิเคราะห์สมการทางคณิตศาสตร์ และทฤษฎีที่ใช้ในการคำนวณเพื่อหาค่าพิกัดภาพ

-เพื่อศึกษาหลักการสร้างภาพ 3มิติ

1.3 ขอบเขตของโครงงาน

- ขอบเขตของการปรับเทียบกล้อง

เป็นการเขียนโปรแกรมในเชิงวิเคราะห์สมการทางคณิตศาสตร์ให้คำนวณพารามิเตอร์การปรับเทียบกล้อง

- ขอบเขตการหาพิกัด 3 มิติ

เป็นการนำภาพที่ได้บันทึกมาหาค่าพิกัด (u , v) ของภาพและพิกัดโลก (x , y , z) มาใส่ matrix calibration และเมื่อแก้สมการออกมา จะได้ค่าพารามิเตอร์ต่างๆที่ใช้ในการหาค่าพิกัด 3 มิติ

- ขอบเขตการสร้างภาพ 3 มิติ

เป็นการนำค่าพิกัด 3 มิติที่ได้มาสร้างเป็นภาพเคลื่อนไหวแบบ 3 มิติโดยใช้โปรแกรม OpenGL ที่ช่วยในการประมวลผลภาพ

1.4 ประโยชน์และผลที่คาดว่าจะได้รับ

จากโครงงานนี้สามารถนำโปรแกรมไปประยุกต์ใช้งานในการสร้างภาพ 2 มิติ เป็น 3 มิติ เพื่อที่จะนำไปประยุกต์ใช้งานในด้านต่างๆ เช่น

- ด้านการแพทย์คือ การวิเคราะห์ท่าทางการเคลื่อนไหวของผู้ป่วย การรักษาผู้ป่วยโดยวิธีกายภาพบำบัด

- ด้านกีฬาคือ ศึกษาการเคลื่อนไหวในการออกกำลังกายที่ถูกต้อง เพื่อลดการบาดเจ็บขณะออกกำลังกาย

บทที่ 2
ระบบการมองเห็น

2.1 ความรู้และกระบวนการต่างๆที่สนับสนุนระบบการมองเห็น

เทคนิคและกระบวนการต่างๆที่นำมารวมกันเพื่อออกแบบระบบการมองเห็นของคอมพิวเตอร์โดยแบ่งเป็นเรื่องหลักๆดังนี้

การประมวลผลทางภาพ (Image Processing) เป็นศาสตร์ที่ว่าด้วยเทคนิคในการแปลงภาพต้นฉบับไปเป็นข้อมูลในรูปแบบอื่น ประกอบด้วยส่วนสำคัญดังนี้ การปรับปรุงภาพให้ชัดเจนขึ้น (Image Enhancement) การบีบอัดข้อมูลภาพให้มีขนาดเล็กลง (Image Compression) และการแก้ไขภาพที่เลือนรางหรือภาพที่มีปัญหาเรื่องการปรับโฟกัสระบบจะจัดการกับภาพที่เข้าและให้ผลลัพธ์ออกมาเป็นข้อมูลในรูปแบบอื่น เช่น ภาพเส้นโครงร่างของวัตถุ (Contour) ดังนั้นความสำคัญของการประมวลผลภาพ (Image Processing) ในระบบการมองเห็นของคอมพิวเตอร์ก็คือการแปลงข้อมูลภาพโดยอัตโนมัติ การปรับปรุงภาพให้ชัดเจนขึ้นและการจำกัดสัญญาณรบกวนที่อาจจะเกิดจากแสงสว่าง

คอมพิวเตอร์กราฟฟิก (Computer Graphics) เป็นศาสตร์ที่ว่าด้วยเทคนิคในการนำข้อมูลที่ได้รับจากเซนเซอร์ชนิดต่างๆมาสร้างเป็นภาพโดยจะแสดงผลเป็นรูปทรงเรขาคณิตแบบง่ายๆ เช่น เส้นตรง , วงกลม และพื้นผิวแบบต่างๆความรู้ทางด้านคอมพิวเตอร์กราฟฟิก จะมีบทบาทในการสร้างภาพเสมือนและสร้างสภาพแวดล้อมเหมือนจริงให้กับระบบการมองเห็นของเครื่องจักร

การรู้จำ (Pattern Recognition) เป็นศาสตร์ที่ว่าด้วยการจำแนกประเภทของวัตถุด้วยข้อมูลที่เป็นตัวเลขและสัญลักษณ์ ซึ่งได้รับการพัฒนามาจากคณิตศาสตร์ที่เกี่ยวข้องกับสถิติและความน่าจะเป็นความรู้ในเรื่องการรู้จำ (Pattern Recognition) จะมีบทบาทในการจดจำรูปแบบของวัตถุที่สนใจซึ่งถูกนำไปใช้อย่างแพร่หลายในการจดจำและแยกแยะวัตถุในภาคอุตสาหกรรม

ปัญญาประดิษฐ์ (Artificial Intelligence) เป็นศาสตร์ที่ว่าด้วยการสอนให้คอมพิวเตอร์มีความสามารถที่จะเรียนรู้และมีความเฉลียวฉลาดในการประมวลผลและตัดสินใจในสถานการณ์ต่างๆ การวิเคราะห์สถานการณ์จะใช้การคำนวณสัญลักษณ์ที่แทนเหตุการณ์ที่เกิดขึ้น เทคนิคนี้ประกอบด้วยส่วนสำคัญสามส่วน คือ ส่วนแรกเรียกว่า "Perception" ทำหน้าที่แปลงเหตุการณ์หรือเงื่อนไขต่างๆให้อยู่ในรูปของสัญลักษณ์ ส่วนต่อมาเรียกว่า "Cognition" ทำหน้าที่จัดการหรือคำนวณข้อมูลสัญลักษณ์ด้วยวิธีที่เหมาะสมซึ่งขึ้นอยู่กับชนิดของข้อมูลและผลลัพธ์ที่ต้องการว่าเป็นแบบใด และส่วนสุดท้ายเรียกว่า "Action" ทำหน้าที่แปลงข้อมูลสัญลักษณ์กลับมาเป็นข้อมูลหรือสัญญาณที่สามารถนำไปใช้งานได้

เมื่อย้อนกลับไปเมื่อประมาณสิบปีที่ผ่านมาจะพบว่าโครงข่ายประสาทเทียม(Neural-network) เริ่มเข้ามามีบทบาทในงานทางด้านนี้ เนื่องจากลักษณะเด่นที่สามารถเรียนรู้และปรับตัวเองได้เหมือนกับการทำงานของสมองมนุษย์ซึ่งมีความยืดหยุ่นและมีประสิทธิภาพในการเรียนรู้และตัดสินใจ จึงได้รับความนิยมตลอดมาจนถึงปัจจุบัน

การมองเห็นของมนุษย์ (Psychophysics) เป็นศาสตร์ที่ว่าด้วยการมองเห็นของมนุษย์โดยคิดออกมาเป็นรูปแบบซึ่งมีรูปแบบดังนี้

การมองเห็น = รูปทรงเรขาคณิต + การวัดขนาด + การแปลความหมาย

การออกแบบระบบการมองเห็นของเครื่องจักรจะอยู่บนพื้นฐานของรูปแบบการมองเห็นของมนุษย์ คือ การหาลักษณะเฉพาะของวัตถุในภาพ , การวัดขนาดของลักษณะเฉพาะนั้นในเชิงของเรขาคณิต และการแปลความหมายของข้อมูลของเรขาคณิตที่ได้ เช่น ขนาด, ตำแหน่ง , รูปทรงและชนิดของวัตถุเป็นต้น

ในเบื้องต้นจะกล่าวถึงระบบประมวลผลภาพที่ใช้ในระบบการมองเห็นของคอมพิวเตอร์ เริ่มจากเรขาคณิตที่เกี่ยวกับการประมวลผลภาพ ,การแซมปลิ้ง (Sampling) และการควอนไตเซชั่น (Quantization) และพื้นฐานการประมวลผลภาพสองระดับ (Binary Image Processing)

2.1.1 การเก็บภาพ

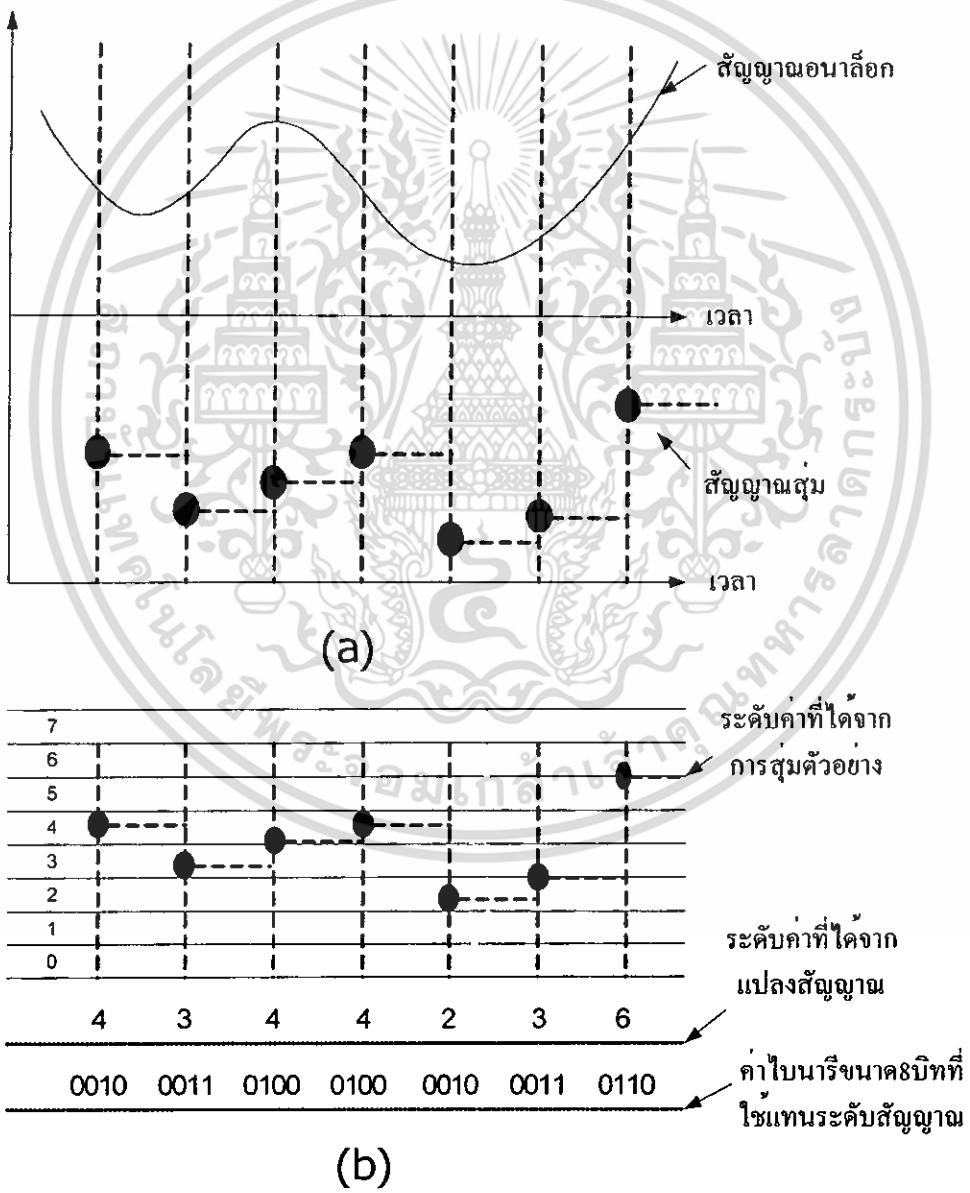
ในการเก็บภาพของวัตถุหนึ่งๆจะต้องอาศัยอุปกรณ์สำหรับภาพได้แก่ กล้องวีดีโอ เพื่อทำการแปลงระดับความสว่างในแต่ละตำแหน่งของภาพ ให้อยู่ในรูปของสัญญาณทางไฟฟ้าแบบอนาล็อก จากนั้นจะมีอุปกรณ์สำหรับแปลงสัญญาณภาพดังกล่าวให้เป็นสัญญาณดิจิตอล หรือ A/D เป็นส่วนประกอบที่สำคัญ และจะมีการจัดเรียงสัญญาณดิจิตอลนี้ลงสู่หน่วยความจำภาพ (Video-Memory) ซึ่งจะแทนค่าความสว่างของภาพในแต่ละจุดให้เกิดเป็นภาพเรียกว่า ภาพดิจิตอล (Digital-Image)

ในภาพหนึ่งๆจะถูกแบ่งออกเป็นจุดย่อยๆจำนวนมากเรียกว่า ส่วนประกอบภาพ (Picture-Elements) หรือ จุดภาพ(Pixel) จำนวนของจุดภาพยิ่งมากจะยิ่งทำให้ภาพมีความละเอียดคมชัดมากยิ่งขึ้น จากนั้นจึงแทนภาพของแต่ละจุดภาพด้วยสัญญาณดิจิตอล หรือแทนด้วยสถานะลอจิก “0” และ “1” การใช้สัญญาณดิจิตอลมีข้อดีคือ การนำข้อมูลไปประยุกต์ใช้สามารถทำได้ง่าย สิ่งที่สำคัญในการเก็บภาพคือ กระบวนการดิจิไตเซชั่นและหน่วยความจำภาพ

กระบวนการดิจิไตเซชั่น (Digitization) กระบวนการดิจิไตเซชั่นเป็นกระบวนการเปลี่ยนสัญญาณภาพจากสัญญาณอนาล็อกให้เป็นข้อมูลทางดิจิตอลแล้วนำไปจัดเก็บไว้ในหน่วยความจำ การเปลี่ยนสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (Analog to Digital หรือ A/D) ประกอบด้วย 2 ขั้นตอน คือ การสุ่มตัวอย่างสัญญาณอนาล็อก (Sampling) และการจัดระดับของสัญญาณที่สุ่มมา

(Quantizing) แต่ละระดับแทนด้วยรหัสดิจิตอล ดังแสดงในรูป 2.1 ซึ่งจะใช้รหัสดิจิตอลขนาด 8 บิท แทนระดับสัญญาณได้ 256 ระดับ ซึ่งพอเพียงสำหรับรายละเอียดภาพ

สัญญาณภาพประกอบด้วยสัญญาณหลายชนิดปะปนกันมา ในการแปลงสัญญาณจะทำการแปลงส่วนข้อมูลภาพจริงๆเท่านั้น ในส่วนของสัญญาณอื่นๆ เช่น สัญญาณแบลิ่งกิ้ง ไม่ได้ถูกแปลงด้วย แต่จะใช้การควบคุมการแปลงสัญญาณและการเขียนข้อมูลที่ถูกแปลงเป็นสัญญาณดิจิตอลแล้วหน่วยความจำ



รูปที่ 2.1 (a) แสดงรูปคลื่นสัญญาณอนาล็อกเมื่อถูกสุ่มตัวอย่างในกระบวนการดิจิไตเซชั่น
(b) แสดงการจัดระดับสัญญาณที่ได้จากการสุ่มตัวอย่างมาแล้ว ให้มีค่าเป็นจำนวนเต็ม และแปลงเป็นรหัสดิจิตอล 4 บิทแทน

เนื่องจากสัญญาณภาพมีการเปลี่ยนแปลงที่เร็วมาก การจะเก็บรายละเอียดของสัญญาณภาพให้ครบ A/D ต้องทำงานที่ความถี่สูงๆและต้องใช้รหัสดิจิตอลขนาด 8 บิทเป็นอย่างน้อย โดยส่วนมาก A/D ที่ใช้เป็นประเภทแฟลช A/D นี้สามารถเปลี่ยนสัญญาณอนาล็อกให้เป็นค่าดิจิตอลได้ภายใน 1 คาบสัญญาณนาฬิกา

สำหรับภาพขาวดำ สัญญาณ Y หรือ ลูมิแนนซ์ เท่านั้นที่ถูกนำมาแปลงเป็นสัญญาณดิจิตอล เพราะเป็นส่วนของข้อมูลภาพ สำหรับภาพสีใช้แต่สัญญาณ Y เพียงอย่างเดียวไม่เพียงพอที่จะอธิบายความหมายของสีได้ จึงจำเป็นต้องใช้ 3 สัญญาณ Y , R-Y , B-Y มาทำการแปลงสัญญาณจะทำให้ต้องใช้ A/D และหน่วยความจำเพิ่มถึง 3 ชุด ถ้าไม่ใช้สัญญาณดังกล่าวสามารถใช้สัญญาณสี RGB ก็ได้ โดยการนำสัญญาณภาพไปเข้าวงจรถอดรหัส (Decoder) เพื่อแยกเอาสัญญาณสี RGB ออกมาจากสัญญาณภาพแล้วจึงนำไปแปลงสัญญาณต่อไป

(a) 32 บิท ต่อ 1 จุดภาพ (b) 4 บิท ต่อ 1 จุดภาพ

(c) 2 บิท ต่อ 1 จุดภาพ (d) 1 บิท ต่อ 1 จุดภาพ

รูปที่ 2.2 แสดงการเปรียบเทียบที่ใช้จำนวนบิทต่อ 1 จุดภาพต่างกัน

สัญญาณภาพอนาล็อกสามารถแสดงความละเอียดและจำนวนสีของภาพอาจเรียกได้ว่าไม่จำกัด เนื่องจากความต่อเนื่องของสัญญาณ แต่เมื่อสัญญาณภาพอนาล็อกถูกเปลี่ยนไปอยู่ในรูปของสัญญาณดิจิตอล ความละเอียดและจำนวนสีของภาพถูกจำกัดด้วยความเร็วของ A/D ,จำนวนบิทที่ใช้ในการแปลงสัญญาณและขนาดของหน่วยความจำ ในภาพระดับเกรย์สเกลแต่ละจุด ภาพสามารถ

แสดงระดับสัญญาณและขนาดของหน่วยความจำ ในภาพระดับเกรย์สเกลแต่ละจุดภาพสามารถแสดงระดับสัญญาณได้ 256 ระดับนั่นคือ 1 จุดภาพจะต้องสร้างจากข้อมูลจำนวน8 บิท แต่หากเป็นภาพสี 1 จุด ภาพจะประกอบด้วยสัญญาณสี RGB ดังนั้นในแต่ละจุดภาพจึงต้องใช้จำนวนบิทเท่ากับ 24 บิท เพื่อให้สามารถเห็นระดับความแตกต่างของสีได้อย่างพอเพียง ซึ่งสามารถแสดงสีที่แตกต่างกันได้ 16,777,216 สี ในรูปที่2.2 แสดงการเปรียบเทียบที่เกิดจากการกำหนดจำนวนบิทต่อ 1 จุดภาพที่แตกต่างกัน จะเห็นได้ว่าภาพที่มีจำนวนบิทต่อ 1 จุดภาพมากจะทำให้ความละเอียดของภาพมากกว่าภาพที่มีจำนวนบิทต่อ 1 จุดภาพน้อย

หน่วยความจำภาพ (Video Memory) สัญญาณภาพอนาล็อกเมื่อถูกแปลงให้เป็นสัญญาณดิจิตอลแล้วจะทำให้เกิดข้อมูลเป็นจำนวนมาก จึงจำเป็นต้องใช้หน่วยความจำในการเก็บข้อมูลไว้ชั่วคราวก่อนที่จะนำไปใช้งาน หน่วยความจำที่ใช้ประเภทอ่านและเขียนข้อมูลใหม่ได้หรือเรียกว่า RAM (Random Access Memory) บางครั้งหน่วยความจำภาพที่สำคัญคือ ต้องใช้เวลาในการเข้าถึงข้อมูล (Access Time) ที่ต่ำมากๆนั่นคือ ความเร็วในการเขียนและอ่านข้อมูลต้องมากเพียงพอ

ขนาดของหน่วยความจำที่ใช้นั้นขึ้นอยู่กับจำนวนข้อมูลดิจิตอลที่ต้องการจัดเก็บส่วนข้อมูลจะมากหรือน้อยขึ้นอยู่กับความละเอียดของภาพ

2.2 การประมวลผลภาพ (Image Processing)

ข้อจำกัดของฮาร์ดแวร์ (Hardware) ที่ใช้ในการเก็บภาพในกระบวนการดิจิไตเซชั่นคือไม่สามารถแยกแยะ หรือกำจัดสัญญาณรบกวน (Noise) ที่เกิดขึ้นกับภาพนั้นได้ยิ่งไปกว่านั้นข้อมูลภาพบางตำแหน่งยังปกปิดข้อมูลที่เราสนใจ ซึ่งหากมองด้วยตาแล้วไม่สามารถจะมองเห็นได้จึงได้มีการนำภาพดังกล่าวนั้นไปดำเนินการด้วยวิธีที่เรียกว่า " การประมวลผลภาพ " ซึ่งจะทำการสร้างภาพใหม่ขึ้นโดยการเปลี่ยนแปลงข้อมูลตัวเลข คือ ในบริเวณที่เราสนใจจะถูกระทำการปรับปรุงรายละเอียดภาพ (Enhancement) และยังเป็นผลให้สัญญาณรบกวนลดลงหรือถูกกำจัดทิ้งไปตัวอย่างของการประมวลผลภาพทางดิจิตอล (Digital Image Processing) ที่นิยมใช้ทั่วไป ได้แก่การกำจัดสัญญาณรบกวน การหาขอบภาพ (Edge Enhancement) การฟิลเตอร์ (Filtering) และการปรับปรุงระดับเกรย์สเกล เป็นต้น

2.2.1 การประมวลผลภาพหรือการเปลี่ยนแปลงข้อมูลของภาพ

2.2.1.1 วิธีประมวลผลทีละจุดใน 1 ภาพ (Point by Point)

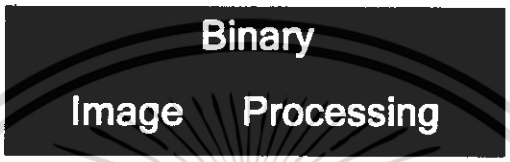
โดยแต่ละจุดในภาพของภาพต้นแบบ (Original Image) จะถูกแปลงไปเป็นภาพใหม่ซึ่งค่าของแต่ละจุดภาพในภาพใหม่นี้จะสัมพันธ์กับค่าของจุดภาพที่ตำแหน่งเดียวกันกับภาพต้นแบบ ตัวอย่างเช่น การเปลี่ยนภาพไบนารี่ (Binary Image) จุดภาพที่มีระดับเป็น "0" ในภาพต้นแบบจะถูก

เปลี่ยนไปเป็นระดับ “1” ในภาพใหม่ และจุดภาพที่มีระดับเป็น “1” ในภาพต้นแบบจะถูกเปลี่ยนไปเป็น “0” ในภาพใหม่ ดังรูปที่ 2.3

Binary

Image Processing

(a) แสดงภาพก่อนการประมวลผลทีละจุดของภาพไบนารี่



(b) แสดงภาพหลังการประมวลผลทีละจุดของภาพไบนารี่

รูปที่ 2.3 แสดงการประมวลผลภาพแบบไบนารี่

2.2.1.2 วิธีการประมวลผลโดยใช้จุดภาพที่สอดคล้องกัน (Corresponding Points)

เป็นวิธีการประมวลผลโดยใช้ภาพที่สอดคล้องกันจากภาพตั้งแต่ 2 ภาพขึ้นไป วิธีนี้จะสร้างภาพใหม่โดยใช้การเทียบเคียง (Correlation) ของแต่ละจุดภาพ เพื่อหาค่าของจุดภาพที่เหมือนกันหรือสอดคล้องกันจากภาพ 2 ภาพ หรือตั้งแต่ 2 ภาพขึ้นไป

ค่าของจุดภาพดังกล่าวสามารถนำมารวมกันด้วยวิธีต่างๆ เช่น ค่าของจุดภาพ 2 ภาพของเมืองๆหนึ่ง ซึ่งเก็บมาที่เวลาต่างกันสามารถนำมาลบกันเพื่อคำนวณหาความเปลี่ยนแปลงที่เกิดขึ้นในช่วงเวลาที่ต่างกัน นอกจากนี้ยังสามารถใช้รวมข้อมูลที่ได้จากตัวตรวจวัด (Sensor) ต่างๆจำนวนหนึ่ง เพื่อแปลงเป็นข้อมูลภาพที่มีความสมบูรณ์หรือมีรายละเอียดของภาพที่สามารถแสดงให้เข้าใจได้ง่ายยิ่งขึ้น ซึ่งตัวตรวจวัดอาจจะเป็นกล้องถ่ายภาพดาวเทียมโดยเก็บภาพของย่านสเปคตรัมต่างๆกัน เช่น รังสีอินฟาเรด รังสีอุลตร้าไวโอเลตและรังสีที่สามารถมองเห็นได้ เป็นต้น

2.2.1.3 วิธีประมวลผลทีละกลุ่มใน 1 ภาพ (Regional Points)

ในการคำนวณจะใช้ค่าของจุดภาพที่อยู่รอบๆจุดภาพหลัก นำมาสร้างภาพใหม่ด้วยวิธีการต่างๆ เช่น การเฉลี่ยข้อมูลจากจุดภาพที่อยู่รอบๆจุดภาพหลัก ซึ่งจะช่วยลดข้อมูลที่มีความผิดพลาดให้ลดลง ดังนั้นค่าของจุดภาพใหม่นี้จะเท่ากับค่าเฉลี่ยของจุดภาพ 9 จุดติดกัน(จุดภาพ 1 จุด และรอบๆจุดภาพหลักอีก 8 จุด) ดังภาพที่ 2.4(a) แสดงภาพต้นแบบจะเห็นว่ามีสัญญาณรบกวนเกิดขึ้นบนภาพ แต่เมื่อนำมาประมวลผลโดยการเฉลี่ยแล้วจะได้ดังภาพที่ 2.4(b) จะเห็นว่าสัญญาณรบกวนลดลงมาก

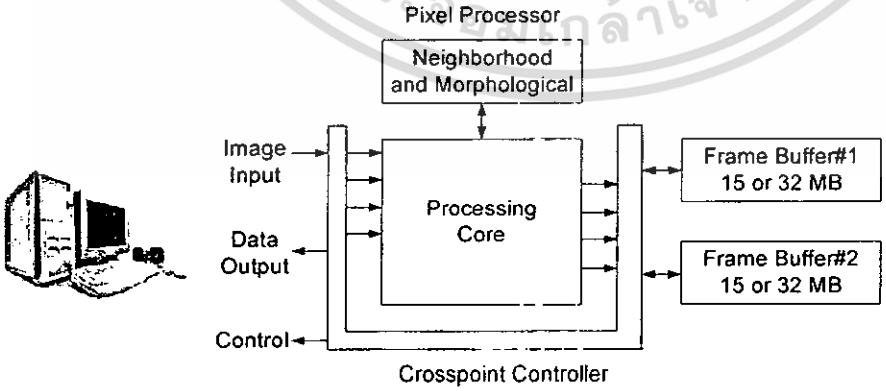
a)ภาพก่อนกำจัดสัญญาณรบกวน (b)ภาพหลังกำจัดสัญญาณรบกวน

รูปที่ 2.4 แสดงการประมวลผลภาพในการกำจัดสัญญาณรบกวน

ในการเลือกใช้วิธีการประมวลผลภาพนั้น ขึ้นอยู่กับว่าจะนำไปใช้งานในลักษณะใด โดยส่วนใหญ่ในการประมวลผลภาพเพื่อเพิ่มรายละเอียดภาพนั้น นิยมใช้วิธีการประมวลทีละกลุ่มใน 1 ภาพ

2.3 การแสดงข้อมูลภาพ

การแสดงข้อมูลภาพจะนำข้อมูลดิจิตอลที่เก็บไว้ในหน่วยความจำมาเปลี่ยนเป็นสัญญาณภาพอนาล็อก(D/A Conversion) แล้วส่งไปยังภาพต่อไป ในการแสดงภาพหนึ่งภาพจะต้องมีจุดภาพบางจุดที่ต้องสว่างและบางจุดต้องมืด การที่จะจัดการให้เกิดภาพตามที่ต้องการได้นั้นมีส่วนประกอบ 3 ส่วนที่จะใช้ในการจัดการนี้ คือ เฟรมบัฟเฟอร์ (Frame Buffer) ตัวควบคุมการแสดงภาพ (Display Controller) และวิธีการแปลงภาพให้เป็นตำแหน่งของจุดภาพที่เหมาะสมในเฟรมบัฟเฟอร์ (Scan Conversion Algorithms) ดังรูปที่ 2.5

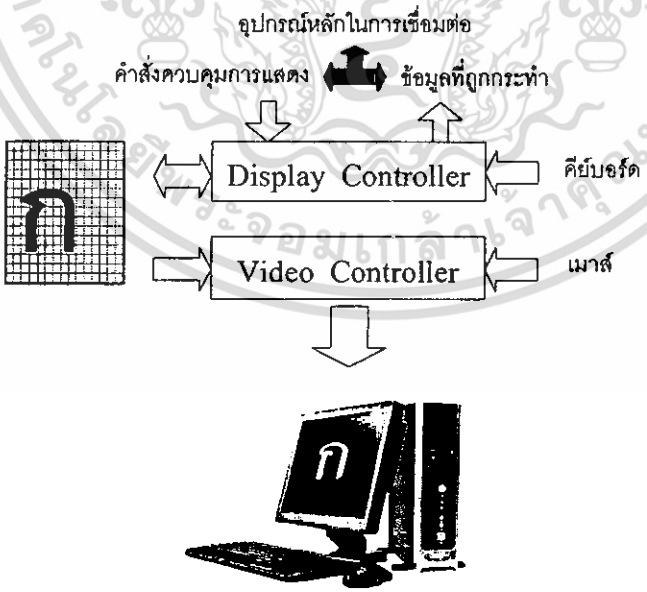


รูปที่ 2.5 แสดงส่วนประกอบในหน่วยแสดงภาพ

2.3.1 เฟรมบัฟเฟอร์

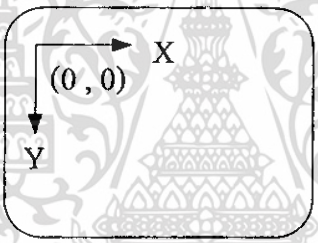
จุดแต่ละจุดที่ปรากฏอยู่บนจอภาพจะสอดคล้องกับค่าบิทที่อยู่ในหน่วยความจำส่วนหนึ่ง ซึ่งเราเรียกหน่วยความจำส่วนนี้ว่าเฟรมบัฟเฟอร์ หรือ บิทแมป (Bit Map) บิทเหล่านี้จะถูกเก็บไว้ในลักษณะตารางหน่วยความจำที่ใช้เป็นบัฟเฟอร์ในปัจจุบันมักจะแยกออกจากหน่วยความจำหลักของเครื่องคอมพิวเตอร์ เพื่อที่จะทำให้สามารถแสดงภาพออกทางจอภาพได้อย่างรวดเร็วมากขึ้น จำนวนแถวและหลักของเฟรมบัฟเฟอร์จะเท่ากับจำนวนของจุดภาพที่จอภาพแสดงได้ การบอกขนาดของหน่วยความจำที่ใช้เป็นเฟรมบัฟเฟอร์อาจบอกในรูปของจำนวนจุดภาพที่สามารถแสดงบนจอภาพ หรืออาจจะบอกในรูปของจำนวนจุดภาพในหลักการคูณจำนวนจุดภาพในแถวก็ได้ ตัวอย่างเช่น 640 x 350 สำหรับจอภาพแบบ EGAHi และสำหรับจอภาพแบบ VGAHi จะเป็น 640x480 เป็นต้น เมื่อมีการใส่บิท 1 ลงในเฟรมบัฟเฟอร์ตรงตำแหน่งใดก็ตาม จะเกิดเป็นจุดสว่างบนจอภาพตรงตำแหน่งที่สอดคล้องกับเฟรมบัฟเฟอร์ เช่น ถ้าให้แถวที่ 2 หลักที่ 3 ของเฟรมบัฟเฟอร์มีค่าเป็น 1 บิท จอภาพที่ตำแหน่งแถวที่ 2 หลักที่ 3 ก็จะเกิดเป็นจุดสว่างเป็นต้น ส่วนตำแหน่งที่มีค่าเป็นบิท 0 ก็จะไม่มีจุดสว่าง ดังรูปที่ 2.6

แต่ละตำแหน่งจุดภาพบนจอภาพและตำแหน่งในหน่วยความจำที่สอดคล้องกันในเฟรมบัฟเฟอร์จะถูกอ้างอิงได้โดยใช้คู่ลำดับ (x , y)โดยที่ x จะแทนตำแหน่งของหลัก ส่วน y แทนตำแหน่งของแถว จุด (0,0) ของระบบพิกัดนี้จะอยู่ที่มุมบนซ้ายของจอภาพดังรูปที่ 2.6



รูปที่ 2.6 แสดงความสัมพันธ์ของเฟรมบัฟเฟอร์กับจอภาพ

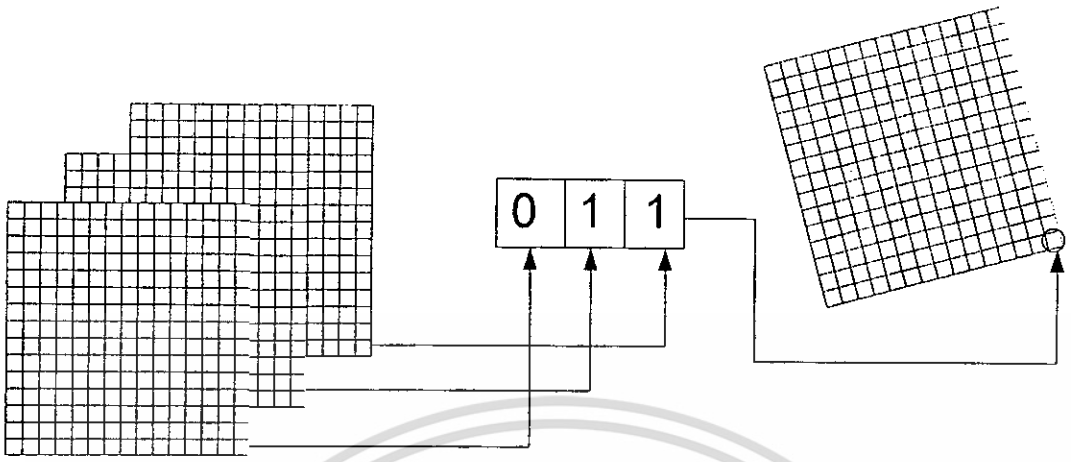
ข้อมูลในเฟรมบัฟเฟอร์ซึ่งใช้แทนจุดภาพแต่ละจุดนั้นจะประกอบด้วยบิทจำนวนหนึ่ง สำหรับจอภาพขาวดำซึ่งมีความเข้มเพียง 2 ระดับ ข้อมูลในเฟรมบัฟเฟอร์จะมีเพียง 1 บิท (1-Bit Plane Frame Buffer) ซึ่งต่างกับจอภาพขาวดำที่มีความเข้มแสงหลายระดับ ข้อมูลสำหรับ 1 จุดภาพจะต้องมีมากกว่า 1 บิท ในภาพที่ 2.8 แสดงเฟรมบัฟเฟอร์ที่ใช้ 3 บิทนั่นคือ ใน 1 จุดภาพจะมีค่าใช้แทนจุดภาพนี้ได้ 8 ค่า (2^3) ซึ่งแต่ละค่าจะแทนความเข้ม 1 ระดับรวมทั้งหมดจะแทนได้ 8 ระดับ จากระดับ 0 ถึงระดับ 2^3 -1 =7 สำหรับจอภาพขาวดำถ้าใช้ข้อมูล 8 บิทสำหรับ 1 จุดภาพก็จะสามารถแสดงระดับความเข้มได้ถึง 2^8 หรือ 256 ระดับ สำหรับระบบจอภาพสีต้องการข้อมูล 24 บิท (24-Bit Plane Frame Buffer) โดยที่จะใช้ 8 บิท สำหรับแต่ละแม่สีคือ แดง , เขียว ,น้ำเงิน ซึ่งตามทฤษฎีแล้วจะสร้างได้ถึง 2^{24} เท่ากับ 16,777,216 สี สำหรับจอภาพที่มีความละเอียด512x512x24= 6,291,456 บิท ซึ่งหน่วยความจำของคอมพิวเตอร์ราคาต่ำไม่สามารถมีหน่วยความจำขนาดนี้ได้ ดังนั้นข้อมูลต่อ 1 จุดภาพจึงมีแค่เพียง 1 ถึง 4 บิท เท่านั้น



รูปที่ 2.7 แสดงระบบพิกัดของจอภาพ

2.3.2 ตัวควบคุมการแสดงภาพ

ในส่วนที่ 2 ของหน่วยการแสดงภาพ คือ ตัวควบคุมการแสดงภาพ ฮาร์ดแวร์ส่วนนี้จะอ่านค่าที่อยู่ในเฟรมบัฟเฟอร์ไปไว้ในวีดีบัฟเฟอร์ (Video Buffer) ซึ่งจะเปลี่ยนค่าบิทเหล่านี้ให้เป็นสัญญาณทางไฟฟ้าซึ่งใช้สำหรับการควบคุมการแสดงภาพบนจอภาพด้วย ตัวอย่างเช่น ถ้าตัวควบคุมการแสดงภาพพบค่าบิท 1 ในเฟรมบัฟเฟอร์ที่มีข้อมูล 1 บิทต่อจุดภาพ ก็จะเกิดการส่งสัญญาณแรงดันสูงไปให้ CRT ซึ่งจะจัดการให้เกิดจุดสว่างบนจอภาพในตำแหน่งที่สอดคล้องกับข้อมูลที่อยู่ในเฟรมบัฟเฟอร์นั่นเอง



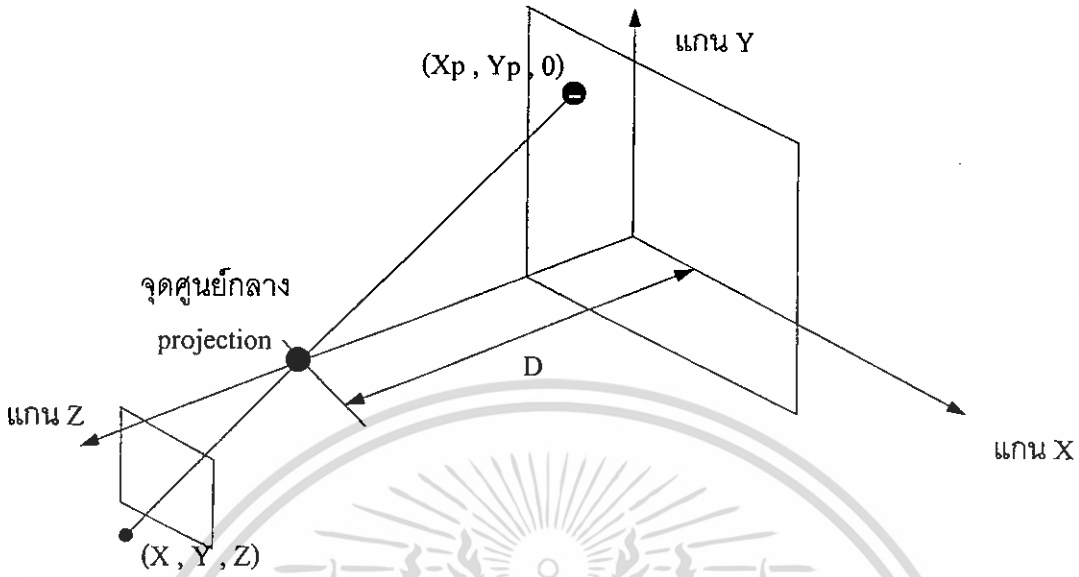
รูปที่ 2.8 แสดงเฟรมบัฟเฟอร์ที่ใช้ 3 บิทต่อ 1 จุดภาพ

2.3.3 วิธีการแปลงภาพให้เป็นตำแหน่งของจุดภาพที่เหมาะสมในเฟรมบัฟเฟอร์

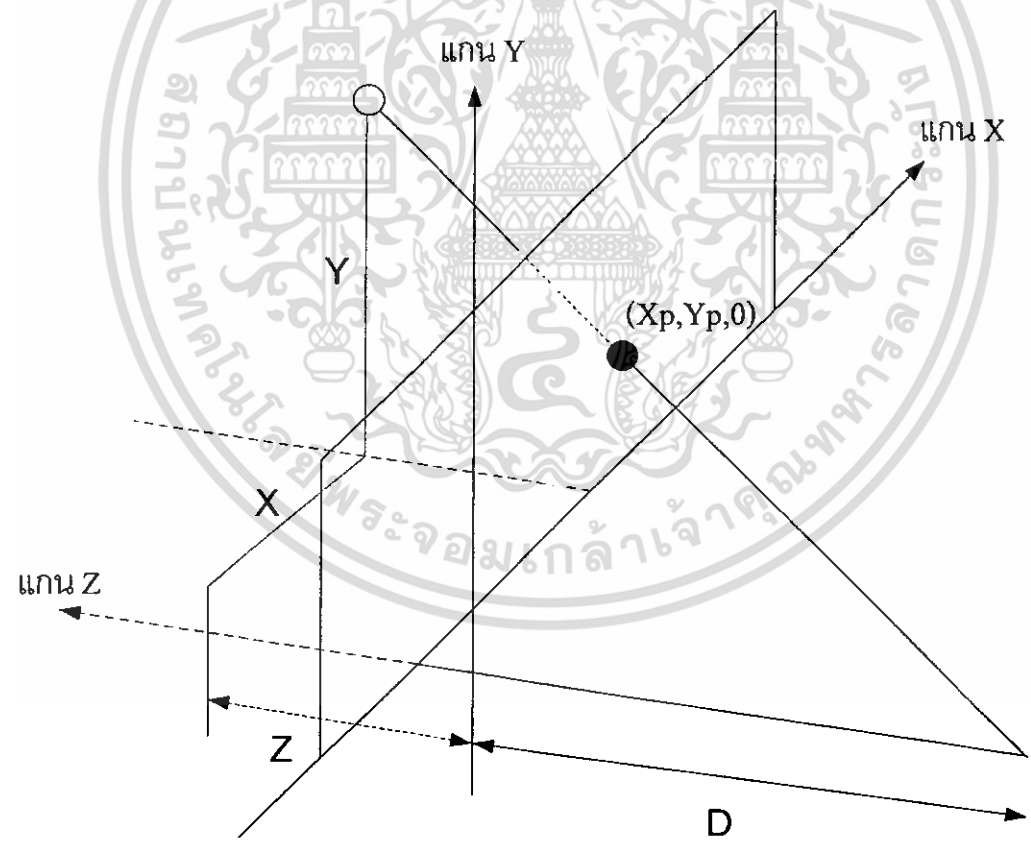
ส่วนนี้เป็นวิธีการหรือกระบวนการที่ใช้การเปลี่ยนคำสั่ง หรือสมการให้เป็นค่าที่เหมาะสม ซึ่งสามารถใช้แทนภาพที่ได้จากสมการหรือคำสั่งนั้นได้แล้วเก็บลงบัฟเฟอร์ เช่น สมการเส้นตรง x + y.= 5 หรือคำสั่ง Line (a ,b) จะต้องผ่านกระบวนการอันหนึ่งเพื่อให้ได้ค่าที่เหมาะสมแล้วเก็บในเฟรมบัฟเฟอร์สำหรับจอภาพที่มีคุณภาพสูงๆจะมีตัวประมวลผล(Processor) จัดการการแสดงภาพโดยเฉพาะเพื่อทำกระบวนการข้างต้นรวมทั้งการจัดการอื่นๆเกี่ยวกับจอภาพ เช่น การลบจอภาพ เป็นต้น ส่วนจอภาพที่ราคาไม่แพงนัก จะใช้ CPU ของเครื่องกับโปรแกรมสำหรับจัดการงานดังกล่าวซึ่งทำให้การทำงานทำได้ช้ากว่ามาก ยิ่งถ้าใช้ในระบบกราฟฟิกยากที่จะทำให้ระบบกราฟฟิกเป็นแบบอินเตอร์แอกทีฟได้ เนื่องจากการเปลี่ยนแปลงภาพไปเพียงเล็กน้อยจะต้องมีการคำนวณมากมายตามมาเสมอ

2.4 การแสดงภาพตามสัดส่วนระยะทาง

การแสดงภาพตามสัดส่วนระยะทางมีอยู่หลายวิธีด้วยกัน ขึ้นอยู่กับการเลือกการนำมาใช้ซึ่งในปริญญานิพนธ์นี้จะใช้หลักการที่เรียกว่า การแปลงแบบเพอร์สเพคทีฟ (Perspective Transformation) ซึ่งนิยมใช้ในการแปลงภาพจากพิกัดโลก หรือ World Coordinate ไปยังฉากรับภาพ (Image Plane) แล้วเก็บลงในหน่วยความจำภาพเพื่อใช้ประมวลผลภาพต่อไป หรือนำไปแสดงบนจอคอมพิวเตอร์ที่เห็นดังที่ใช้งานกันอยู่



(a) การแปลงจุดภาพบนวัตถุในพิกัดโลกไปเป็นจุดภาพบนฉากรับภาพ



(b) การแปลงจุดบนวัตถุผ่านเส้นตรงไปจบที่ศูนย์กลางของ Projection

รูปที่ 2.9 แสดงรูปการแปลงแบบเพอร์สเพคทีฟ

รูปแบบการแปลงแบบเพอร์สเพคทีฟ ซึ่งจะใช้หลักการของกล้องวีดีโอในการอธิบายดังรูปที่ 2.9 (a) ดังการแปลงบนจุดภาพบนวัตถุในพิกัดโลกไปเป็นจุดภาพบนฉากรับภาพ โดยผ่านจุดโฟกัสของเลนส์หรือกำหนดให้เป็นศูนย์กลางของ Projection และเพื่อให้ดูง่ายขึ้นจะย้ายจุดศูนย์กลางของ Projection ไปไว้ที่ส่วนปลายและย้ายจุดฉากรับภาพขึ้นมา ดังแสดงในรูป 2.9 (b)

เมื่อทำการแปลงจุดบนวัตถุผ่านเส้นตรงไปจบที่จุดศูนย์กลางของ Projection ในกรณีของภาพที่ 2.9 (b) ศูนย์กลางของ Projection วางอยู่บนแกน z ทางด้านลบซึ่งห่างจากฉากรับภาพเป็นระยะ D (D คือความยาวโฟกัสของกล้อง) เราสามารถเลือกตำแหน่งอื่นๆให้เป็นศูนย์กลางของ Projection ได้ แต่การเลือกตำแหน่งตามแกน z จะง่ายต่อการคำนวณ

รูปแบบการแปลงแบบเพอร์สเปคทีฟ จะนำมาเขียนเป็นสมการพาราเมตริกซ์ที่เกิดขึ้นบนเส้นตรงจากจุดบนวัตถุ P(X,Y,Z) ไปยังศูนย์กลางของ Projection ได้เป็นสมการต่างๆดังนี้

$$\alpha = X - X\delta \tag{2.1}$$

$$\beta = Y - Y\delta \tag{2.2}$$

$$\gamma = Z - (Z + D)\delta \tag{2.3}$$

เมื่อพารามิเตอร์ δ มีค่าเป็น 0 -1 และโคออดิเนต (α, β, γ)แทนตำแหน่งบนเส้นตรง เมื่อ $\delta = 0$ เป็นผลให้จุดอยู่ที่วัตถุเป็น (X,Y,Z) เมื่อ δ =1 เป็นผลให้จุดไปอยู่ที่ศูนย์กลาง Projection ที่ตำแหน่ง (0,0,-D) เพื่อให้ได้รับโคออดิเนตบนฉากรับภาพ เราจะให้ γ=0 และคำนวณค่าพารามิเตอร์ δ จะได้

$$\delta = \frac{Z}{(Z + D)} \tag{2.4}$$

จากค่าของพารามิเตอร์ δ เป็นผลให้เส้นตรงตัดกับฉากรับภาพที่ตำแหน่ง $(X_p, Y_p, 0)$ แทนค่า δ จากสมการ (2.4) ลงในสมการ (2.1) , (2.2) และ(2.3) เราจะได้สมการของการแปลงแบบเพอร์สเปคทีฟดังนี้

$$X_p = \frac{D}{(Z + D)} \bullet X \tag{2.5}$$

$$Y_p = \frac{D}{(Z + D)} \bullet Y \tag{2.6}$$

$$Z_p = 0 \tag{2.7}$$

จากสมการ (2.5) , (2.6) และ (2.7) เป็นสมการที่ได้จากการแปลงภาพจากพิกัดโลกไปยังฉากรับภาพ แล้วเก็บลงในหน่วยความจำภาพเพื่อใช้ในการประมวลผลต่อไป

ในปริญญานิพนธ์นี้ต้องการรู้ตำแหน่งในพิกัดโลกของวัตถุ เพื่อให้โปรแกรมสามารถคำนวณตำแหน่งของวัตถุได้จริง ดังนั้นจึงต้องใช้ทฤษฎีต่างๆเข้าช่วยซึ่งจะได้กล่าวในบทถัดไป

บทที่ 3
ทฤษฎีและหลักการเบื้องต้น

การเห็นใน 3 มิติ (3D Vision)

ในบทนี้เรากล่าวถึงการเห็น 2 มิติ 3 มิติ มีหลายสาเหตุที่ยากลำบากในการนำข้อมูลการเห็น 2 มิติ เป็น 3 มิติ ได้แก่

1. ระบบภาพของกล้องและของตามนุษย์มีการโปรเจคชันแบบเพอสเป็กทีพ (Perspective - Projection) ซึ่งทำให้มีการสูญเสียข้อมูลอย่างมาก
2. ความสัมพันธ์ระหว่างภาพใน 2 มิติและเรขาคณิตใน 3 มิติของจุดที่สอดคล้องกันในฉากมีความซับซ้อนมาก
3. การบังกันของวัตถุในฉากเป็นการเพิ่มความยุ่งยากของการเห็นภาพใน 3 มิติ
4. การปรากฏอยู่ของนอยส์ในภาพ

3.1 การแปลงเรขาคณิตของวัตถุใน 3 มิติ (Geometric Transformation in 3D)

หมายถึงการแปลงเรขาคณิตสำหรับวัตถุใน 3 มิติ ซึ่งมีพิกัดอยู่ในรูป (x, y, z)

3.1.1 การเคลื่อนย้าย (Translation)

การย้ายจุดที่มีพิกัด (x, y, z) ไปยังจุดใหม่ที่มีระยะเคลื่อนที่ (t_x, t_y, t_z) ดังสมการ

$$\begin{aligned} U &= x + t_x \\ V &= y + t_y \\ W &= z + t_z \end{aligned} \qquad (3.1)$$

สามารถเขียนในระบบพิกัดโฮโมจีเนียสได้เป็น

$$\begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.2)$$

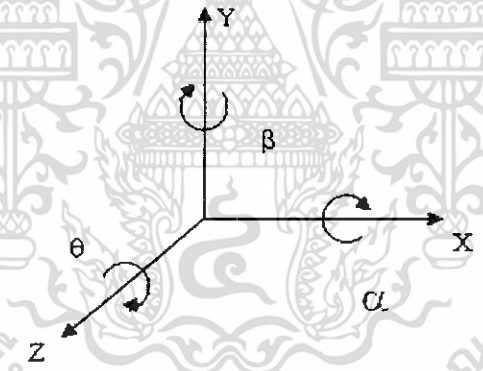
3.1.2 การย่อ/ ขยาย(Scalling)

การย่อ/และการขยายใน 3 มิติ ที่มีพารามิเตอร์ S_x , S_y และ S_z ตามแนวแกน x , y และ z เขียนในรูปเมทริกซ์ได้เป็น

$$\begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 & t_x \\ 0 & S_y & 0 & t_y \\ 0 & 0 & S_z & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.3)$$

3.1.3 การหมุนภาพ (Rotation)

การหมุนใน 3 มิติ สามารถแบ่งได้เป็น 3 ชนิดนั่นคือหมุนรอบแกน x , แกน y และแกน z การหมุนวัตถุรอบจุดใดๆ เราต้องทำการย้ายจุดนั้นมาที่จุดกำเนิดจากนั้นทำการหมุนและย้ายกลับไปที่เดิม



รูปที่ 3.1 แกนพิกัด 3 มิติและการหมุนจุดรอบแกนพิกัด มุมของการหมุนวัดตามเข็มนาฬิกาเมื่อมองจากแกนสู่จุดกำเนิด

การหมุนรอบแกน Z ด้วยมุม θ

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (\text{สมการ } R_\theta) \ (3.4)$$

การหมุนรอบแกน X ด้วยมุม α

$$R_\alpha = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(สมการ } R_\alpha\text{)} \quad (3.5)$$

การหมุนรอบแกน Y ด้วยมุม β

$$R_\beta = \begin{pmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(สมการ } R_\beta\text{)} \quad (3.6)$$

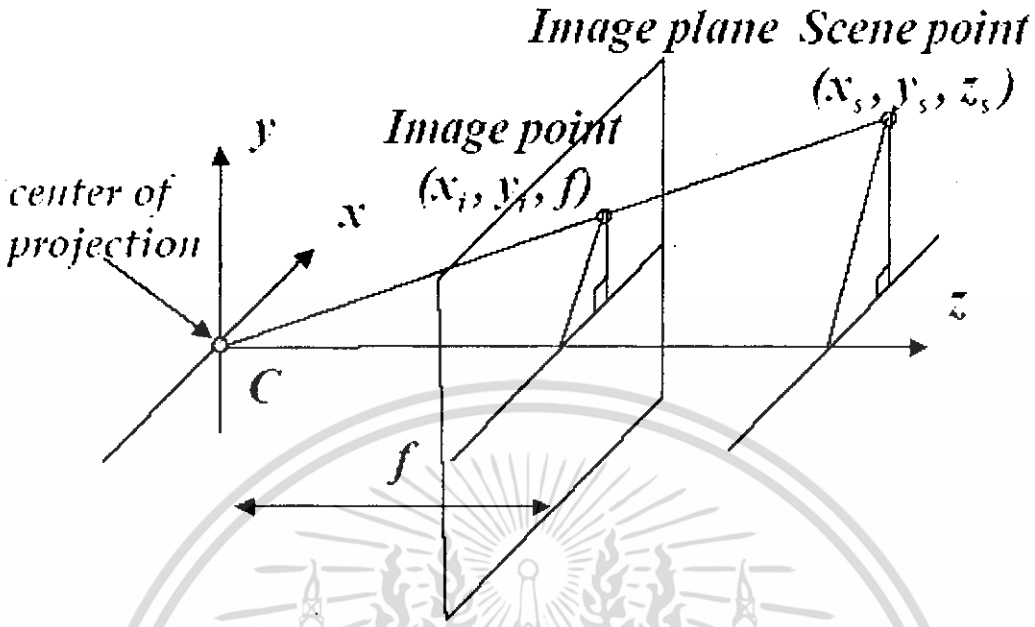
3.2 เรขาคณิตสำหรับการเห็นใน 3 มิติ (Geometry for 3D Vision)

3.2.1 พื้นฐานของการโปรเจ็คชันแบบเพอสเป็กทีพของกล้อง (Basis of Perspective Camera)

ภาพ 2 มิติส่วนใหญ่ได้มาจากการมองวัตถุ 3 มิติผ่านเลนส์ของตาหรือเลนส์ของกล้องภาพสามารถถูกจำลองได้โดยการใช้ Perspective Transformation ดังรูปที่ 3.2 เราแทนแกนพิกัดโลก (World Coordinate System) ด้วยอักษรตัวใหญ่ (X, Y, Z)และแทนแกนของระบบพิกัดของกล้องด้วยตัวอักษรตัวเล็ก (x, y, z) เราสมมติว่าพิกัดของโลกซ้อนทับกับระบบพิกัดของกล้องแสงจากวัตถุใน 3 มิติ ส่องผ่านศูนย์ของการโปรเจคชันแล้วโปรเจ็คลงบนภาพ ให้จุด (x_s, y_s, z_s) แทนจุดพิกัดของวัตถุ 3 มิติ ให้ (x_i , y_i) ให้ f คือโฟกัสของเลนส์ โดยใช้สามเหลี่ยมคล้าย

$$x_i = \frac{fx_s}{z_s}$$

$$y_i = \frac{fy_s}{z_s} \qquad (3.7)$$



รูปที่ 3.2 แบบจำลองระบบสร้างภาพ

ให้

$$v = \begin{pmatrix} X_s \\ Y_s \\ Z_s \end{pmatrix} \tag{3.8}$$

เป็นเวคเตอร์ที่ประกอบจุดพิกัดของวัตถุเวคเตอร์ในระบบโฮโมจีเนียส $\tilde{v}$ ของ v คือ

$$\tilde{v} = \begin{pmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{pmatrix} \tag{3.9}$$

เป็นเวคเตอร์ในระบบพิกัด Cartesian v สามารถได้จากเวคเตอร์ในระบบโฮโมจีเนียสได้โดยการหารสามสมาชิกแรกด้วยสมาชิกที่สี่

พิจารณาเมทริกซ์ของ Perspective Transformation

$$P = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.10}$$

ทำการคูณเวคเตอร์

$$\tilde{w} = P\tilde{v} \tag{3.11}$$

ได้เป็น

$$\tilde{w} = \begin{pmatrix} fx_s \\ fy_s \\ z_s \end{pmatrix} \tag{3.12}$$

ทำการนอร์มอไลซ์สมาชิกที่สามให้เป็น 1 เพื่อแปลงเป็นเวคเตอร์ในระบบพิกัด Cartesian ดังนั้น

$$w = \begin{pmatrix} \frac{fx_s}{z_s} \\ \frac{fy_s}{z_s} \end{pmatrix} \tag{3.13}$$

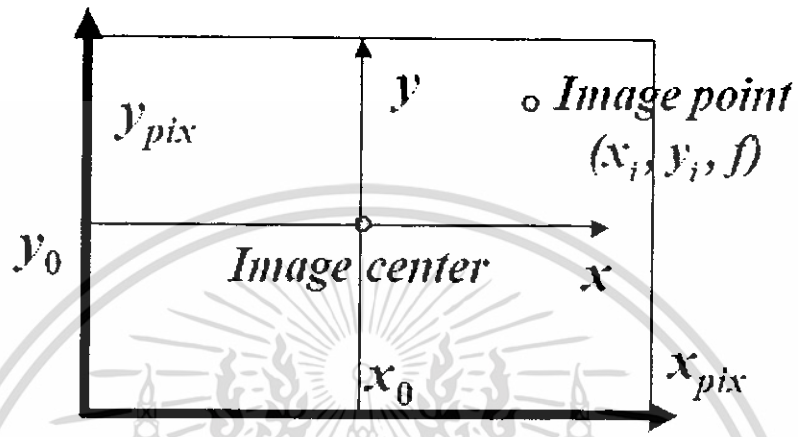
3.2.2 การแปลงจากหน่วยความยาวเป็นพิกเซล

การหาพิกัดของจุดที่ถูกโปรเจ็คชัน (x_i , y_i , f) ในหน่วยของพิกเซล กำหนดให้พิกัดของภาพมีจุดกำเนิดอยู่ที่มุมด้านล่างซ้ายของภาพดังแสดงในรูปที่ 3.3 การแปลงจากความยาวเป็นพิกเซลเราจำเป็นต้องรู้ค่าการปรับสเกล (Aspect Ratio) ของแต่ละแกนของระบบพิกัดภาพ

โดยที่ k_x คือค่าการปรับสเกลในทิศทาง x , k_y คือค่าการปรับสเกลในทิศทาง y และ s แทนพารามิเตอร์ของการเฉือน เราเขียนในรูปเทียบกล้อง (Camera Calibration Proceduce) เป็นจุดตัดระหว่างแกนอ็อบติคกับระนาบภาพ

พิกัดของจุด (x_i , y_i , f) หาได้จากความสัมพันธ์

$$x_{pix} = k_x x_i + s y_i + x_o$$
$$= k_x f \frac{x_s}{z_s} + sf \frac{y_s}{z_s} + x_o \quad (3.14)$$



รูปที่ 3.3 แสดงพิกัดภาพ

$$y_{pix} = k_y y_i + y_o$$
$$= k_y f \frac{y_s}{z_s} + y_o \quad (3.15)$$

เราสามารถอยู่ในรูปของเมทริกซ์ได้เป็น

$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = \begin{pmatrix} \alpha_x & s & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_s \\ y_s \\ z_s \\ 1 \end{pmatrix} \quad (3.16)$$

โดยที่ $\alpha_x = k_x x_s$ ซึ่งเป็นค่าความยาวโฟกัสในทิศ x ในหน่วยของพิกเซ็ลและ $\alpha_x = k_x x_s$ ซึ่งเป็นค่าความยาวโฟกัสในทิศ y ในหน่วยของพิกเซ็ล เราสามารถเขียน

$$\begin{pmatrix} \alpha_x & s & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_x & s & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = K [I_3 | O_3]$$

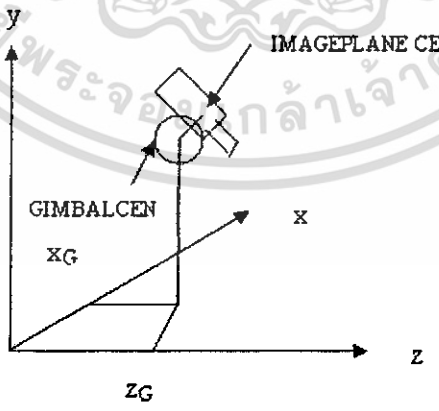
(3.17)

เมทริกซ์ K มีลักษณะเป็นสามเหลี่ยมข้างบน (Upper Triangle) ขนาด 3x3 มีชื่อว่า Calibration Matrix เป็นเมทริกซ์ที่ประกอบพารามิเตอร์ภายใน (Intrinsic Parameter) ที่สำคัญของกล้อง

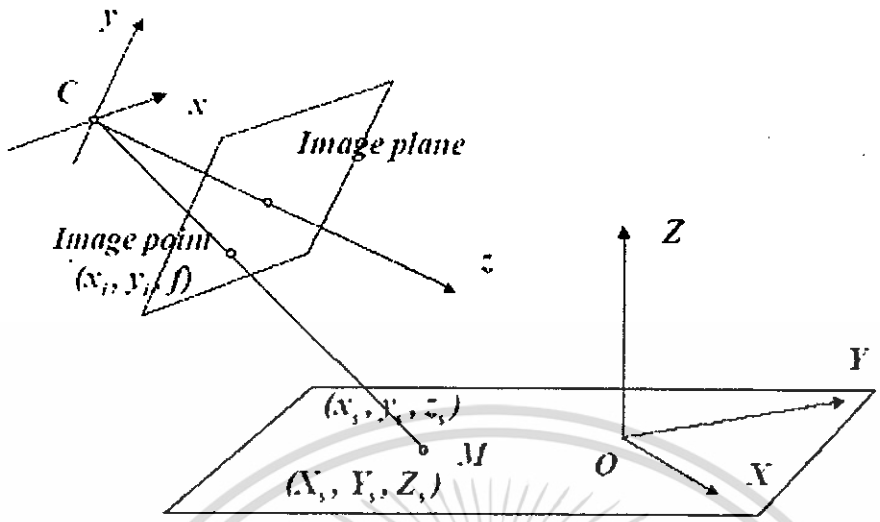
3.3 กรณีที่พิกัดของโลกไม่ซ้อนทับกับพิกัดของกล้อง

แบบจำลองการเกิดภาพจากกล้องที่ไม่มีสมมติฐานรูปที่ 3.4 (a) แสดงกล้องอิเล็กทรอนิกส์ในช่องว่างพิกัดของโลก (World Coordinate Space) กล้องนี้ถูกยึดโดยหัวยึดกล้อง (Gimbal) ที่สามารถหมุนทำส่าย (Pan) มุม θ กับแนวนอน (Horizontal) และเอียง (Tilt) ทำมุม ϕ กับแนวดิ่ง จุดกึ่งกลางของหัวยึดกล้องอยู่ที่จุดพิกัด (X_G, Y_G, Z_G) เทียบกับระบบอ้างอิงของโลก จุดกึ่งกลางของหัวยึดกล้องและจุดกึ่งกลางของการโปรเจ็คชันของระนาบภาพอยู่ห่างกันด้วยระยะ (X_0, Y_0, Z_0) รูป 3.4 (b) แสดงระบบพิกัดโลกและระบบพิกัดกล้องที่ไม่ซ้อนทับกัน

กรณีที่จุดกึ่งกลางของการโปรเจ็คชันของกล้องถูกวางไว้ที่จุดกึ่งกลางของระบบพิกัดอ้างอิงโลก ไม่มีการส่ายหรือเอียงทำมุมกับแกนอ้างอิง จุดกึ่งกลางของหัวยึดกล้องและของ



(a)



(b)

รูปที่ 3.4 (a) แบบจำลองการเกิดภาพจากกล้อง (b) เรขาคณิตของการเกิดภาพ

ระนาบภาพอยู่ที่เดียวกัน แบบจำลองการเกิดภาพ (Imaging Model) ในระบบพิกัดโฮโมจีเนียสนั่นคือ

$$\tilde{w} = M\tilde{v} \tag{3.18}$$

โดยที่ $\tilde{v}$ คือเวคเตอร์ของจุดบนวัตถุในระบบพิกัดโฮโมจีเนียส $\tilde{w}$ คือเวคเตอร์ของจุดบนระนาบภาพในระบบพิกัดโฮโมจีเนียสและ M คือเมทริกซ์ของ Perspective Transformation ในกรณีนี้แบบจำลองการเกิดภาพจากกล้องสามารถหาได้ง่ายโดยการเปลี่ยนแปลง (Modify) สมการ (3.17)

การที่ (หัวยึดกล้อง) กล้องอยู่ที่จุดพิกัด (X_G, Y_G, Z_G) เทียบกับระบบพิกัดอ้างอิงของโลก ดังนั้นแบบจำลองใหม่คือ

$$\tilde{w} = MT_G\tilde{v} \tag{3.19}$$

โดยที่

$$T_G = \begin{pmatrix} 1 & 0 & 0 & -X_G \\ 0 & 1 & 0 & -Y_G \\ 0 & 0 & 1 & -Z_G \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.20}$$

การส่ายหรือการเอียง (Pan and Tilt) โดยใช้เมทริกซ์การหมุนคูณเข้า

$$\tilde{w} = MRT_G\tilde{v} \tag{3.21}$$

โดยที่ $R = R_\phi R_\theta$

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.22}$$

และ

$$R_\phi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.23}$$

ท้ายสุดจุดกึ่งกลางของหัวยึดกล้องและจุดกึ่งกลางของการโปรเจคชันของระนาบภาพอยู่ห่างกันด้วยระยะ(X_o, Y_o, Z_o) ดังนั้น แบบจำลองของการเกิดภาพจากกล้องคือ

$$\tilde{w} = MT_C RT_G\tilde{v} \tag{3.24}$$

โดยที่

$$T_C = \begin{pmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.25}$$

สมการ (3.24) สามารถเขียนใหม่ได้เป็น

$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = K\begin{bmatrix} I_3 \mid O_3 \end{bmatrix}\begin{pmatrix} R & -T \\ O_3^T & 1 \end{pmatrix}\begin{pmatrix} x_s \\ y_s \\ z_s \\ 1 \end{pmatrix} \tag{3.26}$$

โดยที่ R แทนเมทริกซ์ที่เป็นผลรวมของการหมุนระบบพิกัด และ T แทนเวคเตอร์ที่เป็นผลรวมของการย้ายระบบพิกัด เรากล่าวว่าเมทริกซ์

$$\begin{pmatrix} R & -T \\ O_3^T & 1 \end{pmatrix}$$

(เมทริกซ์)

เป็นเมทริกซ์ที่รวมเอาพารามิเตอร์ภายนอก (Extrinsic Parameter) ของขบวนการปรับเทียบกล้อง เราสามารถเขียนสมการ (3.26) ในรูปแบบที่ง่ายได้เป็น

$$x = MX \tag{3.27}$$

โดยที่ $M = [KR \mid -KRT]$ (3.28)

3.4 การปรับเทียบกล้อง (Camera Calibration)

การปรับเทียบกล้องเป็นการหาประมาณเมทริกซ์ M จากจุดในฉาก 3 มิติที่รู้ตำแหน่งและภาพที่เกิดขึ้นของจุด จากนั้นทำการหาพารามิเตอร์ภายในและพารามิเตอร์ภายนอก พิจารณาการปรับเทียบโดยใช้กล้องตัวเดียว

ในการหาเมทริกซ์ M เราทำการหาจุดในฉากที่เราทราบพิกัด $X = [x, y, z]^T$ และจุดในภาพ 3 มิติที่สอดคล้องกัน $[u, v]^T$ ซึ่งจะได้สมการที่อยู่ในรูปสมการ (3.29) ดังนี้

$$\begin{pmatrix} \alpha_U \\ \alpha_V \\ \alpha \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} x_S \\ y_S \\ z_S \\ 1 \end{pmatrix} \tag{3.29}$$

$$\begin{pmatrix} \alpha_U \\ \alpha_V \\ \alpha \end{pmatrix} = \begin{pmatrix} m_{11}x+ & m_{12}y+ & m_{13}z+ & m_{14} \\ m_{21}x+ & m_{22}y+ & m_{23}z+ & m_{24} \\ m_{31}x+ & m_{32}y+ & m_{33}z+ & m_{34} \end{pmatrix} \tag{3.30}$$

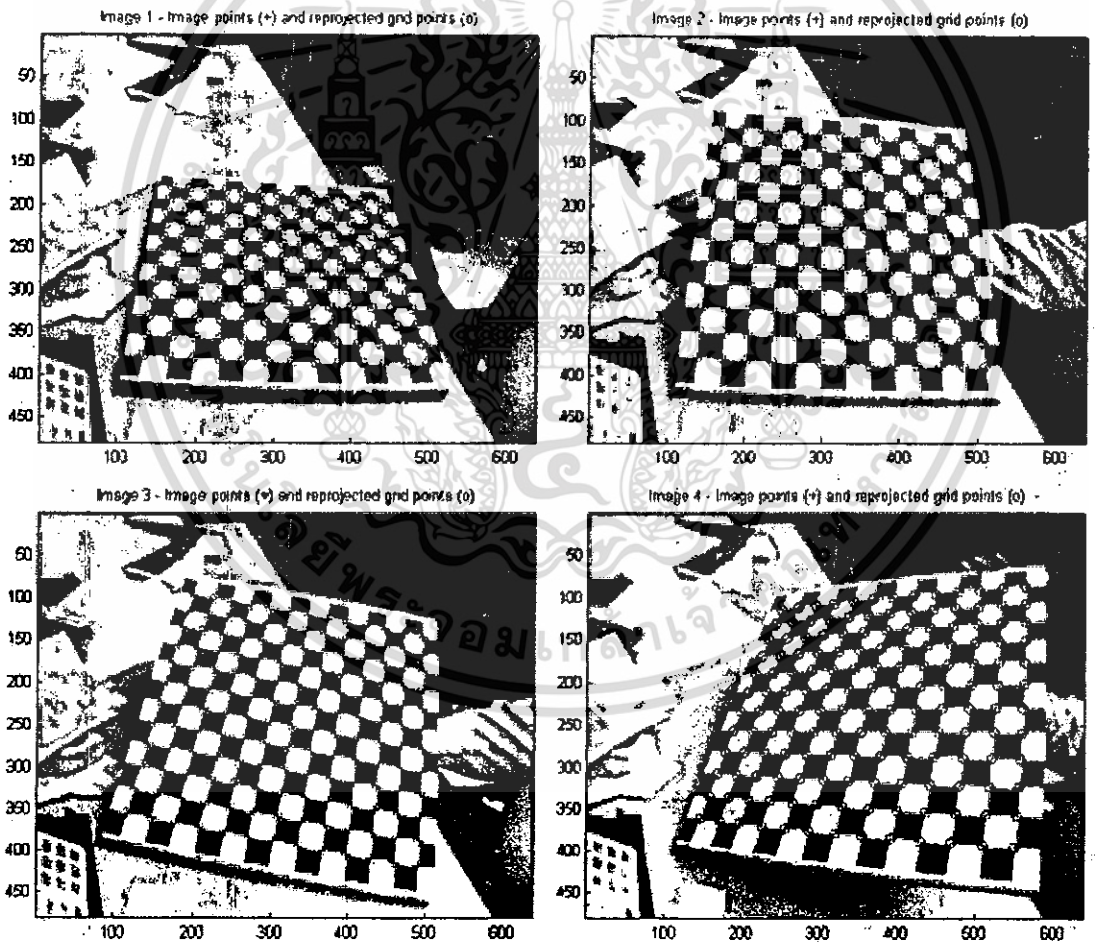
ทำการนอร์มอไลซ์ให้สมาชิกที่ 3 เป็น 1 เราจะได้สมการ

$$\begin{aligned} U\left(m_{31}x + m_{32}y + m_{33}z + m_{34}\right) &= m_{11}x + m_{12}y + m_{13}z + m_{14} \\ V\left(m_{31}x + m_{32}y + m_{33}z + m_{34}\right) &= m_{11}x + m_{12}y + m_{13}z + m_{14} \end{aligned} \tag{3.31}$$

ดังนั้นเราจะได้สมการเชิงเส้น 2 สมการสำหรับจุดในฉาก 3 มิติ หนึ่งจุดและจุดในภาพ 2 มิติที่สอดคล้องกัน ทำการเขียนสมการ(3.31)ใหม่ในรูปของเมทริกซ์ได้เป็น

$$\begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -ux & -uy & -uz & -u \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -vx & -vy & -vz & -v \\ & & & & & \cdot & & & & & & \\ & & & & & \cdot & & & & & & \\ & & & & & \cdot & & & & & & \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ \cdot \\ \cdot \\ \cdot \\ m_{34} \end{bmatrix} = 0 \qquad (3.32)$$

จะเห็นได้ว่าเรามีตัวแปรที่ไม่ทราบค่า 11 ตัวแปร แทนที่จะเป็น 12 ตัวแปรเนื่องจากแฟคเตอร์การย่อ/ขยายที่ไม่สามารถทราบค่าได้ ในการแก้สมการโฮโมจีเนียสเราต้องใช้อย่างน้อย 6 จุด ถ้ามีมากกว่า 6 จุดเราจะได้สมการ Over-determined ซึ่งสามารถแก้ได้โดยใช้วิธี Least Square



รูปที่ 3.5 ภาพของตารางหมากรุกที่นำมาใช้ในการเปรียบเทียบกล้อง

เมื่อได้เมทริกซ์ M แล้วขั้นตอนต่อไปทำการแยกพารามิเตอร์ภายใน (Intrinsic Parameter) เนื่องจาก

$$M=[KR \mid -KRT] \sim [A \mid B] \tag{3.33}$$

ดังนั้นเมทริกซ์ย่อย 3x3 แทนด้วย A เวคเตอร์คอลัมน์ทางขวามือสุดแทนด้วย b เวคเตอร์ที่มีพารามิเตอร์ระยะเคลื่อนที่สามารถแยกออกมาได้ง่ายโดย $t = -A^{-1}b$ เวคเตอร์ t ให้ข้อมูลที่บอกถึงตำแหน่งกึ่งกลางของ Image Plane

จากนั้นพิจารณา A =KR โดยที่ K เป็นเมทริกซ์สามเหลี่ยมบน(Upper Triangle) และเมทริกซ์ R เป็นเมทริกซ์ออโธกอนอล การแยกเมทริกซ์ K และ R สามารถทำได้โดยใช้เทคนิค QR Decomposition สำหรับเมทริกซ์ A

3.5 การหาพิกัด 3 มิติของจุดในฉากโดยใช้กล้อง 2 ตัว

ในหัวข้อนี้เราพิจารณาการหาพิกัด 3 มิติของจุดในฉากโดยใช้กล้อง 2 ตัว โดยที่กล้องทั้งสองตัวผ่านขบวนการปรับเทียบกล้องที่กล่าวในหัวข้อ ในขบวนการปรับเทียบเราให้กล้องทั้งสองถ่ายภาพวัตถุในฉาก 3 มิติที่เราทราบพิกัด ซึ่งอาจเป็นตารางหมากรุกที่เราทราบขนาดของแต่ละตาราง (แสดงในรูปที่ 3.5) จากนั้นทำการหาพิกัดของจุดตัดของตารางในกล้องทั้งสอง ใช้สมการ (3.32) ทำการคำนวณเมทริกซ์ของ Perspective Transformation M สำหรับแต่ละกล้องต่อไปทำการวางวัตถุที่เราต้องการหาพิกัด 3 มิติในฉาก ทำการหาจุดสอดคล้องของจุดในฉาก 3 มิติของกล้องทั้งสองตัว กำหนดให้เมทริกซ์ของ Perspective Transformation M ของกล้องที่ 1 และ 2 เป็น M และ M' ตามลำดับ เราแทนแต่ละแถวของเมทริกซ์ M ด้วย m_1^T , m_2^T , m_3^T ทำนองเดียวกันสำหรับกล้องตัวที่สอง เราแทนแต่ละแถวของเมทริกซ์ M' ด้วย m'^T_1 , m'^T_2 , m'^T_3
สำหรับกล้องที่ 1 เราได้

$$u = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = MX = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} x \tag{3.34}$$

สำหรับกล้องที่ 2 เราได้

$$u' = \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = M'X = \begin{bmatrix} m'^T_1 \\ m'^T_2 \\ m'^T_3 \end{bmatrix} x \tag{3.35}$$

โดยที่ u และ u' แทนพิกัดในภาพของวัตถุในกล้อง 1 และ 2 ตามลำดับ ในการกำจัดค่าสเกลที่ไม่ทราบค่า เราทำการหาสัดส่วนระว่างสามแถวในเมทริกซ์ Perspective Transformation M

$$u : v : w = m_1^T X : m_2^T X : m_3^T X$$
$$u' : v' : w' = m_1'^T X : m_2'^T X : m_3'^T X \tag{3.36}$$

ดังนั้นเราได้สมการสำหรับกล้อง 1 และกล้อง 2

$$\begin{aligned} um_2^T X &= vm_1^T X & u'm_2'^T X &= v'm_1'^T X \\ um_3^T X &= vm_1^T X & u'm_3'^T X &= v'm_1'^T X \\ um_3^T X &= vm_2^T X & u'm_3'^T X &= v'm_2'^T X \end{aligned} \tag{3.37}$$

สมการ (3.37) สามารถเขียนเป็น

$$\begin{bmatrix} um_2^T - vm_1^T \\ um_3^T - vm_1^T \\ um_3^T - vm_2^T \end{bmatrix} X = 0 \tag{3.38a}$$

$$\begin{bmatrix} u'm_2'^T - v'm_1'^T \\ u'm_3'^T - v'm_1'^T \\ u'm_3'^T - v'm_2'^T \end{bmatrix} X = 0 \tag{3.38b}$$

ทำการคูณแถวแรกของสมการ (3.38a) ด้วย w แถวที่สองด้วย –v แล้วทำการบวก เราได้

$$(uwm_2^T - vwm_1^T - uvm_3^T + vwm_1^T)X = (uwm_2^T - uwm_3^T)X = 0 \tag{3.39}$$

จะเห็นว่าสมการ (3.39) มีคุณสมบัติ Linearly Dependent กับแถวที่สามของสมการ (3.38a) ดังนั้นแต่ละกล้องเราใช้ได้เพียง 2 สมการดังนี้

$$\begin{bmatrix} um_3^T - vm_1^T \\ um_3^T - vm_2^T \end{bmatrix} X = 0 \quad \text{และ} \quad \begin{bmatrix} u'm_3'^T - v'm_1'^T \\ u'm_3'^T - v'm_2'^T \end{bmatrix} X = 0 \tag{3.40}$$

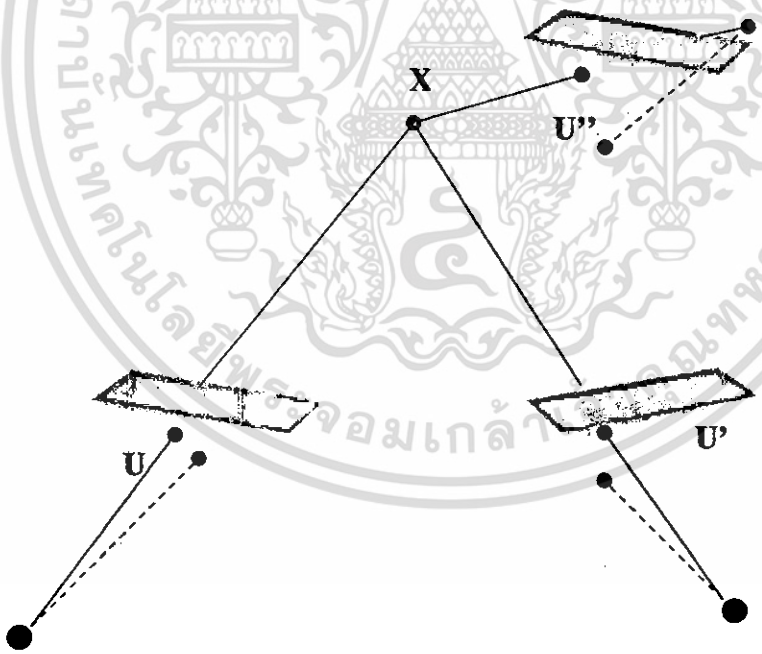
เขียนสมการ (3.39) ใหม่ได้ดังนี้

$$\begin{bmatrix} um_3^T - wm_1^T \\ vm_3^T - wm_2^T \\ um_3^T - wm_1^T \\ vm_3^T - wm_2^T \end{bmatrix} X = AX = 0 \qquad (3.41)$$

ทำการแก้สมการโฮโมจีเนียส (3.41) สำหรับพิกัด 3 มิติ X โดยเราสนใจเฉพาะคำตอบประเภท Non-Trivial หรือกรณี det(A) = 0

3.6 การหาพิกัด 3 มิติของจุดในฉากโดยกล้อง 3 ตัวหรือมากกว่า

สำหรับกรณีของการใช้ 2 กล้องหรือมากกว่าเราได้สมการในลักษณะเดียวกัน เช่นถ้าใช้กล้อง 3 ตัวดังแสดงในรูปที่ 3.6 เราได้สมการ

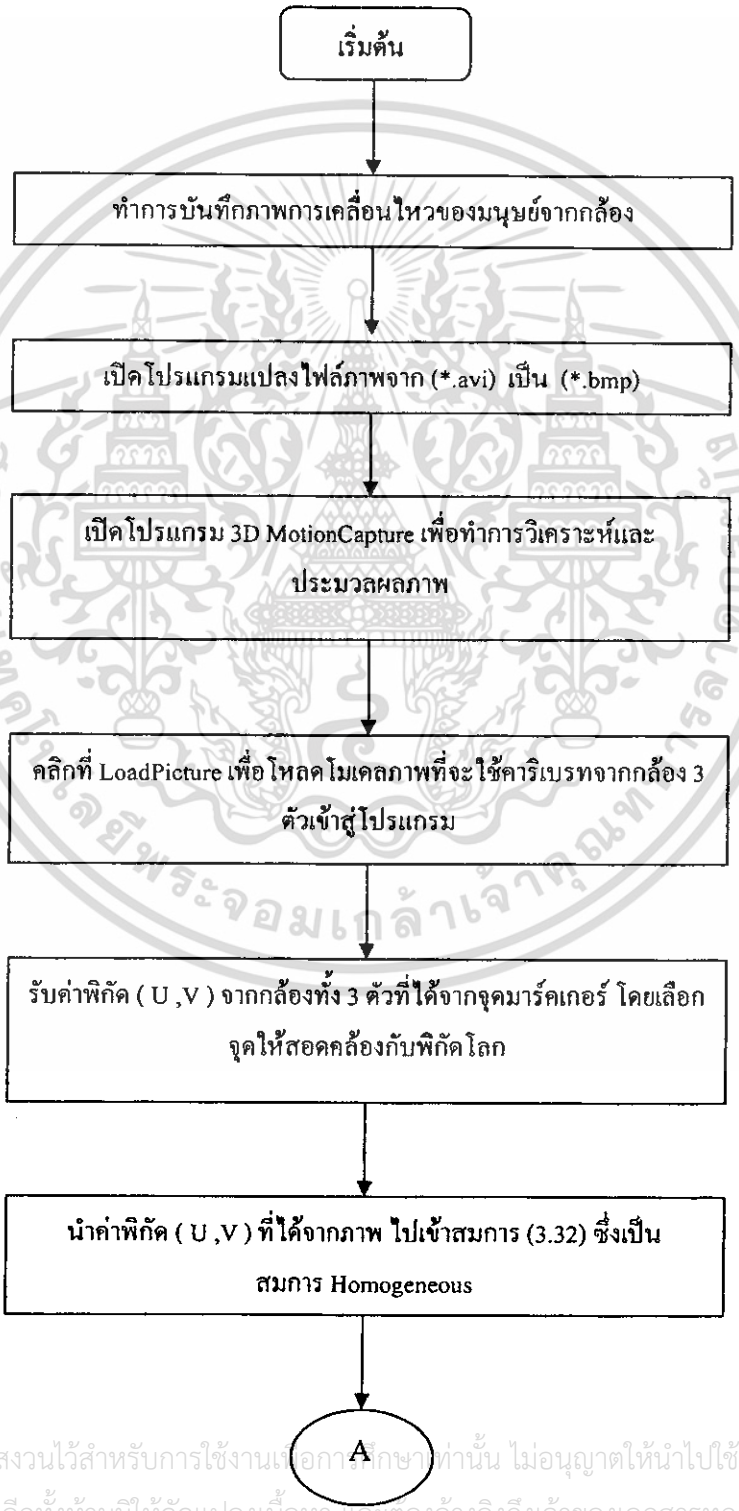


รูปที่ 3.6 การหาพิกัด 3 มิติโดยใช้กล้อง 3 ตัว

$$\begin{bmatrix} um_3^T - wm_1^T \\ vm_3^T - wm_2^T \\ u'm_3^T - w'm_1^T \\ v'm_3^T - w'm_2^T \\ u''m_3^T - w''m_1^T \\ v''m_3^T - w''m_2^T \end{bmatrix} X = AX = 0 \qquad (3.42)$$

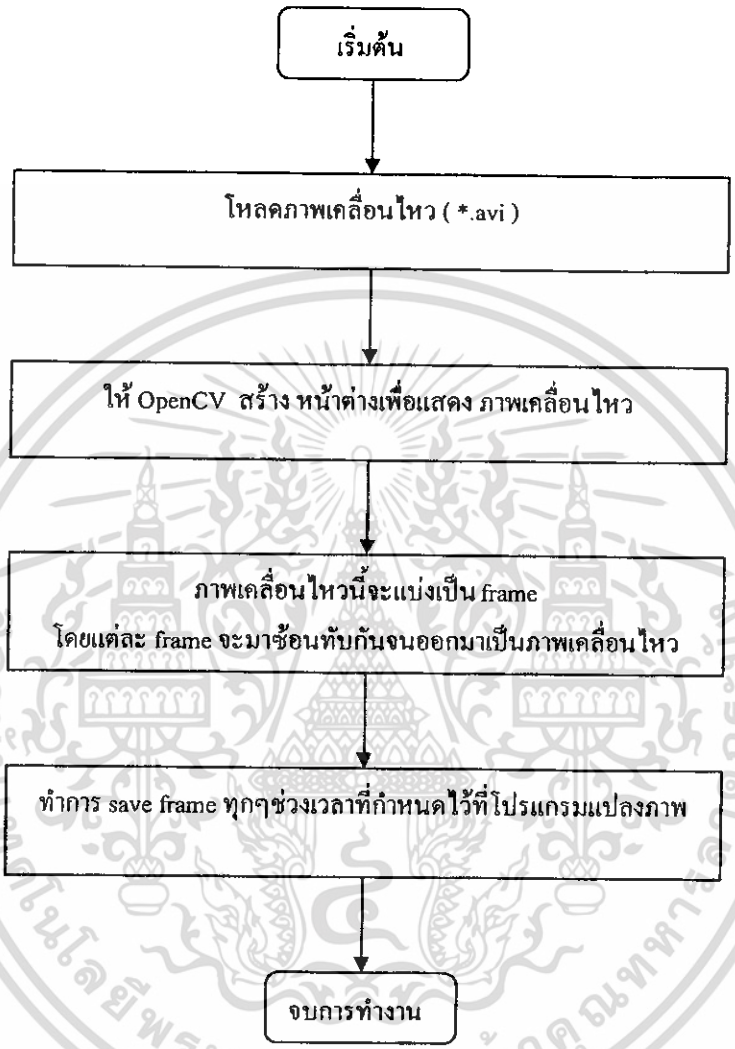
บทที่ 4
การออกแบบโปรแกรม

4.1 โครงสร้างโปรแกรม 3D COORDINATE EXTRACTION



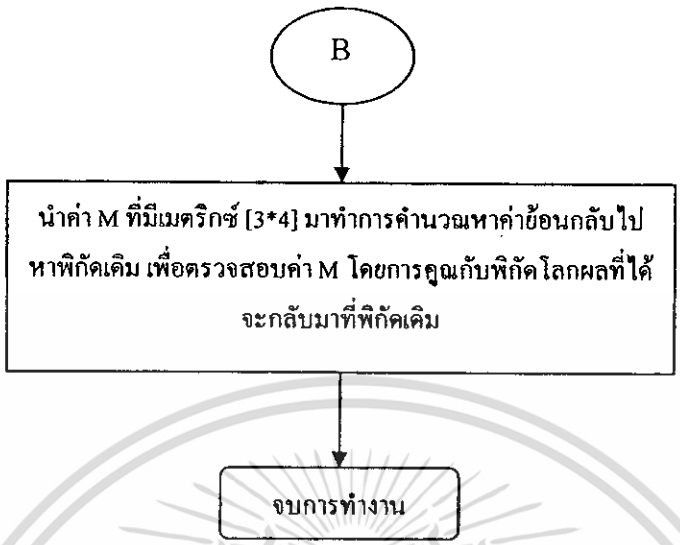
รูปที่ 4.1 Flow Chart โครงสร้างการทำงานของระบบโดยรวม

4.2 โครงสร้างโปรแกรมการแปลงภาพจาก (*.avi) เป็น (*.bmp)



รูปที่ 4.2 Flow Chart แสดงโครงสร้างโปรแกรมการแปลงภาพจาก (*.avi) เป็น (*.bmp)

4.3 โครงสร้างโปรแกรมการปรับเทียบกล้อง (Calibration)



รูปที่ 4.3 Flow Chart แสดงโครงสร้างโปรแกรมปรับเทียบกล้อง

4.4 ขั้นตอนกระบวนการปรับเทียบกล้อง Calibration

1. แปลงไฟล์ภาพของวัตถุที่จะทำการคาริเบรทซึ่งเป็นโครงสี่เหลี่ยมลูกบาศก์ 2 ชั้น จากไฟล์ (*.avi) เป็น (*.bmp)
2. โหลดไฟล์ภาพเข้าสู่โปรแกรมคาริเบรท
3. ทำการคลิกที่จุดมาร์คเกอร์(จุดวงกลมสีเหลือง) เพื่อนำค่าพิกัดภาพ (U,V) ที่ได้จากจุดที่คลิก และพิกัดโลก (X,Y,Z) มาเข้าสมการ (3.32) ซึ่งขนาดของเมทริกซ์จะเป็นไปตามสมการ 2n*12 (n = จำนวนจุดที่คลิก) จากการทดลองจะใช้ทั้งหมด 9 จุด จะได้สมการเมทริกซ์ขนาด [18 × 12]
4. เมทริกซ์ที่ได้จะถูกนำไปแยกเพื่อหาค่าพารามิเตอร์ M โดย ใช้วิธีการ SVD (Singular Value Decomposition)
5. ผลจากการแยกเมทริกซ์โดยใช้วิธี SVD จะถูกกระจายได้เป็น 3 เมทริกซ์ U , W ,V^T
6. ค่าการคาริเบรชั่น(M) จะอยู่ที่คอลัมน์สุดท้ายซึ่งคือ ค่า V^T เขียนอยู่ในรูปเมทริกซ์ได้ขนาด 3×4

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \end{bmatrix}$$

7. นำเมทริกซ์ M ทางด้านซ้ายมือขนาด 3×3 มากระจายออกด้วยวิธี QR Decomposition เพื่อทำการแยกพารามิเตอร์ภายในซึ่งจะได้ค่าเมทริกซ์ของพารามิเตอร์ Q คือ Q*QT = I ซึ่งเป็นเมทริกซ์ออโธกอนอล และ พารามิเตอร์ ที่จะนำไปใช้ในการหาพิกัด 3 มิติ

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

$$R = \begin{bmatrix} fc(1) & alpha\_c * fc(1) & cc(1) \\ 0 & fc(2) & cc(2) \\ 0 & 0 & 1 \end{bmatrix}$$

ตัวแปรภายในกล้อง(Intrinsic camera parameters)

1) Focus Length [fc(1) และ fc(2)] คือความยาวโฟกัสในหน่วยพิกเซล
2) Principal point [cc(1) และ cc(2)] คือค่าพิกัดจุดกึ่งกลางของภาพ
3) Distortion [alpha_c] คือ ค่าสัมประสิทธิ์ความโค้งของมุมระหว่างแกนพิกเซล x และ y ที่เก็บไว้ในรูปขนาด alpha_c
4) Distirtion [kc] คือใช้แบบจำลองทางความเพี้ยนในอัตราส่วนของกล้อง

ตัวแปรภายนอกกล้อง (Extrinsic camera parameters)

อธิบายเกี่ยวกับตำแหน่งและทิศทางของระบบกล้องในพิกัด 3 มิติ ของโลก รวมไปถึง

การเลื่อน (Translation)

$$t = [t_x, t_{yx}, t_z]^T$$

การหมุน (Rotation)

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

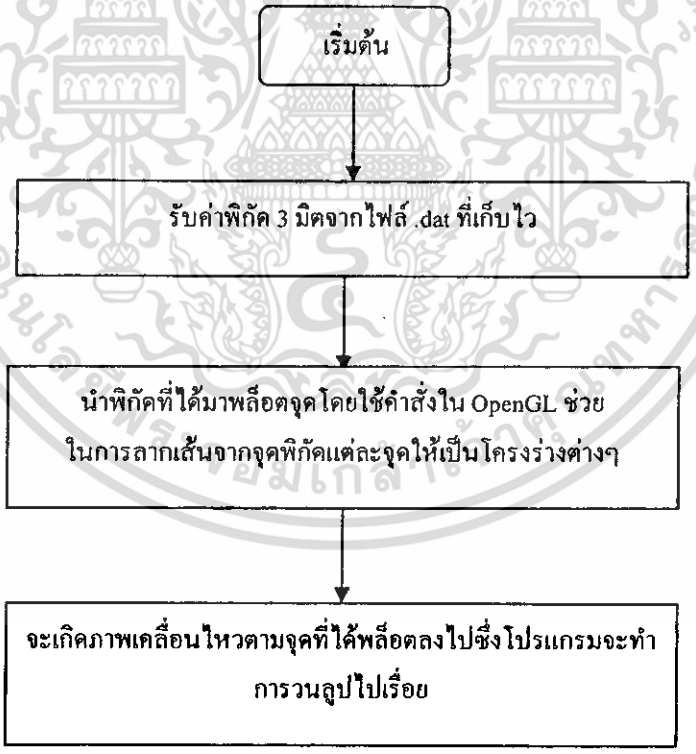
8. หาค่าพารามิเตอร์ระยะเคลื่อนที่ (T) ได้โดย $T = -A^{-1}B$ ซึ่งค่า T จะบอกถึงตำแหน่งกึ่งกลางของ Image Plane

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \end{bmatrix}$$

A B

9. ทำแบบเดียวกันจนครบ 3 กล้อง

4.5 ขั้นตอนการสร้างภาพการเคลื่อนไหว 3 มิติ



รูปที่ 4.4 Flow Chart แสดงโครงสร้างการสร้างภาพ3มิติ

บทที่ 5

การทดลองและผลการทดลอง

5.1 โปรแกรมการปรับเทียบกล้อง

ใช้ในการวิเคราะห์และประมวลภาพ 3 มิติของวัตถุในส่วนของการวิเคราะห์ภาพวัตถุที่ทำการเคลื่อนไหวจะทำการบันทึกภาพโดยใช้กล้องทั้งหมด 3 กล้อง วางในตำแหน่งที่เห็นทุกจุดพิกัดของวัตถุ สำหรับแต่ละกล้องจะใช้สมการ Calibration (3.32) ในการวิเคราะห์ภาพถ่ายของวัตถุ ซึ่งเราจะได้ค่าพารามิเตอร์ภายใน และ ภายนอกเพื่อนำมาพิจารณาการปรับเทียบโดยใช้กล้องหลายตัว

โดยค่าที่แสดงในโปรแกรมนี้ประกอบด้วย ค่า M ซึ่งเป็นการคำนวณ projection matrix โดยค่า M นี้สามารถแยกออกได้เป็นค่า K ซึ่งเป็นค่าที่แสดงพารามิเตอร์ที่อยู่ภายในกล้องแต่ละตัว และ ค่า R ซึ่งแสดงข้อมูลในการหมุนของวัตถุ ส่วนค่าที่แสดงอีกค่าคือค่า T ซึ่งจะบอกข้อมูลในการเลื่อนในแนวแกน X ,Y และ Z

ในการบอกพิกัดตำแหน่งของวัตถุ จะใช้ลูกวงกลมขาว จำนวน 12 ลูก ติดไว้ที่มุมของวัตถุที่ใช้ในการปรับเทียบกล้อง

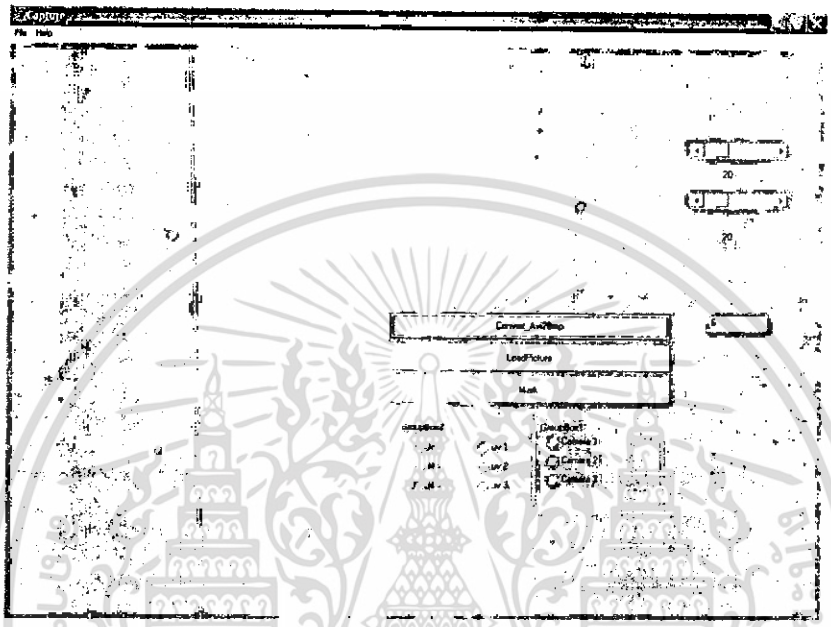
5.1.1 วัตถุคาริเบรท

รูปที่ 5.1 แสดงโมเดลโครงสี่เหลี่ยมลูกบาศก์ 2 ชั้น

ขั้นตอนการทดลอง

1) เปิดโปรแกรม Capture เพื่อแปลงภาพเคลื่อนไหว (*.avi) เป็น ไฟล์รูปภาพ (*.bmp) โดยโปรแกรม แปลงภาพนี้จะใช้ โปรแกรมช่วยคือ OpenCV เป็นไลบรารีสำหรับใช้งานด้านการประมวลผลภาพ (image processing) โดย OpenCV จะนำภาพเคลื่อนไหวมาทำเป็น frame แล้ว

frame มาโชว์ซ้อนกันไปเรื่อยๆๆจนมองเหมือนเป็นภาพเคลื่อนไหว ในการ Convert ภาพนั้นเราจะใช้ Timer ในการจับเวลาให้ทำการsave ภาพทุกๆๆกี่มิลลิวินาทีแล้วแต่เวลาที่เรากำหนดไว้บนโปรแกรม

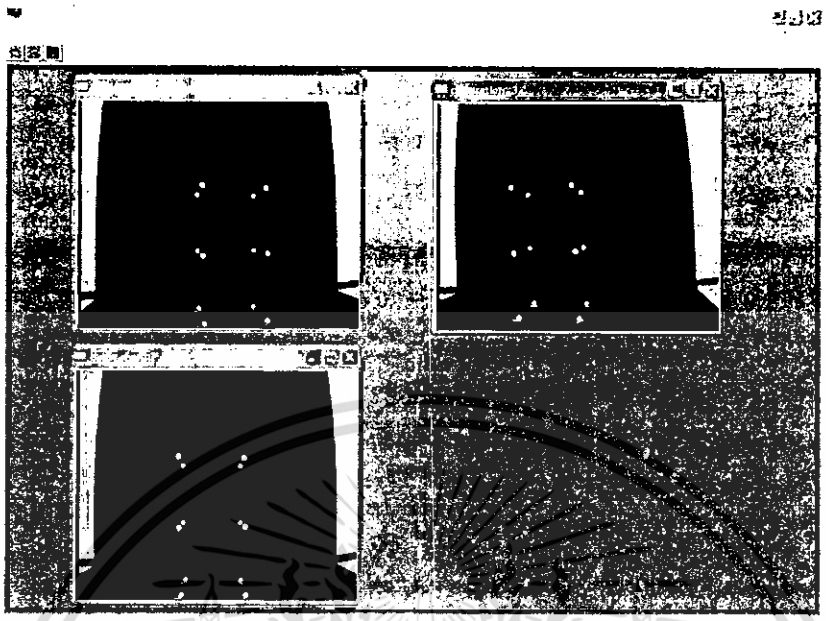


รูปที่ 5.2 แสดงโปรแกรมการแปลงไฟล์ (*.avi) เป็น (*.bmp)

2) โหลดไฟล์ (*.avi) ที่ต้องการแปลง ให้ครบทั้ง 3 กล้อง แล้วคลิกที่ปุ่ม Convert_avi2bmp

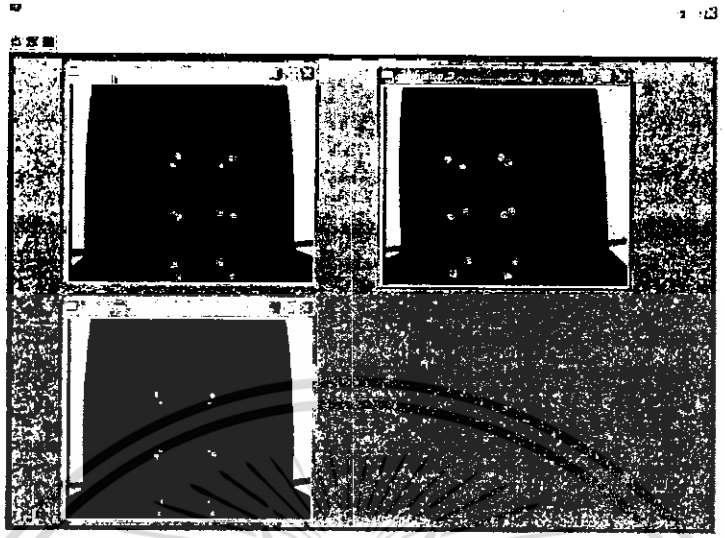
3) ทำการปรับเทียบกล้อง (Calibration)

การปรับเทียบกล้องเป็นกระบวนการในการใช้กล้อง 3 ตัวในการ Capture ภาพนิ่งโดยอ้างอิงวัตถุในพิกัดโลกจุดที่อ้างอิงจะอ้างอิงเป็นกรณีพิเศษคือให้มุมใดมุมหนึ่ง เป็นจุดกำเนิด จากนั้น ทำการตรวจสอบโดยการคำนวณ projection matrix M และทำการ reprojection (การคำนวณย้อนกลับไปหาภาพเดิม)สมการ projection matrix M (3.32) เพื่อทำการคำนวณจุดอ้างอิงของภาพและแสดงผลพิกัดซ้อนทับของภาพที่ทำการ Capture



รูปที่ 5.3 แสดงโปรแกรมการปรับเทียบกล้อง

4) คลิกที่ปุ่ม File -> Open โปรแกรมจะทำการโหลดไฟล์ภาพ (*.bmp) ที่ได้บันทึกไว้จากทั้ง 3 มุมกล้องใช้เมาท์คลิกจุดมาร์คเกอร์ของโมเดล โดยคลิกให้อยู่ตรงกลางจุดเพื่อจะนำเอาพิกัดภาพ (U ,V) ไปคำนวณในสมการ 3.32
5) เมื่อทำการคลิกที่จุดมาร์คเกอร์ทุก ตำแหน่งแล้ว โปรแกรมที่ได้ออกแบบไว้จะทำการประมวลผลซึ่งจะคำนวณค่าพารามิเตอร์ต่างๆ แล้วจะทำการคำนวณย้อนกลับ เพื่อหาพิกัดภาพ ซึ่งจากผลการทดลองเห็นว่าค่าที่ได้ (รูปวงกลม) ทับกับตำแหน่งพิกัดภาพเดิมที่ได้กำหนดไว้ ตอนแรกครบทุกจุดของมาร์คเกอร์พอดี แสดงให้เห็นว่าการคาริเบรทถูกต้อง แสดงได้ดังรูป 5.4
6) ส่วนภาพที่ได้จากกล้องอีก 2 ภาพ ก็จะทำลักษณะเดียวกันดังที่ได้กล่าวมาข้างต้น
7) เนื่องจากตัวโปรแกรมจะเป็นการคาริเบรทโดยการคลิกที่รูปที่โหลดมาเพียงครั้งเดียวหลังจากนั้นโปรแกรมจะทำการ autolandmark เพื่อตรวจจับหาจุดมาร์คเกอร์ของรูปถัดไป

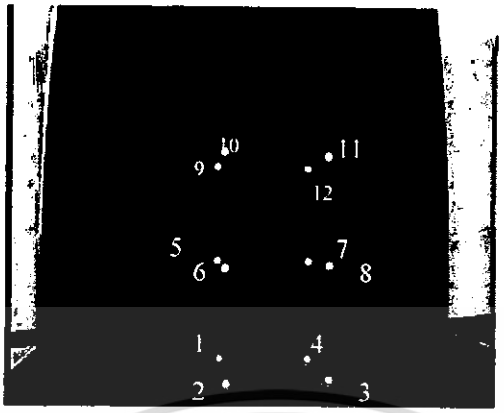


(a)

(b)

รูปที่ 5.4 ผลการ Calibration โดยใช้ model ซึ่งเป็นโครงสี่เหลี่ยมลูกบาศก์ 2 ชั้น
(a) กำหนดพิกัดถูกต้อง (b) กำหนดพิกัดไม่ถูกต้อง

- ผลการทดลองการ Calibration โดยกำหนดจุดพิกัดตามจุดที่กำหนด



รูปที่ 5.5 แสดงตำแหน่งจุดในคลิกเมาส์เพื่อระบุพิกัดภาพของวัตถุ

กล้องที่	M				K			R			T		
1	-0.019	0	0	0.545	-0.019	0	0	-0.998	0.064	0	13.313	91.917	-25.558
	0	0.001	0.017	-0.838	0	0.017	0	0.064	0.998	0			
	0	0	0	0.009	0	0	0	0	0	1.000			
2	-0.017	0.016	0	0.02	0.024	0	0	0.998	0.067	0	48.354	49.989	-26.514
	0.001	0.001	0.021	-0.999	0	0.021	0	-0.067	0.998	0			
	0	0	0	0.01	0	0	0	0	0	1.000			
3	-0.01	0.021	0	-0.173	0.024	0	0	1.000	0.008	0	60.224	35.004	-28.790
	0.002	0.001	0.021	-0.984	0	0.021	0	0.008	-1.000	0			
	0	0	0	0.010	0	0	0	0	0	1.000			

ตารางที่ 5.1 ตารางแสดงค่าพารามิเตอร์ภายในและพารามิเตอร์ภายนอกของกล้องทั้ง 3 ตัว กรณี Calibartion ที่ถูกต้อง

- ผลการทดลองการ Error Calibartion โดยกำหนดจุดพิกัดไม่ตรงตามที่กำหนด

กล้องที่	M				K			R			T		
1	-0.003	-0.003	-0.006	0.602	0.028	0	0	-0.145	-0.989	-0.006	154.982	-256.370	92.186
	0.023	0.015	0	-0.798	0	0.006	0	0.989	-0.146	0.003			
	-0	-0	-0	0.010	0	0	0	-0.004	-0.006	1.000			
2	-0	-0	-0.013	0.655	0.035	0	0	-0.010	1.000	0	138.337	-175.340	96.283
	0.031	0.015	0	-0.754	0	0.013	0	1.000	0.010	0			
	0	0	-0	0.009	0	0	0	1.000	0	-1.000			
3	0.001	0.001	0.012	-0.580	0.035	0	0	-0.036	0.999	-0.002	152.762	-315.05	34.317
	-0.031	-0.016	-0	0.813	0	0.012	0	0.999	0.036	-0.001			
	0	0	-0	-0.008	0	0	0	-0.001	-0.002	-1.000			

ตารางที่ 5.2 ตารางแสดงค่าพารามิเตอร์ภายในและพารามิเตอร์ภายนอกของกล้องทั้ง 3 ตัว กรณี Calibration ไม่ถูกต้อง

5.2 การหาพิกัด 3 มิติของมนุษย์

ในการทดลองนี้เป็นการทดสอบโปรแกรมการหาพิกัด 3 มิติในการหาค่าพิกัดแกน(X,Y,Z) ของมนุษย์ ซึ่งจะเป็นการหาพิกัดจุด 3 มิติของวัตถุที่มีการเคลื่อนไหว

ขั้นตอนการทดลอง

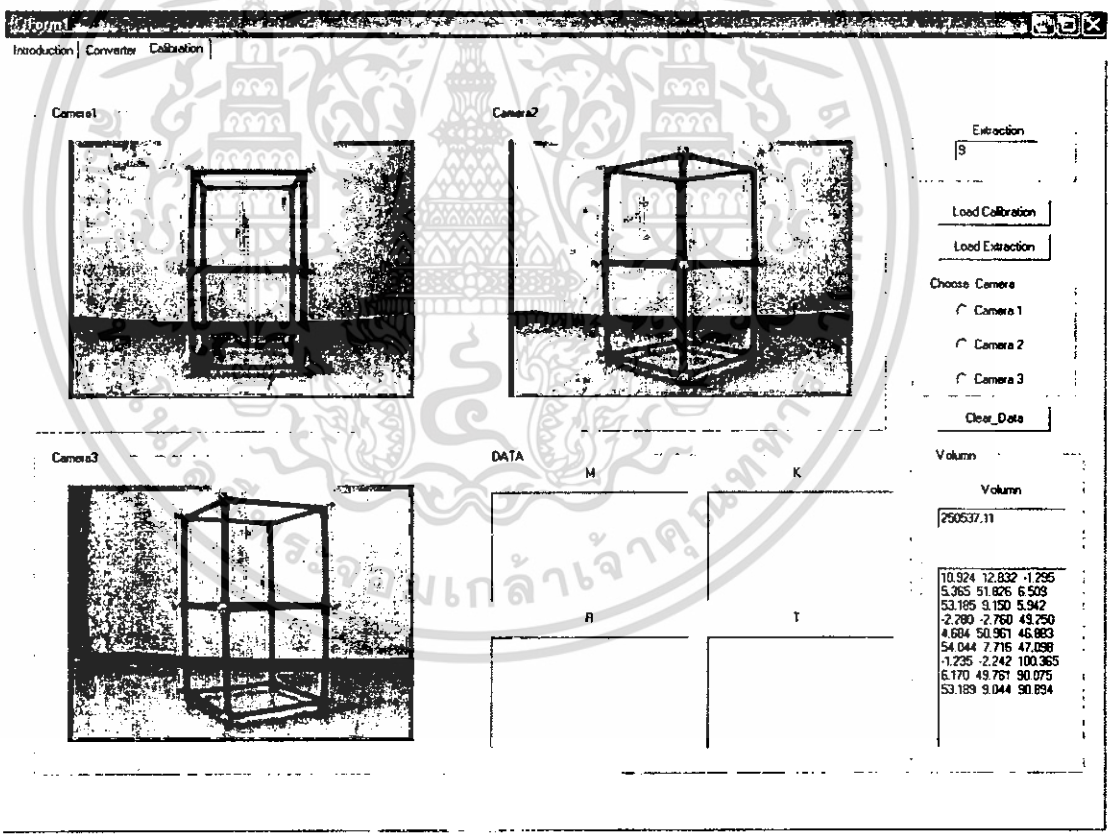
1) หลังจากทำการ Capture ภาพ model แล้วอย่าพึ่งขยับกล้อง จากนั้นให้เปลี่ยนวัตถุจาก model เป็นวัตถุที่ต้องการหาพิกัด 3 มิติและทำการ Capture อีกครั้ง
2) ทำการแปลงไฟล์ที่ทำการ Capture ภาพจาก (*.avi) เป็น (*.bmp) และ save ไฟล์ลงที่ไดร์ฟ ที่ต้องการโดยจะเลือกเก็บที่ไดร์ฟ C

3) ทำการคลิกปุ่ม LoadPicture จะปรากฏรูปวัตถุที่ทำการแคปเจอร์
4) จากนั้นทำการใช้เมาส์คลิก ตรงตำแหน่งต่างๆของจุดมาร์คเกอร์ให้ครบทุกจุดแล้วกดปุ่ม Mark
5) วนกลับไปทำในข้อ 4 ใหม่จนครบทั้ง 3 กล้อง
6) เนื่องจากตัวโปรแกรมจะเป็นการหาพิกัดภาพโดยจะทำการคลิกที่รูปซึ่งโหลดมาเพียงครั้งเดียว ดังนั้นหลังจากคลิกที่จุดมาร์คเกอร์จากทุกกล้องเสร็จแล้วโปรแกรมจะทำการ autolandmark เพื่อตรวจจับหาจุดสีที่เป็นมาร์คเกอร์แต่ละจุดสีของรูปถัดไปจนกว่าจะครบ

การทดลอง

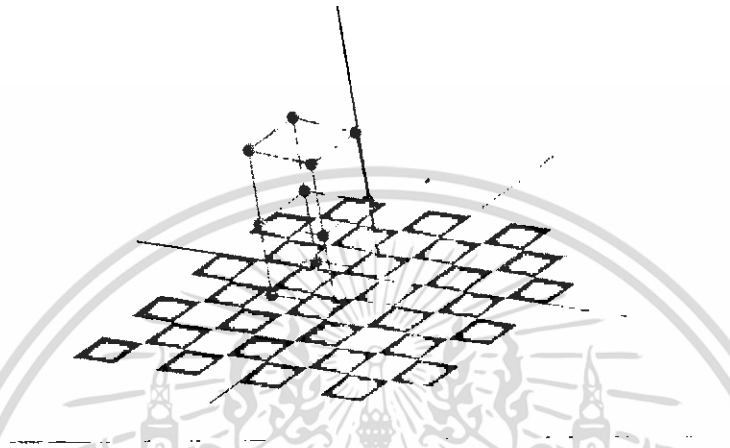
การหาพิกัดของวัตถุที่ไม่เคลื่อนไหว

5.2.1 วัตถุทรงเรขาคณิตแบบที่ 1



รูปที่ 5.6 แสดงภาพการคลิกตำแหน่งพิกัดภาพของวัตถุที่ 1 ด้วยเมาส์ทั้ง 3 มุมกล้องและแสดงค่าพิกัดจุด (X,Y,Z) ของจุดทั้ง 12 จุดของวัตถุแบบที่ 1

- นำค่าพิกัดที่ได้ไปพล็อตในโปรแกรม OpenGL จะได้ภาพ 3 มิติ ดังแสดงตามรูปที่ 5.7



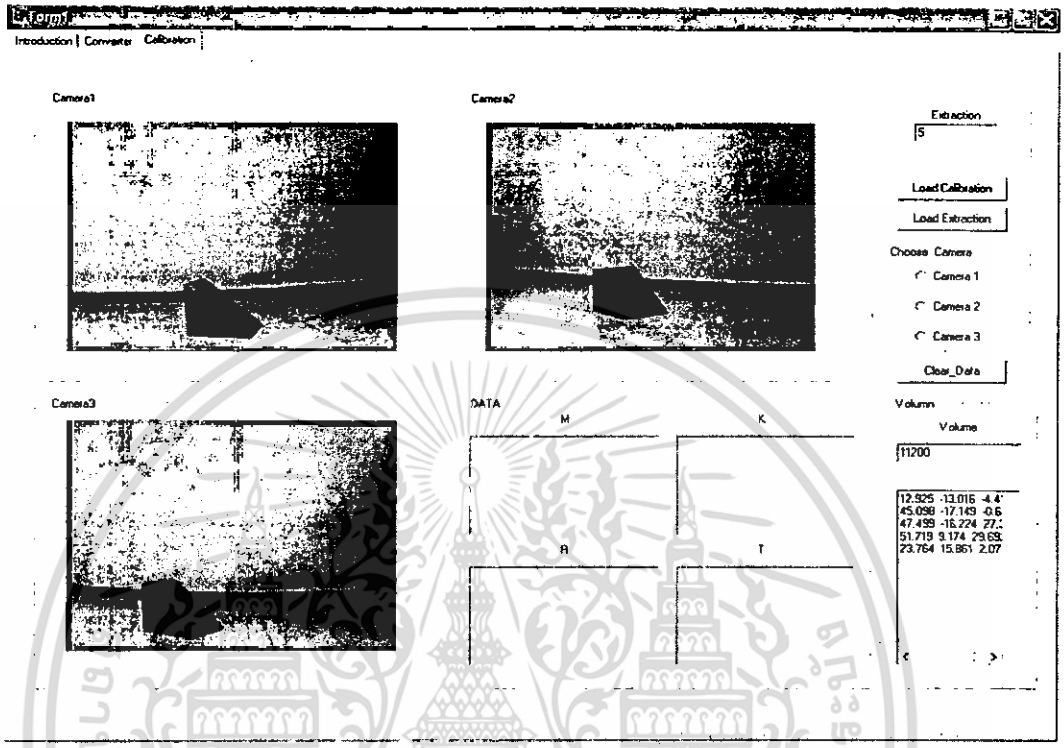
รูปที่ 5.7 แสดงภาพ 3 มิติ ของโครงลูกบาศก์ 2 ชั้น

- ผลการทดลองประสิทธิภาพการหาปริมาตรของวัตถุแบบที่ 1

	ค่าจริง	ค่าจากการทดลอง	%ความผิดพลาด
ความกว้าง(X)	48.00 cm	50.23 cm	4.64
ความยาว(Y)	52.00 cm	51.42 cm	1.12
ความสูง(Z)	96.00 cm	88.53 cm	7.78

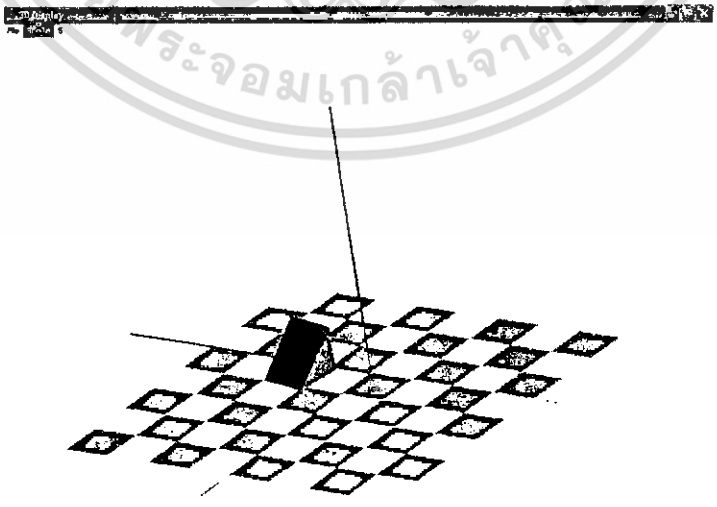
ตารางที่ 5.3 ตารางแสดงประสิทธิภาพการหาปริมาตรของวัตถุแบบที่ 1

5.2.2 วัตถุทรงเรขาคณิตแบบที่ 2



รูปที่ 5.8 แสดงค่าพิกัดจุด(X,Y,Z) ของจุดทั้ง 5 จุดของวัตถุแบบที่ 2

- นำค่าพิกัดที่ได้ไปพล็อตในโปรแกรม OpenGL ดังแสดงตามรูปที่ 5.18



รูปที่ 5.9 แสดงภาพ 3 มิติ ของกล่องสามเหลี่ยม

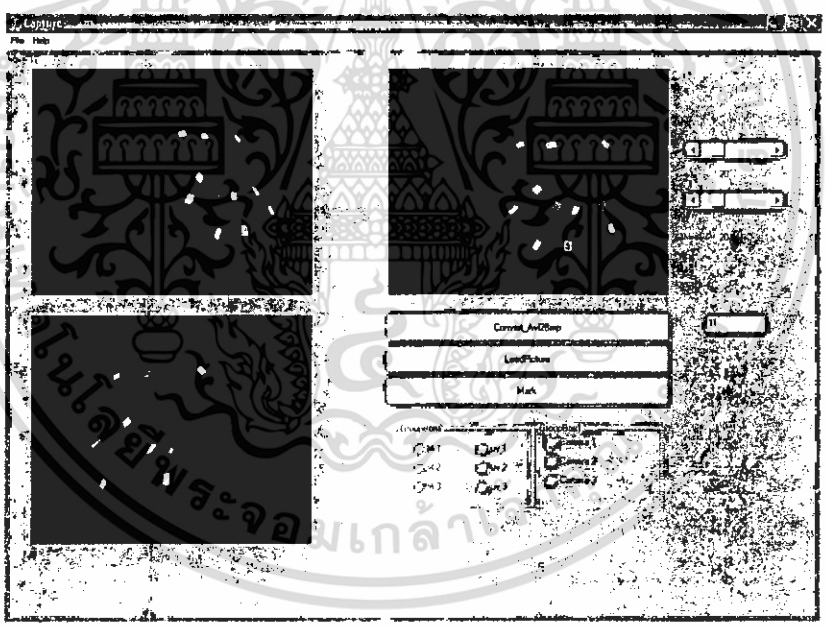
- ผลการทดลองประสิทธิภาพหาปริมาตรของวัตถุแบบที่ 2

	ค่าจริง	ค่าจากการทดลอง	%ความผิดพลาด
ความกว้าง(X)	30.00 cm	32.17 cm	7.23
ความยาว(Y)	30.00 cm	28.40 cm	5.33
ความสูง(Z)	30.00 cm	28.06 cm	6.46

ตารางที่ 5.4 ตารางแสดงประสิทธิภาพการหาปริมาตรของวัตถุแบบที่2

การหาพิกัดการเคลื่อนไหวของร่างกาย

5.2.3 การขยับนิ้วมือ



รูปที่ 5.10 การหาพิกัดภาพโดยใช้มือ

จากตารางเป็นตัวอย่างการรับค่าพิกัด U ,V ในแต่ละกล้องแล้วใช้โปรแกรมคำนวณเพื่อเปลี่ยนจากค่าพิกัด 2 มิติ เป็น พิกัด 3 มิติ

ค่า UV1 = ค่าพิกัด 2 มิติที่ได้จากภาพของกล้องที่ 1 ค่า UV2 = ค่าพิกัด 2 มิติที่ได้จากภาพของกล้องที่ 2

ค่า UV3 = ค่าพิกัด 2 มิติที่ได้จากภาพของกล้องที่ 3 โดยที่ u = พิกัดแกน X และ v = พิกัดแกน Y

UV1		UV2		UV3		XYZ		
U	V	U	V	U	V	X	Y	Z
189	82	166	98	108	76	11.90	10.85	14.87
217	86	206	98	144	75	16.55	11.68	14.89
197	166	158	180	81	162	9.98	14.77	4.53
211	139	188	154	121	135	14.15	12.88	7.83
234	209	187	226	91	212	12.51	18.33	0.21
234	158	211	176	134	160	16.25	15.09	5.37
268	205	228	227	116	222	16.15	20.65	0.06
255	163	236	180	152	167	18.56	16.79	4.94
301	181	281	204	171	208	21.43	21.90	2.65
281	156	270	174	179	161	21.55	18.84	5.87
259	90	273	97	213	68	24.03	13.39	15.10

ตารางที่ 5.5 ตารางแสดงค่าพิกัดของนิ้วมือ

5.2.4 ลักษณะการเดิน

รูปที่ 5.11 การหาพิกัดภาพโดยใช้ขา

จากตารางเป็นค่าพิกัด 3 มิติของขา

ค่า UV1 = ค่าพิกัด 2 มิติที่ได้จากภาพของกล้องที่ 1 ค่า UV2 = ค่าพิกัด 2 มิติที่ได้จากภาพของกล้องที่ 2

ค่า UV3 = ค่าพิกัด 2 มิติที่ได้จากภาพของกล้องที่ 3 โดยที่ u = พิกัดแกน X และ v = พิกัดแกน Y

UV1		UV2		UV3		XYZ		
U	V	U	V	U	V	X	Y	Z
216	40	134	34	118	61	2.30	-0.56	14.28
267	43	195	40	168	56	8.04	-0.74	14.33
218	107	137	105	130	122	2.90	-0.85	8.20
255	101	180	105	165	114	7.15	-0.94	8.73
213	169	134	177	140	180	2.98	-2.09	1.40
251	168	175	177	171	179	7.23	-1.09	1.89

ตารางที่ 5.6 ตารางแสดงค่าพิกัดของขา

5.2.5 การเคลื่อนไหวร่างกาย

รูปที่ 5.12 การหาพิกัดภาพโดยใช้คน

จากตารางเป็นค่าพิกัด 3 มิติของคน

ค่า UV1 = ค่าพิกัด 2 มิติที่ได้จากภาพของกล้องที่ 1 ค่า UV2 = ค่าพิกัด 2 มิติที่ได้จากภาพของกล้องที่ 2

ค่า UV3 = ค่าพิกัด 2 มิติที่ได้จากภาพของกล้องที่ 3 โดยที่ u = พิกัดแกน X และ v = พิกัดแกน Y

UV1		UV2		UV3		XYZ		
U	V	U	V	U	V	X	Y	Z
177	22	144	23	105	23	2.20	5.88	24.10
141	59	114	62	82	65	-0.34	3.42	20.73
128	88	104	90	73	95	-1.27	3.15	17.83
104	136	85	138	57	142	-3.02	2.59	12.93
206	53	179	48	146	48	4.86	2.83	21.62
214	99	187	93	153	93	5.52	3.99	16.87
233	141	210	132	179	136	7.36	3.15	12.55
180	106	147	102	108	105	2.51	7.19	15.95
147	165	120	161	81	166	0.20	7.15	10.10
142	205	120	199	84	204	0.00	5.54	6.02
129	241	113	235	81	238	-0.84	3.42	2.03
192	168	164	161	125	168	3.70	6.77	9.74
193	204	170	197	134	205	4.08	5.02	5.89
195	244	179	237	146	247	4.66	2.64	1.27

ตารางที่ 5.7 ตารางแสดงค่าพิกัดของคน

5.3 การสร้างภาพการเคลื่อนไหวแบบ 3 มิติ

เมื่อหาค่าพิกัดภาพ 3 มิติได้แล้วก็จะนำค่าพิกัดที่ได้นั้นมาทำการวาดเป็นรูปภาพการเคลื่อนไหว แบบ 3 มิติโดยจะใช้โปรแกรม OpenGL ช่วยในการวาดโดยจะใช้คำสั่ง gluSphere() ช่วยในการพล็อตจุด และใช้คำสั่ง glBegin (GL_LINES) , glVertex3f () ที่ช่วยในการลากเส้นเชื่อมเพื่อให้ดูเป็นโครงร่าง 3 มิติของทุกๆเฟรม

ขั้นตอนการทดลอง

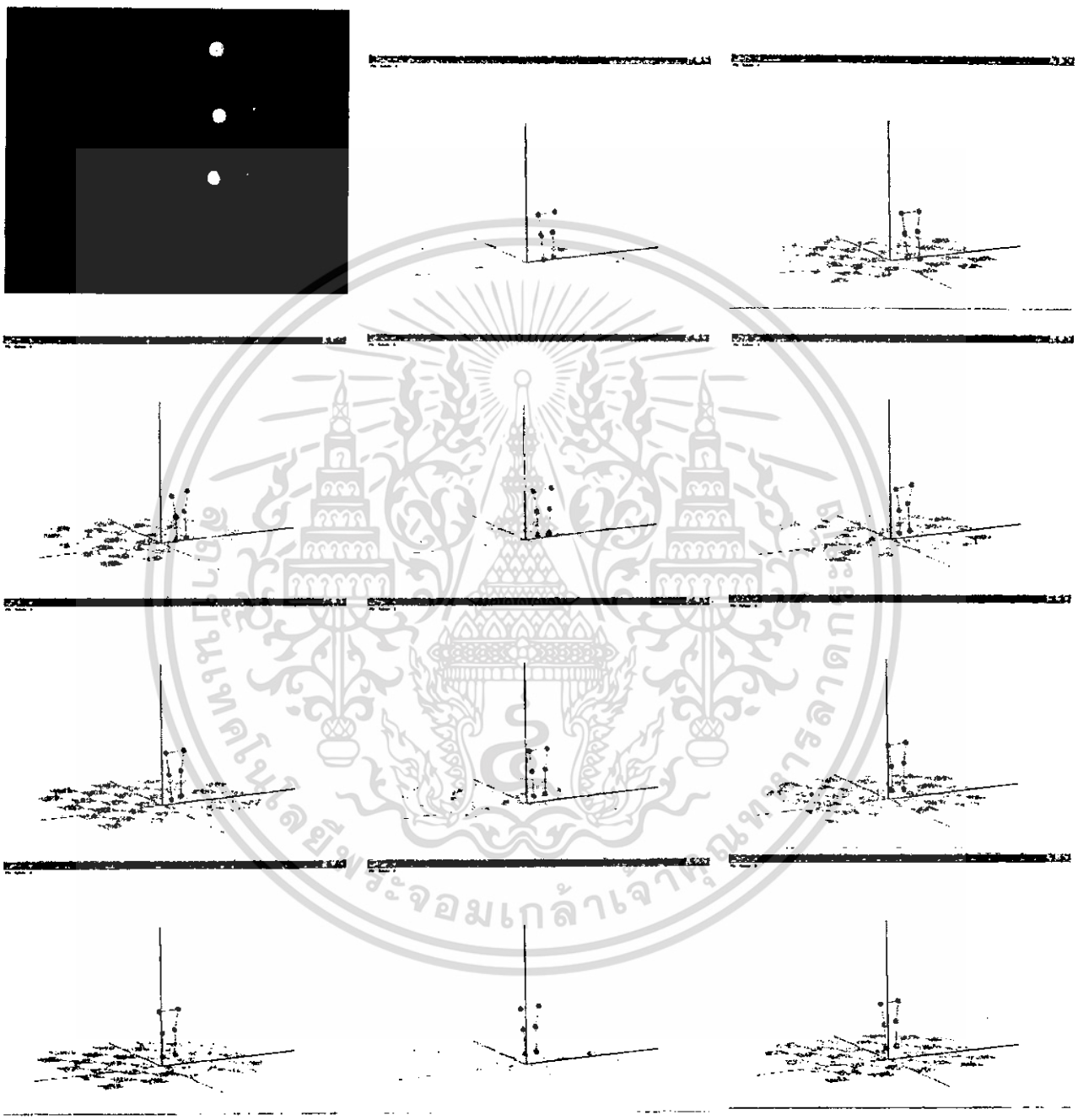
1) กดปุ่ม file->RUN ซึ่งภาพที่ได้จะแสดงอยู่ในรูป 3 มิติโดยสามารถที่จะดูภาพได้ทุกมุมเมื่อกดเมาส์ค้างแล้วหมุนรูปไปยังมุมที่ต้องการ

การทดลอง

5.3.1 การขยับนิ้วมือ

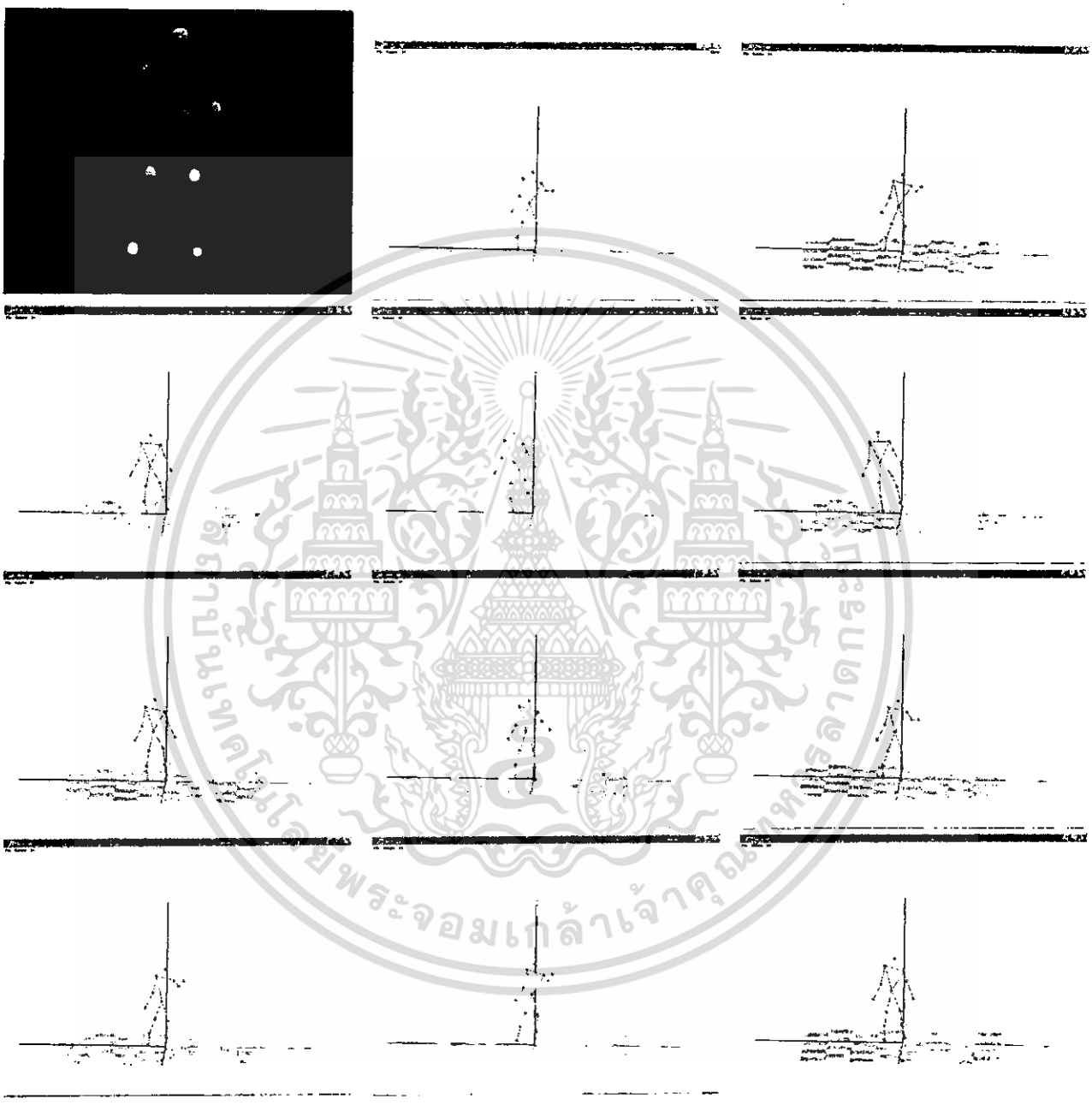
รูปที่ 5.13 แสดงการเคลื่อนไหวของนิ้วมือ

5.3.2 ลักษณะการเดิน



รูปที่ 5.14 แสดงการเคลื่อนไหวของขา

5.3.3 การเคลื่อนไหวร่างกาย



รูปที่ 5.15 แสดงการเคลื่อนไหวของมนุษย์

บทที่ 6
สรุปและวิจารณ์

6.1 วิเคราะห์และสรุปผลที่ได้จากการทดลอง

1) จากการทดลองการแปลงภาพจากภาพเคลื่อนไหวให้เป็นรูปภาพ

การทดลองนี้เราได้ใช้ โปรแกรม OpenCV ซึ่งเป็นไลบรารีที่ใช้ในงานประมวลผลทางภาพ โดยโปรแกรมนี้จะช่วยให้เราสามารถแปลงภาพเคลื่อนไหวให้เป็นรูปภาพได้โดยที่โปรแกรม OpenCV นี้จะทำการสร้างหน้าต่างขึ้นมาอีกหน้าต่างหนึ่งเพื่อทำการแสดงภาพเคลื่อนไหว ซึ่งการที่แสดงเป็นภาพเคลื่อนไหวนั้นเกิดมาจากการนำแต่ละ Frame รูปภาพมาวางต่อกันเรื่อยๆ จนเรามองเห็นเป็นภาพเคลื่อนไหว ซึ่งความเร็วในการแปลงภาพขึ้นอยู่กับเวลาที่เรากำหนดไว้เป็นหน่วยมิลลิวินาที

2) จากการทดลอง Calibration

การทดลองนี้เป็นการหาค่าพารามิเตอร์ภายในและภายนอกของกล้องแต่ตัว โดยเริ่มจากการหาค่าของ Perspective Transformation M ด้วยการนำค่าพิกัดภาพและพิกัดโลกมาเข้าสมการที่ 3.32 แล้วใช้ฟังก์ชั่น SVD แยกค่าออกมาก็จะได้ค่าของ Perspective Transformation M มีทั้งหมด 12 ค่า ซึ่งค่า M ที่ได้มานี้สามารถตรวจสอบได้โดยการคำนวณย้อนกลับ ให้เมทริกซ์พิกัดโลกคูณเมทริกซ์ M ผลที่ได้จะได้พิกัดภาพ เมื่อนำค่าพิกัดภาพที่ได้จากการคำนวณไปพล็อตลงในภาพจะเห็นว่าตำแหน่งที่พล็อตลงไปมีค่าใกล้เคียงกับตำแหน่งที่คลิกเพื่อหาพิกัดภาพ แสดงว่าค่า M ที่ได้ถูกต้อง เมื่อได้ค่า M มาแล้วสามารถนำมาแยกได้เป็น ค่า K (Calibration Matrix) , ค่า R (Rotation) และค่า T(Translatin) โดยค่า K เป็นเมทริกซ์ที่บอกค่าพารามิเตอร์ภายในของกล้องแต่ละตัว ส่วนค่า R เป็นเมทริกซ์ที่เป็นผลรวมของการหมุนของระบบพิกัด ส่วนค่า T เป็นผลรวมของการย้ายของระบบพิกัด

3) การทดลองการหาค่าพิกัด 3 มิติ

การทดลองนี้เราใช้กล้อง 3 ตัว โดยกล้องทั้ง 3 ตัวนี้ได้ผ่านกระบวนการปรับเทียบกล้องดังที่ได้กล่าวข้างต้นนี้แล้ว แล้วนำวัตถุที่ต้องการหาพิกัด 3 มิติ มาไว้ในตำแหน่งที่กล้องทั้ง 3 ตัวสามารถมองเห็นจุดมาร์คที่ติดไว้บนวัตถุได้ครบทั้ง 3 กล้อง ซึ่งจุดมาร์คที่เราใช้ติดจะต้องมีหลายๆสี เนื่องจากเราจะหาพิกัดภาพของจุดมาร์คโดยการ autolandmark แบบการปรับเทียบสีของจุดมาร์คแต่ละจุด จากนั้นทำการหาพิกัดภาพของจุดมาร์คของกล้องทั้ง 3 ตัวแล้วนำค่าพิกัดภาพที่ได้กับค่า Perspective Transformation (M) มาเข้าสมการที่ 2.42 เพื่อหาค่าพิกัด 3 มิติ

ในการทดลองเราได้ทำการหาค่าพิกัด 3 มิติ จากภาพที่มีการเคลื่อนไหวโดยเราจะหาค่าพิกัดภาพของจุดมาร์คของแต่ละภาพแล้วมาเข้าสมการหาพิกัด3 มิติ จะได้พิกัด 3 มิติ ของแต่ละภาพ เมื่อเรานำค่าพิกัด 3 มิติมาต่อกันก็สามารถมองเห็นเป็นภาพเคลื่อนไหว

ภาพเคลื่อนไหวที่ใช้ในการทดลองประกอบด้วย

- การขยับนิ้วมือ
- ลักษณะการเดิน
- การเคลื่อนไหวร่างกาย

ในการทดลองนี้เราจะหาพิกัดภาพของจุดมาร์คด้วยวิธีการปรับเทียบสี ซึ่งในบางครั้งสีที่ใช้ที่อยู่ในจุดใกล้เคียงกัน มีช่วงสีอยู่ในช่วงเดียวกัน จะทำให้การ autolandmark เกิดความผิดพลาดได้ ซึ่งจะต้องปรับช่วงสีให้เกิดความพอดี

เมื่อได้ค่าของพิกัด 3 มิติ ของแต่ละภาพแล้ว เราจึงนำค่าที่ได้มาพล็อต โดยใช้ Open GL (open graphics library) ซึ่งเป็นมาตรฐานการอินเตอร์เฟซโปรแกรมประยุกต์ (application program interface หรือ API) สำหรับการกำหนดภาพ 2 มิติ และ 3 มิติ มาช่วยในการพล็อต ซึ่งจะเห็นว่าเมื่อทำการพล็อตค่าทั้งหมดแล้วจะทำให้จุดที่พล็อตเกิดเกิดเป็นภาพเคลื่อนไหว

6.2 ปัญหาที่พบและข้อจำกัดของโปรแกรม

1) วัตถุที่ใช้เป็นจุดมาร์คแต่ละจะต้องมองเห็นด้วยกล้องที่ใช้ทั้งหมด และเมื่อถ่ายออกมาต้องไม่เห็นจุดมาร์คเล็กเกินไป เพราะจะทำให้เกิดความผิดพลาดในการ autolandmark หาพิกัดภาพ

2) วัตถุที่ใช้เป็นจุดมาร์คต้องมีสีอยู่คนละโทนสี หรือในตำแหน่งที่ติดจุดมาร์คในบริเวณที่ใกล้กันไม่ควรมีสีในโทนเดียวกัน

3) ในขณะที่ทำการถ่ายภาพเคลื่อนไหวจะต้องทำการเคลื่อนไหวอย่างช้าๆเนื่องจากกล้องที่ใช้จับภาพไม่ทันทำให้ภาพที่ได้ไม่ต่อเนื่องเกิดการกระโดดของภาพเป็นผลให้การทำ autolandmark หาจุดมาร์คไม่พบ

เอกสารอ้างอิง

1. Alan Watt (1994) , " 3D Computer Graphic " , Addision-Wesley.
2. ผศ.ดร. ชูชาติ ปิณฑวิรุจน์ , เอกสารประกอบการเรียนวิชา Digital Image Processing
3. William K. Pratt (1991) , "Digital Image Processing " , Second Edition , A Wiley - Interscience Publication

ภาคผนวก

อุปกรณ์ที่ใช้เป็นจุดมาร์คเกอร์

ค่า RGB ของจุดมาร์คเกอร์

COLOR	R	G	B
(แดง)	255	0	0
(น้ำเงิน)	0	0	255
(เขียว)	0	128	0
(ขาว)	255	255	255
(ครีม)	255	255	128
(ส้ม)	255	128	0
(เหลือง)	255	255	0

```
//******* FORM1 *********
#include <vcl.h>
#pragma hdrstop
#include <dos.h>
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include <stdio.h>
#define NRANSI
#include "nr.h"
#include "nrutil.h"
#include "nrutil.c"
#include "PYTHAG.C"
#include "qr.C"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
#define NUM 8                    // number calib marker
#define SIZEW 300.0
TForm1 *Form1;
CvMat* A;
unsigned char CountPoint=0,Num_Pic=0;
CvPoint startPointMouse = {0, 0};
CvPoint endPointMouse = {0, 0};
IplImage *img_1,*img_2,*img_3;
FILE* fp;
int nP;                          // number marker pic
float **u1, **u2,**u3;
int NumPrt;
```

```
void cvSetMouseCallbackI_1(int event, int x, int y, int flags, void* param)
{
    if( event==CV_EVENT_FLAG_LBUTTON )
    {
      Num_Pic=1;
//********************Set Coorddinate of picture*********************************
      cvmSet(A ,CountPoint , 0 , x);
      cvmSet(A ,CountPoint , 1 , y);
//*********************Marker Click*************************************************
      startPointMouse.x=x-2;
      startPointMouse.y=y-2;
      endPointMouse.x=x+2;
      endPointMouse.y=y+2;
      cvRectangle(img_1,startPointMouse,endPointMouse,CV_RGB(0, 0, 255 ),1,8,0);
      cvShowImage("calibration_1", img_1 );
//*****************************************************************************
      CountPoint++;
      if(CountPoint==8){
                  CountPoint=0;
                  Form1->PerspectiveTransfomation();
                        }
    }
}
void cvSetMouseCallbackI_2(int event, int x, int y, int flags, void* param)
{
    if( event==CV_EVENT_FLAG_LBUTTON )
    {
      Num_Pic=2;
```

```
//**********Set Coorddinate of picture**********
        cvmSet(A ,CountPoint , 0 , x);
        cvmSet(A ,CountPoint , 1 , y);
//**********************************************************************
//**********Marker Click**********
        startPointMouse.x=x-2;
        startPointMouse.y=y-2;
        endPointMouse.x=x+2;
        endPointMouse.y=y+2;
        cvRectangle(img_2,startPointMouse,endPointMouse,CV_RGB(0, 0, 255 ),1,8,0);
        cvShowImage("calibration_2", img_2 );
//**********************************************************************
        CountPoint++;
        if(CountPoint==8){
                CountPoint=0;
                Form1->PerspectiveTransfomation();
                }
    }
}
void cvSetMouseCallbackI_3(int event, int x, int y, int flags, void* param)
{
    if( event==CV_EVENT_FLAG_LBUTTON )
    {
      Num_Pic=3;
//**********Set Coorddinate of picture**********
        cvmSet(A ,CountPoint , 0 , x);
        cvmSet(A ,CountPoint , 1 , y);
```

```
//********************Marker Click*********************************************
        startPointMouse.x=x-2;
        startPointMouse.y=y-2;
        endPointMouse.x=x+2;
        endPointMouse.y=y+2;
        cvRectangle(img_3,startPointMouse,endPointMouse,CV_RGB(0, 0, 255 ),1,8,0);
        cvShowImage("calibration_3", img_3 );
//*****************************************************************************
        CountPoint++;
        if(CountPoint==8){
                        CountPoint=0;
                         Form1->PerspectiveTransfomation();
                        }
    }
}
//*****************************************************************************//
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
        A =cvCreateMat( NUM, 2, CV_32F); //coordinate of Image
        NumPrt=0;
        // build folder
        //ShellExecute( NULL, "open", NULL, NULL, "C:\\V1", SW_SHOW );
        system("md c:\\M1");
        system("md c:\\M2");
        system("md c:\\M3");
        system("md c:\\uv1");
        system("md c:\\uv2");
        system("md c:\\uv3");
```

```
        nP = 6;
        u1=matrix(0,nP,0,2);
        u2=matrix(0,nP,0,2);
        u3=matrix(0,nP,0,2);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::PerspectiveTransfomation()
{
        CvMat* P =cvCreateMat( NUM*2, 12, CV_32F);
        CvMat* M =cvCreateMat( 3, 4, CV_32F);
        CvMat* w =cvCreateMat( 1, 12, CV_32F);
        CvMat* u =cvCreateMat( NUM*2, 12, CV_32F);
        CvMat* v =cvCreateMat( 12, 12, CV_32F);
        CvMat* X =cvCreateMat( NUM, 4, CV_32F);
        CvMat* XT=cvCreateMat( 4, NUM, CV_32F);
        CvMat* reproject=cvCreateMat( 3, NUM, CV_32F);
        CvMat* ReProjection=cvCreateMat( NUM, 2, CV_32F);
        int b,k,row=0,count;
        double a[] ={   0, 0 ,0, 1,
                        10.5, 0, 0, 1,
                        10.5, 10.5, 0, 1,
                        0, 10.5, 0, 1,
                        0, 0, 10.5, 1,
                        10.5, 0, 10.5, 1,
                        10.5, 10.5, 10.5, 1,
                        0, 10.5, 10.5, 1
                    };
         CvMat XYZ;
         cvInitMatHeader(&XYZ, 8, 4, CV_64FC1, a);
        for(int i=0;i<NUM;i++)
```

```
{
  cvmSet(P, 2*i, 0 , cvmGet(&XYZ, i, 0));
  cvmSet(P, 2*i, 1 , cvmGet(&XYZ, i, 1));
  cvmSet(P, 2*i, 2 , cvmGet(&XYZ, i, 2));
  cvmSet(P, 2*i, 3 , 1);
  cvmSet(P, 2*i, 4 , 0);
  cvmSet(P, 2*i, 5 , 0);
  cvmSet(P, 2*i, 6 , 0);
  cvmSet(P, 2*i, 7 , 0);
  cvmSet(P, 2*i, 8 ,-cvmGet(A, i, 0)*cvmGet(&XYZ, i, 0));
  cvmSet(P, 2*i, 9 ,-cvmGet(A, i, 0)*cvmGet(&XYZ, i, 1));
  cvmSet(P, 2*i, 10 ,-cvmGet(A, i, 0)*cvmGet(&XYZ, i, 2));
  cvmSet(P, 2*i, 11 ,-cvmGet(A, i, 0));

  cvmSet(P, (2*i)+1, 0 , 0);
  cvmSet(P, (2*i)+1, 1 , 0);
  cvmSet(P, (2*i)+1, 2 , 0);
  cvmSet(P, (2*i)+1, 3 , 0);
  cvmSet(P, (2*i)+1, 4 , cvmGet(&XYZ, i, 0));
  cvmSet(P, (2*i)+1, 5 , cvmGet(&XYZ, i, 1));
  cvmSet(P, (2*i)+1, 6 , cvmGet(&XYZ, i, 2));
  cvmSet(P, (2*i)+1, 7 , 1);
  cvmSet(P, (2*i)+1, 8 ,-cvmGet(A, i, 1)*cvmGet(&XYZ, i, 0));
  cvmSet(P, (2*i)+1, 9 ,-cvmGet(A, i, 1)*cvmGet(&XYZ, i, 1));
  cvmSet(P, (2*i)+1, 10 ,-cvmGet(A, i, 1)*cvmGet(&XYZ, i, 2));
  cvmSet(P, (2*i)+1, 11 ,-cvmGet(A, i, 1));
}
  cvSVD(P, w, u, v, 0 );
  //CV_SVD_V_T
```

```
//////////////////copy matrix v last colum to matrix M//////////////////////
            for(b=0;b<3;b++)
            {
                for(k=0;k<4;k++)
                {
                 cvmSet(M, b, k , cvmGet(v, row, 11) );
                 row++;
                 }
            }
                row=0;
////////////////////////////////////////////////////////////////////////
//**********************Reproject Draw marker on image**************************//
      for(int i=0;i<8;i++)
      {
        cvmSet(XT, 0, i , cvmGet(&XYZ, i, 0));
        cvmSet(XT, 1, i , cvmGet(&XYZ, i, 1));
        cvmSet(XT, 2, i , cvmGet(&XYZ, i, 2));
        cvmSet(XT, 3, i , cvmGet(&XYZ, i, 3));
      }
      cvMatMulAdd(M, XT, 0, reproject);
      for(b=0;b<NUM;b++)
            for(k=0;k<2;k++)
            cvmSet(ReProjection,b,k,cvmGet(reproject,k,b)/cvmGet(reproject,2,b));
if(Num_Pic==1)
{
      for(b=0;b<NUM;b++)
            cvCircle(img_1, cvPoint(cvmGet(ReProjection,b,0),cvmGet(ReProjection,b,1)), 5,
cvScalar(0,255,0), 1);
      cvShowImage("calibration_1", img_1 );
```

```
        fp = fopen("C:\\M1\\m1.dat", "w");
        for(b=0;b<3;b++)
        fprintf(fp,"%f %f %f%f\n",cvmGet(M,b,0),cvmGet(M,b,1),cvmGet(M,b,2),cvmGet(M,b,3));
        fclose(fp);
    }
    if(Num_Pic==2)
    {
        for(b=0;b<NUM;b++)
            cvCircle(img_2, cvPoint(cvmGet(ReProjection,b,0),cvmGet(ReProjection,b,1)), 5,
    cvScalar(0,0,255), 1);
        cvShowImage("calibration_2", img_2 );
        fp = fopen("C:\\M2\\m2.dat", "w");

     for(b=0;b<3;b++)
        fprintf(fp,"%f %f %f%f\n",cvmGet(M,b,0),cvmGet(M,b,1),cvmGet(M,b,2),cvmGet(M,b,3));
        fclose(fp);
    }
    if(Num_Pic==3)
    {
        for(b=0;b<NUM;b++)
            cvCircle(img_3, cvPoint(cvmGet(ReProjection,b,0),cvmGet(ReProjection,b,1)), 5,
    cvScalar(255,255,0), 1);
        cvShowImage("calibration_3", img_3 );
        fp = fopen("C:\\M3\\m3.dat", "w");
        for(b=0;b<3;b++)
        fprintf(fp,"%f %f %f%f\n",cvmGet(M,b,0),cvmGet(M,b,1),cvmGet(M,b,2),cvmGet(M,b,3));
        fclose(fp);
    }
    //*************************************************************************//
```

```
//////////////////////////Release Memory/////////////////////////////////////
        cvReleaseMat( &M );
        cvReleaseMat( &P );
        cvReleaseMat( &w );
        cvReleaseMat( &u );
        cvReleaseMat( &v );
        cvReleaseMat( &X );
        cvReleaseMat( &XT );
        cvReleaseMat( &reproject );
        cvReleaseMat( &ReProjection );
//////////////////////////////////////////////////////////////////////////
}
void __fastcall TForm1::Open1Click(TObject *Sender)
{
     img_1=cvLoadImage("C:\\M1\\m1_0.bmp",1);
     img_2=cvLoadImage("C:\\M2\\m2_0.bmp",1);
     img_3=cvLoadImage("C:\\M3\\m3_0.bmp",1);

     cvNamedWindow("calibration_1", CV_WINDOW_AUTOSIZE);
     cvNamedWindow("calibration_2", CV_WINDOW_AUTOSIZE);
     cvNamedWindow("calibration_3", CV_WINDOW_AUTOSIZE);

     cvShowImage("calibration_1", img_1 );
     cvShowImage("calibration_2", img_2 );
     cvShowImage("calibration_3", img_3 );

     cvSetMouseCallback("calibration_1",cvSetMouseCallbackI_1);
     cvSetMouseCallback("calibration_2",cvSetMouseCallbackI_2);
     cvSetMouseCallback("calibration_3",cvSetMouseCallbackI_3);
}
```

```
__fastcall TForm1::~TForm1()
{
      cvReleaseMat(&A);
      cvReleaseImage(&img_1);
      cvReleaseImage(&img_2);
      cvReleaseImage(&img_3);
}
void __fastcall TForm1::Extration(TObject *Sender)
{
      float m1[3][4],m2[3][4],m3[3][4];//,u1[nP][2],u2[nP][2],u3[nP][2];
      int b,k,i,j;
       fp = fopen("c:\\mark.dat","r");
        fscanf(fp,"%d",&nP);
       fclose(fp) ;
       u1=matrix(0,nP,0,2);
       u2=matrix(0,nP,0,2);
       u3=matrix(0,nP,0,2);
//@@@@@@@@ load memory @@@@@@@@@@@@@
      fp=fopen("C:\\M1\\m1.dat","r");
          for(b=0;b<3;b++){
               for(k=0;k<4;k++)
           fscanf(fp,"%f",&m1[b][k]);
           fprintf(fp,"\n");
                    }
      fclose(fp);
      fp=fopen("C:\\M2\\m2.dat","r");
          for(b=0;b<3;b++){
               for(k=0;k<4;k++)
           fscanf(fp,"%f",&m2[b][k]);
           fprintf(fp,"\n");                }
```

```
        fclose(fp);
        fp=fopen("C:\\M3\\m3.dat","r");
            for(b=0;b<3;b++){
                for(k=0;k<4;k++)
             fscanf(fp,"%f",&m3[b][k]);
             fprintf(fp,"\n");
                    }
        fclose(fp);
////////////////////read image point ////////////////////////
for(int s=0;s<NumPrt;s++)
 {
      char U1[20];
      sprintf(U1,"C:\\uv1\\uv1_%d.dat",s );
      fp=fopen(U1,"r");
          for(b=0;b<nP;b++){
               for(k=0;k<2;k++)
           fscanf(fp,"%f",&u1[b][k]);
           fprintf(fp,"\n");
                   }
      fclose(fp);

      char U2[20];
      sprintf(U2,"C:\\uv2\\uv2_%d.dat", s);
      fp=fopen(U2,"r");
          for(b=0;b<nP;b++){
               for(k=0;k<2;k++)
           fscanf(fp,"%f",&u2[b][k]);
           fprintf(fp,"\n");
                   }
      fclose(fp);
```

```
char U3[20];
sprintf(U3,"C:\\uv3\\uv3_%d.dat",s );
fp=fopen(U3,"r");
    for(b=0;b<nP;b++){
        for(k=0;k<2;k++)
      fscanf(fp,"%f",&u3[b][k]);
      fprintf(fp,"\n");
            }
fclose(fp);

//@@@@@@@3-D@@@@@@@@@@@@@@@@@@@@
CvMat* G   = cvCreateMat(6,4,CV_32FC1);
CvMat* w   = cvCreateMat(4,4,CV_32FC1);
CvMat* u   = cvCreateMat(6,4,CV_32FC1);
CvMat* v   = cvCreateMat(4,4,CV_32FC1);
CvMat* z   = cvCreateMat(4,nP,CV_32FC1);
CvMat* zT  = cvCreateMat(nP,4,CV_32FC1);
int  step_G  = G->step/sizeof(float);
float *data_G = G->data.fl;
int  step_z  = z->step/sizeof(float);
float *data_z = z->data.fl;
int  step_v  = v->step/sizeof(float);
float *data_v = v->data.fl;
```

```
for(i=0;i<nP;i++){
        for(k=0;k<4;k++){
        (data_G+0*step_G)[k]=(u1[i][0]*m1[2][k])-m1[0][k];      //matrix fine 8x4
        (data_G+1*step_G)[k]=(u1[i][1]*m1[2][k])-m1[1][k];
        (data_G+2*step_G)[k]=(u2[i][0]*m2[2][k])-m2[0][k];
        (data_G+3*step_G)[k]=(u2[i][1]*m2[2][k])-m2[1][k];
        (data_G+4*step_G)[k]=(u3[i][0]*m3[2][k])-m3[0][k];
        (data_G+5*step_G)[k]=(u3[i][1]*m3[2][k])-m3[1][k];
              }
 cvSVD(G, w, u, v, 0 );
            (data_z+0*step_z)[i]=(data_v+0*step_v)[3] ;
            (data_z+1*step_z)[i]=(data_v+1*step_v)[3] ;
            (data_z+2*step_z)[i]=(data_v+2*step_v)[3] ;
            (data_z+3*step_z)[i]=(data_v+3*step_v)[3] ;
            } //for(i=0;i<NUM;i++)
   cvTranspose(z,zT);

//++++++++++++save parameter for verify++++++++++++++++++++++++++
    char xd[20] ;
    sprintf(xd,"C:\\XYZ_%d.dat",s );
    fp=fopen(xd,"w");
    for(i=0;i<nP;i++)
    fprintf(fp,"%f %f %f\n",cvmGet(zT, i, 0)/cvmGet(zT, i, 3),cvmGet(zT, i, 1)/cvmGet(zT, i,
3),cvmGet(zT, i, 2)/cvmGet(zT, i, 3));
    fclose(fp);
}// for NumPrt
}
//-------------------------------------------------------------------------
```

```
void __fastcall TForm1::Exit1Click(TObject *Sender)
{
    Form1->Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ToolButton3Click(TObject *Sender)
{
     Form2 = new TForm2(this);
 //    Form2->ShowModal();
}
void __fastcall TForm1::avi2jpg1Click(TObject *Sender)
{
     Form3 = new TForm3(this);
     // Form3->ShowModal();
}
void __fastcall TForm1::FormActivate(TObject *Sender)
{
       fp = fopen("c:\\mark.dat","r");
        fscanf(fp,"%d",&nP);
       fclose(fp) ;
}
```

```
//*******FORM2*********
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
#include "Unit1.h"
#include "Unit3.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
//#define NuPt 11
TForm2 *Form2;
float grid=100;
float SizeAxial = 120;
float ScaleSize=2.5;
//***OpenGL****
GLUquadricObj *theObj;
float mat_specular[] = {1.0,1.0,1.0,1.0};
float mat_shininess[] = {50.0};
float light_position[] = {1.0,1.0,1.0,0.0 };
float light_ambient[] = {0.5,0.5,0.5,0.5};
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{       ResetPosition();
        MouseActive = false;
        KEY=false;
        lock=true;
        m=0;
         Application->OnIdle = IdleLoop;
```

```
  pfile=fopen("C:\\NumericF.dat","r");
      fscanf(pfile,"%d",&NumericF);
      fclose(pfile);
      Form1->NumPrt= NumericF;
       pfile=fopen("C:\\mark.dat","r");
      fscanf(pfile,"%d",&NuPt);
      fclose(pfile);
      xxx1->Caption = NuPt;
}
void __fastcall TForm2::FormClose(TObject *Sender, TCloseAction &Action)
{
      Action =caFree;
}
void __fastcall TForm2::FormCreate(TObject *Sender)
{
      hdc = GetDC(Handle);
      SetPixelFormatDescriptor();
      hrc = wglCreateContext(hdc);
      wglMakeCurrent(hdc, hrc);
      SetupRC();
//****light  opengl*******
 glLightfv(GL_LIGHT0,GL_POSITION,light_position);
 glLightfv(GL_LIGHT0,GL_AMBIENT,light_ambient);
 glEnable(GL_LIGHT0);
 glEnable(GL_LIGHTING);
 glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
 glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);
 glColorMaterial(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE);
 glEnable(GL_COLOR_MATERIAL);
```

```
  glEnable(GL_DEPTH_TEST);
  theObj = gluNewQuadric();
}
void __fastcall TForm2::FormDestroy(TObject *Sender)
{
      ReleaseDC(Handle, hdc);
      wglMakeCurrent(hdc, NULL);
      wglDeleteContext(hrc);
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormResize(TObject *Sender)
{
   GLfloat nRange = 250.0f;
   glViewport(0, 0, ClientWidth, ClientHeight);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   if (ClientWidth <= ClientHeight)
     glOrtho(-nRange, nRange, -nRange*ClientHeight/ClientWidth,
                          nRange*ClientHeight/ClientWidth, -nRange, nRange);
   else
     glOrtho(-nRange*ClientWidth/ClientHeight, nRange*ClientWidth/ClientHeight,
                           -nRange, nRange, -nRange, nRange);

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm2::FormPaint(TObject *Sender)
{
    if(KEY)
    {
     RenderGLScene();
     SwapBuffers(hdc);
       }
}
void __fastcall TForm2::SetPixelFormatDescriptor()
{
     PIXELFORMATDESCRIPTOR pfd = {
     sizeof(PIXELFORMATDESCRIPTOR),
     1,
     PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER,
     PFD_TYPE_RGBA,
     24,
     0,0,0,0,0,0,
     0,0,
     0,0,0,0,0,
     32,
     0,
     0,
     PFD_MAIN_PLANE,
     0,
     0,0,0
     };
   PixelFormat = ChoosePixelFormat(hdc, &pfd);
   SetPixelFormat(hdc, PixelFormat, &pfd); //PixelFormat
}
void __fastcall TForm2::SetupRC()
```

```
{
        glClearColor(0, 0, 0, 0);
        glClear(GL_COLOR_BUFFER_BIT);
        glFlush();
}
void __fastcall TForm2::RenderGLScene()
{
if(lock)
{
        glClearColor(0,0,0,0);
        glColor3f(0, 0, 0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glRotatef(-80, 1.0, 0.0, 0.0); // (-100, 1.0, 0.0, 0.0)
        glRotatef(25, 0.0, 0.0, 1.0);//(35, 0.0, 0.0, 1.0)
        glRotatef(10, 0.0, 1.0, 0.0);//(35, 0.0, 0.0, 1.0)
        glScalef(ScaleSize, ScaleSize, ScaleSize);
        glTranslatef(0,0,-60); //0,0,-190
       lock=0;
}
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        gl_PolarView(mAzimuth, mElevation, mTwist);
//////////////////////////Plot Point ////////////////////////////////////////
        ReadPosition();
        PlotSkeleton();
//////////////////////////////////////////////////////////////////////////////
        glLineWidth(0.5);
          glColor3f(1, 1, 1);
           for(int i=-3;i<5;i++)
         {  for (int j=-3;j<5;j=j+2) //x
           {  if (i%2==0)
```

```
        { glBegin(GL_POLYGON);
       glVertex3f(j*20,i*20,0);
       glVertex3f((j+1)*20,i*20,0);
       glVertex3f((j+1)*20,(i+1)*20,0);
       glVertex3f(j*20,(i+1)*20,0);
        glEnd();
        }
        else  {
         glBegin(GL_POLYGON);
       glVertex3f((j-1)*20,i*20,0);
       glVertex3f(j*20,i*20,0);
       glVertex3f(j*20,(i+1)*20,0);
       glVertex3f((j-1)*20,(i+1)*20,0);
        glEnd();  }
      }
   }
////////create axis x y z //////////
     glLineWidth(2);
     glBegin(GL_LINES);
     glColor3f(1, 0, 0);
glVertex3f(0, 0, 0);
glVertex3f(SizeAxial,0, 0);
glColor3f(1, 1, 0);
glVertex3f(0, 0, 0);
glVertex3f(-SizeAxial,0, 0);
glColor3f(0, 1, 0);
glVertex3f(0, 0, 0);
glVertex3f(0,SizeAxial, 0);
glColor3f(0, 1, 1);
glVertex3f(0, 0, 0);
```

```
        glVertex3f(0,-SizeAxial, 0);
        glColor3f(0, 0, 1);
        glVertex3f(0, 0, 0);
        glVertex3f(0,0, SizeAxial);
          glEnd();
        Sleep(400);
         glFlush();
     mAzimuth = 0;
     mElevation = 0;}
void __fastcall TForm2::IdleLoop(TObject*, bool& done)
{
       glColor3f(0, 0, 0);
       glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
       done = false;
       if(KEY)
       RenderGLScene();
       SwapBuffers(hdc);
}
void __fastcall TForm2::gl_PolarView(GLint mAzimuth, GLint mElevation, GLint mTwist)
{
     glRotatef(mTwist, 0.0, 0.0, 1.0);
     glRotatef(mAzimuth, 0.0, 1.0, 0.0);
     glRotatef(mElevation, 1.0, 0.0, 0.0);
}
void __fastcall TForm2::RotateModel()
{
      inc = 3;
      mTwist = (mTwist- inc) % 360;
      FormPaint(NULL);
}
```

```
void __fastcall TForm2::FormMouseDown(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y)
{
    if(Button == mbLeft)
    {       iRot = true;
    startx = X;  starty = Y;  }
    else{
    iZoom = true;
    glScalef(2, 2, 2);
    }
}
void __fastcall TForm2::FormMouseUp(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y)
{
    if(Button == mbLeft)
    {
      iRot = false;
    } //
    else iZoom = false;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormMouseMove(TObject *Sender, TShiftState Shift,
    int X, int Y)
{if(iRot)
 {
  mElevation = (mElevation + Y - starty) ;
  mAzimuth = (mAzimuth + X - startx) ;
  startx = X; starty = Y;
  FormPaint(NULL);
 }
```

```
  else if(iZoom)
  {
   sRange +=  - Y + starty;
   starty = Y;
   FormResize(NULL);
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm2::ResetPosition()
{
      vMax=221;
      mElevation = 0; mAzimuth = 0; mTwist = 0;
      mDistance = -vMax/2;    //Distance from camera to center of model
      mHorz = 0-10; mVert = 0;    //Model position
      sRange = vMax/2;        //Frame size
}
void __fastcall TForm2::Timer1Timer(TObject *Sender)
{
      RotateModel();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::ReadPosition()
{
      char a[30];
   sprintf(a,"C:\\XYZ_%d.dat",m);
   pfile=fopen(a,"r");
   for(int b=0;b<NuPt;b++){
         for(int k=0;k<3;k++){fscanf(pfile,"%f",&u[b][k]);}
      fprintf(pfile,"\n");
                              }
```

```
        m++;
        if(m==NumericF)m=0;
                fclose(pfile);
}
void __fastcall TForm2::PlotSkeleton()
{
      switch(NuPt)
      {
       case 11: ///// hand
       {
        for(int a=0;a<NuPt;a++)
      {
      glColor3f(1, 0.5, 0.4);
      glTranslatef(3*u[a][0],3*u[a][1],3*u[a][2]);
      gluQuadricDrawStyle(theObj,GLU_FILL);
      gluSphere(theObj,1.5,20,20);
      glTranslatef(-3*u[a][0],-3*u[a][1],-3*u[a][2]);
       }
        glBegin(GL_LINES); // line marker
      for(int i=0;i<5;i++)
      {      glColor3f(1, 0.5, 0.9);
           glVertex3f(3*u[2*i][0],3*u[2*i][1],3*u[2*i][2]);
           glVertex3f(3*u[2*i+1][0],3*u[2*i+1][1],3*u[2*i+1][2]);
            if (i<4)
            {
             glColor3f(1, 0, 0.9);
             glVertex3f(3*u[2*i+1][0],3*u[2*i+1][1],3*u[2*i+1][2]);
             glVertex3f(3*u[2*i+3][0],3*u[2*i+3][1],3*u[2*i+3][2]);  }
            }
```

```
      glColor3f(1, 0, 0.9);
       glVertex3f(3*u[1][0],3*u[1][1],3*u[1][2]);
       glVertex3f(3*u[10][0],3*u[10][1],3*u[10][2]);
       glVertex3f(3*u[10][0],3*u[10][1],3*u[10][2]);
       glVertex3f(3*u[9][0],3*u[9][1],3*u[9][2]);
      glEnd();
   }break;
   case 6:          // leg
   {
 for(int a=0;a<NuPt;a++)
 {
 glColor3f(1, 0.5, 0.4);
 glTranslatef(3*u[a][0],3*u[a][1],3*u[a][2]);
 gluQuadricDrawStyle(theObj,GLU_FILL);
 gluSphere(theObj,2.5,20,20);
 glTranslatef(-3*u[a][0],-3*u[a][1],-3*u[a][2]);
 }
glBegin(GL_LINES); // line marker
 for(int i=0;i<4;i++)
 {      glColor3f(1, 0.5, 0.9);
      glVertex3f(3*u[i][0],3*u[i][1],3*u[i][2]);
      glVertex3f(3*u[i+2][0],3*u[i+2][1],3*u[i+2][2]);
 }
      glVertex3f(3*u[1][0],3*u[1][1],3*u[1][2]);
      glVertex3f(3*u[0][0],3*u[0][1],3*u[0][2]);
     glEnd();
  }break;
  case 14:          // body
  {
 for(int a=0;a<NuPt;a++)
```

```
{
glColor3f(1, 0.5, 0.4);
glTranslatef(3*u[a][0],3*u[a][1],3*u[a][2]);
gluQuadricDrawStyle(theObj,GLU_FILL);
gluSphere(theObj,1.5,20,20);
glTranslatef(-3*u[a][0],-3*u[a][1],-3*u[a][2]);
 }
glBegin(GL_LINES); // line marker
for(int i=1;i<3;i++)
{      glColor3f(1, 0.5, 0.9);
     glVertex3f(3*u[i][0],3*u[i][1],3*u[i][2]);
     glVertex3f(3*u[i+1][0],3*u[i+1][1],3*u[i+1][2]);
     glVertex3f(3*u[i+3][0],3*u[i+3][1],3*u[i+3][2]);
     glVertex3f(3*u[i+4][0],3*u[i+4][1],3*u[i+4][2]);
     glVertex3f(3*u[i+7][0],3*u[i+7][1],3*u[i+7][2]);
     glVertex3f(3*u[i+8][0],3*u[i+8][1],3*u[i+8][2]);
     glVertex3f(3*u[i+10][0],3*u[i+10][1],3*u[i+10][2]);
     glVertex3f(3*u[i+11][0],3*u[i+11][1],3*u[i+11][2]);}
     glVertex3f(3*u[1][0],3*u[1][1],3*u[1][2]);
     glVertex3f(3*u[7][0],3*u[7][1],3*u[7][2]);
     glVertex3f(3*u[4][0],3*u[4][1],3*u[4][2]);
     glVertex3f(3*u[7][0],3*u[7][1],3*u[7][2]);
     glVertex3f(3*u[8][0],3*u[8][1],3*u[8][2]);
     glVertex3f(3*u[7][0],3*u[7][1],3*u[7][2]);
     glVertex3f(3*u[11][0],3*u[11][1],3*u[11][2]);
     glVertex3f(3*u[7][0],3*u[7][1],3*u[7][2]);
     glVertex3f(3*u[1][0],3*u[1][1],3*u[1][2]);
     glVertex3f(3*u[4][0],3*u[4][1],3*u[4][2]);
    glEnd();  }break;
}//end switch}
```

```
void __fastcall TForm2::Run2Click(TObject *Sender)
{
      KEY=true;
      FormPaint(NULL);
}
void __fastcall TForm2::R1Click(TObject *Sender)
{
      Timer1->Enabled = true;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::stop1Click(TObject *Sender)
{
      Timer1->Enabled = false;
      mElevation = 0; mAzimuth = 0; mTwist = 0;
}
//---------------------------------------------------------------------------
```

```
//*******FORM3*********
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit3.h"
#include <stdio.h>
#include "Unit2.h"
#define NRANSI
#include "nr.h"
#include "nrutil.h"
#include "nrutil.c"
#include "PYTHAG.C"
#include "qr.C"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
//#define Num_point 6
TForm3 *Form3;
IplImage* frame ;
FILE* fpp,*data;
CvMat *co_uv,*rgb,*D_uv;
int Num_point;
int camera;
int i;
double r,g,b;
int N_pic,N_click;
int N_pic_max ;
int D_x,D_y;
double P_x,P_y;
double R_r,R_g,R_b;
```

```
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
         //i-0;
        N_pic=0;
        Num_point=6;
        camera=0;
        Edit3->Text = IntToStr(Num_point);
        co_uv = cvCreateMat( Num_point, 2, CV_32F);  //coordinate of Image
        D_uv = cvCreateMat( Num_point, 2, CV_32F);
        rgb = cvCreateMat( Num_point, 3, CV_32F);
        ScrollBar1->Position = 20;
        Label1->Caption = ScrollBar1->Position ;
        ScrollBar2->Position = 20;
        Label2->Caption = ScrollBar1->Position ;
         ScrollBar3->Position = 5;
        Label3->Caption = ScrollBar3->Position ;
        i=0;
        x=0;
        fpp = fopen("C:\\mark.dat", "w");
              fprintf(fpp, "%d\n",Num_point);
              fclose(fpp);
}
//---------------------------------------------------------------------------
void __fastcall TForm3::FormClose(TObject *Sender, TCloseAction &Action)
{
      Action =caFree;
}
```

```
void __fastcall TForm3::LoadFileavi1Click(TObject *Sender)
{
      OpenDialog1->Filter="avi (*.avi)|*.avi";
      if( OpenDialog1->Execute() )
                        {
                         OpenDialog1->FileName;
                        }
}
//---------------------------------------------------------------------
void __fastcall TForm3::Convert(TObject *Sender)
{
       char a[20];
       frame = 0;
       CvCapture* capture = cvCaptureFromAVI(OpenDialog1->FileName.c_str());
       cvNamedWindow( "mywindow", CV_WINDOW_AUTOSIZE );
while( 1 )
{
              // Get one frame
             frame = cvQueryFrame( capture );
     if( !frame ) {
              fprintf( stderr, "ERROR: frame is null...\n" );
              getchar();
              break;
               }
             cvShowImage( "mywindow", frame );
            // Do not release the frame!
            //If ESC key pressed, Key=0x10001B under OpenCV 0.9.7(linux version),
            //remove higher bits using AND operator
```

```
            if( (cvWaitKey(10) & 255) == 27 ) break;
            if(RadioButton1->Checked==true)
              sprintf(a,"C:\\M1\\m1_%d.bmp",x);
            if(RadioButton2->Checked==true)
              sprintf(a,"C:\\M2\\m2_%d.bmp",x);
            if(RadioButton3->Checked==true)
              sprintf(a,"C:\\M3\\m3_%d.bmp",x);
            if(RadioButton4->Checked==true)
              sprintf(a,"C:\\uv1\\uv1_%d.bmp",x);
            if(RadioButton5->Checked==true)
              sprintf(a,"C:\\uv2\\uv2_%d.bmp",x);
            if(RadioButton6->Checked==true)
              sprintf(a,"C:\\uv3\\uv3_%d.bmp",x);
            cvSaveImage( a, frame );
            x++;
}
  fpp = fopen("c:\\NumericF.dat", "w");
  fprintf(fpp, "%d\n",x);
  fclose(fpp);
 x=0;
 cvReleaseCapture( &capture );
 cvDestroyWindow( "mywindow" );
}
void __fastcall TForm3::Load(TObject *Sender) //loadpicture
{  sprintf(I_Ex1,"C:\\uv1\\uv1_0.bmp");
   Image1->Picture->LoadFromFile(I_Ex1);
   sprintf(I_Ex2,"C:\\uv2\\uv2_0.bmp");
   Image2->Picture->LoadFromFile(I_Ex2);
   sprintf(I_Ex3,"C:\\uv3\\uv3_0.bmp");
   Image3->Picture->LoadFromFile(I_Ex3);}
```

```
void __fastcall TForm3::Image1MouseDown(TObject *Sender,
    TMouseButton Button, TShiftState Shift, int X, int Y)
{
      int xx,yy;
     int r1=0,g1=0,b1=0,n_pixels=0;
     RadioButton7->Checked = true;
     for (yy=Y-5;yy<Y+5;yy++)
     {
     for (xx=X-5;xx<X+5;xx++)
     {
     r1=GetRValue(Image1->Canvas->Pixels[xx][yy]);
     g1=GetGValue(Image1->Canvas->Pixels[xx][yy]);
     b1=GetBValue(Image1->Canvas->Pixels[xx][yy]);
     r=r+r1;
     g=g+g1;
     b=b+b1;
     n_pixels= n_pixels++;
     }
    }
    r=r/n_pixels;
    g=g/n_pixels;
    b=b/n_pixels;
//****rgb ref**************
    cvmSet(rgb, i, 0 , r);
    cvmSet(rgb, i, 1 , g);
    cvmSet(rgb, i, 2 , b);
//****uv ref**************
    cvmSet(co_uv, i, 0 , X);
    cvmSet(co_uv, i, 1 , Y);
    i++;
```

```
        if(i==Num_point){
            char Q_[20];
            sprintf(Q_,"C:\\uv1\\ref_uv.dat");
            fpp = fopen(Q_, "w");
            for(int count=0;count<Num_point;count++)
            fprintf(fpp, "%.3f\t  %.3f\n",cvmGet(co_uv, count, 0),cvmGet(co_uv, count, 1));
            fclose(fpp);
            sprintf(Q_,"C:\\uv1\\uv1_0.dat");
            fpp = fopen(Q_, "w");
            for(int count=0;count<Num_point;count++)
            fprintf(fpp, "%.3f\t  %.3f\n",cvmGet(co_uv, count, 0),cvmGet(co_uv, count, 1));
            fclose(fpp);
            sprintf(Q_,"C:\\uv1\\ref_rgb.dat");
            fpp = fopen(Q_, "w");
            for(int count=0;count<Num_point;count++)
            fprintf(fpp, "%.3f\t %.3f\t %.3f\n",cvmGet(rgb, count, 0),cvmGet(rgb, count,
1),cvmGet(rgb, count, 2));
            fclose(fpp);
            i=0;}//end if
}
void __fastcall TForm3::Image2MouseDown(TObject *Sender,
    TMouseButton Button, TShiftState Shift, int X, int Y)
{
     int xx,yy;
     int r1=0,g1=0,b1=0,n_pixels=0;
     RadioButton8->Checked = true;
```

```
 for (yy=Y-5;yy<Y+5;yy++)
      {
      for (xx=X-5;xx<X+5;xx++)
      {
       r1=GetRValue(Image2->Canvas->Pixels[xx][yy]);
       g1=GetGValue(Image2->Canvas->Pixels[xx][yy]);
       b1=GetBValue(Image2->Canvas->Pixels[xx][yy]);
       r=r+r1;
       g=g+g1;
       b=b+b1;
       n_pixels= n_pixels++;
      }
     }
     r=r/n_pixels;
     g=g/n_pixels;
     b=b/n_pixels;

     cvmSet(rgb, i, 0 , r);
     cvmSet(rgb, i, 1 , g);
     cvmSet(rgb, i, 2 , b);
     cvmSet(co_uv, i, 0 , X);
     cvmSet(co_uv, i, 1 , Y);
     i++;
if(i==Num_point){
          char Q_[20];
          sprintf(Q_,"C:\\uv2\\ref_uv.dat");
          fpp = fopen(Q_, "w");
          for(int count=0;count<Num_point;count++)
          fprintf(fpp, "%.3f\t  %.3f\n",cvmGet(co_uv, count, 0),cvmGet(co_uv, count, 1));
          fclose(fpp);
```

```
            sprintf(Q_,"C:\\uv2\\uv2_0.dat");
            fpp = fopen(Q_, "w");
            for(int count=0;count<Num_point;count++)
            fprintf(fpp, "%.3f\t  %.3f\n",cvmGet(co_uv, count, 0),cvmGet(co_uv, count, 1));
            fclose(fpp);
            sprintf(Q_,"C:\\uv2\\ref_rgb.dat");
            fpp = fopen(Q_, "w");
            for(int count=0;count<Num_point;count++)
            fprintf(fpp, "%.3f\t %.3f\t %.3f\n",cvmGet(rgb, count, 0),cvmGet(rgb, count,
1),cvmGet(rgb, count, 2));
            fclose(fpp);
            i=0;
          }
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Image3MouseDown(TObject *Sender,
    TMouseButton Button, TShiftState Shift, int X, int Y)
{    int xx,yy;
     int r1=0,g1=0,b1=0,n_pixels=0;
     RadioButton9->Checked = true;
     for (yy=Y-5;yy<Y+5;yy++)
     {for (xx=X-5;xx<X+5;xx++)
      { r1=GetRValue(Image3->Canvas->Pixels[xx][yy]);
        g1=GetGValue(Image3->Canvas->Pixels[xx][yy]);
        b1=GetBValue(Image3->Canvas->Pixels[xx][yy]);
      r=r+r1;
      g=g+g1;
      b=b+b1;
      n_pixels= n_pixels++ }}
```

```
        r=r/n_pixels;
        g=g/n_pixels;
        b=b/n_pixels;
        cvmSet(rgb, i, 0 , r);
        cvmSet(rgb, i, 1 , g);
        cvmSet(rgb, i, 2 , b);
        cvmSet(co_uv, i, 0 , X);
        cvmSet(co_uv, i, 1 , Y);
        i++;
        if(i==Num_point){
            char Q_[20];
            sprintf(Q_,"C:\\uv3\\ref_uv.dat");
            fpp = fopen(Q_, "w");
            for(int count=0;count<Num_point;count++)
            fprintf(fpp, "%.3f\t %.3f\n",cvmGet(co_uv, count, 0),cvmGet(co_uv, count, 1));
            fclose(fpp);
            sprintf(Q_,"C:\\uv3\\uv3_0.dat");
            fpp = fopen(Q_, "w");
            for(int count=0;count<Num_point;count++)
            fprintf(fpp, "%.3f\t %.3f\n",cvmGet(co_uv, count, 0),cvmGet(co_uv, count, 1));
            fclose(fpp);
            sprintf(Q_,"C:\\uv3\\ref_rgb.dat");
            fpp = fopen(Q_, "w");
            for(int count=0;count<Num_point;count++)
            fprintf(fpp, "%.3f\t %.3f\t %.3f\n",cvmGet(rgb, count, 0),cvmGet(rgb, count,
1),cvmGet(rgb, count, 2));
            fclose(fpp);
            i=0;
        }
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm3::DetermineFrame(TObject *Sender)
{
  //count_pic= (StrToInt(Edit1->Text)-1);
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Threshold()
{
      char file[40] ;
      int redchk_min,redchk_max,greenchk_min,greenchk_max,bluechk_min,bluechk_max;
      bool redchk1,greenchk1,bluechk1;
      bool redchk2,greenchk2,bluechk2;
      int r_y ,r_x,r_x_ref,r_y_ref;
      int Ymin=0,Ymax=0,Xmin=0,Xmax=0;
      int D_Ymin,D_Ymax,D_Xmin,D_Xmax;
      int N_color=0;
      int N_color_max=3;
      int mmm=1;
     Graphics::TBitmap *pBitmap = new Graphics::TBitmap();
try
 {
     sprintf(file,"c://uv%d//uv%d_%d.bmp",camera,camera,N_pic) ;
     pBitmap->LoadFromFile(file);
     bool First;
////***********color ******************************
     redchk_min =  R_r-StrToInt(Label2->Caption) ;
     redchk_max =  R_r+StrToInt(Label2->Caption);
     greenchk_min =  R_g-StrToInt(Label2->Caption);
     greenchk_max =  R_g+StrToInt(Label2->Caption);
```

```
bluechk_min =  R_b-StrToInt(Label2->Caption);
bluechk_max =  R_b+StrToInt(Label2->Caption);

if (redchk_min<0){redchk_min=0;}
if (redchk_max>255){redchk_max=255;}
if (greenchk_min<0){greenchk_min=0;}
if (greenchk_max>255){greenchk_max=255;}
if (bluechk_min<0){bluechk_min=0;}
if (bluechk_max>255){bluechk_max=255;}
  First = true ;
    for (int j=P_y-(StrToInt(Label1->Caption));j<=P_y+(StrToInt(Label1->Caption));j++)
    {
    for (int i=P_x-(StrToInt(Label1->Caption));i<=P_x+(StrToInt(Label1->Caption));i++)
    {
    redchk1= (GetRValue(pBitmap->Canvas->Pixels[i][j])>=redchk_min) &&
(GetRValue(pBitmap->Canvas->Pixels[i][j])<=redchk_max) ;
    greenchk1=(GetGValue(pBitmap->Canvas->Pixels[i][j])>=greenchk_min) &&
(GetGValue(pBitmap->Canvas->Pixels[i][j])<=greenchk_max) ;
    bluechk1=(GetBValue(pBitmap->Canvas->Pixels[i][j])>=bluechk_min) &&
(GetBValue(pBitmap->Canvas->Pixels[i][j])<=bluechk_max) ;
    if (redchk1 && greenchk1 && bluechk1)
    {
     for (int k=i-5;k<=i+5;k++)
     {
     redchk2= (GetRValue(pBitmap->Canvas->Pixels[k][j])>=redchk_min) &&
(GetRValue(pBitmap->Canvas->Pixels[k][j])<=redchk_max) ;
     greenchk2=(GetGValue(pBitmap->Canvas->Pixels[k][j])>=greenchk_min) &&
(GetGValue(pBitmap->Canvas->Pixels[k][j])<=greenchk_max) ;
     bluechk2=(GetBValue(pBitmap->Canvas->Pixels[k][j])>=bluechk_min) &&
(GetBValue(pBitmap->Canvas->Pixels[k][j])<=bluechk_max) ;
```

```
if (redchk2 && greenchk2 && bluechk2)
{
N_color++;
}
}
if (N_color>=N_color_max)
{
N_color = 0;
  if (First==true)
   {
   Ymin=j;
   Xmin=i;
   First=false;
   }
   else
   {
   if (j<Ymin)  Ymin=j;
   if (j>Ymax)  Ymax=j;
   if (i<Xmin)  Xmin=i;
   if (i>Xmax)  Xmax=i;
   }
}
}
} //loop i
} //loop j
 D_x=(Xmin+Xmax)/2;
 D_y=(Ymin+Ymax)/2;
```

```
if ((abs(P_x-D_x)>=2)||(abs(P_y-D_y)>=2))
{ for (int j=P_y-10;j<=P_y+10;j++)
 { for (int i=P_x-10;i<=P_x+10;i++)
 { redchk1= (GetRValue(pBitmap->Canvas->Pixels[i][j])>=redchk_min-5) &&
(GetRValue(pBitmap->Canvas->Pixels[i][j])<=redchk_max+5) ;
 greenchk1=(GetGValue(pBitmap->Canvas->Pixels[i][j])>=greenchk_min-5) &&
(GetGValue(pBitmap->Canvas->Pixels[i][j])<=greenchk_max+5) ;
 bluechk1=(GetBValue(pBitmap->Canvas->Pixels[i][j])>=bluechk_min-5) &&
(GetBValue(pBitmap->Canvas->Pixels[i][j])<=bluechk_max+5) ;
 if (redchk1 && greenchk1 && bluechk1)
 {if (First==true)
 {Ymin=j;
 Xmin=i;
 First=false; }
 else
 { if (j<Ymin) Ymin=j;
 if (j>Ymax) Ymax=j;
 if (i<Xmin) Xmin=i;
 if (i>Xmax) Xmax=i;}
 }
 } //loop i
 } //loop j
 D_x=(Xmin+Xmax)/2;
 D_y=(Ymin+Ymax)/2;
 } else
 { D_x=P_x ;
 D_y=P_y; }
 if ((abs(P_x-D_x)>6)&&(abs(P_y-D_y)>6))
 { D_x=P_x ;
 D_y=P_y; }
```

```
//********************** finish Threshold*********************************
            cvmSet(D_uv, N_click, 0 , D_x);
            cvmSet(D_uv, N_click, 1 , D_y);
          switch (camera){
            case 1:
            {
            Image1->Picture->Bitmap->Canvas->Pen->Color =clRed;
            Image1->Picture->Bitmap->Canvas->Pen->Width = 3;
            Image1->Picture->Bitmap->Canvas->Brush->Style = bsClear;
            Image1->Picture->Bitmap->Canvas->Ellipse(D_x-(ScrollBar3->Position ),D_y-
(ScrollBar3->Position ),D_x+(ScrollBar3->Position ),D_y+(ScrollBar3->Position )) ;
            }break;
            case 2:
            { Image2->Picture->Bitmap->Canvas->Pen->Color =clRed;
            Image2->Picture->Bitmap->Canvas->Pen->Width = 3;
            Image2->Picture->Bitmap->Canvas->Brush->Style = bsClear;
            Image2->Picture->Bitmap->Canvas->Ellipse(D_x-(ScrollBar3->Position ),D_y-
(ScrollBar3->Position ),D_x+(ScrollBar3->Position ),D_y+(ScrollBar3->Position )) ;
            }break;
            case 3:
            {  Image3->Picture->Bitmap->Canvas->Pen->Color =clRed;
            Image3->Picture->Bitmap->Canvas->Pen->Width = 3;
            Image3->Picture->Bitmap->Canvas->Brush->Style = bsClear;
            Image3->Picture->Bitmap->Canvas->Ellipse(D_x-(ScrollBar3->Position ),D_y-
(ScrollBar3->Position ),D_x+(ScrollBar3->Position ),D_y+(ScrollBar3->Position )) ;
            }break;
          } //switch}
 catch (...)
 { ShowMessage("Could not load or alter bitmap"); }
 delete pBitmap;}
```

```
void __fastcall TForm3::Button4Click(TObject *Sender)
{
        N_pic=0;
        Timer1->Enabled = true ;
}
__fastcall TForm3::~TForm3()
{
     cvReleaseMat( &co_uv );
     cvReleaseMat( &rgb );
}
//---------------------------------------------------------------------
void __fastcall TForm3::Timer1Timer(TObject *Sender)
{
     char buffer[30];
     int  NumericF=0 ;
      fpp = fopen("c:\\NumericF.dat","r");
      fscanf(fpp,"%d",&NumericF);
      N_pic_max = NumericF-1;
      fclose(fpp) ;
     if (N_pic<N_pic_max)
     {
     N_pic++;
     sprintf(buffer,"c://uv%d//uv%d_%d.bmp",camera,camera,N_pic);
     switch (camera){
      case 1:
      { Image1->Picture->LoadFromFile(buffer);
      }break;
     case 2:
      {Image2->Picture->LoadFromFile(buffer);
      }break;
```

```
        case 3:
         {Image3->Picture->LoadFromFile(buffer);
         }break;
        }//end switch
        Timer1->Enabled = false ;
        mask();
        }
}
//---------------------------------------------------------------------------
void __fastcall TForm3::mask()
{
        int oo,gg;
        char buffer[100];
        float **ref_uv=matrix(1,Num_point,1,2);
        float **ref_rgb=matrix(1,Num_point,1,3);
         //uv_ref
        sprintf(buffer,"C:\\uv%d\\ref_uv.dat",camera)  ;
        fpp = fopen(buffer,"r");
        for (oo=1;oo<Num_point+1;oo++ )
        {
        for (gg=1;gg<3;gg++ )
        {
        fscanf(fpp,"%f",&ref_uv[oo][gg]);
        }
        cvmSet(co_uv, oo-1, 0 , ref_uv[oo][1]);
        cvmSet(co_uv, oo-1, 1 , ref_uv[oo][2]);
        fprintf(fpp,"\n");
        }
        fclose(fpp) ;
```

```
//color
    sprintf(buffer,"C:\\uv%d\\ref_rgb.dat",camera);
    fpp = fopen(buffer,"r");
     for (oo=1;oo<Num_point+1;oo++ )
    { for (gg=1;gg<4;gg++ )
      {fscanf(fpp,"%f",&ref_rgb[oo][gg]);}
    cvmSet(rgb, oo-1, 0 , ref_rgb[oo][1]);
    cvmSet(rgb, oo-1, 1 , ref_rgb[oo][2]);
    cvmSet(rgb, oo-1, 2 , ref_rgb[oo][3]);
    }
     fclose(fpp) ;
    for (N_click=0;N_click<Num_point;N_click++)
    {
    P_x = cvmGet(co_uv, N_click, 0);
    P_y = cvmGet(co_uv, N_click, 1);
   R_r = cvmGet(rgb, N_click, 0);
    R_g = cvmGet(rgb, N_click, 1);
    R_b = cvmGet(rgb, N_click, 2);
    Threshold();   }
        sprintf(buffer,"C:\\uv%d\\uv%d_%d.dat",camera,camera,N_pic);
        data = fopen(buffer, "w");
        for(int count=0;count<Num_point;count++)
        fprintf(data, "%.3f %.3f \n",cvmGet(D_uv, count, 0),cvmGet(D_uv, count, 1));
        fclose(data);
        sprintf(buffer,"C:\\uv%d\\ref_uv.dat",camera);
        data = fopen(buffer, "w");
        for(int count=0;count<Num_point;count++)
        fprintf(data, "%.3f %.3f \n",cvmGet(D_uv, count, 0),cvmGet(D_uv, count, 1));
        fclose(data);
        Timer1->Enabled = true ;}
```

```
void __fastcall TForm3::RadioButton7Click(TObject *Sender)
{
    camera = 1;
}
void __fastcall TForm3::RadioButton8Click(TObject *Sender)
{
    camera = 2;
}
void __fastcall TForm3::RadioButton9Click(TObject *Sender)
{
    camera = 3;
}
void __fastcall TForm3::ScrollBar1Change(TObject *Sender)
{
    Label1->Caption = ScrollBar1->Position ;
}
void __fastcall TForm3::ScrollBar2Change(TObject *Sender)
{
    Label2->Caption = ScrollBar2->Position ;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Image1MouseMove(TObject *Sender, TShiftState Shift,
    int X, int Y)
{
    Image1->Cursor =  crCross;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Image2MouseMove(TObject *Sender, TShiftState Shift,
    int X, int Y)
{Image2->Cursor =  crCross;}
```

```
void __fastcall TForm3::Image3MouseMove(TObject *Sender, TShiftState Shift,
    int X, int Y)
{
     Image3->Cursor =  crCross;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Edit3Change(TObject *Sender)
{
     Num_point=StrToInt(Edit3->Text);
     fpp = fopen("C:\\mark.dat", "w");
          fprintf(fpp, "%d\n",Num_point);
          fclose(fpp);

     co_uv = cvCreateMat( Num_point, 2, CV_32F);  //coordinate of Image
     D_uv = cvCreateMat( Num_point, 2, CV_32F);
     rgb = cvCreateMat( Num_point, 3, CV_32F);
}
//---------------------------------------------------------------------------
void __fastcall TForm3::ScrollBar3Change(TObject *Sender)
{
     Label3->Caption = ScrollBar3->Position ;
}
//---------------------------------------------------------------------------
```