ระบบการตรวจหาทิศทาง และ ความยาวโฟกัสของกล้องวิดีโอเพื่อการแทรกวัตถุเสมือน

Calibration System for Augmented reality

โดย

นาย วิชิต นันทรัตนพงศ์

รหัส 43067058

อาจารย์ที่ปรึกษา

ผศ.ดร. นพพร โชติกกำธร

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคเรียนที่ 1 ปีการศึกษา 2545
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	ระบบการตรวจหาทิศทาง และ ความยาวโฟกัสของกล้องวิดีโอเพื่อการแทรกวัตถุเสมือน
นักศึกษา	นาย วิชิต นันทรัตนพงศ์
อาจารย์ที่ปรึกษา	ผศ.ดร. นพพร โชติกกำธร
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2545

บทคัดย่อ

การแทรกวัตถุเสมือน (virtual object) ลงในภาพจริงโดยเฉพาะภาพเคลื่อนไหว เช่น ภาพวิดีโอแบบเรียลไทม์ (real time video) มีสิ่งสำคัญที่จะทำให้ภาพวิดีโอที่ปรากฏมีความสมจริงคือ ระยะทางจากกล้องวิดีโอถึงวัตถุจริง เช่น คนหรือสิ่งของในภาพ และ ความยาวโฟกัส (focal length) ของกล้องวิดีโอ ซึ่งในระหว่างการบันทึกภาพปัจจัยทั้งสองนี้มีการเปลี่ยนแปลงได้ตลอดเวลา ระยะทางจากกล้องวิดีโอถึงวัตถุจริงเป็นตัวกำหนดขนาดของวัตถุเสมือน กล่าวคือเมื่อเคลื่อนกล้องวิดีโอเข้าใกล้วัตถุจริง ภาพวัตถุจริงจะมีขนาดใหญ่ขึ้นเพราะฉะนั้นวัตถุเสมือนจะต้องมีขนาดใหญ่ขึ้นในสัดส่วนเดียวกัน ในขณะที่ความยาวโฟกัสก็มีผลต่อขนาด และ ลักษณะ perspective ของภาพด้วย เช่น เมื่อมีการซูม ขนาดของวัตถุในภาพจะต้องมีขนาดใหญ่ขึ้น และ ผู้ที่ดูภาพจะรู้สึกเสมือนภาพเข้าใกล้ตามากขึ้น

การหาระยะทางจากกล้องวิดีโอถึงวัตถุจริงใช้วิธีการติดเซ็นเซอร์ไว้ที่กล้องวิดีโอเพื่อหาระยะทางที่เปลี่ยนแปลงไปเมื่อกล้องวิดีโอมีการเคลื่อนที่เทียบกับจุดเริ่มต้นใดๆ ส่วนการหาระยะโฟกัสของกล้องวิดีโอใช้วัตถุทรงกลมเป็นตัววัด โดยเมื่อภาพวัตถุทรงกลม project บนเซ็นเซอร์รับภาพของกล้องวิดีโอ รัศมีของภาพวงกลมที่เกิดขึ้นมีผลเปลี่ยนแปลงมาจากระยะทางจากกล้องวิดีโอถึงวัตถุจริง และ ระยะโฟกัส ดังนั้นเมื่อทราบระยะทางจากกล้องวิดีโอถึงวัตถุจริงด้วยวิธีการข้างต้นจึงสามารถคำนวณหาระยะโฟกัสได้

Title	Calibration System for Augmented Reality
Student	Mr. Wichit Nuntharattanapong
Advisor	Asst. Prof. Dr. Nopporn Chotikakamthorn
Level of study	Master of Science in Information Technology
Major	Information Science
Academic year	2002

ABSTRACT

The important factors when we want to insert virtual objects to real video are a distance from video camera to real objects and a focal length. In real time situation, these two factors are always changed. The distance told that what size of virtual object should be in video image. For instance, when the camera moves forward to real object, its virtual object must be proportionally enlarged. With regard to the focal length, it gave information about the perspective of the virtual object.

In my project, sphere is used to find out the focal length. When image of sphere is projected on the camera it would be a circle. The radian of the circle is affected by two factors that I mentioned above. The distance derives from attachment a sensor to the camera. It finds out how far from the camera to the origin. By using this method, you know the distance, therefore you will able to calculate the focal length.

กิตติกรรมประกาศ

โครงงานพัฒนาระบบการตรวจหาทิศทาง และ ความยาวโฟกัสของกล้องวิดีโอเพื่อการแทรกวัตถุเสมือน จะไม่สามารถสำเร็จลุล่วงไปได้ด้วยดีถ้าขาดความร่วมมือจากบุคคลหลายฝ่าย ดังนี้

ขอขอบคุณพระคุณ ผศ.ดร. นพพร โชติกกำธร ที่ได้ให้ทั้งโอกาส และ คำแนะนำในการพัฒนาโครงงานนี้ รวมถึง ดร.ธนารัตน์ ชลิดาพงศ์ ที่ได้ให้คำแนะนำเบื้องต้นเกี่ยวกับการใช้งานชุดพัฒนา Microsoft Vision SDK

ขอขอบคุณน้องๆ ที่ Multimedia and Virtual Lab อันได้แก่ คุณ วรวิทย์ วีระพันธ์ และ คุณ พรชัย กาญจนสุภัคร์ ที่ได้สละเวลาอันมีค่าในการช่วยทำให้งานสำเร็จลุล่วงไปได้ด้วยดี

และสุดท้ายที่มิอาจลืมที่จะกล่าวถึง คือกำลังใจจาก บิดา มารดา และ ภรรยา ของข้าพเจ้า

วิชิต นันทรัตนพงศ์

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญภาพ	VII
สารบัญตาราง	X
บทที่	
1. บทนำ	1
1.1 ความเป็นมา	1
1.2 วัตถุประสงค์ของการพัฒนาระบบงาน	2
1.3 ขอบเขตของการพัฒนาระบบงาน	2
1.4 ประโยชน์คาดว่าจะได้รับจากโครงการ	3
2. ทฤษฎีพื้นฐานที่ใช้ในการสร้าง Augmented reality application	4
2.1 Augmented Reality	4
2.2 Monitor-based Augmented Reality system	9
2.3 หลักการเขียนภาษาเชิงวัตถุ และ การสร้างโปรแกรมด้วย MS Visual C++	11
2.4 Microsoft Vision System Development Kit	12
2.4.1 โครงสร้างของ MS Vision SDK.	16
2.4.2 คลาส CVisImage	17

2.4.2.1 Pixel และ Image Types 19

2.4.2.2 การสร้าง CVisImage Object 23

2.4.2.3 Operations พื้นฐานของคลาส CVisImage 24

2.4.3 การแสดงข้อมูลภาพทางจอมอนิเตอร์ 25

2.4.3.1 แสดงผลโดยใช้ Windows HDC 25

2.4.3.2 แสดงผลโดยใช้คลาส CVisPane และ VisDisplayImage 26

2.4.3.3 แสดงผลโดยการสร้าง Windows bitmap 26

2.4.4 การจัดการติดต่อกับอุปกรณ์ต่อพ่วง 27

2.5 การตรวจหาตำแหน่งของกล้องด้วยอุปกรณ์ ISOTRAK®II 28

2.6 การตรวจหาความยาวโฟกัสโดยใช้วัตถุทรงกลม 31

2.6.1 ความยาวโฟกัส (focal length) 31

2.6.2 การใช้วัตถุทรงกลมสีสำหรับการหาค่าความยาวโฟกัส 33

2.7 OpenGL และ การสร้างรูปสามมิติ 36

3. การออกแบบและการสร้างระบบ 42

3.1 การ Capture Frame picture 43

3.2 การตรวจหาวัตถุที่ใช้ในการ calibrate 45

3.3 การหาเส้นผ่านศูนย์กลางของรูปวงกลมที่เกิดจากการ project ของวัตถุทรงกลม 46

3.4 การดึงค่าตำแหน่งของกล้องวิดีโอด้วยอุปกรณ์ ISOTRAK®II 48

3.5 การคำนวณหาความยาวโฟกัส 50

3.6 การสร้าง การกำหนดตำแหน่ง และ ขนาดของวัตถุสังเคราะห์สามมิติใน OpenGL 52

4. ผลการทดลอง 54

4.1 ผลการทดลองเมื่อมีการเปลี่ยนเฉพาะความยาวโฟกัสเพียงอย่างเดียว 55

4.2 แสดงผลการทำงานเมื่อสั่งให้ระบบ tracker ทำงาน 60

5. สรุปและแนวทางในการพัฒนาโปรแกรมในอนาคต 65

บรรณานุกรม 67

ภาคผนวก 68

ภาคผนวก ก 69

ทรัพยากรที่ใช้ในการพัฒนาโปรแกรม 69

การติดตั้งโปรแกรม MS Vision SDK 69

ภาคผนวก ข 75

แผนผังคลาสภายใน MS Vision SDK 75

สารบัญภาพ

หน้า

ภาพที่

1. แสดงภาพโต๊ะทำงานจริงที่ถูกซ้อนด้วย โคมไฟเสมือน และ เก้าอี้เสมือนสองตัว 4

2. แสดงอุปกรณ์ Head-Mounted Displays 5

3. แสดงการสร้างรูปมะเร็งเต้านมเทียมซ้อนทับลงบนอุปกรณ์การฝึกการผ่าตัด 6

4. แสดงการใส่อุปกรณ์ Head-Mounted Displays เพื่อช่วยในการซ่อม laser printer 6

5. แสดงมุมมองผ่านผู้ที่สวมใส่อุปกรณ์ Head-Mounted Displays เพื่อการซ่อม laser printer 7

6. แสดงคำอธิบายถึงวัตถุต่างๆในภาพโดยซ้อนลงไปในภาพที่ปรากฏผ่านระบบ Augmented Reality 7

7. คำอธิบายเกี่ยวกับชิ้นส่วนเครื่องจักรผ่านการมองของผู้ใช้เมื่อผู้ใช้ชี้ไปที่ชิ้นส่วนที่ต้องการ 8

8. แสดงองค์ประกอบของระบบ monitor-based Augmented Reality system 9

9 แสดงส่วนประกอบของระบบ Augmented Reality และ coordinates ของวัตถุในโลกจริง และ ในโลกเสมือนที่ซ้อนทับกัน 10

10. Windows Driver Model (WDM) 14

11. การสร้าง applications ที่ไม่สนับสนุน WDM 14

12. แสดงโครงสร้างของ MS Vision SDK 16

13. โครงสร้างของการเก็บภาพใน Vision SDK 18

14. แสดงส่วนประกอบของระบบ ISOTRAK II tracking system 29

15. แสดง ISOTREK II Controls/Connections 29

16. แสดงอุปกรณ์ Transmitter 30

17. แสดงอุปกรณ์ Receiver 30

18. แสดงค่าความยาวโฟกัสที่มีผลต่อขนาดของวัตถุบนภาพที่ project บน Image plane 31

19. แสดงผลของระยะห่างกับวัตถุที่จะ project ลงบน Image plane 32

20. ลักษณะทาง perspective ที่ปรากฏซึ่งทำให้เกิดความลึกของภาพ 33

21. แสดง Flow Diagram ของกระบวนการ Image Analysis 34

22. การหาความยาวโฟกัสโดยใช้ภาพวงกลมที่ได้จากการ project วัตถุทรงกลม 35

23. รูปวัตถุสังเคราะห์สามมิติที่สร้างโดย Open GL 37

24. ความสัมพันธ์ระหว่าง Field of view กับความยาวโฟกัส 38

25. ความยาวโฟกัสเปลี่ยนแปลงสัมพันธ์กับ Filed of view 40

26. Block Diagram ของระบบงาน 42

27. แสดงหน้าต่าง Vision AppWizard 43

28. แสดงหน้าต่างของ application 44

29. แสดงวัตถุทรงกลมสีน้ำเงินที่โปรแกรมตรวจสอบได้ 46

30. แสดงอัลกอริทึมในการหาเส้นผ่านศูนย์กลางของวงกลมในภาพ 47

31. เมื่อการหาครั้งแรกล้มเหลวให้ทำการแบ่งพื้นที่ในการหาให้ละเอียดยิ่งขึ้น 48

32. แสดงการเรียกค่าจากอุปกรณ์ ISOTRAK II 50

33. หน้าต่างในการทำงานของ application 54

34. เรียกเมนู Set Keyboard สำหรับการกำหนดขนาด และ ตำแหน่งของวัตถุสังเคราะห์ให้เหมาะสม 55

35. การจัดวางวัตถุสังเคราะห์ลงในภาพ 56

36. การกำหนดค่าความยาวโฟกัสเริ่มต้น 57

37. เลือกเมนูให้ application มีการตอบสนองต่อการเปลี่ยนแปลงความยาวโฟกัส 58

38. ผลที่ได้เมื่อมีการเปลี่ยนแปลงความยาวโฟกัสของกล้องวิดีโอ 59

39. ตรวจหาระยะระหว่างวัตถุที่ใช้ในการ calibrate กับ จุด origin 60

40. บันทึกระยะห่างระหว่างวัตถุที่ใช้ในการ calibrate กับ จุด origin 61

41. ทำการบันทึกค่าระยะห่างระหว่าง กล้อง และ วัตถุเริ่มต้น 62

42. ฟังก์ชั่นตรวจสอบระยะห่าง 63

43. เปิดระบบ Tracker 63

44. ผลลัพธ์ที่ได้จากการเปิดระบบ Tracker 64

45. เลือก Systems ใน control panel 70

46. กำหนด Path โดยเลือกจาก Environment Variables 70

47. กำหนด Path โดยเลือกจาก Environment Variables 71

48. เลือก Options จากเมนู Tools ใน work space ของ VisSDK 72

49. กำหนด "inc" path ที่ include files 72

50. แสดงการ Set Active Configuration 73

51. แสดงการ Rebuild ALL-Win32 Debug 74

สารบัญตาราง

หน้า

ตารางที่

1. แสดงรายละเอียดของ pixel แต่ละชนิดในกลุ่ม Gray Scale Pixel Types 19

2. แสดงรายละเอียดของ pixel แต่ละชนิดในกลุ่ม RGBA Color Pixel Types 21

3. แสดงรายละเอียดของ pixel แต่ละชนิดในกลุ่ม YUVA Color Pixel Types 22

4. เปรียบเทียบผลเมื่อมีการเปลี่ยนแปลงความยาวโฟกัส 59

บทที่ 1
บทนำ

1.1 ความเป็นมา

Augmented Reality คือการผสมวัตถุเสมือน (virtual object) ที่ได้จากการสังเคราะห์โดยเครื่องคอมพิวเตอร์ลงไปในภาพสภาพแวดล้อมจริงที่มนุษย์สามารถมองเห็นได้เสมือนวัตถุนั้นมีอยู่ในสิ่งแวดล้อมนั้นจริงๆ ซึ่งมีการนำไปใช้ประโยชน์กันอย่างแพร่หลาย เช่น ภาพโฆษณาที่ซ้อนทับลงบนพื้นสนามฟุตบอลในระหว่างการแข่งขัน การสร้างภาพอวัยวะซ้อนทับภาพที่เกิดจากการทำ CT scan ที่ใช้ในทางการแพทย์ เป็นต้น

จุดมุ่งหมายสำคัญสำหรับการซ้อนทับวัตถุเสมือนในงาน Augmented Reality คือจะทำอย่างไรให้ผู้ใช้งาน หรือ ผู้ดูภาพที่ผ่านการซ้อนทับแล้วไม่เห็นความแตกต่างระหว่างภาพเสมือนกับภาพจริงที่ซ้อนทับกันได้ กล่าวคือดูเสมือนหนึ่งเป็นภาพเดียวกัน เพราะฉะนั้นนอกจากจะต้องใช้คอมพิวเตอร์ประสิทธิภาพสูงสำหรับการสร้างวัตถุเสมือนสามมิติเพื่อให้ได้ภาพเสมือนที่ดูสมจริงแล้ว วัตถุเสมือนจะต้องสามารถตอบสนองต่อสิ่งแวดล้อมในแบบทันทีทันใด (real-time response) เช่น การเปลี่ยนขนาดและมุมมองของวัตถุเสมือนเมื่อกล้องวิดีโอมีการเคลื่อนที่ไปมา และ การเปลี่ยนขนาดเมื่อมีการเปลี่ยนระยะโฟกัสของกล้อง เป็นต้น

จากปัญหาดังกล่าวโดยเฉพาะอย่างยิ่งในงานที่เกี่ยวกับการซ้อนภาพเสมือนลงไปในการบันทึกภาพวิดีโอ การที่จะทำให้ได้ภาพมีความสมบูรณ์จึงต้องมีระบบตรวจสอบตำแหน่งของกล้องวิดีโอ และ ระบบตรวจสอบความยาวโฟกัส (focal length) มาช่วยในการกำหนดขนาด และ คุณสมบัติในทาง perspective ของภาพเสมือนที่จะปรากฏบนจอภาพ ระบบช่วยเหลือนี้ในงาน Augmented Reality เรียกว่า Calibration System การCalibrate (ทำกับ image source ส่วนใหญ่คือ กล้องถ่ายรูป หรือ กล้องถ่ายวิดีโอ) ทำเพื่อหาปัจจัยที่เกี่ยวข้องกับกล้องสองอย่างคือ ปัจจัยภายนอก (extrinsic parameter) ที่เกี่ยวข้องอันได้แก่ ระยะห่างระหว่างกล้องกับวัตถุจริงในภาพ เช่น คน หรือสิ่งของที่อยู่ภายในภาพ เป็นต้น และ ปัจจัยภายใน (intrinsic parameter) ที่เกี่ยวข้องอันได้แก่ จุดศูนย์กลางของเซ็นเซอร์ที่จอรับภาพของกล้องจริง (optical center) ความยาวโฟกัส (focal length) เป็นต้น แต่เนื่องจาก optical center มีขนาดคงที่ในกล้องตัวหนึ่งๆในระบบที่สร้างนี้จึงไม่ได้ทำการหา ในขณะที่ระยะห่างระหว่างกล้องกับวัตถุจริงมีการเปลี่ยนตลอดเวลาเมื่อกล้องวิดีโอมีการเคลื่อน

ที่ และ เมื่อผู้ใช้กล้องวิดีโอมีการปรับความยาวโฟกัส (zoom out หรือ zoom in) ความยาวโฟกัสก็จะมีการเปลี่ยนแปลงไปด้วย จึงถือสองปัจจัยสำคัญที่ระบบจะต้องสามารถหาได้โดยอัตโนมัติ การหาระยะห่างระหว่างกล้องกับวัตถุจริง ทำได้โดยการใช้เซ็นเซอร์ติดกับตัวกล้องโดยครั้งแรกให้ตั้งค่าระยะห่างระหว่างกล้องกับวัตถุเมื่อเริ่มต้นให้เป็นตำแหน่ง origin (x_0 , y_0 , z_0) เมื่อกล้องมีการเคลื่อนที่ ระบบจะทำการหาค่าตำแหน่งเทียบกับตำแหน่ง origin ส่วนการหาความยาวโฟกัส หาโดยการใช้วัตถุทรงกลม (เพราะวัตถุทรงกลมมีความคุณลักษณะสมมาตรทุกทิศทาง) โดยเมื่อกล้องถ่ายภาพวัตถุทรงกลมก็จะเกิดรูปวงกลมซึ่งรัศมีของวงกลมที่เกิดขึ้นครั้งแรกจะตั้งให้เป็นค่าเริ่มต้นจากนั้นเมื่อกล้องมีการเคลื่อนที่ และ มีการปรับความยาวโฟกัส รัศมีของวงกลมจะมีการเปลี่ยนแปลงซึ่งการเปลี่ยนนี้เกิดจากระยะห่างระหว่างกล้องกับวัตถุจริงที่เปลี่ยนแปลงกับความยาวโฟกัส ดังนั้นเมื่อทราบระยะห่างระหว่างกล้องกับวัตถุจริง เราก็สามารถคำนวณหาความยาวโฟกัสได้ เมื่อได้ค่าทั้งสองเราจึงสามารถนำวัตถุสังเคราะห์ที่สร้างไว้แล้วและต้องการนำไปซ้อนทับในภาพจริงไปทำการ transformation เพื่อให้ได้ภาพที่ได้ดูเหมือนจริงมากที่สุด

1.2 วัตถุประสงค์ของการพัฒนาระบบงาน

1. เพื่อสร้างโปรแกรมประยุกต์ที่มีความสามารถในการ capture ภาพต่อเนื่องเป็นเฟรมในแบบทันทีทันใด (real time) บนเครื่องคอมพิวเตอร์ทั่วไปที่ใช้ระบบปฏิบัติการวินโดว์
2. เพื่อสร้างโปรแกรมประยุกต์ที่สามารถนำภาพที่ได้ในแต่ละเฟรมจากข้อ 1 ไปทำขบวนการทาง image processing เพื่อหาความยาวโฟกัสที่มีการเปลี่ยนแปลงของกล้องวิดีโอในขณะที่กำลังบันทึกภาพในแบบทันทีทันใด (real time)
3. เพื่อใช้งานอุปกรณ์ ISOTRAK®II เป็นอุปกรณ์ช่วยในการหาระยะห่างของกล้องกับวัตถุ เพื่อที่จะทำให้โปรแกรมประยุกต์มีการทำงานที่รวดเร็วยิ่งขึ้น
4. เพื่อนำข้อมูลจากที่ได้จากข้อ 3 และ ข้อ 4 ไปทำการสร้างวัตถุสังเคราะห์ที่มีลักษณะตอบสนองต่อทิศทาง ระยะห่าง และ ความยาวโฟกัส ของกล้องวิดีโอในแบบอัตโนมัติ
5. เพื่อสร้างโปรแกรมประยุกต์ที่สามารถนำผลลัพธ์ที่ได้จาก ข้อ 4 ไปทำการแทรกลงในภาพที่ได้จาก ข้อ 1 แล้วแสดงผลทางหน้าจอมอนิเตอร์ในแบบต่อเนื่อง ให้ดูเสมือนหนึ่งว่าวัตถุนั้นมีอยู่ในภาพจริงมิได้ถูกสร้างขึ้น และ ทั้งหมดต้องในสภาพแวดล้อมในแบบทันทีทันใด (real time)

1.3 ขอบเขตของการพัฒนาระบบงาน

Augmented reality เป็นงานที่เกี่ยวข้องกับหลักการ และ ทฤษฎีหลายอย่าง เช่น Image Processing , Computer Graphic เป็นต้น ในการสร้างระบบงานนี้มีขอบเขตดังนี้

1. ศึกษาการเขียนโปรแกรมในแบบ การเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming) ในที่นี้ใช้ภาษา C++ ในการพัฒนาระบบ
2. ศึกษาการใช้งานบน Microsoft Visual C++ ในชุดงาน Microsoft Visual Studio
3. ศึกษาการใช้ Microsoft Vision System development Kit (Vision SDK) ในการสร้าง application สำหรับการ Capture ภาพที่ได้จากกล้องวิดีโอ การซ้อนภาพเสมือนกับภาพจริง และ การแสดงภาพที่ได้ทางหน้าจอมอนิเตอร์
4. ศึกษาหลักการทาง image processing เกี่ยวกับ feature extraction , geometric transformation

1.4 ประโยชน์คาดว่าจะได้รับจากโครงการ

1. ได้ฝึกฝนการใช้ภาษาในแบบภาษาเชิงวัตถุในการสร้าง application ที่สามารถนำไปใช้งานได้
2. ได้นำทฤษฎีด้าน image processing และ computer vision ไปใช้ในการสร้าง application ที่สามารถทำงานได้จริง
3. ได้ทดลอง application ในแบบ Augmented reality ซึ่งจะนำไปสู่การสร้าง application ที่ใช้งานด้านอื่นๆต่อไป

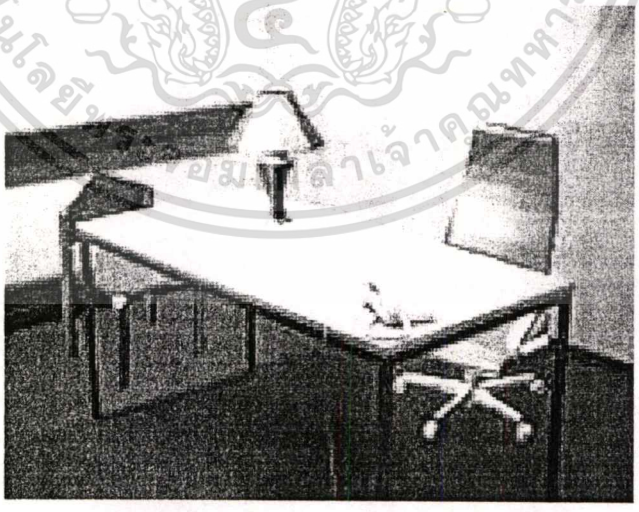
บทที่ 2

ทฤษฎีพื้นฐานที่ใช้ในการสร้าง Augmented reality application

ในบทนี้จะกล่าวถึงหลักการ และ ทฤษฎีที่ได้นำมาใช้ในการพัฒนาโปรแกรมประยุกต์

2.1 Augmented reality

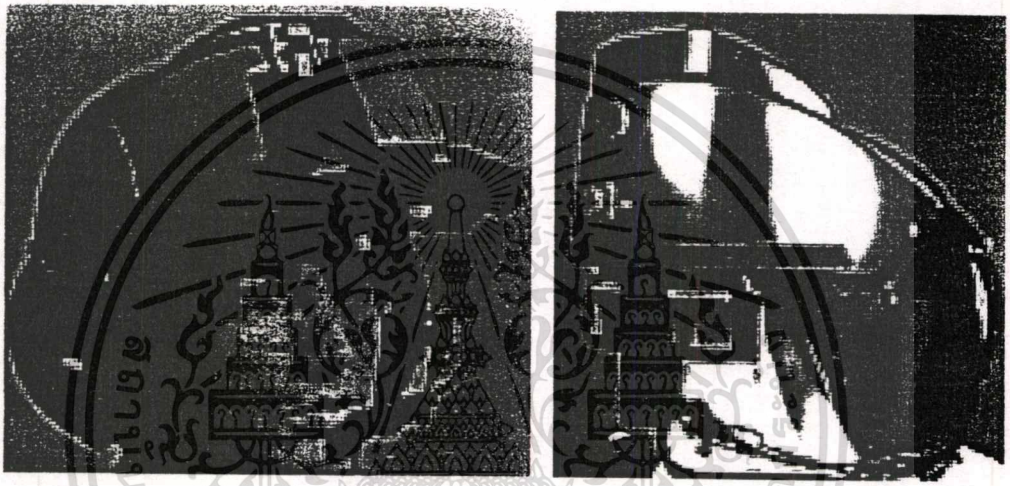
Augmented reality เป็นวิทยาการแขนงหนึ่งที่มีผู้ศึกษาควบคู่มากับ Virtual Reality ในขณะที่ Virtual Reality คือการที่จะพยายามที่จะผสมวัตถุจริง (real object) โดยเฉพาะผู้ใช้งานให้ลงไปในสภาวะแวดล้อมที่สังเคราะห์ (synthetic environment) ขึ้นโดยคอมพิวเตอร์ เช่น โครงการมหาวิทยาลัยเสมือน (Virtual University) เป็นต้น แต่ Augmented reality กลับพยายามทำในสิ่งตรงกันข้ามคือพยายามผสมวัตถุสังเคราะห์ (synthetic object) ลงไปในสภาพแวดล้อมจริง หรือ คือการทำให้ผู้ใช้งานที่มองเห็นสภาพแวดล้อมจริง (real world) ในขณะที่มีวัตถุสังเคราะห์ซ้อนทับหรือ แทรกตัวอยู่ อาทิเช่น ในงานภาพยนตร์เรื่อง “Who Framed Roger Rabbit” ที่พยายามสร้างตัวการ์ตูนที่ปรากฏอยู่ในฟิล์มภาพยนตร์เสมือนหนึ่งตัวการ์ตูนนั้นมีอยู่จริง เป็นต้น



ภาพที่ 1. แสดงภาพโต๊ะทำงานจริงที่ถูกซ้อนด้วย โคมไฟเสมือน และ เก้าอี้เสมือนสองตัว

นักวิจัยหลายท่านที่ทำงานด้าน Augmented reality มักจะจำกัดการใช้งานที่เกี่ยวข้องกับการใช้อุปกรณ์ Head-Mounted Displays แต่ขอบเขตการใช้งานของ Augmented reality มีมากกว่า จึงอาจให้คำจำกัดความถึงลักษณะเฉพาะของ Augmented reality ได้ดังนี้

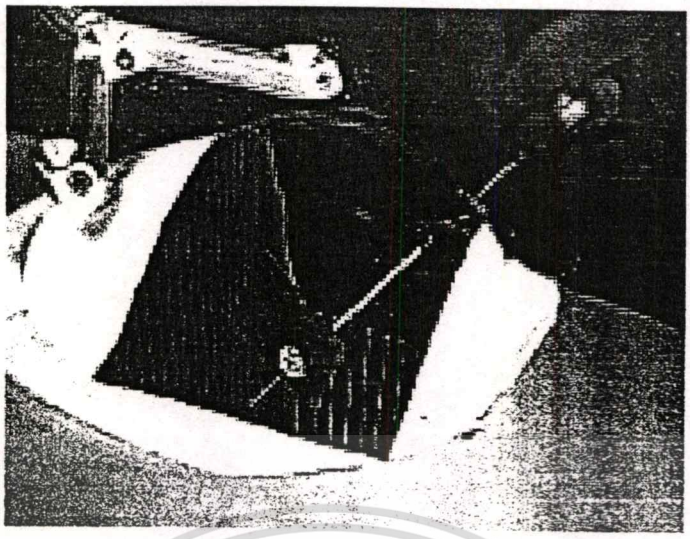
1. เป็นการทำงานที่เกี่ยวข้องกับการผสมระหว่างภาพจริง และ ภาพเสมือน
2. applications ที่สร้างขึ้นต้องมีการตอบสนองในแบบทันทีทันใด (real time)
3. วัตถุสังเคราะห์ที่สร้างต้องมีลักษณะเป็นสามมิติ



ภาพที่ 2. แสดงอุปกรณ์ Head-Mounted Displays

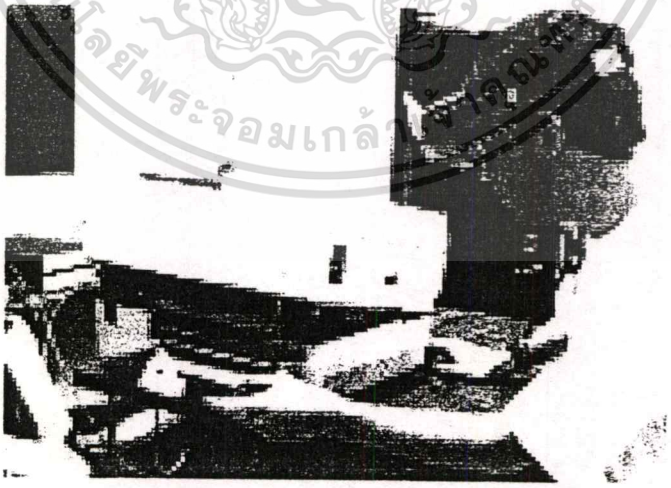
ด้วยคำจำกัดความทั้งสามข้อทำให้ภาพของงาน Augmented reality มีความชัดเจนมากขึ้น การสร้าง applications และการใช้งานในด้าน Augmented reality มีมากมาย และ หลากหลายชนิด ดังอาจแบ่งได้ดังนี้

1. ด้านการแพทย์ เช่น การทำภาพจำลองเพื่อช่วยในการฝึกฝนในห้องผ่าตัด หรือ อย่างเช่น การทำภาพจำลองซ้อนทับลงในภาพที่ได้จากการทำ CT Scan เป็นต้น เนื่องจากคุณสมบัติในแง่ที่ว่าวัตถุสังเคราะห์ที่ปรากฏมีลักษณะเป็นสามมิติ ดังนั้น แพทย์ที่ใช้งานสามารถมีอิสระในการมองในมุมต่างๆได้โดยไม่จำกัด

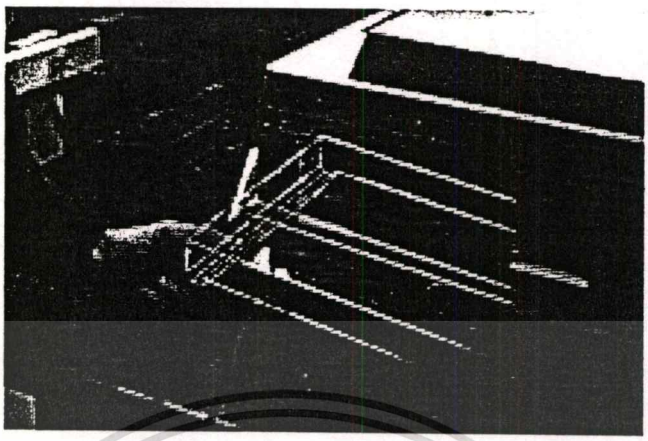


ภาพที่ 3. แสดงการสร้างรูปมะเร็งเต้านมเทียมซ้อนทับลงบนอุปกรณ์การฝึกการผ่าตัด

2. ด้านการผลิต และ การดูแลซ่อมแซมเครื่องจักร ประโยชน์ที่จะได้ในงานนี้คือ ในการซ่อมแซมเครื่องจักรที่มีความซับซ้อนการใช้คู่มือที่เป็นสองมิติอยู่ในกระดาษอาจทำให้ช่างที่ดูแลเสียเวลาในการทำความเข้าใจมาก เพราะฉะนั้นถ้าสามารถแสดงข้อมูล (ข้อความ หรือ รูป) ลงไปบนอุปกรณ์จริงก็จะทำให้การทำงานมีประสิทธิภาพมากยิ่งขึ้น



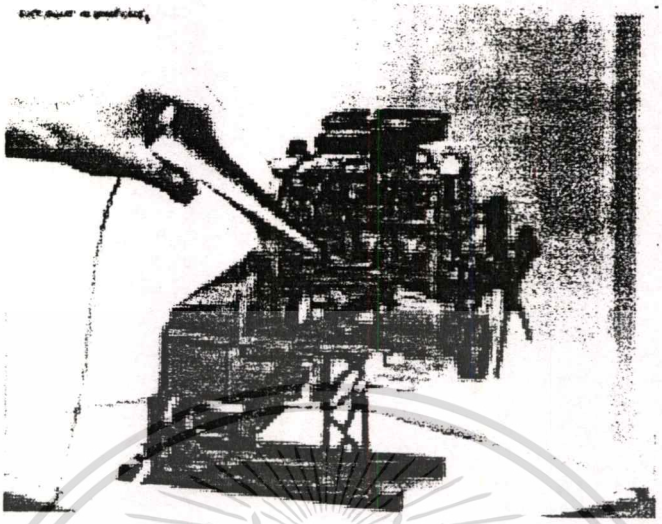
ภาพที่ 4. แสดงการใส่อุปกรณ์ Head-Mounted Displays เพื่อช่วยในการซ่อม laser printer



ภาพที่ 5. แสดงมุมมองผ่านผู้ที่สวมใส่อุปกรณ์ Head-Mounted Displays เพื่อการซ่อม laser printer

3. Annotation และ Visualization ปัจจุบันมีการใช้งานอุปกรณ์มือถือ (hand-held display) อยู่มากมาย เช่น พนักงานใช้อุปกรณ์ช่วยบันทึกการใช้ไฟฟ้าตามบ้าน เป็นต้น การนำเอา Augmented Reality มาผสมทำให้การใช้งานมีประสิทธิภาพสูงยิ่งขึ้น

ภาพที่ 6. แสดงคำอธิบายถึงวัตถุต่างๆในภาพโดยซ้อนลงไปในภาพที่ปรากฏผ่านระบบ Augmented Reality

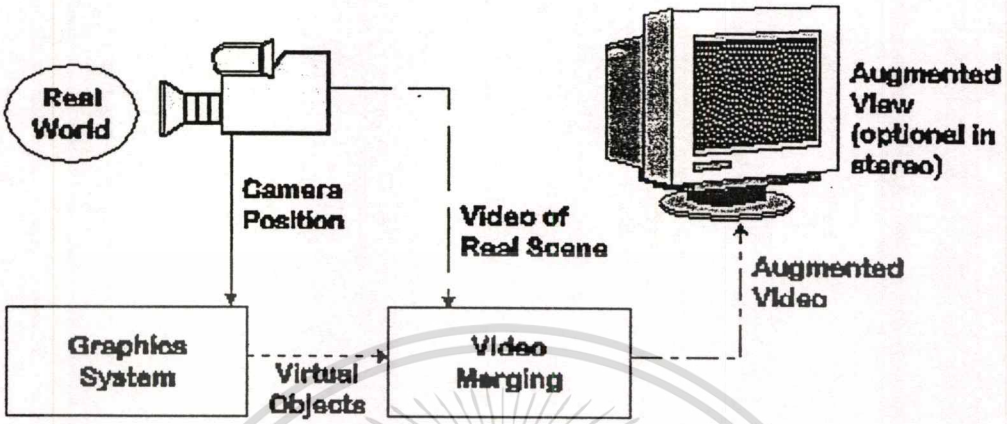


ภาพที่ 7. คำอธิบายเกี่ยวกับชิ้นส่วนเครื่องจักรผ่านการมองของผู้ใช้เมื่อผู้ใช้ชี้ไปที่ชิ้นส่วนที่ต้องการ

4. ด้านบันเทิง มีการใช้งานอย่างกว้างขวาง เช่นงานด้าน visual effect ที่ปรากฏในภาพยนตร์ ทั้งใน และ ต่างประเทศ มีการนำวัตถุสังเคราะห์ที่สร้างเครื่องคอมพิวเตอร์ซึ่งมักเป็นวัตถุที่ไม่มีอยู่ หรือ ไม่สามารถนำมารวมในภาพหลักได้ในเวลาเดียวกัน หรือ อย่างเช่น การรายงานอากาศที่มีการซ้อนภาพกราฟิกลงไปด้วยในขณะที่มีการรายงานสด
5. ด้านการทหาร มีการใช้จำลองแบบฝึกในทางการทหาร เช่น การจำลองการบิน (flight simulator) เป็นต้น

Augmented Reality อาจแบ่งด้วยอุปกรณ์ที่ใช้ในการนำเสนอได้เป็นสองประเภท ได้แก่ การนำเสนอภาพผ่านอุปกรณ์ Head-Mounted Displays (HMD-based Augmented Reality system) และ การนำเสนอภาพผ่านจอมอนิเตอร์ (monitor-based Augmented Reality system) ในหัวข้อต่อไปจะกล่าวถึง หลักการ และ องค์ประกอบของการพัฒนา monitor-based Augmented Reality system ซึ่งใช้ในการทดลองสร้างระบบในการพัฒนาระบบงานชิ้นนี้

2.2 Monitor-based Augmented Reality system



ภาพที่ 8. แสดงองค์ประกอบของระบบ monitor-based Augmented Reality system

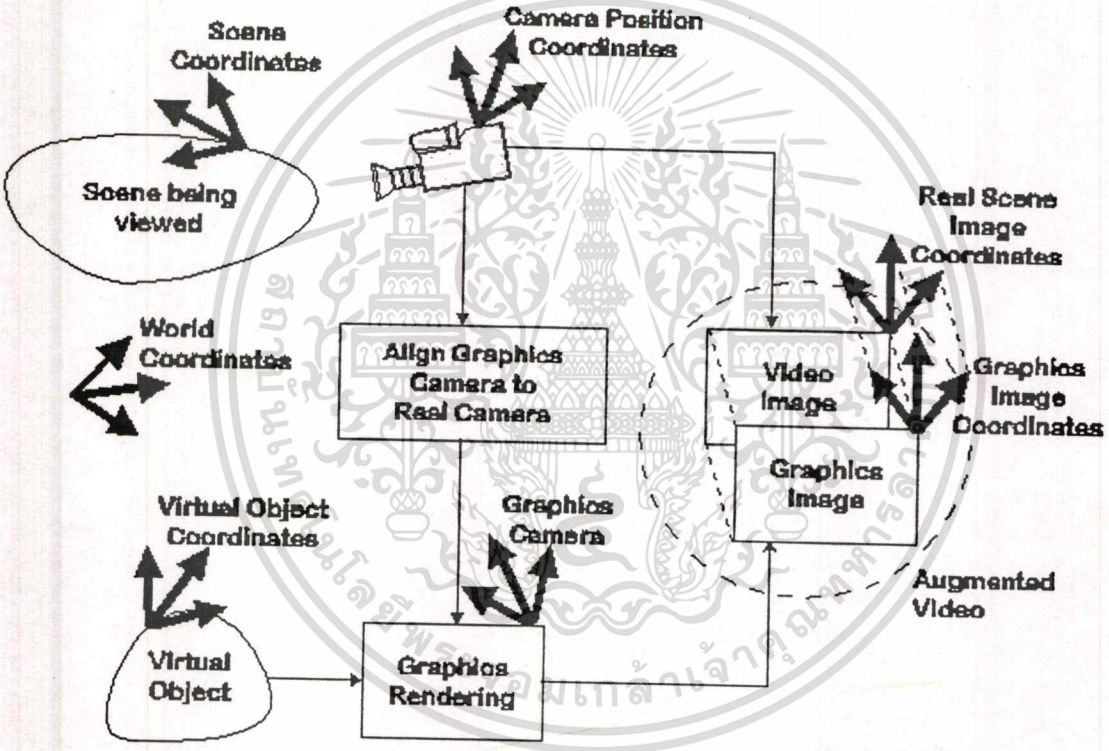
ระบบ Augmented Reality ที่แสดงภาพผ่านทางจอมอนิเตอร์มีส่วนประกอบตามรูปที่ 8 คือ

- กล้องวิดีโอเป็นส่วนที่นำสัญญาณภาพจากสิ่งแวดล้อมจริง
- ระบบตรวจหาตำแหน่งของกล้องวิดีโอ (Camera Position) เพื่อบอกตำแหน่งของกล้องให้ระบบสังเคราะห์ภาพกราฟิก (หรือ วัตถุเสมือน)
- ระบบการสังเคราะห์ภาพกราฟิก เพื่อนำเสนอไปผสมรวมกับภาพจริง (Graphics System) ระบบนี้ยังมีส่วนที่ทำการหาขนาดระยะโฟกัสของกล้องวิดีโอในแบบ real-time ด้วย
- ระบบการผสานภาพที่ได้จากวิดีโอกับภาพที่ได้จากการสังเคราะห์ขึ้น (Video Merging) ผลที่ได้คือ ภาพวิดีโอที่นำเสนอทางจอมอนิเตอร์ (Augmented Video)
- มอนิเตอร์ที่แสดงภาพที่ผ่านขบวนการทั้งหมดแล้ว

จุดมุ่งหมายสำคัญคือ จะทำอย่างไรให้ภาพที่ผ่านระบบ Augmented Reality จะต้องทำให้ผู้ที่ดูภาพไม่สามารถแยกระหว่างภาพจริงกับวัตถุสังเคราะห์ในภาพได้ และ วัตถุสังเคราะห์ต้องมีการตอบสนองในแบบทันทีทันใด (real-time) เสมือนหนึ่งเป็นวัตถุจริงที่มีอยู่ในสิ่งแวดล้อม อาทิเช่น สามารถเปลี่ยนตามระยะห่าง หรือ ตามระยะโฟกัสของกล้องที่เปลี่ยนไป สามารถตอบสนองต่อการถูกบังจากวัตถุจริง เป็นต้น

การบันทึกภาพโดยอุปกรณ์บันทึกภาพ อาทิเช่น กล้องถ่ายวิดีโอ กล้องถ่ายนิ่ง และ digital camera เป็นการ transform จากโลกสามมิติ (3D world) เป็น สองมิติที่จอรับภาพ (2D image plane)

อาจเรียกว่าอุปกรณ์บันทึกภาพทำหน้าที่ perspective projection ซึ่งปัจจัย (parameter) ของตัวอุปกรณ์ซึ่งแบ่งเป็นสองอย่างคือ ปัจจัยภายใน (intrinsic parameters) ได้แก่ ความยาวโฟกัส (focal length) ความเพี้ยนของเลนส์ (lens distortion) และ ปัจจัยภายนอก (extrinsic parameter) ได้แก่ ตำแหน่งของกล้อง (position) มุมกล้อง (pose) ซึ่งทั้งหมดรวมกันเป็นตัวกำหนดลักษณะของภาพที่จะ project ลงใน image plane ซึ่งก็จะเป็นข้อมูลที่ระบบสังเคราะห์วัตถุใช้ในการสร้าง และ ผสมวัตถุลงในภาพด้วย



รูปที่ 9 แสดงส่วนประกอบของระบบ Augmented Reality และ coordinates ของวัตถุในโลกจริง และ ในโลกเสมือนที่ซ้อนทับกัน

ในบทต่อไปจะกล่าวถึงวิธีการหาตำแหน่ง และ มุมกล้องของกล้องวิดีโอซึ่งอาศัยเซ็นเซอร์ในการตรวจหาตำแหน่งของกล้องทั้งสามแกนคือ X Y และ Z รวมทั้งการตรวจสอบมุมกล้องทั้งสามแกนที่เปลี่ยนแปลงนับเป็นจำนวนองศา

2.3 หลักการเขียนภาษาเชิงวัตถุ และ การสร้างโปรแกรมด้วย MS Visual C++

Object Oriented Programming (OOP) เป็นแนวคิดในการเขียนโปรแกรมแบบหนึ่ง โดยไม่มุ่งเน้นไปในการเขียนโปรแกรมเพียงอย่างเดียว แต่เป็นการมองภาพรวมของสิ่งที่อยู่รอบๆ ตัวเรา โดยให้คำจำกัดความว่า "วัตถุแต่ละอย่างนั้น ต่างก็จะมีลักษณะและวิธีการใช้งานเป็นของตัวมันเอง" ประโยคนี้มีความหมายว่า วัตถุแต่ละชนิด หรือ แต่ชิ้น ต่างก็มีรูปร่างลักษณะ และ การใช้งานที่แตกต่างกันออกไป เราจะเรียกคุณลักษณะของวัตถุว่า Attribute และ เราจะเรียกวิธีการใช้งานวัตถุว่า Method ยกตัวอย่างเช่น จักรยานคันหนึ่งมีสีแดง สามารถใช้ขับขี่ได้ สีแดงจัดเป็น Attribute ในขณะที่การใช้งานขับขี่ถือเป็น Method เป็นต้น

จะเห็นได้ว่าแนวคิดในแบบ OOP เป็นแนวคิดที่คล้ายธรรมชาติของสิ่งของสิ่งหนึ่งซึ่งเราสามารถแบ่งแยกสิ่งต่างๆ ออกเป็นประเภทๆ ได้ ถ้าเราได้นำเอาแนวคิดของ OOP มาใช้ในการเขียนโปรแกรม และ การจัดการข้อมูล เราจะพบว่าโปรแกรม หรือ ฟังก์ชั่น จะมีความเป็นอิสระต่อกันอย่างเห็นได้ชัด อธิบายง่ายๆก็คือ โปรแกรม หรือ ฟังก์ชั่นแต่ละตัวถึงแม้จะมาจากที่เดียวกันแต่มันสามารถทำงานในคนละหน้าที่ เก็บข้อมูลคนละค่าได้ โดยจะไม่มายุ่งเกี่ยวกันแต่อย่างใด

คลาสในแนวคิดของ OOP คือกลุ่มของวัตถุที่คุณลักษณะพื้นฐาน และ มีฟังก์ชั่นพื้นฐานเหมือนกัน เช่น คลาสคน ไม่ว่าคนนั้นจะมีชื่อใด มีอาชีพใด หรือ มีส่วนสูงเท่าไหร่ก็จัดอยู่ในคลาสคนซึ่งจะมีคุณลักษณะ และ ฟังก์ชั่นพื้นฐานแบบเดียวกัน ผู้เขียนโปรแกรมไม่สามารถนำคลาสไปใช้งานได้โดยตรง ต้องมีการสร้าง object (หรือเรียกว่า Instance ก็ได้) ซึ่ง object ที่สร้างจากคลาสใดจะมีคุณลักษณะพื้นฐานมาจากคลาสด้วย เพราะฉะนั้นการสร้างโปรแกรมด้วยวิธีการแบบ OOP จะทำให้การสร้างโปรแกรมมีความยืดหยุ่น และ มีประสิทธิภาพสูง

คุณสมบัติพิเศษอีกอย่างหนึ่งที่ทำให้การเขียนโปรแกรมในแบบ OOP คือคุณสมบัติ การสืบทอด (Inheritance) การที่ OOP นำแนวคิดนี้มาใช้ทำให้ผู้เขียนโปรแกรมสามารถนำเอา source code (อาจเป็นคลาส) ที่เขียนไว้แล้วสามารถนำกลับมาใช้ใหม่ หรือ มาเปลี่ยนแปลงแก้ไขให้เหมาะกับงานเฉพาะอย่างเป็นไปอย่างมีประสิทธิภาพมากยิ่งขึ้น (แนวคิด Reusability)

เนื่องจากการเขียนโปรแกรมในแบบ OOP มีความยืดหยุ่นสูง การรักษาความปลอดภัยภายในจึงเป็นสิ่งที่สำคัญ เพราะฉะนั้นการกำหนดระดับการเข้าถึงข้อมูลภายในคลาสจึงเป็นสิ่งสำคัญ ซึ่งเป็นอีกคุณสมบัติอีกอย่างที่มีในแนวคิดแบบ OOP การเข้าถึงข้อมูลภายใน (member variable) หรือ ฟังก์ชั่นภายใน (member function หรืออาจเรียกว่า method) สามารถกำหนดระดับได้สามระดับ คือ

- ระดับ private (default คือถ้าผู้เขียนคลาสไม่ใส่ compiler จะเข้าใจว่าเป็น private ไว้ก่อน) เป็นการป้องกันระดับสูงสุด กล่าวคือ เป็นการป้องกันไม่ให้กระบวนการใดๆ ที่อยู่นอกคลาสเรียกใช้สิ่งที่อยู่ในคลาสไม่ได้ การจะเข้าถึงข้อมูลภายในจะต้องผ่านจากฟังก์ชั่นที่เป็นของคลาสนั้นอนุญาตให้ใช้ได้
- ระดับ public ซึ่งเป็นระดับที่ตรงข้ามกับระดับ private อย่างสิ้นเชิง กล่าวคือ กระบวนการใดๆ ที่อยู่นอกคลาสนั้นสามารถเข้าถึงข้อมูลภายใน หรือ ฟังก์ชั่นภายในได้อย่างเต็มที่
- สุดท้ายคือระดับ protected ก็คล้ายในระดับ private ข้อแตกต่างคือระดับ protected คลาสลูกที่สืบทอดคุณลักษณะจากคลาสแม่สามารถเข้าถึงข้อมูลภายในคลาสแม่ได้โดยตรง

ความสามารถในการกำหนดระดับการเข้าถึงข้อมูลภายในคลาสอาจเรียกว่า คุณสมบัติการห่อหุ้มวัตถุ (Data Hiding)

การเขียนโปรแกรมแบบ OOP ผู้เขียนโปรแกรมสามารถใช้คุณสมบัติของฟังก์ชั่นคอนสตรัคเตอร์ (Constructor) และ ดีสตรัคเตอร์ (Destructor) เพื่อเพิ่มความยืดหยุ่นในกับโปรแกรมได้ ในการสร้างคลาสแต่ละครั้ง เราไม่จำเป็นต้องสร้างคอนสตรัคเตอร์และดีสตรัคเตอร์ทุกครั้งไป การสร้างคอนสตรัคเตอร์และดีสตรัคเตอร์นั้นขึ้นอยู่กับความจำเป็นในแต่ละสถานการณ์และการดำเนินงานของคลาส และ ในคลาสหนึ่งสามารถมีคอนสตรัคเตอร์ได้หลายตัวโดยฟังก์ชั่นคอนสตรัคเตอร์ต้องมีชื่อฟังก์ชั่นเหมือนชื่อของคลาส และ ไม่มีการคืนค่าใดๆออกมาเลย ฟังก์ชั่นคอนสตรัคเตอร์จะถูกเรียกโดยอัตโนมัติเมื่อมีการสร้าง object และ การสร้าง object สามารถผ่านพารามิเตอร์ (หรือไม่ผ่านก็ได้เป็น default) ซึ่ง complier จะเรียกฟังก์ชั่นคอนสตรัคเตอร์ได้ตรงกัน คุณสมบัตินี้เรียกว่า สามารถโอเวอร์โหลดฟังก์ชั่นได้

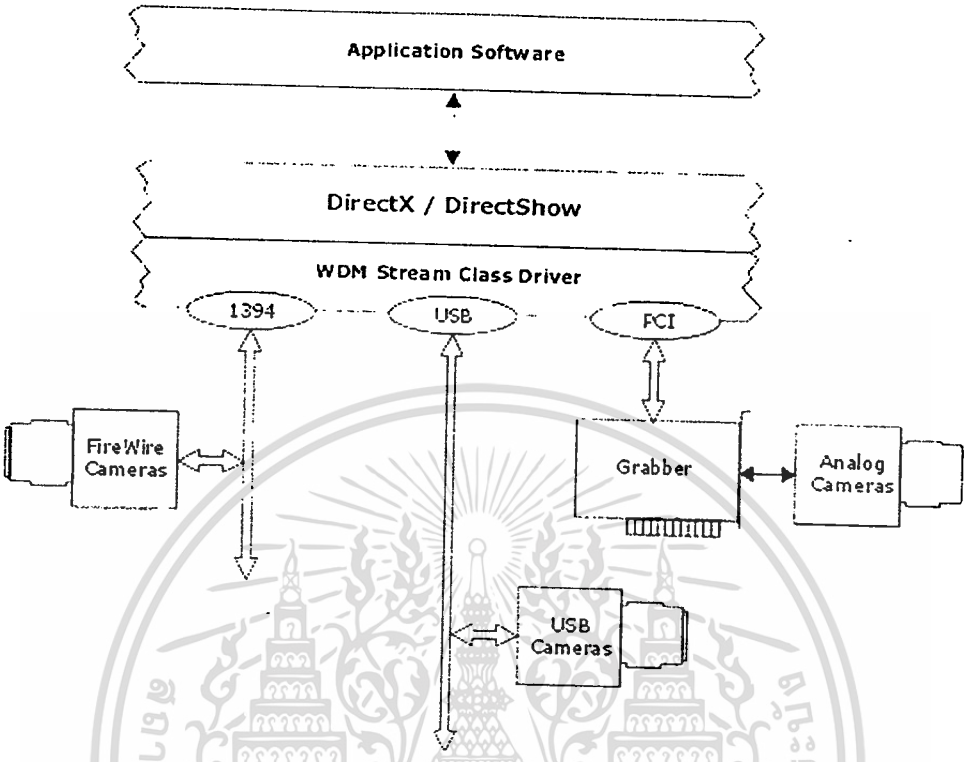
ยังมีคุณสมบัติ Dynamic Binding (Late Binding) และ Polymorphism ซึ่งเป็นคุณสมบัติพิเศษที่ทำให้การเขียนโปรแกรมแบบ OOP แตกต่างจากการเขียนแบบโครงสร้าง ซึ่งจะไม่ขอกล่าวในรายงานฉบับนี้ ต่อไปจะเป็นการกล่าวถึง การใช้ชุดพัฒนา Microsoft Vision System Development Kit ซึ่งใช้งานบน MS Visual C++ Version 6.0

2.4 Microsoft Vision System Development Kit

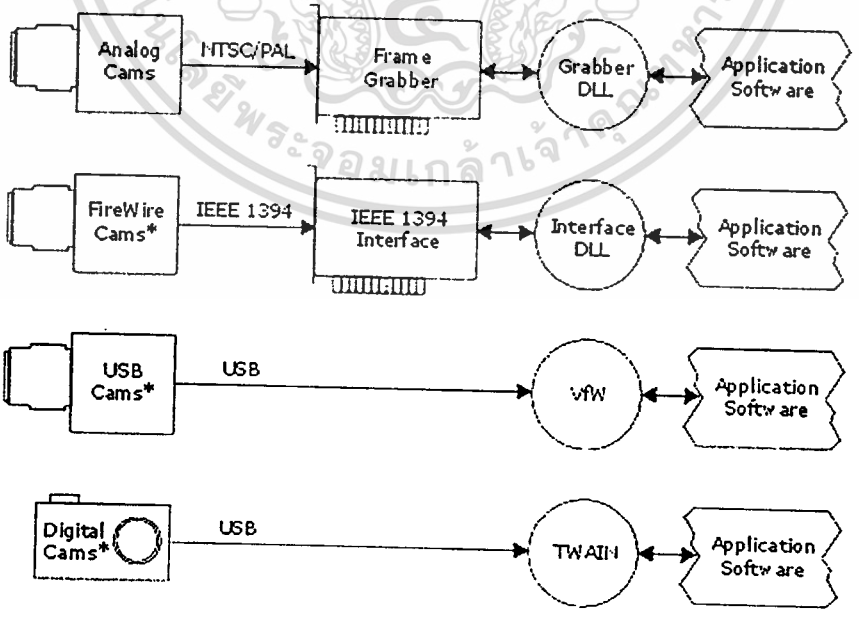
Microsoft Vision System Development Kit (MS Vision SDK) เป็น library ที่ช่วยในการสร้างโปรแกรมสำหรับการจัดการ และ วิเคราะห์ ข้อมูลภาพที่ทำงานบนระบบปฏิบัติการวินโดว์ พัฒนาโดย Vision Technology Research Group ซึ่งเป็นส่วนหนึ่งของ Microsoft Research เพื่อสนับสนุนการวิจัยในงานทางด้าน Image Processing , real-time Image Processing และ Computer

Vision ที่ต้องการสร้าง applications บนระบบปฏิบัติการวินโดว์ MS Vision SDK ได้เตรียมคลาส และ ฟังก์ชั่นในภาษา C++ สำหรับการทำงานที่เกี่ยวข้องกับงานด้านภาพ ผู้สร้าง applications ที่ติดตั้ง MS Vision SDK สามารถพัฒนา applications โดยใช้ MS Visual C++ ในชุด Visual Studio ได้อย่างอิสระโดยไม่ขึ้นกับอุปกรณ์ต่อเชื่อมที่นำสัญญาณเข้าเครื่องคอมพิวเตอร์ เพราะ MS Vision SDK อาศัยหลัก Windows Driver Model (WDM) ที่ทำหน้าที่เป็นตัวกลางในการติดต่อระหว่าง application program กับ อุปกรณ์เชื่อมต่อ เพราะฉะนั้นอุปกรณ์ที่เป็นตัวนำข้อมูลภาพทั้งภาพนิ่งและ ภาพเคลื่อนไหว เช่น กล้องถ่ายรูปดิจิตอล กล้องถ่ายวิดีโอดิจิตัล ที่ต่อกับอุปกรณ์คอมพิวเตอร์ผ่านทาง USB port หรือ ผ่านทาง port IEEE 1394 (Fire Wire) และ รวมทั้งกล้องวิดีโออนาล็อกที่ต่อผ่านทาง capture card (เช่น Metrox card , Meteor card เป็นต้น) ที่ต่อกับคอมพิวเตอร์ผ่านทาง PCI bus ดังแสดงในภาพที่ 10. ส่วนภาพที่ 11 แสดงการสร้าง applications ที่ไม่มี WDM ผู้สร้าง applications ต้องสร้างส่วนที่ติดต่อกับอุปกรณ์ต่อเชื่อมแต่ละชนิดเอง และเมื่อมีการนำอุปกรณ์ใหม่มาใช้จะต้องมีการแก้ไข source code เพื่อทำให้ระบบสามารถทำงานได้

บริษัท Microsoft ได้ทำการแจกจ่าย MS Vision SDK ในรูป source code (download ได้ที่ http://www.research.microsoft.com/research/vision/) ซึ่งสามารถนำไปใช้งานบน MS Visual C++ version 6 ขึ้นไป MS Vision SDK ได้สร้างคลาสหลายชนิดเป็นแบบ template ซึ่งทำให้การเรียกใช้งาน member function กับข้อมูลภาพที่มี data type ชนิดต่างๆกันมีความสะดวก เช่น data type ที่เป็นแบบ integer , byte และ float เป็นต้น และ รวมถึงลักษณะ pixels ต่างๆกัน เช่น RGB , YUV และ Grey Scale เป็นต้น เมื่อเปรียบเทียบข้อดี และ ข้อเสียของ MS Vision SDK กับชุดพัฒนา applications ทางด้าน image processing อื่นๆอาจพอสรุปได้ดังนี้



รูปที่ 10. Windows Driver Model (WDM)



รูปที่ 11 การสร้าง applications ที่ไม่สนับสนุน WDM

ข้อดี

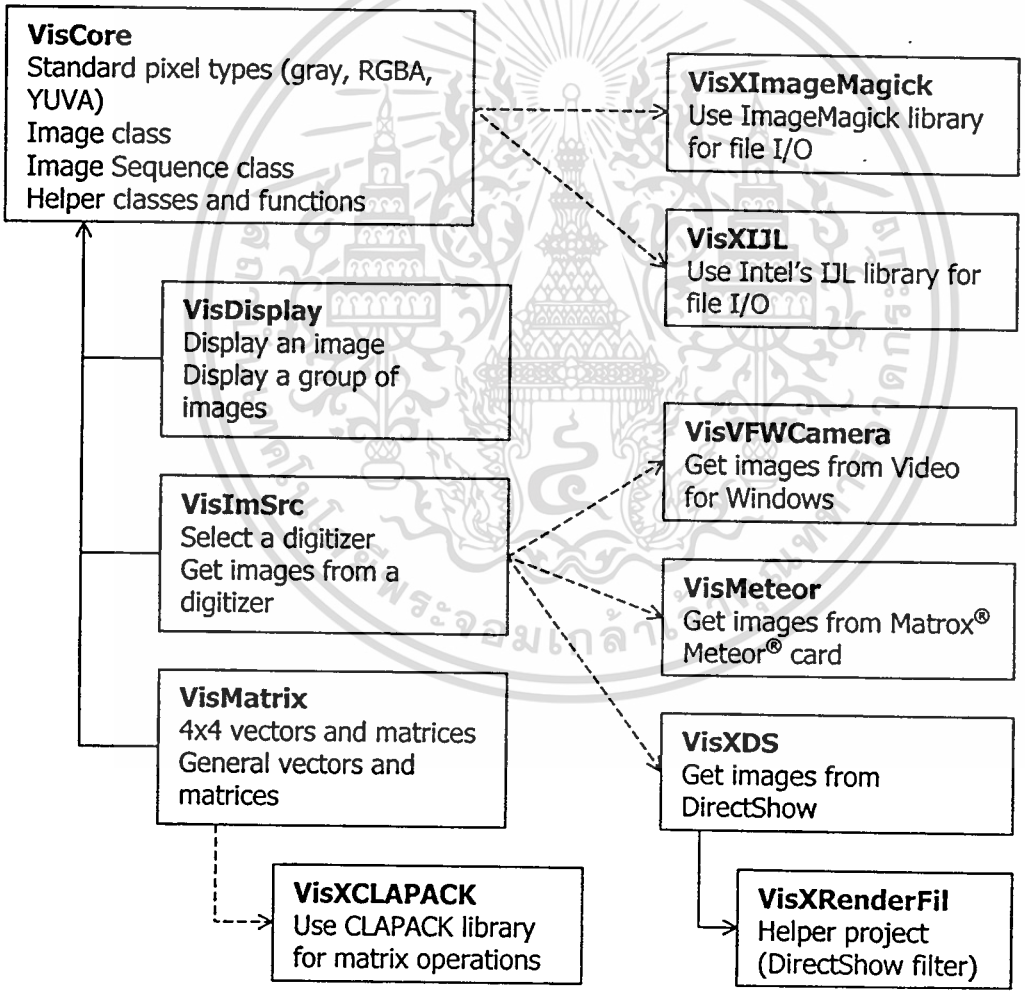
- ประมวลผลได้รวดเร็วเหมาะสมกับงานแบบ real-time โดยเฉพาะกับ application ที่ใช้งานบนระบบปฏิบัติการวินโดว์
- MS Vision SDK เป็น low-level library ทำให้การสร้าง applications มีเสถียรภาพสูง
- การติดต่อระหว่าง applications กับระบบปฏิบัติการวินโดว์เป็นไปได้อย่างมีประสิทธิภาพอย่างเช่น การใช้ image memory ร่วมกันระหว่าง processes ต่างๆภายใน applications รวมทั้งสามารถใช้งาน Graphic Device Interface (GDI) บน Microsoft Foundation Class (MFC) ด้วย
- ผู้ใช้งานสามารถสร้าง pixel data type ได้เอง (User-definable pixel) ซึ่งเหมาะสมกับงานวิจัยทางด้าน image processing
- ผู้พัฒนาสามารถสร้าง applications โดยไม่ต้องคำนึงอุปกรณ์ต่อเชื่อมที่ใช้ในการนำข้อมูลภาพเข้ามา ซึ่งรวมถึงสามารถต่อเชื่อมอุปกรณ์ใหม่ๆได้โดยไม่ต้องแก้ไข source code (ทั้งนี้อุปกรณ์นั้นต้องสนับสนุน WDM เช่น DirectX 8.0 หรือ Video for Window)

ข้อเสีย

- ข้อมูลภาพทั้งหมดจะอยู่ในหน่วยความจำ (RAM) ทำให้ไม่สามารถรองรับแฟ้มข้อมูลภาพ (image file) ชนิดที่มีขนาดใหญ่เกินกว่าที่จะอยู่ในหน่วยความจำได้หมด MS Vision SDK สามารถรองรับได้ขนาดประมาณ 2 Gigabyte ใน window NT และ ขนาด 1 Gigabyte window 9x (เป็นการชดเชยกับความเร็วที่เพิ่มขึ้นในการประมวลผล)
- MS Vision SDK ไม่ได้เตรียมฟังก์ชั่นสำหรับการทำ image processing เช่น การทำ threshold , erosion , dilation เป็นต้น ผู้สร้างจะต้องเขียนฟังก์ชั่นเหล่านี้ไว้ใช้งานเอง หรือ อาจใช้ library ของ Intel's Image Processing Library (download ได้ที่ http://developer.intel.com/) ซึ่งได้เตรียม function พื้นฐานสำหรับงานด้าน image processing โดย Intel's Image Processing Library ใช้ IPLImage เป็นโครงสร้างของข้อมูลภาพ เพราะฉะนั้นถ้าต้องการใช้ฟังก์ชั่นของ Intel's Image Processing Library ผู้ใช้ต้องเปลี่ยน CVisImageObjects (object ที่ใช้เก็บข้อมูลภาพใน MS Vision SDK) ไปเป็น IPLImage strucrures โดยสามารถเรียกใช้ฟังก์ชั่นที่ประกาศไว้ใน header file "VisXIpl.h"

MS Vision SDK สามารถอ่าน และ เขียน ข้อมูลภาพชนิด Device Independent Bitmap (BMP) สำหรับภาพนิ่ง และ ข้อมูลภาพชนิด AVI สำหรับภาพต่อเนื่อง อีกทั้งยังสามารถอ่านข้อมูลภาพชนิด GIF และ PNG ได้ด้วย สำหรับถ้าต้องการอ่าน และ เขียน ข้อมูลภาพชนิดอื่นเช่น JPEG สามารถสร้างใช้ VisXIJL Project สำหรับสร้าง Dynamic Link Library (DLL) เพื่อติดต่อกับ Intel's Joint Photographic Experts Group Library (IJL) หรือ ใช้ VisXImageMagick Project สำหรับการสร้าง DLL ในการติดต่อกับ ImageMagick library เพื่อการอ่านและเขียนแฟ้มข้อมูลภาพชนิดอื่น

2.4.1 โครงสร้างของ MS Vision SDK.



รูปที่ 12 แสดงโครงสร้างของ MS Vision SDK

รูปที่ 12 แสดงโครงสร้างหลักของ MS Vision SDK ซึ่งประกอบไปด้วย header files ที่ใช้สำหรับการ #include เพื่อให้ precomplier ทราบว่าจะมีการเรียกใช้ components หรือ คลาสที่ได้ define ไว้มาใช้งานในโปรแกรม header filesที่ใช้ในโครงการพัฒนาระบบงานนี้ได้แก่

- VisWin.h ซึ่งไม่ได้อยู่รวมกับ MS Vision SDK แต่จะอยู่ใน Microsoft Foundation Class (MFC) ซึ่งเป็น header file ที่ define macros พื้นฐาน และ คลาสที่ต้องการใช้ใน MS Vision SDK
- VisCore.h ซึ่งมี Classes จำนวนมากแต่โดยหลักที่นำมาใช้ได้แก่ คลาส CVisImage , CVisSequence และ ฟังก์ชั่น DisplayInHdcโดยเฉพาะ CVisImage ซึ่งเป็นหัวใจหลักเพราะเป็นตัวที่เก็บข้อมูลภาพซึ่งใช้สำหรับการจัดเก็บ การนำเอาข้อมูลไปใช้ในการประมวลผล รวมทั้งการนำเสนอข้อมูลที่ผ่านการประมวลผลดังกล่าวแล้ว ซึ่งในหัวข้อต่อไปจะกล่าวถึง คลาส CVisImage อย่างละเอียด ในส่วน คลาส CVisSequence จะใช้สำหรับการจัดเก็บข้อมูลภาพในแบบต่อเนื่อง และ การบันทึกภาพต่อเนื่องที่ผ่านการ process แล้วในรูปแบบ AVI file และ สุดท้ายฟังก์ชั่น DisplayInHdc เป็นการนำเสนอข้อมูลภาพที่ผ่านการ process แล้วด้วย Window GDI
- VisImSrc.h สนับสนุนการ capture ภาพจากอุปกรณ์ต่อเชื่อม (เช่น กล้องวิดีโอที่ต่อผ่าน USB port) โดยมี คลาส CVisImageSource และ ฟังก์ชั่น VisFIndImageSource ในการค้นหาแหล่งที่มาของสัญญาณภาพ
- VisDisplay.h ใช้ในการนำเสนอภาพที่ผ่านการ process แล้วเพื่อนำเสนอผ่านทางหน้าจอมอนิเตอร์ซึ่งมีคลาส CVisPane , CVisPaneArray รวมถึงฟังก์ชั่น VisDisplayImage

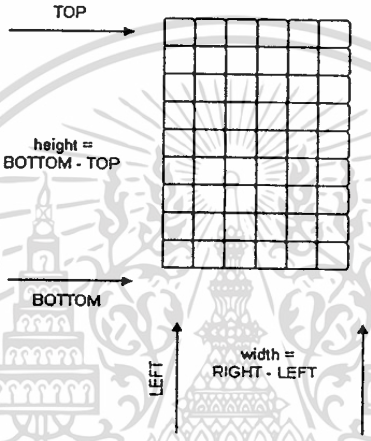
สำหรับรายละเอียดที่เกี่ยวกับคลาสต่างๆ รวมทั้ง methods และ attributes ของแต่ละคลาส จะมีการกล่าวในภาคผนวก ต่อไป

2.4.2 คลาส CVisImage

คลาส CVisImage คือ คลาสหลักที่ใช้ในการทำงานของ Vision SDK คลาสนี้ถูกกำหนดไว้ใน project "VisCore.h" คลาส CVisImage คล้ายคลึงกับ Windows' bitmap header กล่าวคือ คลาส CVisImage มีหน้าที่เก็บคุณสมบัติหลายอย่างเกี่ยวกับภาพ อีกทั้งยังเก็บ pointer ที่ชี้ไปยังหน่วยความจำที่เก็บข้อมูลภาพจริง ข้อแตกต่างจาก bitmap คือ คลาสนี้ถูกเขียนโดยใช้ภาษา C++ และเขียนเป็นแบบ templates ที่สามารถเก็บ และ สามารถ process pixel format ชนิดต่างๆกันได้

ข้อมูลภาพที่จัดเก็บใน Vision SDK จัดเก็บในรูป array ของ pixels ซึ่ง pixel ทั้งหมดจะมี data type ชนิดเดียวกันขึ้นกับผู้เขียนโปรแกรมว่าต้องการใช้ data type แบบใด pixel type (data type ของ pixel นั้นๆ) มีหลายแบบ และ มีทั้งแบบที่ Vision SDK กำหนดไว้แล้ว (Predefined Pixel and Image Types) และ ผู้เขียนโปรแกรมกำหนดเองภายหลัง (Adding a New Pixel Type)

Vision SDK ใช้โครงสร้างแบบ standard Windows RECT structure สำหรับการระบุข้อมูลเกี่ยวกับพื้นที่ของข้อมูลภาพ ดังแสดงในรูปที่ 13



รูปที่ 13 โครงสร้างของการเก็บภาพใน Vision SDK

ตำแหน่ง origin (x_0, y_0) คือ ตำแหน่ง TOP , LEFT ในขณะที่ pixel สุดท้ายคือตำแหน่ง BOTTOM , RIGHT ส่วนค่า height ได้จากนำค่า BOTTOM ลบด้วยค่า TOP ส่วนค่า width ได้จาการนำค่า RIGHT ลบด้วยค่า LEFT

ผู้เขียนโปรแกรมสามารถอ้างถึงตำแหน่งของแต่ละ Pixel ได้โดยตรง โดย Vision SDK จะกำหนดค่า x เป็น คอลัมน์ของภาพ และ ค่า y เป็น แถวของภาพ โดยค่า x จะเริ่มต้นที่ 0 และ เพิ่มค่าขึ้นที่ละ 1 จากด้านซ้ายไปขวา ในขณะที่ค่า y ก็เหมือนค่า x เพียงแต่เพิ่มค่าขึ้นจากบนลงล่าง (ดูตามรูปที่ 13) แต่จุด origin เป็นเพียงจุดสมมุติผู้เขียนสามารถกำหนดจุด origin ได้เองที่ตำแหน่งใดๆก็ได้ในภาพ ซึ่งทำให้การทำงานมีความยืดหยุ่นมากขึ้น

Object (Instance) ที่สร้างจากคลาส CVisImage จริงๆไม่ได้เก็บข้อมูลของแต่ละ pixel จริง แต่จะเก็บ pointer ที่ชี้ไปยัง block ข้อมูลจริงอีกทีหนึ่ง เพราะฉะนั้นผู้เขียนสามารถสร้าง objects หลายๆ objects โดยที่แต่ละ objects สามารถอ้างอิงไปที่ข้อมูลจริงโดยไม่ทำการคัดลอกข้อมูลไปยังที่ใหม่ โดยเมื่อไม่มีการใช้ object นั้นๆแล้ว Vision SDK จะทำการจัดการว่าถ้าไม่การอ้างอิงถึงข้อมูลจริงอีกต่อไป Vision SDK จะทำการคืนหน่วยความจำที่เก็บข้อมูลภาพนั้นๆเสีย

นอกจากนี้ในคลาส CVisImage ยังสามารถเลือกที่จะ process เพียงแค่บางพื้นที่ของภาพเท่านั้นโดย method SubImage() ซึ่งจะกำหนดให้มีการ process เฉพาะตำแหน่งที่กำหนด

2.4.2.1 Pixel และ Image Types

CVisImage ถูกสร้างในแบบ template ในโครงสร้างของภาษา C++ ซึ่งเราสามารถกำหนดชนิดของ object ที่สร้างว่าจะให้เป็นแบบชนิดใดแล้วแต่ความต้องการโดยใช้คำสั่ง CVisImage <pixeltype> โดยชนิดของแต่ละ pixel จะได้กล่าวต่อไป

- Gray Scale Pixel Types แต่ละ pixel ประกอบด้วยค่าความสว่างของแต่ละ pixel ซึ่งมีระดับความสว่าง 256 ระดับ (โดยปกติใช้หน่วยความจำขนาด 8 บิตพอดี) ตารางที่ 1 แสดงรายละเอียดของ pixel แต่ละชนิดในกลุ่มนี้ Component Size

Name	Component Size	Data type	Pixel Size	Display	ImageSource
CVis(Gray)BytePixel	8 bits	Unsigned char	8 bits	Yes	Yes
CVis(Gray)CharPixel	8 bits	Signed char	8 bits	No	No
CVis(Gray)ShortPixel	16 bits	Signed short	16 bits	No	No
CVis(Gray)UShortPixel	16 bits	Unsigned short	16 bits	Yes	No
CVis(Gray)IntPixel	32 bits	Signed int	32 bits	No	No
CVis(Gray)LongPixel	32 bits	Signed long	32 bits	No	No
CVis(Gray)UIntPixel	32 bits	Unsigned int	32 bits	Yes	Yes
CVis(Gray)ULongPixel	32 bits	Unsigned long	32 bits	Yes	Yes
CVis(Gray)FloatPixel	32 bits	Float	32 bits	No	No
CVis(Gray)DoublePixel	64 bits	Double	64 bits	No	No

ตารางที่ 1 แสดงรายละเอียดของ pixel แต่ละชนิดในกลุ่ม Gray Scale Pixel Types

ของแต่ละชนิดมีค่าขึ้นกับ Data Type เช่น CVis(Gray)BytePixel มีค่า 8 บิต ตาม Data Type ชนิด Unsigned char เป็นต้น ผู้เขียนโปรแกรมสามารถเข้าถึงค่าความสว่างได้โดยใช้ method Pixel(x,y) ได้โดยตรงโดยจะได้ค่าเป็น Unsigned char เพราะฉะนั้นเราสามารถ cast ค่าที่ได้เป็น int ได้เพื่อการคำนวณทางคณิตศาสตร์ เช่น

```
CVisByteImage Img;
int x =  (int)Img.Pixel(0,0);
```

สามารถนำค่า x ซึ่งเป็นค่าความสว่างที่ตำแหน่ง pixel ที่จุด (0,0) ไปคำนวณทางคณิตศาสตร์ได้

- RGBA Color Pixel Types จะประกอบด้วย component ย่อยๆ ภายในแต่ละ pixel อีก 4 ค่าได้แก่ ค่าความสว่างในย่านสีแดง (method R()) ค่าความสว่างในย่านสีน้ำเงิน (method B()) ค่าความสว่างในย่านสีเขียว (method G()) และ ค่าอัลฟ่าซึ่งเป็นค่าที่กำหนดระดับการ transparency (method A()) ตารางที่ 2 แสดงรายละเอียดภายในกลุ่ม RGBA Color pixel Types โดยลำดับในการเก็บคือ B-G-R-A ซึ่งตรงกับค่า RGBQUAD ที่ใช้ใน 32-bit Windows bitmaps การเข้าถึงค่าความสว่างของแต่ละสีก็ให้ใช้ method R() , G() , B() หรือ A() โดยเรียกผ่าน method Pixel() ที่อ้างอิงถึงจุดนั้นๆ อีกทางหนึ่ง เช่น

```
CVisRGBAByteImage ColorImg;
int IntR = (int)ColorImg.Pixel(0,0).R();
```

สามารถนำค่า IntR ไปคำนวณได้ และ ยังสามารถใช้ method SetR(value) สำหรับกำหนดค่าความสว่างของสีแดงให้กับจุดนั้นๆ ได้ โดยสามารถกำหนดเป็นค่า int หรือ Unsinged char ก็ได้ (เช่น เดียวกับสีอื่นๆ)

Name	Component Size	Data type	Pixel Size	Display	ImageSource
CVisRGBABytePixel	8 bits	Unsigned char	32 bits	Yes	Yes
CVisRGBACharPixel	8 bits	Signed char	32 bits	No	No
CVisRGBAShortPixel	16 bits	Signed short	64 bits	No	No
CVisRGBAUShortPixel	16 bits	Unsigned short	64 bits	No	No
CVisRGBAIntPixel	32 bits	Signed int	128 bits	No	No
CVisRGBALongPixel	32 bits	Signed long	128 bits	No	No
CVisRGBAUIntPixel	32 bits	Unsigned int	128 bits	No	No
CVisRGBAULongPixel	32 bits	Unsigned long	128 bits	No	No
CVisRGBAFloatPixel	32 bits	Float	128 bits	No	No
CVisRGBADoublePixel	64 bits	Double	256 bits	No	No

ตารางที่ 2 แสดงรายละเอียดของ pixel แต่ละชนิดในกลุ่ม RGBA Color Pixel Types

- YUVA Color Pixel Types คล้ายกับกลุ่ม RGBA Color Pixel Types ที่ได้กล่าวมาแล้วเพียงแต่ค่าความสว่างจัดเก็บในแบบ YUV component แทนที่ RGB เท่านั้น ตารางที่ 3 แสดงรายละเอียดภายในกลุ่มนี้

Name	Component Size	Data type	Pixel Size	Display	ImageSource
CVisYUVABytePixel	8 bits	(Un)signed char	32 bits	Yes	Yes
CVisYUVACharPixel	8 bits	(Un)signed char	32 bits	Yes	Yes
CVisYUVAShortPixel	16 bits	(Un)signed short	64 bits	No	Yes
CVisYUVAIntPixel	32 bits	(Un)signed int	128 bits	No	No
CVisYUVALongPixel	32 bits	(Un)signed long	128 bits	No	No
CVisYUVAULongPixel	32 bits	(Un)signed long	128 bits	No	No
CVisYUVAUIntPixel	32 bits	(Un)signed int	128 bits	No	No
CVisYUVAFloatPixel	32 bits	Float	128 bits	No	No
CVisYUVADoublePixel	64 bits	Double	256 bits	No	No

ตารางที่ 3 แสดงรายละเอียดของ pixel แต่ละชนิดในกลุ่ม YUVA Color Pixel Types

คุณลักษณะการสร้างคลาสที่เป็นแบบ template ยังมีการใช้ในคลาสอื่นๆ เช่น คลาส CVisSequence ซึ่งใช้มากในการจัดการกับข้อมูลภาพที่ได้จากการนำภาพที่ได้จากการถ่ายสดด้วยกล้องวิดีโอ ตัวอย่างการเขียนโปรแกรมโดยการใช้ template เพื่อจัดการกับ pixel ชนิดต่างๆกันได้แก่

```
CVisSequence<CVisRGBABytePixel> sequence;
Sequence.ConnectToSource(imagesource, true, true);
```

จากตัวอย่างโปรแกรมผู้เขียนสามารถสร้างโปรแกรมสำหรับการ capture ภาพจากกล้องวิดีโอ (หรืออาจจะเป็นแหล่งอื่นก็ได้) เพียงแต่ pixel บางชนิดมี หรือ ไม่มีความสามารถในการใช้งาน ImageSource หรือ การ Display ภาพซึ่งผู้เขียนสามารถตรวจคุณสมบัติได้จากตารางทั้งสามตาราง

ซึ่งผู้เขียนสามารถเปลี่ยนแปลง pixel ไปมาได้เพื่อความสะดวกในการใช้งาน โดยการใช้ method CopyPixelTo() หรือ method Copy() ซึ่งจะได้กล่างถึงในหัวข้อต่อๆไป

2.4.2.2 การสร้าง CVisImage Object

การสร้าง CVisImage Object สามารถระบุขนาดของภาพได้โดยใช้ CVisImage constructor ในการกำหนดขนาดคอลัมน์ และ แถวของภาพได้ เช่น

ใช้คำสั่ง CVisByteImage image(100,50);

เป็นการกำหนดขนาดของ Object ให้มีขนาด 100 คอลัมน์ และ 50 แถว (หรือ กว้าง 100 pixels สูง 50 pixels) แต่ผู้เขียนโปรแกรมสามารถสร้าง object ก่อนแล้วระบุขนาดในภายหลังโดยใช้ method Allocate เช่น

```
CVisByteImage image;
Image.Allocate(100,50);
```

จากบทก่อนๆได้กล่าวว่า CVisImage เป็นเพียงการเก็บ pointer ที่อ้างถึงตำแหน่งของหน่วยความจำที่เก็บข้อมูลภาพจริง ในบางครั้งผู้เขียนโปรแกรมต้องการเก็บข้อมูลภาพชุดนั้นไปไว้ที่ตำแหน่งแยกกันต่างหาก method Copy() จะเป็นการสร้างหน่วยความจำที่ตำแหน่งใหม่และมีการสร้างชุดข้อมูลใหม่สำหรับเก็บข้อมูลภาพเดิมโดยที่การแก้ไข การ process ที่ข้อมูลชุดนี้จะไม่มีผลกระทบกับข้อมูลต้นฉบับแต่อย่างใด แต่ method Copy() จะทำงานได้เฉพาะกับข้อมูลในชนิดเดียวกัน ถ้าเป็นกรณีเช่น เป็น pixel type ต่างชนิดกันต้องใช้คำสั่งอื่น คือ method CopyPixelsTo() ดังตัวอย่างข้างล่าง

```
// assign imageFaceOriginal to the subimage in the original data.
CVisByteImage imageFaceOriginal = imageOriginal.SubImage(rectFace);
// fill imageFaceCopy with a new image data block containing just the subimage
CVisByteImage imageFaceCopy.Copy(imageFaceOriginal);
// convert and copy the data into an RGBA version of the face subimage
CVisRGBAByteImage imageFaceRGBA(rectFace);
imageFaceOriginal.CopyPixelsTo(imageFaceRGBA);
```

จากตัวอย่างมีการใช้ทั้ง method Copy() และ method CopyPixelTo() จะเห็นได้ว่า Copy() สามารถใช้ได้เฉพาะ pixel ในแบบเดียวกัน ส่วน CopyPixelsTo() สามารถใช้เปลี่ยนแปลงข้ามกันปมาได้ซึ่งคุณลักษณะนี้จะมีประโยชน์มากในการทำขบวนการทาง image processing อย่างการทำ binary threshold เป็นต้น

แต่ method CopyPixelsTo() ก็ไม่สามารถเปลี่ยน pixel จากกลุ่ม RGBA หรือ YUVA ไปเป็นกลุ่ม grayscale ได้ Vision SDK มีฟังก์ชั่นในการคำนวณโดยใช้สูตร

G (ค่าความสว่างในแถบ grayscale) = .299R + .587G + .114B

2.4.2.3 Operations พื้นฐานของคลาส CVisImage

method Pixel(x , y) เป็น method ในการเข้าถึงค่าความสว่างของแต่ละ pixel ในข้อมูลภาพทุกๆชนิด แต่ในบางครั้งการเขียนโปรแกรมต้องมีการเข้าถึงทุกๆ pixel ในภาพ method RowPointer (row) จึงมีความสะดวก และ รวดเร็วกว่าในการทำงาน ดังตัวอย่างข้างล่าง

```
float FindAverageGray(CVisFloatImage image)
{
        assert(image.NBands() == 1);
        float fltTotalIntensity = 0;
        for (int y = image.Top(); y < image.Bottom();y++)
        {
            float *pflt = image.RowPointer(y);
            for (int x = image.Left(); x < image.Right();x++)
            fltTotalIntensity += pflt[x];
        }
        return (fltTotalIntensity / image.NPoints());
}
```

pflt เป็น pointer ที่รับค่าตำแหน่งของแถวของข้อมูลภาพที่ละแถว เวลาต้องการเข้าถึงค่าในแต่ละ pixel จริงก็สามารถสั่งให้ pflt เลื่อนไปทีละตำแหน่งจนสุดแนวแกน x ได้

คลาส CVisByteImage และ CVisRGBAByteImage มีความคล้ายกับ Windows bitmaps เพราะฉะนั้น คุณสมบัติพื้นฐานบางอย่างจึงสามารถใช้ร่วมกันได้ อย่างเช่น การใช้ Window GDI functions ในการแก้ไข การเสริมแต่งภาพได้โดยตรง การเรียกใช้ Window GDI functions นั้นต้องมีการใช้ method Hdc เพื่อให้ได้ object ของคลาส HDC ซึ่งสามารถเรียกใช้ Window GDI functions ได้ดังตัวอย่างข้างล่าง

```
CVisRGBAByteImage image(100, 100);
HDC hdcImage = image.Hdc();
if (hdcImage != 0)
{
```

```
        // Note that we use (0, 0) to refer to the top-left corner of the
        // image when working with Windows GDI functions.
        RECT rect;
        rect.left = rect.top = 0;
        rect.right = image.Width();
        rect.bottom = image.Height();

        HBRUSH hBrush = CreateSolidBrush((COLORREF) 0xff0000);
        if (hBrush != 0)
        {
                FillRect(hdcImage, &rect, hBrush);
                DeleteObject(hBrush);
        }

        DrawText(hdcImage, "Hello World!", - 1, &rect,
                        DT_CENTER | DT_SINGLELINE | DT_VCENTER);

        image.DestroyHdc();
}
```

เป็นการสร้าง object ของคลาส CVisRGBAByteImage ขนาดกว้าง 100 คอลัมน์ สูง 100 แถว โดยให้สีพื้นภายในเป็นสีน้ำเงิน และ เขียนตัวอักษร "Hello World!" ไว้ตรงกลาง สังเกตุว่ามีการใช้ Windows GDI functions เช่น DrawText() โดยต้องมีการสร้าง object ในคลาส HDC โดยรับค่าจาก method Hdc()

2.4.3 การแสดงข้อมูลภาพทางจอมอนิเตอร์

เมื่อมีการสร้าง object ในคลาส CVisImage และ มีการ process ข้อมูลภายในแล้ว การจะแสดงภาพที่ได้สามารถทำได้สามวิธีได้แก่

2.4.3.1 แสดงผลโดยใช้ Windows HDC

ถ้า object ที่สร้างจากคลาส CVisImage มีคุณสมบัติเหมือนใน Windows bitmaps เช่น CVisRGBAByteImage เป็นต้น ผู้เขียนโปรแกรมสามารถใช้ method DisplayInHdc() สำหรับการ

แสดงภาพบน window ได้ (เช่นที่ Application Wizard ของ Vision SDK ใช้) โดยจะต้องมีสร้าง object ของคลาส HDC (ใน Microsoft Foundation Class) จากนั้นผ่านค่า window's HWND ไปให้กับฟังก์ชั่น GetDC() (ใน Microsoft Foundation Class)

DisplayInHdc สนับสนุนชนิดของข้อมูลภาพต่อไปนี้

- CVisRGBAByteImage
- CVisByteImage
- CVisUShortImage(RGB555)
- CVisUIntImage
- CVisYUVACharPixel(Gray Scale)

ถ้าผลที่ได้จากขบวนการทาง image processing ที่ได้มีชนิดของข้อมูลภาพแตกต่างออกไป ผู้เขียนโปรแกรมสามารถใช้ method CopyPixelsTo() สำหรับการเปลี่ยนชนิดของข้อมูลภาพได้

2.4.3.2 แสดงผลโดยใช้คลาส CVisPane และ VisDisplayImage

คลาส CvisPane และ CVisPaneArray กำหนดอยู่ใน VisDisplay.h ผู้เขียนโปรแกรมสามารถ include VisDisplay.h และ สามารถสร้าง object จากคลาส CVisPane และ CVisPaneArray โดยสามารถนำ cbject ที่สร้างจากคลาส CVisImage มาแสดงผลได้ ดังตัวอย่างข้างล่าง

```
m_pPaneArray = new CVisPaneArray(1,1, InsW, InsH,
                                 "",evispaneDefaultAutoDestroy);
m_pPaneArray->Pane(0).Display(InsertImage, evispanedispAliasOrigNoScrnBuf);
```

จากตัวอย่าง InsertImage เป็น object ในคลาส CVisRGBAByteImage ที่ผ่านการ process มาแล้วเราสร้าง m_pPaneArray เป็น pointer ที่สร้างชี้ไปยัง Instance ในคลาส CviaPaneArray ด้วยคำสั่ง new() จากนั้นเรียก method display() ในการแสดง InsertImage ลงบนหน้าจอมอนิเตอร์

ผู้เขียนโปรแกรมสามารถแสดงภาพผ่านฟังก์ชั่น VisDisplayImage() ได้ด้วย โดยฟังก์ชั่นนี้อยู่ใน VisDisplay Project

2.4.3.3 แสดงผลโดยการสร้าง Windows bitmap

ผู้เขียนสามารถ Windows bitmap จากข้อมูลภาพแต่ pixel ได้โดยตรง ซึ่งวิธีการนี้ต้องอาศัยวิธีการเขียนโปรแกรมบนวินโดว์โดยตรง จะไม่กล่าวในที่นี้

2.4.4 การจัดการติดต่อกับอุปกรณ์ต่อพ่วง

Vision SDK สามารถติดต่อกับอุปกรณ์ต่อเชื่อมจากภายนอกได้โดยที่ไม่ขึ้นกับอุปกรณ์ โดย Vision SDK ได้เตรียม VisImSrc DLL เอาไว้ โดยผู้ใช้โปรแกรมสามารถเลือกอุปกรณ์ต่อเชื่อมในขณะที่กำลังต่ออยู่กับคอมพิวเตอร์ได้

คลาส CVisImageSource ใน VisImSrc Project โดยสามารถใช้ฟังก์ชั่น VisFindImageSource() ซึ่ง return เป็น instance ของคลาส CVisImageSource ดังตัวอย่าง

CVisImageSource ImSrc = VisFindImageSource("");

จากคำสั่งข้างบน ImSrc จะทำการติดต่อกับอุปกรณ์ต่อเชื่อมโดยจะเก็บข้อมูลภาพในลักษณะข้อมูลคิบเพื่อรอการ process ต่อไป ในกรณีต้องการติดต่อกับอุปกรณ์ที่เก็บภาพต่อเนื่อง เช่น กล้องถ่ายวิดีโอ สามารถใช้ method SetUseContinuousGrab()

คลาส CVisSequence เป็นคลาสที่มีการใช้มากในการจัดการเกี่ยวกับ process ภาพต่อเนื่อง รวมถึงการอ่าน และ การเขียนข้อมูลภาพที่เป็นภาพต่อเนื่อง เช่น AVI file เป็นต้น คลาส CVisSequence อยู่ใน VisCore Project เช่นเดียวกับคลาส CVisImage มีลักษณะเป็น template เช่นเดียวกัน เพราะฉะนั้นจึงสามารถทำงานกับ data type ในชนิดต่างๆกันได้ การทำงานของคลาส CVisSequence จะทำงานกับข้อมูลภาพที่มีลักษณะเป็นชุดโดยในแต่ละชุดจะต้องเก็บข้อมูลในประเภทเดียวกัน คลาส CVisSequence ใช้ Standard Template library deque class ในการจัดเก็บกลุ่มของข้อมูลภาพ

deque เป็นโครงสร้างข้อมูลที่เก็บ objects ไว้ภายใน และ สามารถเข้าถึงข้อมูลโดยการใช้ดัชนี (index) สามารถเพิ่ม และ สามารถลบข้อมูลออกจากตำแหน่งหัว หรือ ท้ายของ deque ได้ คุณสมบัติคล้ายในโครงสร้างข้อมูลแบบคิว

การใช้คลาส CVisSequence สามารถกำหนดขนาดของจำนวน objects ที่จะบรรจุไว้ใน deque ได้โดยค่า default คือ ค่า 0 ซึ่งเวลาดึง object ไปใช้จะได้ object เป็นชุดล่าสุดที่เข้ามาใน deque การกำหนดว่าจะใช้ buffer เท่าไรขึ้นกับชนิดของการทำงาน อย่างเช่น ถ้าต้องการ capture ภาพที่ได้สดๆมาแสดงบนจอมอนิเตอร์ควรตั้งไว้ในขนาดที่ไม่มากคือ ประมาณ 0 ถึง 5 แต่ถ้าต้องการเก็บภาพจากส่วนอื่นของโปรแกรมไว้ก่อนทำการ process ก็ควรตั้งค่า buffer ขนาดปานกลางคือ ประมาณ 30 ถึง 200 และ การตั้งค่า buffer ขนาดสูงคือ ประมาณมากกว่า 200 สำหรับการเก็บภาพแต่ละเฟรมเพื่อนำมาสร้างเป็นไฟล์วิดีโอ สำหรับ method ในการตั้งค่าขนาดของ buffer ของคลาส CVisSequence คือ SetLenghtMax() และ ยังมี method อื่นๆเช่น ReadStream() , InsertStream() , AppendStream() และ WriteStream() ในการจัดการเกี่ยวกับไฟล์ AVI ด้วย

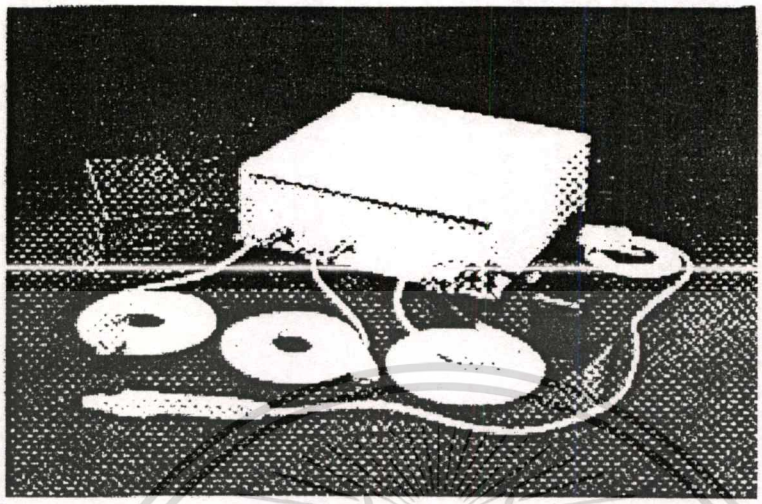
การนำเอา object แต่ละ object ที่อยู่ในคลาส CVisSequence มาทำการ process สามารถใช้ method Pop() โดยผู้เขียนสามารถสร้าง object ในคลาส CVisImage มารับไป process ต่ออีกทางหนึ่ง ดังตัวอย่าง

```
CVisImageSource imagesource = VisFindImageSource("");
if (imagesource.IsValid()){
        CVisSequence<CVisRGBABytePixel> sequence;
        Sequence.ConnectToSource(imagesource, true, false);

        CVisRGBAByteImage imageT;
        if (sequence.Pop(image, 2000){
                imageT.FWriteFile("out.bmp");
        }
        sequence.DisconnectFromSource();
}
```

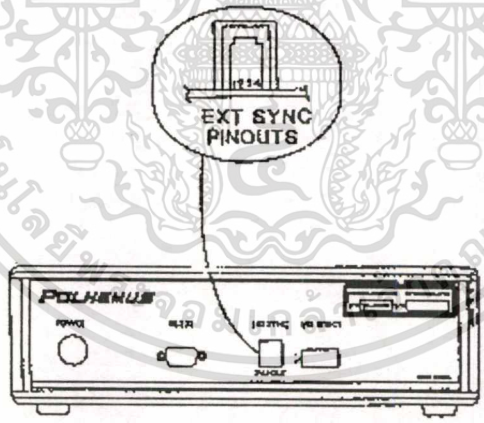
จากตัวอย่างโปรแกรมข้างต้นมีการใช้ method ConnectToSource() เพื่อการเชื่อมต่อ object ในคลาส CVisSequence กับ Image Source จากนั้นมีการใช้ method Pop() สำหรับการนำข้อมูลภาพภายใน deque ของ CVisSequence มาเก็บไว้ใน object ของคลาส CvisRGBAByteImage เพื่อทำการ process ต่อไป

2.5 การตรวจหาตำแหน่งของกล้องด้วยอุปกรณ์ ISOTRAK®II

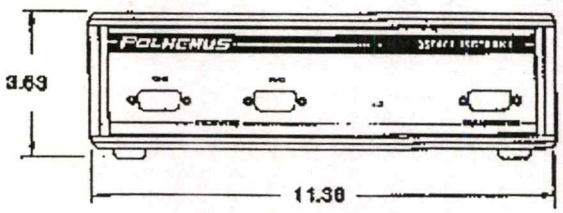
ISOTRAK®II tracking system เป็นระบบสำหรับการตรวจหาตำแหน่งของวัตถุ โดยระบบจะประกอบด้วยอุปกรณ์ System Electronic Unit (SEU) , receiver และ transmitter ระบบ ISOTRAK II มีส่วนติดต่อกับระบบคอมพิวเตอร์โดยผ่าน port RS-232 ระบบ ISOTRAK II จะทำการสร้างสนามแม่เหล็กโดยเมื่อผู้ใช้ต้องการทราบตำแหน่งของวัตถุเคลื่อนที่ ผู้ใช้ต้องทำการติด receiver ไว้เมื่อวัตถุเคลื่อนผ่านสนามแม่เหล็กระบบจะทำการตรวจสอบตำแหน่งเทียบกับ transmitter และ แจ้งค่าตำแหน่ง (six degree of freedom) ผ่านทาง port RS-232 ผู้ใช้งานสามารถเขียนโปรแกรมเรียกฟังก์ชั่นเพื่อดึงค่าจากอุปกรณ์ที่ส่งค่ามาให้ ระบบ ISOTRAK II สามารถครอบคลุมพื้นที่ในการตรวจสอบได้แม่นยำไม่เกินรัศมี 28 นิ้ว ถ้ารัศมีเกินกว่า 28 นิ้วแต่ไม่เกิน 60 นิ้ว ความแม่นยำจะลดลงไป รูปที่ 14 ถึง 17 แสดงส่วนประกอบของชุดอุปกรณ์ ISOTRAK II



รูปที่ 14 แสดงส่วนประกอบของระบบ ISOTRAK II tracking system

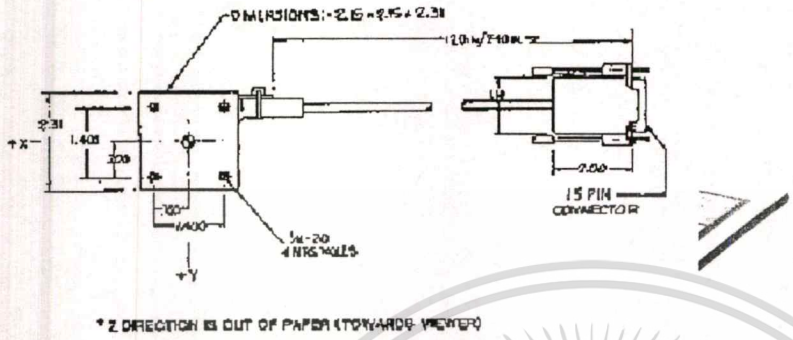


REAR

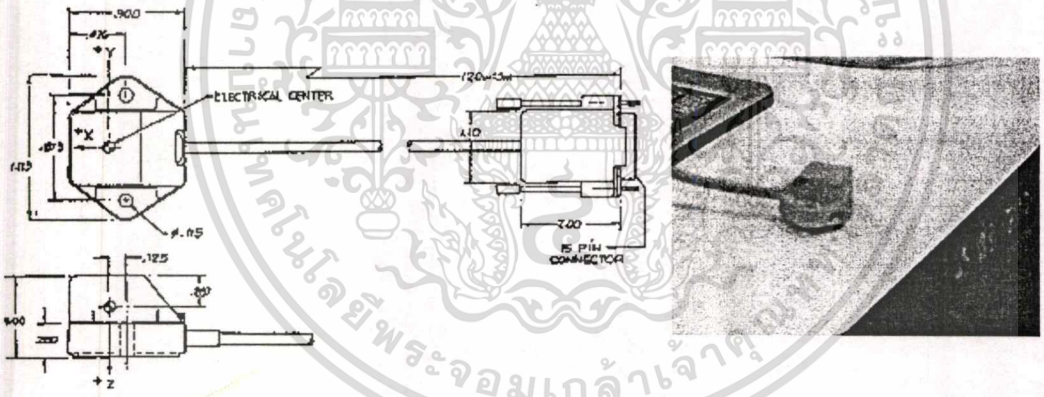


FRONT

รูปที่ 15 แสดง ISOTREK II Controls/Connections



รูปที่ 16 แสดงอุปกรณ์ Transmitter



รูปที่ 17 แสดงอุปกรณ์ Receiver

ชุดอุปกรณ์ ISOTRAK®II สามารถครอบคลุมพื้นที่ในการตรวจสอบได้แม่นยำไม่เกินรัศมี 28 นิ้ว ถ้ารัศมีเกินกว่า 28 นิ้วแต่ไม่เกิน 60 นิ้วความแม่นยำจะลดลงไป รูปที่ 14 ถึง 17 แสดงส่วนประกอบของชุดอุปกรณ์ ISOTRAK®II

ค่าที่ได้จากชุดอุปกรณ์ ISOTRAK®II จะประกอบไปด้วยค่าหกค่าคือ ค่า Six degree-of-freedom ซึ่งประกอบด้วยตำแหน่งของวัตถุที่ต้องการตรวจสอบในแกน x , y และ z เทียบกับจุดสมมุติจุดหนึ่ง (จุด origin คือ ตำแหน่งที่ transmitter เท่ากับ receiver พอดี) และ องศาของการหมุนของวัตถุรอบแกนทั้งสาม คือ ค่า yaw , pitch และ roll เพราะฉะนั้นจากค่า Six degree of freedom ดังกล่าว ทำให้เราสามารถทราบระยะห่างระหว่าง กล้องวีดีโอ และ วัตถุที่ใช้สำหรับการ calibrate

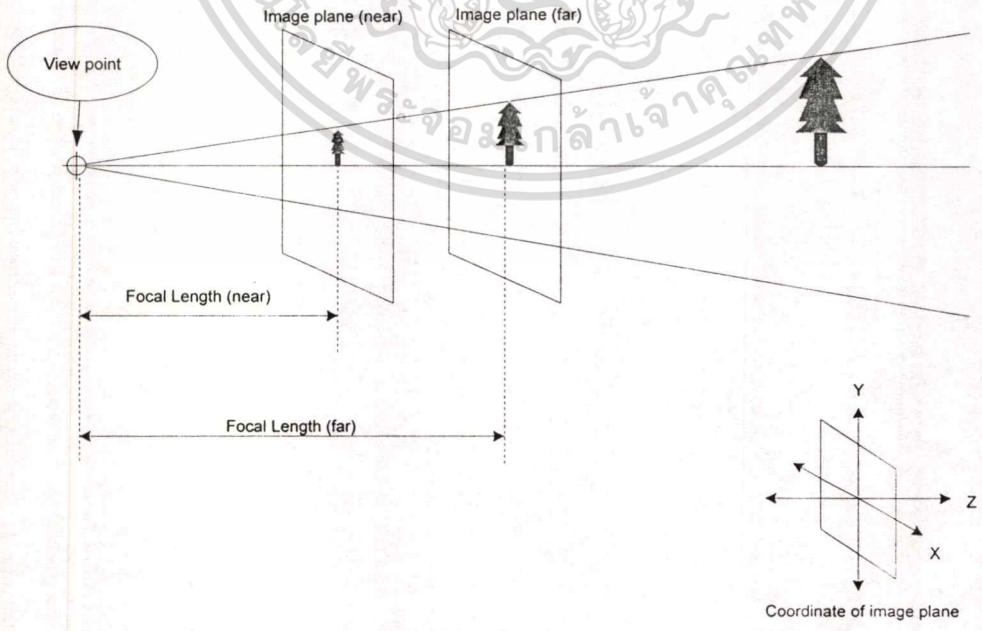
ได้ตลอดเวลาที่กล้องวิดีโอมีการเคลื่อนที่ไปมา และเมื่อประกอบกับการที่เราสามารถหาค่ามุมที่เปลี่ยนแปลงระหว่างสามแกนจะทำให้เราสามารถกำหนดลักษณะของวัตถุสังเคราะห์ทั้งขนาด (ซึ่งได้จากระยะห่างของกล้องกับวัตถุที่ใช้ calibrate) และ มุมมอง (ได้จากตำแหน่งของกล้อง และ มุมกล้องรอบแกนทั้งสาม) แต่ปัจจัยที่สำคัญอีกปัจจัยหนึ่งที่เป็นตัวกำหนดทั้งขนาดวัตถุ และ ลักษณะทาง perspective ของวัตถุ คือ ค่าความยาวโฟกัสของกล้อง (focal length) โดยในส่วนขนาดของวัตถุนั้น ความยาวโฟกัส และ ระยะห่างของกล้องกับวัตถุ เป็นค่าที่นำมาใช้คำนวณเพื่อการกำหนดขนาดของวัตถุสังเคราะห์ร่วมกัน ในหัวข้อต่อไปจะเป็นการกล่าวถึงวิธีการหาค่าความยาวโฟกัสเพื่อนำค่ามาใช้ร่วมกับค่า Six degree of freedom ที่ได้จากหัวข้อนี้

2.6 การตรวจหาความยาวโฟกัสโดยใช้วัตถุทรงกลม

ในหัวข้อนี้จะกล่าวถึง ความหมายของค่าความยาวโฟกัส และ วิธีการในการหาค่าความยาวโฟกัสซึ่งใช้วิธีการทางด้าน Image processing

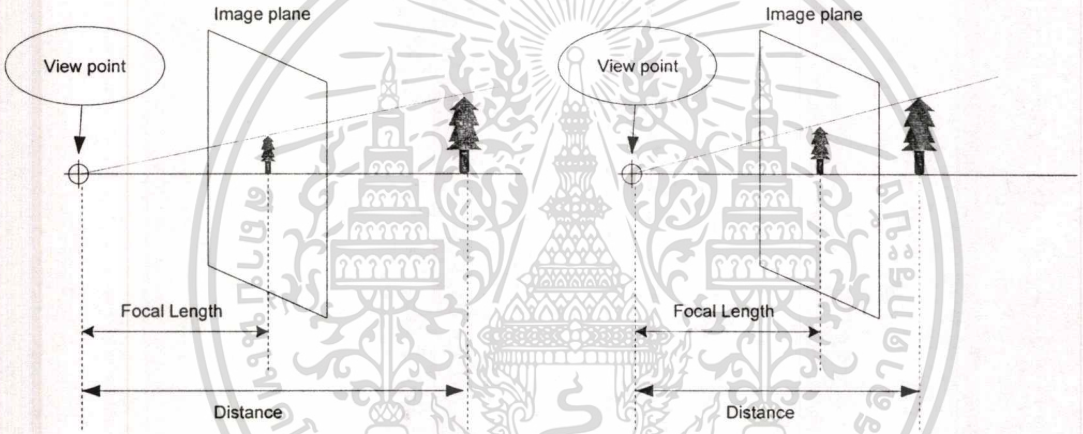
2.6.1 ความยาวโฟกัส (focal length)

ความยาวโฟกัสคือ ระยะทางจากจุดรวมแสงของเลนส์ที่เป็นจุดสมมุติอยู่ด้านหลังเลนส์ (View point) ถึงบริเวณฉากรับภาพ (ถ้าเป็นกล้องถ่ายภาพแบบเดิมก็คือ ฟิล์มถ่ายรูป แต่ถ้าเป็นกล้องถ่ายรูปดิจิตอล หรือ กล้องถ่ายวิดีโอก็คือ แผงเซ็นเซอร์รับภาพ) ดังรูปที่ 18



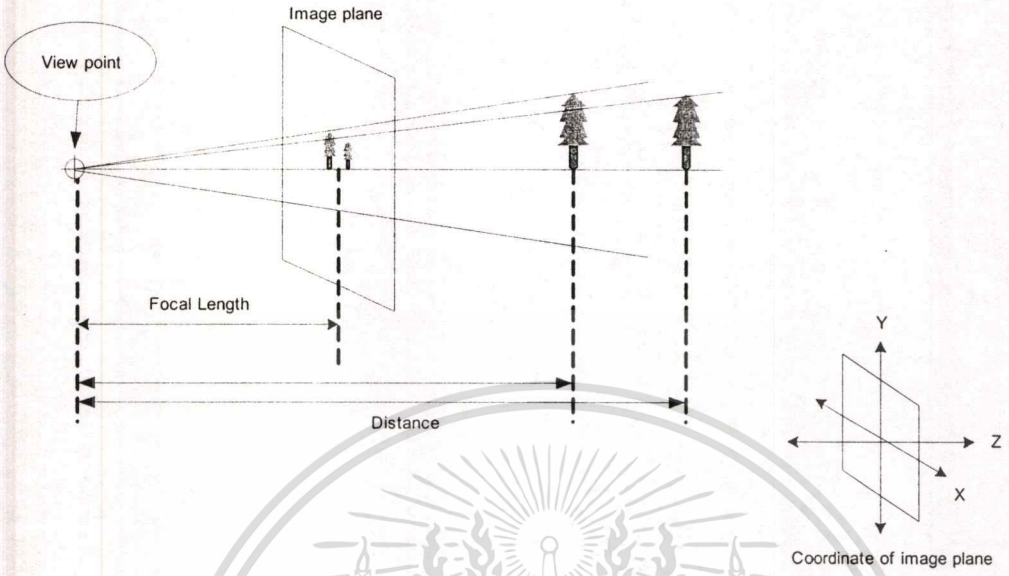
รูปที่ 18 แสดงค่าความยาวโฟกัสที่มีผลต่อขนาดของวัตถุบนภาพที่ project บน Image plane

เมื่อผู้ใช้กล้องวิดีโอมีการซูมกล้อง เข้า-ออก ผลที่ได้คือ ความยาวโฟกัสมีการเปลี่ยนแปลง ดังรูปที่ 18 เมื่อ plane รับภาพเคลื่อนที่ออกจาก view point ผลที่ได้คือภาพจะมีขนาดใหญ่ขึ้น นอกจากนั้นลักษณะทาง perspective ก็จะมีการเปลี่ยนแปลง และ ในทางตรงข้ามกับกรณีถ้า plane รับภาพเคลื่อนที่เข้าหา view point ภาพจะมีขนาดเล็กลง ในส่วนระยะทางจาก view point ถึงวัตถุนั้นผลที่ได้แสดงในรูปที่ 19 กล่าวคือเมื่อกล้องวิดีโอเคลื่อนที่เข้าใกล้วัตถุภาพที่ project ลงบน Image plane ก็จะมีขนาดใหญ่ขึ้นตามไปด้วย ดังรูปขวามือ ภาพต้นไม้ที่ปรากฏจะมีขนาดใหญ่กว่าทางซ้ายมือ



รูปที่ 19 แสดงผลของระยะห่างกับวัตถุที่จะ project ลงบน Image plane

ความยาวโฟกัส และ ระยะห่างของกล้องวิดีโอกับวัตถุ จะเป็นสองปัจจัยหลักที่เป็นตัวกำหนดร่วมว่าวัตถุที่ปรากฏลงบน Image plane ควรมีขนาดเท่าใด และ เมื่อกล้องวิดีโอมีการเคลื่อนที่ ประกอบกับมีการเปลี่ยนอัตราการซูม วัตถุที่จะปรากฏใน Image plane ควรมีขนาดเท่าใด ข้อแตกต่างระหว่างสองปัจจัยคือ ความยาวโฟกัสจะมีผลต่อคุณสมบัติทาง perspective ด้วยในขณะที่ระยะห่างระหว่างกล้องวิดีโอกับวัตถุจะส่งต่อเฉพาะขนาดของวัตถุที่มีการเปลี่ยนแปลงเท่านั้น รูปที่ 20 แสดงคุณสมบัติในทาง perspective กล่าวคือ วัตถุที่มีขนาดเท่ากันเมื่ออยู่ห่างจากกล้องวิดีโอไม่เท่ากันภาพที่ปรากฏเมื่อ project ลงบน Image plane จะมีขนาดไม่เท่ากัน คือ วัตถุที่อยู่ไกลกว่าจะมีขนาดเล็กกว่า ลักษณะนี้ทำให้เกิดความลึกในการมองภาพ



รูปที่ 20 ลักษณะทาง perspective ที่ปรากฏซึ่งทำให้เกิดความลึกของภาพ

จากหัวข้อที่ 2.5 เราสามารถใช้อุปกรณ์ ISOTRAK®II ในการตรวจหาตำแหน่งของกล้องวิดีโอได้ภายในรัศมีของอุปกรณ์ ISOTRAK®II เพราะฉะนั้น เราสามารถหาระยะทางของกล้องวิดีโอกับวัตถุได้ตลอดเวลา ส่วนในการหาความยาวโฟกัสซึ่งเป็นปัจจัยภายในของกล้องวิดีโอเราต้องวิธีการทาง Image processing ในการหาซึ่งจะกล่าวถึงในหัวข้อต่อไป

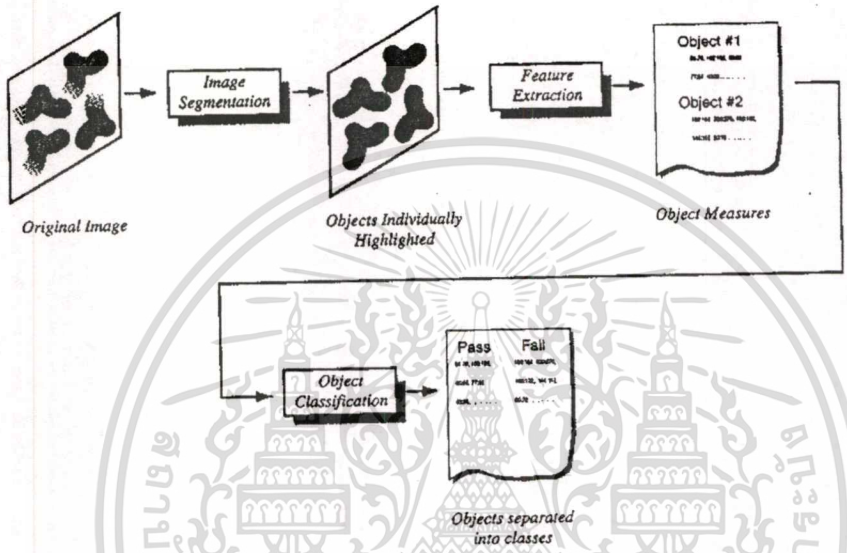
2.6.2 การใช้วัตถุทรงกลมสีสำหรับการหาค่าความยาวโฟกัส

Image Analysis เป็นกระบวนการทาง Image processing วิธีหนึ่งซึ่งแตกต่างจากกระบวนการทาง Image Processing อื่นๆ คือ ผลลัพธ์ที่ได้จะไม่ได้เป็นภาพ หน้าที่หลักของ Image Analysis คือ ต้องการวัดค่าเฉพาะบางอย่างของแต่ละองค์ประกอบในภาพ ผลที่ได้จึงมักเป็นตัวเลขที่ได้จากการวัด หรือ คำนวณหาปริมาณบางอย่างจากภาพ ทั้งนี้เนื่องจากในภาพๆหนึ่งมีข้อมูลแฝงอยู่ซึ่งสามารถตีความเป็นสิ่งที่วัดได้ เช่น ในภาพๆหนึ่งแต่ละ pixel ของภาพนั้นจะประกอบไปด้วยค่าความสว่างซึ่งสามารถเทียบเป็นตัวเลขได้ เพราะฉะนั้นเราสามารถดึงคุณลักษณะเฉพาะบางอย่างออกมาจากภาพๆนั้นได้ รูปที่ 21 แสดงขั้นตอนของกระบวนการ Image Analysis

กระบวนการ Image Analysis มีขั้นตอนดังต่อไปนี้คือ

- ขั้นตอนแรกของกระบวนการ Image Analysis คือการทำ Image Segmentation ซึ่งวัตถุประสงค์คือ การแยกวัตถุที่มีอยู่ในภาพออกมาจากกัน ทำให้วัตถุที่อาจผสมหรือ ซ้อนทับกันแยกออกมาเป็นวัตถุที่มีขอบเขตเป็นของตัวเอง ซึ่งก็อาศัยเทคนิค

ในทาง Image Processing หลายอย่าง เช่น Erosion , Dilation , Outlining และ Skeletonization เป็นต้น



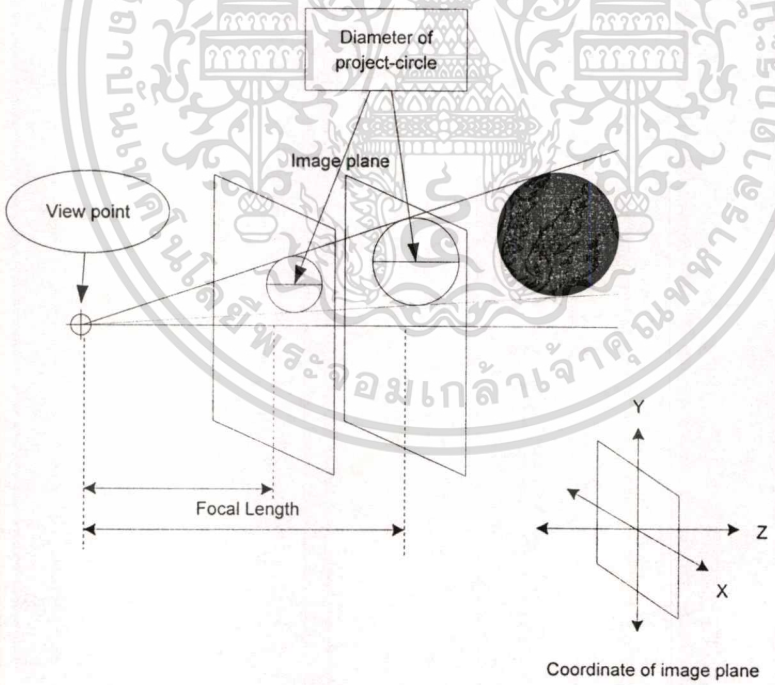
รูปที่ 21 แสดง Flow Diagram ของกระบวนการ Image Analysis

- ขั้นตอนต่อไปคือการทำ Feature Extraction กระบวนการนี้คือ การพยายามหาเอกลักษณ์เฉพาะตัวของวัตถุที่เราสนใจ เพื่อแยกวัตถุนั้นออกจากวัตถุอื่นๆในภาพ หรือ จากภาพพื้นเอง ซึ่งเอกลักษณ์ที่เราสามารถเลือกใช้ได้มีหลายอย่าง เช่น การใช้สีที่ต่างกันของวัตถุในภาพ การใช้รูปทรงที่ต่างกัน หรือ การใช้ลวดลายภายในที่แตกต่างกัน เป็นต้น การจะเลือกใช้เอกลักษณ์ใดก็ขึ้นอยู่กับปัจจัยหลายๆอย่างโดยวัตถุประสงค์ที่สำคัญนั้นคือ เอกลักษณ์ที่เลือกต้องสามารถทำให้การแยกวัตถุออกมามีความถูกต้องแม่นยำสูงสุด และ จะต้องใช้วิธีในการแยกที่เรียบง่ายคำนวณที่สุดเป็นสำคัญ
- สุดท้ายเป็นการทำ Object Classification เป็นการใช้ความรู้ทางสถิติ หรือ วิธีการอื่นๆ เช่น Neural Network เป็นต้น เพื่อจะได้ตัดสินใจว่าวัตถุที่ผ่านกระบวนการทั้งสองสมควรจัดอยู่ในกลุ่มใด

จากวิธีการ Image Analysis ที่กล่าวมาข้างต้นเราสามารถประยุกต์ใช้วิธีการดังกล่าวในการความยาวโฟกัสของกล้องวิดีโอได้ โดยทำการวางวัตถุสำหรับการวัดความยาวโฟกัสเข้าไปใน

บริเวณที่ทำการถ่ายภาพ โดยอาศัยความจริงที่ว่าเมื่อกล้องวิดีโอมีการเปลี่ยนขนาดความยาวโฟกัส (โดยที่ระยะห่างระหว่างกล้องวิดีโอกับวัตถุมีค่าคงที่ไม่เปลี่ยนแปลง) ขนาดของวัตถุนั้นจะการเปลี่ยนแปลงในสัดส่วนเดียวกัน ทำให้เราสามารถวัดขนาดที่เปลี่ยนแปลงแล้วทำการคำนวณหาค่าความยาวโฟกัสที่ปรากฏ ณ. เฟรมของภาพนั้นๆได้

สำหรับในการพัฒนาระบบงานนี้เราเลือกวัตถุทรงกลมเป็นวัตถุที่ใช้สำหรับการตรวจสอบ เพราะ วัตถุทรงกลมมีลักษณะเฉพาะคือ มีความสมมาตรทุกแกนทำให้มุมกล้องไม่มีผลต่อการคำนวณหาความยาวโฟกัส เราสามารถเลือกเอกลักษณ์ของวัตถุในการทำ Feature Extraction ได้หลายอย่างอาทิเช่น ใช้รูปทรง ใช้สี หรือ ใช้ขอบของวัตถุ เป็นต้น ในการพัฒนาระบบงานนี้เราเลือกเอาสีของวัตถุทรงกลมดังกล่าวมาใช้ในการแยกวัตถุออกจากภาพพื้น หรือ จากวัตถุอื่นๆในภาพ ในรูปที่ 22 แสดงการใช้วัตถุทรงกลมสีน้ำเงินในการคำนวณหาความยาวโฟกัสเมื่อมีการซูมกล้องวิดีโอเข้า-ออก



รูปที่ 22 การหาความยาวโฟกัสโดยใช้ภาพวงกลมที่ได้จากการ project วัตถุทรงกลม

การที่ application สามารถตรวจหารูปวงกลมของวัตถุทรงกลมได้เพราะเราใช้สีที่มีความแตกต่างจากสีอื่นๆ ที่อยู่ในเฟรม อาทิเช่น ใช้สีน้ำเงินในการตรวจสอบ เพราะฉะนั้นในเฟรมของภาพจะต้องไม่มีวัตถุชนิดอื่น หรือ สีพื้นที่มีสี (ค่าความสว่างของจุดสีนั้นๆ) ใกล้เคียงกัน โดย

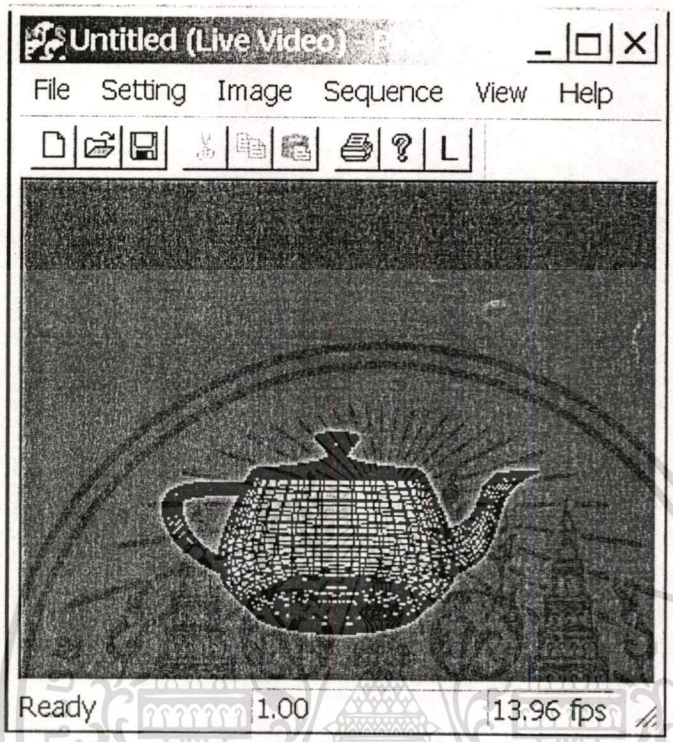
หลักการที่ว่าที่จุดหนึ่งในเฟรมจะประกอบไปด้วยค่าความสว่างของสีสามสี (ในกรณี MS Vision SDK มีค่า A() ซึ่งคือค่า อัลฟ่า เป็นค่าที่ใช้สำหรับการทำ transparent ซึ่งในการทดลองครั้งนี้เราไม่ได้ใช้) โดยที่ถ้าจุดนั้นเป็นแม่สี และ มีความอิ่มตัวสูงการที่จะทำให้เครื่องคอมพิวเตอร์มีความสามารถในการจดจำ และ แยกแยะจุดนั้นๆได้ ก็อาศัยหลักการเปรียบเทียบค่าความสว่างของสีทั้งสาม อย่างเช่น ถ้าเรากำหนดว่าวัตถุที่จะนำมาใช้ในการ calibrate เป็นสีน้ำเงิน การตรวจหาค่าสีน้ำเงิน ก็คือการที่จุดใดๆ มีค่าความสว่างของสีน้ำเงินมากกว่าอีกสองสีที่เหลือ แต่ปัญหา คือ แล้วจะแยกสีขาว หรือ สีดำบางค่าที่ในจุดนั้นมีค่าความสว่างของสีน้ำเงินมากกว่าอีกสองสีได้อย่างไร เนื่องจากเราสามารถเลือกสีของวัตถุให้มีความอิ่มตัวสูง ซึ่งเมื่อมีความอิ่มตัวสูงทำให้ค่าความสว่างที่แตกต่างจะมีค่ามากกว่า สีขาว และ สีดำ (รวมทั้งสีอื่นๆ ที่เป็นสีน้ำเงินเทียม) การจะแยกความแตกต่างนี้ทำได้โดยการหา ค่าความเบี่ยงเบนมาตรฐาน (standard deviation) หรือ ค่าความแปรปรวน (variance) โดยเราสามารถกำหนดไว้เป็นค่าคงที่ค่าหนึ่งซึ่งถ้ากำหนดไว้ต่ำเกินไปจะเกิด noise อันเนื่องจากการที่มีสีน้ำเงินเทียมมาผสม แต่ถ้ากำหนดไว้สูงเกินไปค่าความสว่างบริเวณขอบของรูปวงกลมจะตรวจหาไม่พบทำให้เส้นผ่านศูนย์ที่หาได้มีขนาดสั้นกว่าความเป็นจริง

ส่วนค่าที่จะนำมาคำนวณหาความยาวโฟกัสก็มีได้หลายค่า เช่น ขนาดของเส้นผ่านศูนย์ที่มีการเปลี่ยนแปลง ขนาดของรัศมี หรือ ขนาดของพื้นที่วงกลมที่เกิดจากการ project ของวัตถุทรงกลม เป็นต้น ทั้งการเลือกค่าต่างๆ ขึ้นกับว่าต้องการความถูกต้องมากน้อยเพียงใด และ ความเร็วในการคำนวณด้วย เพราะการทำงานต้องเป็นแบบ real time

2.7 OpenGL และ การสร้างรูปสามมิติ

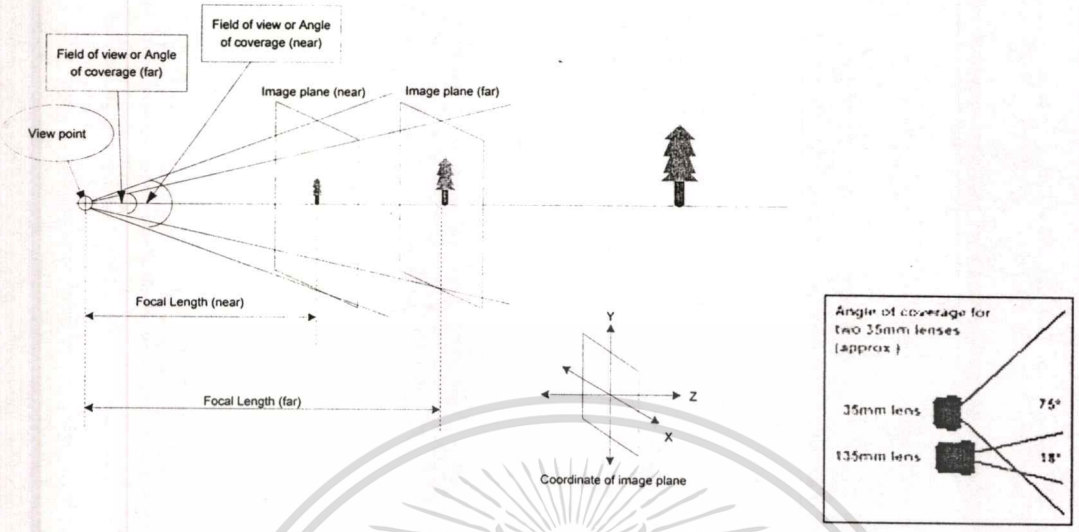
Open GL คือ เป็น Application programming interface (API) ที่ใช้ในการสร้างวัตถุสังเคราะห์สามมิติที่มีความสามารถในการตอบสนอง (interactive) ถูกสร้างขึ้นโดยบริษัท Silicon Graphic Inc. ในแบบ Open Source และ สามารถรองรับกับคอมพิวเตอร์ที่มีแพลตฟอร์มต่างๆกัน เช่น สามารถสร้างโปรแกรมประยุกต์ที่ใช้ OpenGL บนระบบปฏิบัติการ วินโดว์ หรือ ระบบยูนิกซ์ เป็นต้น

จุดสำคัญที่เราใช้ OpenGL ในการสร้างวัตถุสามมิติเพราะ การที่วัตถุสังเคราะห์ที่ได้จากการสร้างโดย OpenGL มีลักษณะที่เป็นวัตถุที่มีความสามารถในการตอบสนอง (interactive) ได้ กล่าวคือ ในโลกของ OpenGL วัตถุสังเคราะห์ที่สร้างขึ้นจะมีลักษณะของการเป็นวัตถุสามมิติ มุมมองต่อวัตถุในโลกของ OpenGL จะเสมือนหนึ่งว่าวัตถุนั้นกำลังถูกบันทึกภาพไว้โดยกล้องในโลกเสมือน (Virtual camera) ซึ่งเราสามารถสั่งให้กล้องในโลกเสมือนเคลื่อนเข้าหาวัตถุสังเคราะห์ เปลี่ยนมุมกล้อง หรือ แม้กระทั่งการซูมเข้า-ออก จากวัตถุสังเคราะห์ได้โดยการผ่านค่าพารามิเตอร์ที่ได้ข้อ 2.5 และ ข้อ 2.6 เข้าไปยังฟังก์ชั่นของ OpenGL ได้



รูปที่ 23 รูปวัตถุสังเคราะห์สามมิติที่สร้างโดย Open GL

ข้อดีอีกประการหนึ่งคือ วัตถุสังเคราะห์ที่สร้างจาก OpenGL และ ผ่านขบวนการตรวจสอบหาขนาดที่เกิดจากการเคลื่อนที่ของกล้องวิดีโอ และ การซูมของกล้องวิดีโอ รวมถึงทิศทาง และ มุมของกล้องวิดีโอ สามารถจะนำไปรวมกับภาพที่ได้จากการ capture ด้วย MS Vision SDK ได้โดยใช้คุณสมบัติที่ OpenGL สามารถ transparence ลงบนภาพแต่ละเฟรมนั้นๆได้ รูปที่ 23 แสดงภาพกาน้ำชาสามมิติที่สร้างโดย OpenGL (เป็นวัตถุสังเคราะห์ที่ OpenGL สร้างไว้เป็นตัวอย่างอยู่แล้ว)



รูปที่ 24 ความสัมพันธ์ระหว่าง Field of view กับความยาวโฟกัส

การเปลี่ยนอัตราการซูมใน OpenGL อาศัยหลักการในการเปลี่ยนแปลงค่า Field of view (Angle of coverage) จากรูปที่ 24 ด้านซ้ายมือ เมื่อเราเปลี่ยนแปลงค่าความยาวโฟกัส เช่น เพิ่มความยาวโฟกัส (ซึ่งทำให้ image plane เคลื่อนที่เข้าใกล้วัตถุ) ค่า Field of view จะมีค่าลดลง หรือ ตามรูป 24 ด้านขวามือ เลนส์ขนาดความยาวโฟกัส 35 มิลลิเมตรมีค่า Field of view เท่ากับ 75 องศา ในขณะที่ เลนส์ขนาด 135 มิลลิเมตรจะมีค่า Field of view ลดลงเหลือเพียง 18 องศา เพราะฉะนั้นอาจกล่าวได้ว่า ค่าความยาวโฟกัส แปรผกผันกับ ค่า Field of view

จากหัวข้อที่ 2.6 เราสามารถหาเส้นผ่านศูนย์กลางของรูปวงกลมที่เกิดจากการ project ของวัตถุทรงกลม ซึ่งเส้นผ่านศูนย์กลางที่มีการเปลี่ยนแปลงนั้นอาจเกิดได้จาก ระยะห่างที่เปลี่ยนแปลงไปจากเดิม หรือ เกิดจากผู้ใช้มีการเปลี่ยนแปลงความยาวโฟกัส (มีการซูมเข้า-ออก) ในระหว่างการใช้กล้องวิดีโอ หรือ เกิดจากทั้งสองปัจจัย เพราะฉะนั้นเส้นผ่านศูนย์กลางของรูปวงกลมเกิดจากผลที่มาจากการเปลี่ยนระยะห่าง และ ความยาวโฟกัส เพราะฉะนั้นจากสมการที่ 1 เส้นผ่านศูนย์กลาง Dia' เกิดจากเส้นผ่านศูนย์กลางที่เกิดจากผลของระยะห่าง Dia_d คูณกับเส้นผ่านศูนย์กลางที่เกิดจากผลของความยาวโฟกัส Dia_f คูณกับค่าเส้นผ่านศูนย์กลางเริ่มต้น

$$Dia' = Dia_d * Dia_f * Dia \quad (1)$$

แต่เนื่องจากเราใช้อุปกรณ์ ISOTRAK®II ซึ่งทำให้เราหาระยะห่างของกล้องวิดีโอกับวัตถุทรงกลม เพราะฉะนั้นจากสมการที่ 2 เมื่อ D คือ ระยะห่างเริ่มต้น Dia คือ เส้นผ่านศูนย์กลางเริ่มต้น และ เมื่อ

กล้องวิดีโอมีการเคลื่อนที่เราหาค่าระยะห่างที่มีการเปลี่ยนแปลงได้คือ D' ซึ่งทำให้เราสามารถหาค่าเส้นผ่านศูนย์กลางที่น่าจะเป็นได้คือ Dia_d'

$$D / D' = Dia / Dia_d' \quad (2)$$

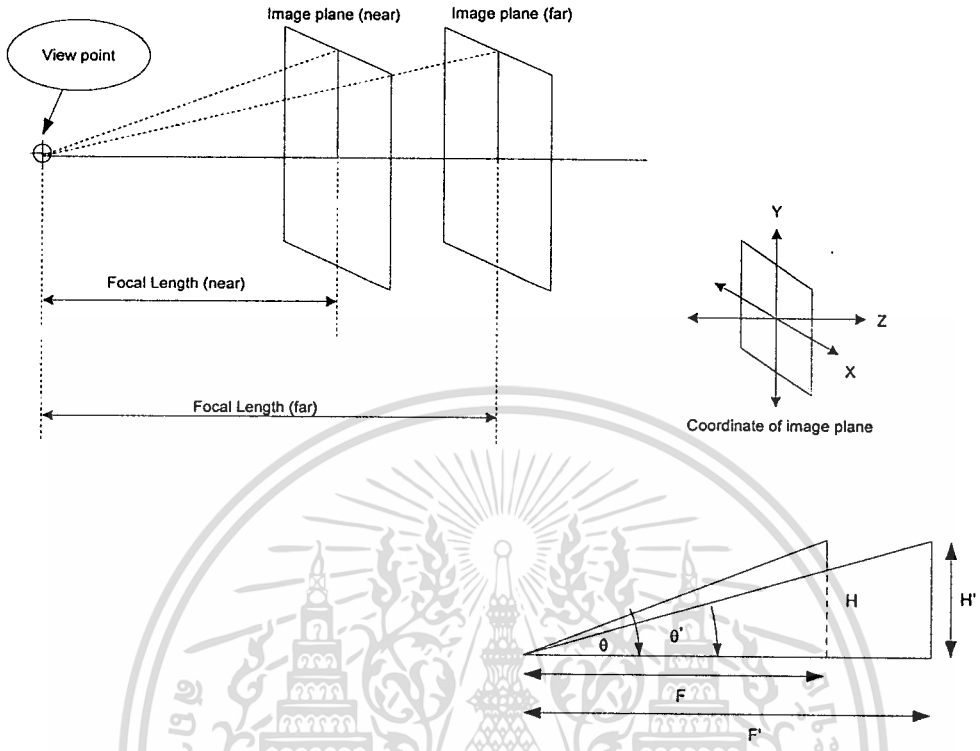
$$Dia_d' = Dia * D' / D \quad (3)$$

เพราะฉะนั้นถ้าเราทราบค่า D ซึ่งคือระยะห่างตอนเริ่มต้น เราทราบค่า D' จากอุปกรณ์ ISOTRAK®II และ เราทราบสามารถวัดค่าเส้นผ่านศูนย์กลางได้ทำให้เราทราบค่า Dia และ ค่า Dia_d'ทำให้เราสามารถหาค่า Dia_f' ได้จากสมการที่ 4 โดยที่ค่า Dia' คือ ค่าเส้นผ่านศูนย์ที่เปลี่ยนแปลงจากค่าเริ่มต้นที่วัดได้

$$Dia_f' = Dia * Dia' / Dia_d' \quad (4)$$

ความสัมพันธ์ระหว่างค่าความยาวโฟกัสที่เปลี่ยนแปลง และ เส้นผ่านศูนย์กลางที่เปลี่ยนแปลงเป็นปฏิภาคตรงเพราะฉะนั้นสามารถเขียนเป็นสมการที่ 5

$$F / F' = Dia / Dia_f' \quad (5)$$



รูปที่ 25 ความยาวโฟกัสเปลี่ยนแปลงสัมพันธ์กับ Filed of view

จากรูปที่ 25 สามเหลี่ยมที่มุมขวาสามารถเขียนเป็นสมการที่ 6

$$\text{Tan}(\theta) = H / F \quad (6)$$

และ เนื่องจากด้าน H เป็นส่วนสูงของ image plane เพราะฉะนั้นจึงได้สมการที่ 7 ซึ่งด้าน H และ H' ของสามเหลี่ยมทั้งสองเท่ากันและมีมุมที่ปลายสามเหลี่ยมเป็น θ และ θ'

$$\text{Tan}(\theta) * F = \text{Tan}(\theta') * F' \quad (7)$$

$$\text{Tan}(\theta') = \text{Tan}(\theta) * (F / F') \quad (8)$$

$$\text{Tan}(\theta') = \text{Tan}(\theta) * (\text{Dia} / \text{Dia}_f') \quad (9)$$

เพราะฉะนั้นผลลัพธ์ที่ได้จากสมการที่ 10 ทำให้เราสามารถผ่านค่า Field of view (θ) ไปยัง virtual camera ใน OpenGL เพื่อการปรับขนาดของวัตถุสังเคราะห์ตามการเปลี่ยนแปลงของกล้องวิดีโอจริงได้อย่างถูกต้อง

$$\theta' = \text{Tan}^{-1}(\text{Tan}(\theta) * (\text{Dia} / \text{Dia}_f') \quad (10)$$

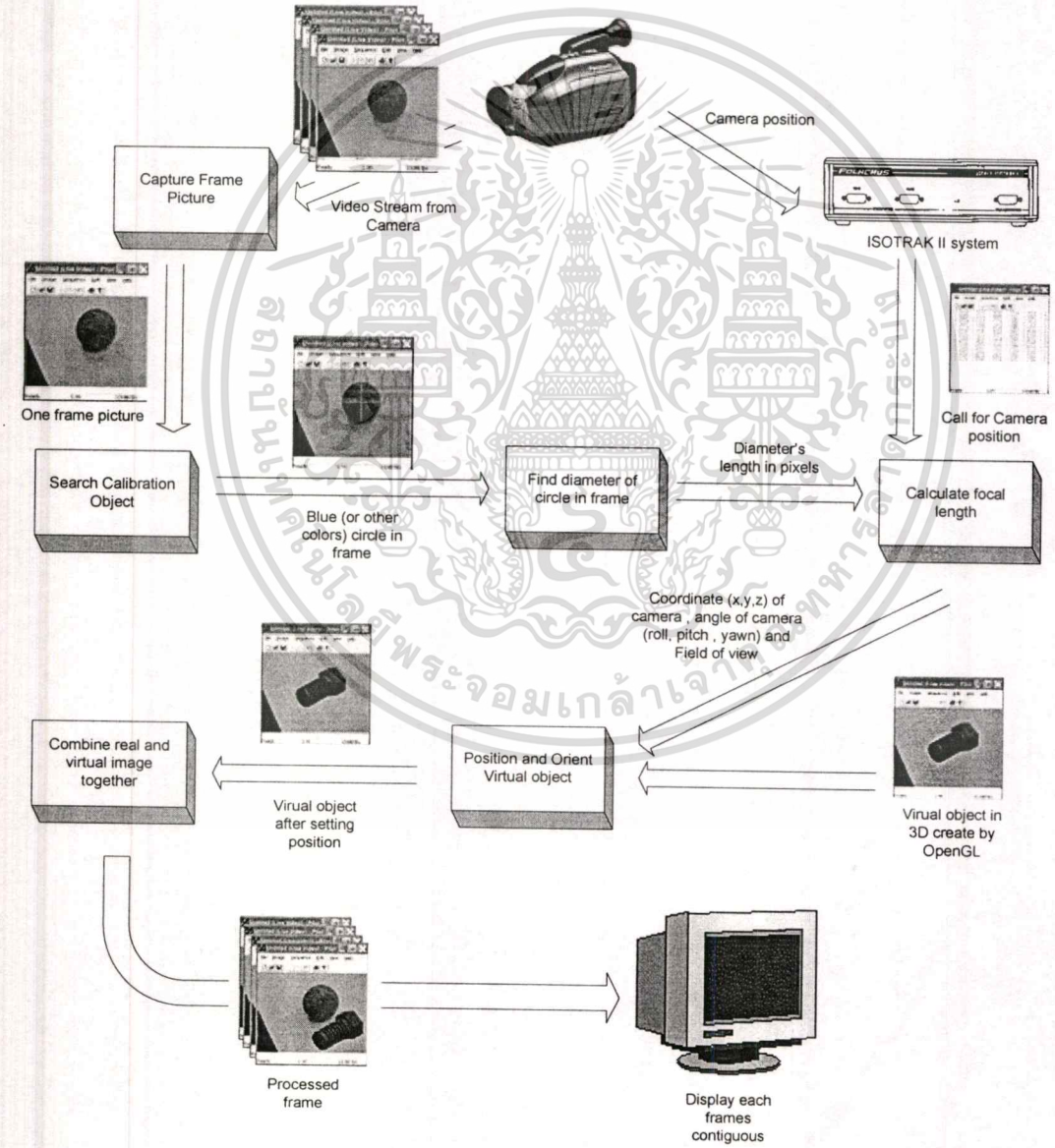
และ เมื่อเราแทนค่า Dia_f' ด้วยค่าจากสมการที่ 4 เราจะได้เป็นสมการที่ 11 ที่ค่า

$$\theta' = \text{Tan}^{-1}(\text{Tan}(\theta) * (\text{Dia} * D' / \text{Dia}' * D) \quad (11)$$

โดยเมื่อเรากำหนดค่า Field of view (θ) เป็นค่าเริ่มต้นสำหรับ virtual camera ใน OpenGL และ ค่า D และ ค่า Dia คือ ค่าเริ่มต้น D′ คือ ค่าที่วัดได้เมื่อผู้ใช้มีการเปลี่ยนระยะห่างของกล้อง และ ค่า Dia′ คือ ค่าเส้นผ่านศูนย์กลางของวงกลมที่วัดได้ขณะนั้น สังเกตว่าถ้ากล้องไม่มีการเคลื่อนที่ ค่า D และ D′ จะมีค่าเท่ากัน ทำให้ค่า Dia′ ที่ได้รับอิทธิพลจากการซูม (มีการเปลี่ยนแปลงความยาวโฟกัส) เพียงอย่างเดียว

บทที่ 3

การออกแบบและการสร้างระบบ

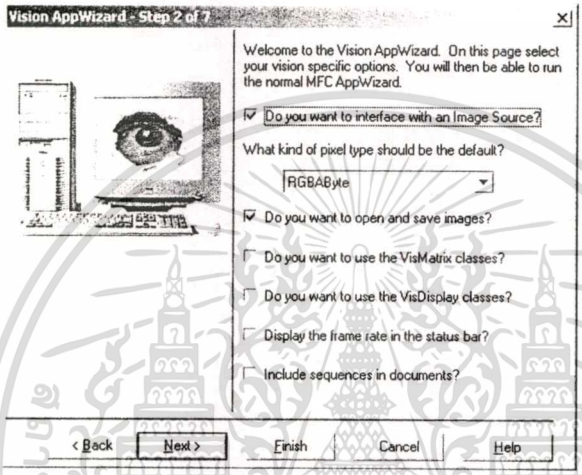


รูปที่ 26 Block Diagram ของระบบงาน

ในบทนี้จะเป็นการนำเอาหลักการ และ ทฤษฎีที่ได้กล่าวเมื่อบทที่แล้วมาทำการทดลองสร้างเป็นโปรแกรมประยุกต์เพื่อการใช้งานด้าน Augmented reality รูปที่ 26 แสดง Block Diagram ในการสร้าง AR application

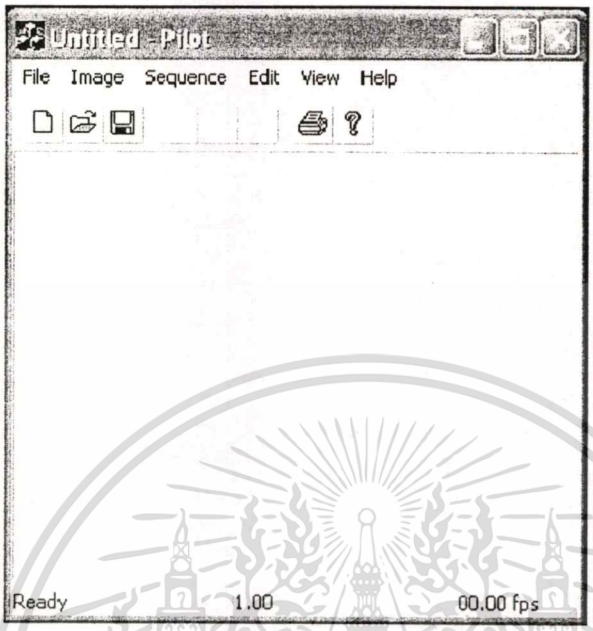
3.1 การ Capture Frame picture

บทนี้จะได้กล่าวถึงการ capture ภาพจากกล้องวิดีโอด้วย MS Vision SDK



รูปที่ 27 แสดงหน้าต่าง Vision AppWizard

โครงงานนี้ใช้ MS Vision SDK ในส่วน Vision AppWizard ในส่วนของการสร้าง application โดยให้เปิดโปรแกรม Visual C++ แล้วใช้คำสั่ง new จากนั้นเลือก icon Vision AppWizard จะปรากฏหน้าต่างดังรูปที่ 27 การสร้าง application ทำเหมือนการสร้าง AppWizard ทั่วๆไป เมื่อสร้างเสร็จทดลองรันโปรแกรมจะปรากฏหน้าต่างดังรูปที่ 28



รูปที่ 28 แสดงหน้าต่างของ application

ส่วนการเขียนคำสั่งเพิ่มเติมจะอยู่ในส่วนคำสั่ง OnDraw() ในคลาส CView ตัวอย่างคำสั่ง เช่น

```
void CView::OnDraw(CDC* pDC){
        CDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);

        pDoc->Lock();

        // TODO: add draw code for native data here
        // (Note that this is within the document's critical section.)
        CVisRGBAByteImage image = pDoc->GetImage();

/*  แทรกคำสั่งสำหรับการ process ได้ที่ตำแหน่งนี้  */

        if (image.IsValid(){
                image.DisplayInHdc(pDC->m_hDC);
```

```
        }

        // Tell the document that this image was displayed.
        // (This is needed when processing images in the foreground thread.
                pDoc->OnImageDisplayed();

                pDoc->Unlock();
        }
```

มีการสร้าง object image ในคลาส CVisRGBABytelmage โดยให้ reference ไปที่ pointer pDoc จากนั้นใช้ method GetImage() ทำการดึงภาพมาจากกล้องวิดีโอ เพราะฉะนั้นเราสามารถแทรกการ process ภาพลงไปได้ โดยเมื่อการ process เสร็จโปรแกรมจะใช้วิธีการแสดงผ่านโดยใช้ method DisplayInHdc() (ซึ่งได้อธิบายวิธีการในหัวข้อ 2.4.3.1)

3.2 การตรวจหาวัตถุที่ใช้ในการ calibrate

จากหัวข้อที่แล้วเราสามารถ capture ภาพต่อเนื่องได้มาเป็นเฟรม ลำดับถัดมาคือ การนำภาพที่ได้ในแต่ละเฟรมมาทำการหารูปของวัตถุในภาพที่ใช้สำหรับการ calibrate จากบทที่สอง ได้กล่าวถึงวิธีการใช้ภาพวงกลมที่ได้จากการ project ของวัตถุทรงกลมมาทำการความยาวโฟกัสต่อไป

วัตถุทรงกลมมีลักษณะเฉพาะ คือมีความสมมาตรทุกทิศทาง เพราะฉะนั้นเราสามารถเคลื่อนไหวกล้องวิดีโอไปมาได้ทุกทิศทาง โดยมีข้อจำกัดเพียงวัตถุที่ใช้ในการ calibrate ต้องอยู่ในภาพเต็มเฟรม ส่วนในการตรวจหาวัตถุทรงกลมนั้นเราอาศัยของสีของวัตถุในการแยกวัตถุทรงกลมออกจากวัตถุอื่น หรือ ภาพพื้น โดยในการทดลองเราใช้วัตถุทรงกลมสีน้ำเงินที่มีความอิ่มตัวของสีสูง เพราะฉะนั้นเราสามารถตรวจสอบได้ว่า pixel นั้นๆในเฟรมที่ปรากฏมีวัตถุสีน้ำเงินอยู่ที่ใดโดยตรวจสอบโดยใช้ตรวจสอบจากค่าความสว่างของสีแต่ละสีเทียบกันในแต่ละ pixel จากตัวอย่าง

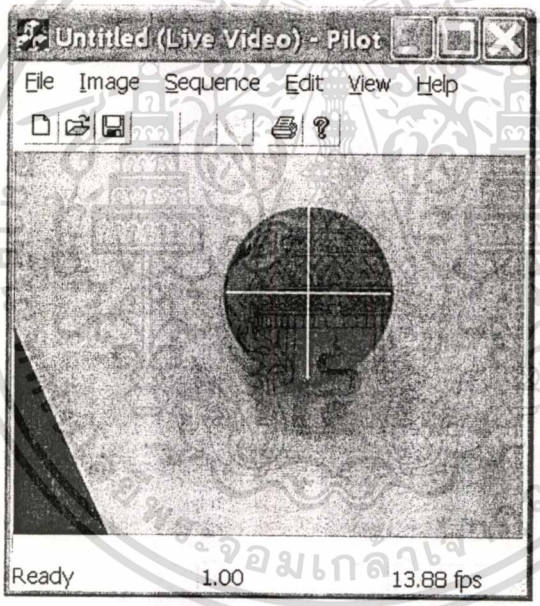
```
int ir = (int)image.Pixel(pixelx,pixely).R();
int ig = (int)image.Pixel(pixelx,pixely).G();
int ib = (int)image.Pixel(pixelx,pixely).B();
float imean = (float)(ir+ig+ib)/3;
float ivar = (float)(((ir-imean)*(ir-imean))+
        ((ig-imean)*(ig-imean))+((ib-imean)*( ib-imean)))/3;
```

จากโปรแกรมตัวอย่างสมมุติฐานมีอยู่ว่าถ้าจุดนั้นเป็นสีน้ำเงิน ค่าความสว่างของสีน้ำเงินเมื่อเทียบกับค่าความสว่างของสีแดง และ สีเขียวจะต้องมีค่ามากกว่า แต่ปัญหาคือ มีสีขาว หรือ สีดำ

บางค่าที่ค่าของสีน้ำเงินในจุดนั้นมีค่ามากกว่าอีกสองสีในจุดเดียวกัน จึงต้องอาศัยค่าความแปรปรวน (Variance) เพราะ ค่าความแปรปรวนเป็นตัวบอกว่าที่จุดนั้นสีน้ำเงินมีค่ามากกว่าสีอีกสองสีแค่ไหน โดยตามความเป็นจริงสีดำ หรือ สีขาวที่ไม่ใช่สีแท้อาจมีค่าความสว่างของสีน้ำเงินมากกว่าอีกสองสีแต่จะมากกว่าไม่มากเท่ากับสีน้ำเงินแท้

```
if(ib>ir && ib>ig && ivar>IVAR && ib>50){ // Do Something …
```

จากการทดลองเราทดสอบกับวัตถุทรงกลมสีน้ำเงินที่ใช้ทดสอบพบว่า ค่าความแปรปรวนที่ขนาดสูงกว่า 50 ให้ผลที่ถูกต้องเมื่อผ่านกระบวนการแยกวัตถุที่ใช้สำหรับการ calibrate จะได้ตามรูปที่ 29



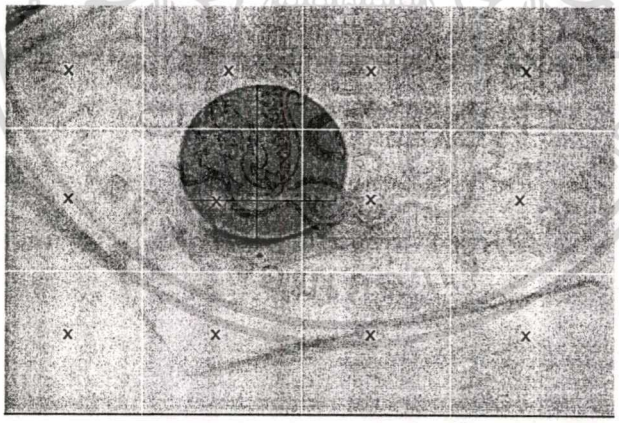
รูปที่ 29 แสดงวัตถุทรงกลมสีน้ำเงินที่โปรแกรมตรวจสอบได้

3.3 การหาเส้นผ่านศูนย์กลางของรูปวงกลมที่เกิดจากการ project ของวัตถุทรงกลม

เนื่องจากในการพัฒนาระบบงานนี้สิ่งที่สำคัญคือ ความรวดเร็วในการประมวลผล เพราะโปรแกรมประยุกต์ที่สร้างขึ้นมาต้องสามารถใช้งานได้ในแบบทันทีทันใด (real time) เพราะฉะนั้นนอกจากประสิทธิภาพของอุปกรณ์นำมาใช้งานแล้ว เช่น หน่วยประมวลผลกลางของคอมพิวเตอร์ (CPU) และ การ์ดจอสามมิติ เป็นต้น อัลกอริทึมสำหรับการคำนวณก็มีเป็นสิ่งสำคัญในการคำนวณหาเส้นผ่านศูนย์กลางของวงกลมในภาพมีหลายวิธีในการหา ซึ่งแต่ละวิธีก็มีข้อดี ข้อเสียแตกต่างกันไป เช่น ถ้าจะหาเส้นที่เกิดจากจุดสีน้ำเงินทุกจุดเรียงต่อกันในแนวแกนนอน แล้ว

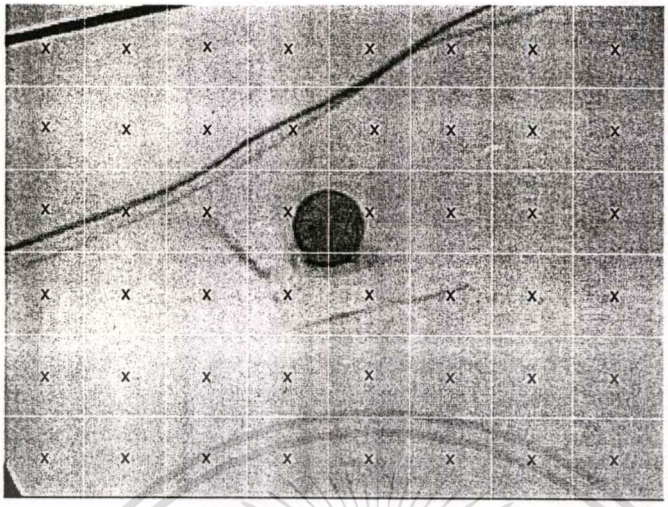
ตรวจสอบดูว่าเส้นใดยาวที่สุด เส้นนั้นน่าจะเป็นเส้นผ่านศูนย์กลางของวงกลม ซึ่งน่าจะเป็นวิธีที่มีความถูกต้องสูงแต่ใช้เวลาในการคำนวณมาก เป็นต้น

วิธีที่ใช้หาเส้นผ่านศูนย์กลางของวงกลมในภาพของการพัฒนาโปรแกรมประยุกต์ชิ้นนี้เราอาศัย วิธีแบ่งภาพที่ได้เป็นสี่เหลี่ยมที่มีพื้นที่เท่าๆกัน เริ่มจากแบ่งส่วนไม่มากก่อน แล้วเริ่มหาสีของจุดศูนย์กลางของแต่ละส่วนเมื่อพบสีที่ต้องการ (เช่น ในการทดลองใช้สีน้ำเงิน) ก็ให้ทำการหาจุดที่ติดต่อกันทั้งด้านซ้าย และ ขวาตามแนวนอน ทำทุกๆส่วน แล้วทำการเปรียบเทียบว่าเส้นตรงที่ได้เส้นใดยาวที่สุด โดยจะต้องกำหนดค่าขั้นต่ำที่ยอมได้เพราะในบางครั้งอาจพบ Noise ซึ่งจะทำให้เราตรวจสอบค่าเส้นผ่านศูนย์กลางผิดได้ เช่น ในการทดลองกำหนดให้ค่าที่ได้มากกว่า 10 ซึ่งคือ ค่าเส้นผ่านศูนย์กลางเมื่อทำการซูมวัตถุทรงกลมออกมากที่สุด (ความยาวโฟกัสสั้นที่สุด) และ กล้องห่างจากวัตถุมากที่สุดที่เรายอมได้ นำค่าที่ได้มาหารด้วย 5 เส้นที่ได้คือเส้นตรงที่ลากผ่านวงกลมจากขอบด้านหนึ่งไปยังขอบอีกด้านไปตามแนวนอน เพราะฉะนั้นถ้าเราทำการหาจุดกึ่งกลางของเส้นที่ได้ เส้นตรงที่ตั้งฉากกับเส้นตรงดังกล่าว (เส้นตรงตามแนวดิ่ง) และ ลากจากขอบของวงกลมด้านหนึ่ง ไป ยังอีกด้านหนึ่งก็คือ เส้นผ่านศูนย์กลางของวงกลมนั้นนั่นเองตามรูป 30 เส้นสีดำในภาพคือ



รูปที่ 30 แสดงอัลกอริทึมในการหาเส้นผ่านศูนย์กลางของวงกลมในภาพ

เส้นที่ลากตั้งฉากกับเส้นที่หาได้ในแนวนอน และ ลากจากขอบของวงกลมในแนวดิ่งจากด้านบนลงจนถึงด้านล่าง



รูปที่ 31 เมื่อการหาครั้งแรกล้มเหลวให้ทำการแบ่งพื้นที่ในการหาให้ละเอียดยิ่งขึ้น

ถ้าทำตามขบวนการที่กล่าวทั้งหมดแล้วไม่พบจุดสี หรือ เส้นที่ได้มีค่าต่ำกว่าค่า Threshold ที่ได้ตั้งไว้ก็ให้แบ่งที่ให้ละเอียดขึ้นแล้วทำการตามขบวนการทั้งหมดตามรูปที่ 31 แบ่งพื้นที่มากขึ้นเป็นสองเท่า เราสามารถแบ่งพื้นที่ได้มากเท่าที่ต้องการ โดยในที่สุดการแบ่งที่มากที่สุดก็คือการหาทุกจุดนั่นเอง ประสิทธิภาพในแบบนี้ดีขึ้นมากกว่าวิธีหาทุกจุดมาก

จากหัวข้อนี้ทำให้เราสามารถหาเส้นผ่านศูนย์กลางของวงกลมที่เกิดขึ้นได้ในทุกเฟรมที่ capture เข้ามาได้ ซึ่งจะใช้ประกอบกับค่าระยะห่างที่ได้จากอุปกรณ์ ISOTRAK®II

3.4 การดึงค่าตำแหน่งของกล้องวิดีโอด้วยอุปกรณ์ ISOTRAK®II

การจะหาความยาวโฟกัสจำต้องอาศัยข้อมูลจากสองส่วนคือ การตรวจหาเส้นผ่านศูนย์กลางของวัตถุทรงกลมที่ project ลงบนเฟรมของภาพ และ การรับค่าตำแหน่งของกล้องจากอุปกรณ์ ISOTRAK II จากหัวข้อที่แล้วเราสามารถหาความยาวของเส้นผ่านศูนย์กลางได้ในบทนี้เราจะกล่าวถึงการใช้อุปกรณ์ ISOTRAK®II ในการหาระยะห่างของกล้องกับวัตถุทรงกลมดังกล่าว

อุปกรณ์ ISOTRAK®II เป็นอุปกรณ์ตรวจหาค่าตำแหน่งของวัตถุที่ติด receiver (จากบทที่ 2.5) โดยการติดต่อระหว่างอุปกรณ์ ISOTRAK®II กับคอมพิวเตอร์จะติดต่อผ่าน port RS-232 โดยอุปกรณ์จะส่งค่าตำแหน่งที่เปลี่ยนแปลงมาเป็นระยะ ผู้เขียนโปรแกรมที่ต้องการใช้งานเพียงแต่เรียกใช้ฟังก์ชั่นที่จะทำการเรียกค่ามาไว้ในโปรแกรมเท่านั้นโดยใช้ VirtualHand Software library ซึ่งเป็น library ในภาษา C ดังในโปรแกรมตัวอย่าง

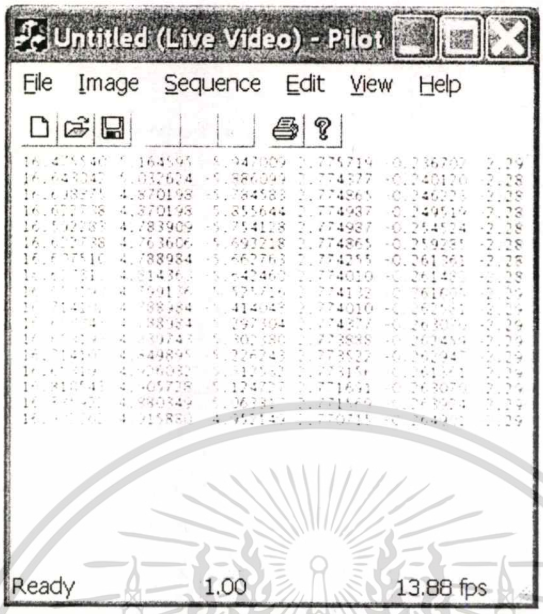
```
#include "vt_globals.h"
#include "vt_virtual_hand.h"
```

```
#include "vt_graphics_OGL.h"
#include "vt_panel.h"
```

ทำการ include library ที่ต้องการจากนั้นใช้คำสั่ง

```
tracker = vt_tracker_get();
if(tracker){
        printf("%f %f %f %f %f %f\n",
        tracker->state_vec[0][0],
        tracker->state_vec[0][1],
        tracker->state_vec[0][2],
        tracker->state_vec[0][3],
        tracker->state_vec[0][4],
        tracker->state_vec[0][5]);

} else {
        printf("Tracker 's not response.\n");
}
vt_update_hand_state();
vt_request_new_hand_data();
```

ค่าตำแหน่งที่ได้จะเป็นค่า float 6 ค่า คือ ค่าตำแหน่งในแกน x , y และ z ที่เปลี่ยนแปลง และ ค่ามุมของกล้องที่มีการเปลี่ยนแปลงดังรูปที่ 32 แสดงค่าที่ได้จากการใช้อุปกรณ์ ISOTRAK®II สามค่าแรกในรูปแสดงตำแหน่งในแนวแกน x , y และ z เทียบกับตำแหน่ง origin (x_0 , y_0, z_0) ส่วนสามค่าหลังเป็นค่ามุมหมุนที่มีการเปลี่ยนแปลงรอบแกน x , y และ z ค่าที่ได้เป็นค่าตั้ง π ถึง $-\pi$



รูปที่ 32 แสดงการเรียกค่าจากอุปกรณ์ ISOTRAK®II

3.5 การคำนวณหาความยาวโฟกัส

ตามคำนิยามเกี่ยวกับความยาวโฟกัสซึ่งได้กล่าวไว้ในหัวข้อ 2.6.1 ซึ่งวัดได้เป็นหน่วยมิลลิเมตรซึ่งเป็นค่าสมบูรณ์ (Absolute) แต่ในการทดลองนี้เราต้องเพียงค่าสัมพัทธ์ (Relative) ซึ่งต้องมีการตั้งค่าเริ่มต้นไว้ก่อนจากนั้นทำการวัดค่าที่มีการเปลี่ยนแปลงเทียบกับค่าเริ่มต้นจากสูตรที่ 11 ในหน้าที่ 40

$$\theta' = \mathrm{Tan}^{-1}(\mathrm{Tan}(\theta) * (\mathrm{Dia} * D' / \mathrm{Dia}' * D)$$

ค่า θ เป็นค่า Field of view เริ่มต้นที่ผู้ใช้ต้องตั้งค่าไว้เมื่อเริ่มใช้ application (สาเหตุที่ต้องใช้ค่า Field of view เพราะใน OpenGL กล้องเสมือนภายในมีการเปลี่ยนแปลงอัตราการซูมโดยใช้ค่า Field of view ซึ่งค่านี้แปรผันตรงกับค่าความยาวโฟกัส) ส่วนค่า Dia คือ ค่าเส้นผ่านศูนย์กลางของรูปวงกลมที่เป็นค่าเริ่มต้นที่ทำการบันทึกไว้ก่อน รวมทั้ง ค่า D คือ ค่าระยะห่างระหว่างจุด origin (ซึ่งตอนเริ่มต้นกล้องวิดีโอตั้งไว้ที่จุด origin) กับตำแหน่งของวัตถุทรงกลมที่ใช้เป็นวัตถุสำหรับการ calibrate ส่วนค่า D′ ได้จากอุปกรณ์ ISOTRAK®II ส่วนค่า Dia′ สามารถหาได้จากวิธีที่ได้กล่าวมาแล้วข้างต้น ตัวอย่าง Source code ข้างล่างเป็นฟังก์ชั่นที่ใช้ในการซูมกล้อง

```
void CPilotView::zoomVR()
{
double pi = 3.1415926535;
```

```
/* แสดงค่าเส้นผ่านศูนย์กลางที่มีการเปลี่ยนที่หน้าต่างข้างล่าง (ดูได้จากบทผลการทดลอง)
*/
                printf("m_setfocal = %f\n",m_setfocal);
                printf("oldDiameter = %f\n",oldDiameter);
                printf("newDiameter = %f\n",newDiameter)

/* ต้องการเปลี่ยนค่า m_setfocal เป็นค่าในช่วง π ถึง -π ด้วยการคูณด้วยค่า pi และ หาร
ด้วยค่า 180 */

                m_setfocal=180/pi*atan(tan(m_setfocal*pi/180)*
                oldDiameter/newDiameter * newDistance/oldDistance);

/* คำนวณโดยใช้สูตรการหาความสัมพันธ์ของ Field of view กับความยาวโฟกัส */

                printf("Theta = %f\n ",m_setfocal);
                ((CMainFrame *) AfxGetMainWnd())
                                ->CalculateScale(newDiameter,m_setfocal);
        glViewport(0,0,320,240);        // Make our viewport the whole window
        glViewport(0,0,400,300);        // Make our viewport the whole window
                                        // We could make the view smaller inside
                                        // Our window if we wanted too.
                                        // The glViewport takes (x, y, width, height)
                        // This basically means, what our our drawing boundries
        glMatrixMode(GL_PROJECTION);    // Select The Projection Matrix
        glLoadIdentity();               // Reset The Projection Matrix
                                // Calculate The Aspect Ratio Of The Window
                                // The parameters are:
                                // (view angle, aspect ration of the width to the height,
                                //  The closest distance to the camera before it clips,
```

```
                                    // FOV          // Ratio
                                    //  The farthest distance before it stops drawing)
        gluPerspective(m_setfocal,(GLfloat)320/(GLfloat)240, 0.5f, 150.0f);
        glMatrixMode(GL_MODELVIEW);      // Select The Modelview Matrix
        glLoadIdentity();                // Reset The Modelview Matrix
}
```

3.6 การสร้าง การกำหนดตำแหน่ง และ ขนาดของวัตถุสังเคราะห์สามมิติใน OpenGL

เมื่อได้ข้อมูลทั้ง ระยะห่างระหว่างกล้องกับวัตถุในโลกจริง มุมกล้อง รวมทั้งความยาวโฟกัส เราสามารถผ่านค่าทั้งหมดให้กับกล้องในโลกเสมือนที่กำลังบันทึกภาพวัตถุสามมิติที่สร้างโดย OpenGL ซึ่งใน OpenGL ได้เตรียมฟังก์ชั่นเหล่านี้ไว้แล้ว เพราะฉะนั้นเมื่อกล้องวิดีโอมีการเคลื่อนที่ เปลี่ยนมุมกล้อง หรือ มีการซูม เข้า-ออก ซึ่งเราสามารถตรวจสอบค่าเหล่านี้ได้ เราเพียงแต่ผ่านไปให้กล้องเสมือนใน OpenGL จากนั้นนำภาพที่ได้ไป transparent ลงบนเฟรมที่กำลัง capture มาได้ก็จะได้ผลลัพธ์ตามหลักการของ Augmented reality

Source code ข้างล่างแสดงตัวอย่างของฟังก์ชั่นใน OpenGL ที่เกี่ยวข้อง

```
void CPilotView::RenderScene(CDC *pDC)
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
// Clear The Screen And The Depth Buffer
glClearColor(0.0f,0.0f,1.0f,1);
glLoadIdentity();                                   // Reset The matrix
g_Camera.Look();
glPushMatrix();
glTranslatef(0, 0, 0);
glColor3ub(0, 0, 0);
glutWireTeapot(2.0);
glPopMatrix();

SwapBuffers(m_hDC);          // Swap the backbuffers to the foreground
}
```

ฟังก์ชั่น RenderScene(); ใช้สำหรับการสร้างวัตถุสังเคราะห์

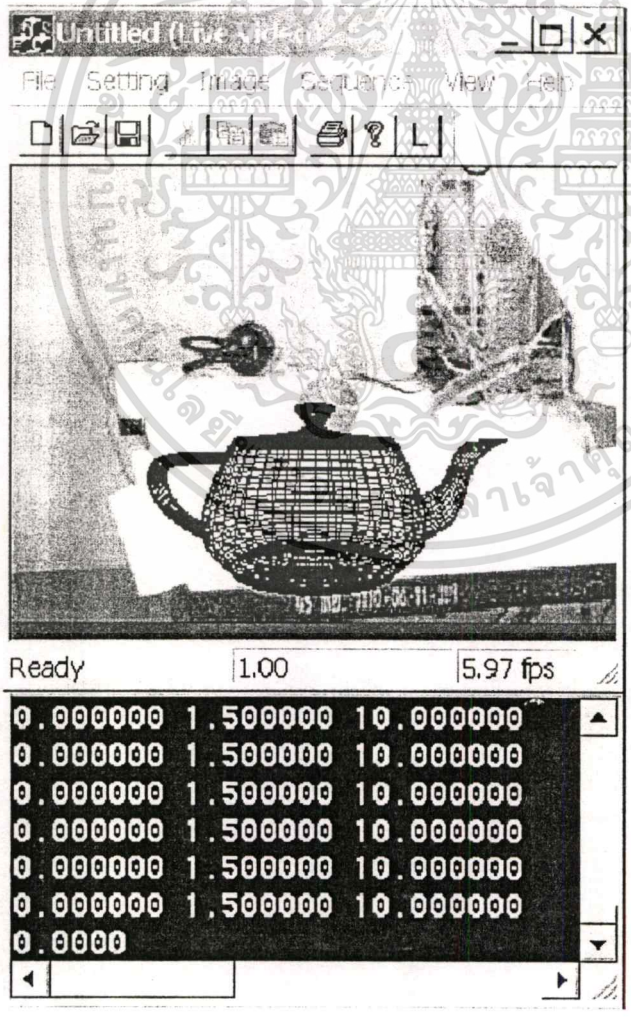
```
void CPilotView::MapTransparent(CDC *pDC)
{
        CDC memCDC;
        CBitmap bitmap;
        memCDC.CreateCompatibleDC ( pDC );
        bitmap.CreateCompatibleBitmap ( pDC ,320,240 );
        memCDC.SelectObject(&bitmap);

        memCDC.BitBlt(0, 0, 320, 240, pDC, 0, 0, SRCCOPY);

        BitBlt(pDC->m_hDC, 0, 0, 320, 240, image.Hdc(), 0, 0, SRCCOPY);
        TransparentBlt(
            pDC->m_hDC,      // handle to destination DC
            0,  // x-coord of destination upper-left corner
            0,  // y-coord of destination upper-left corner
            320,    // width of destination rectangle
            240,    // height of destination rectangle
            memCDC,      // handle to source DC
            0,   // x-coord of source upper-left corner
            0,   // y-coord of source upper-left corner
            320,     // width of source rectangle
            240,    // height of source rectangle
            RGB(0,0,255)  // color to make transparent
        );

}
```

ฟังก์ชั่น MapTransparent(); ใช้สำหรับการซ้อนภาพวัตถุสังเคราะห์ที่ผ่านการ process แล้วทำการซ้อนทับลงบนเฟรมภาพจริง

บทที่ 4

ผลการทดลอง

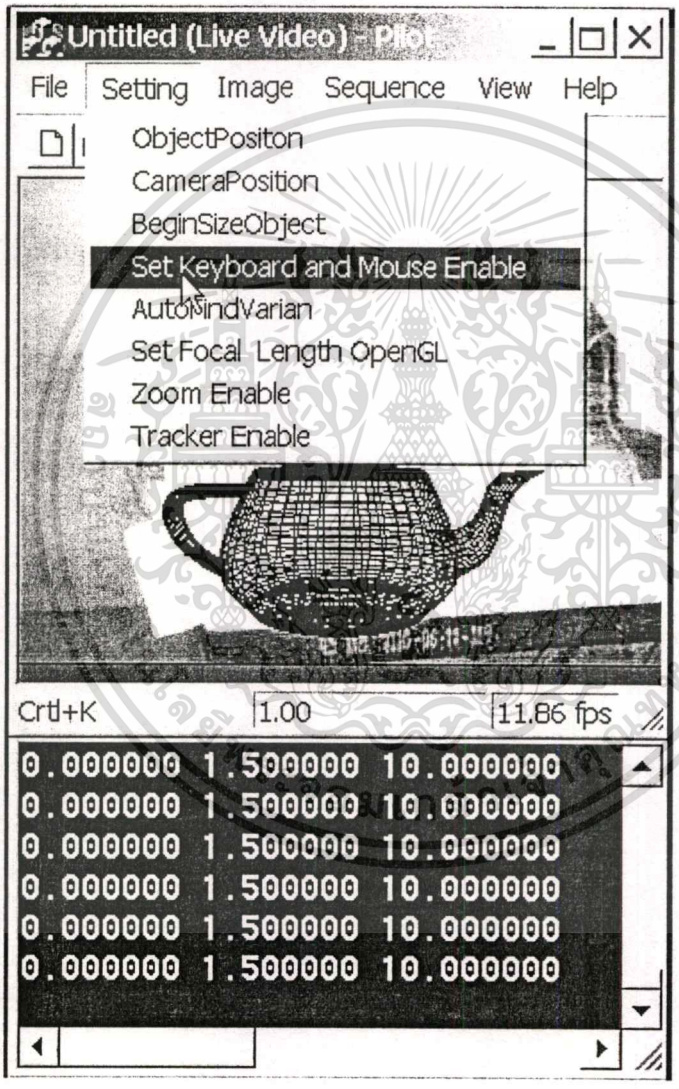
เริ่มต้นใช้งาน application จะได้ผลดังรูปที่ 33 ในรูปใช้ทรงกลมสีน้ำเงินเป็นวัตถุที่ใช้ในการ calibrate รูปกาน้ำชาสีดำเป็นวัตถุสังเคราะห์โดย OpenGL ซึ่งในตอนเริ่มต้นยังไม่ได้มีการจัดขนาด และ ตำแหน่งที่เหมาะสมของวัตถุสังเคราะห์ ซึ่งจะมีฟังก์ชั่นในการจัดต่อไป ส่วนใน



รูปที่ 33 หน้าต่างในการทำงานของ application

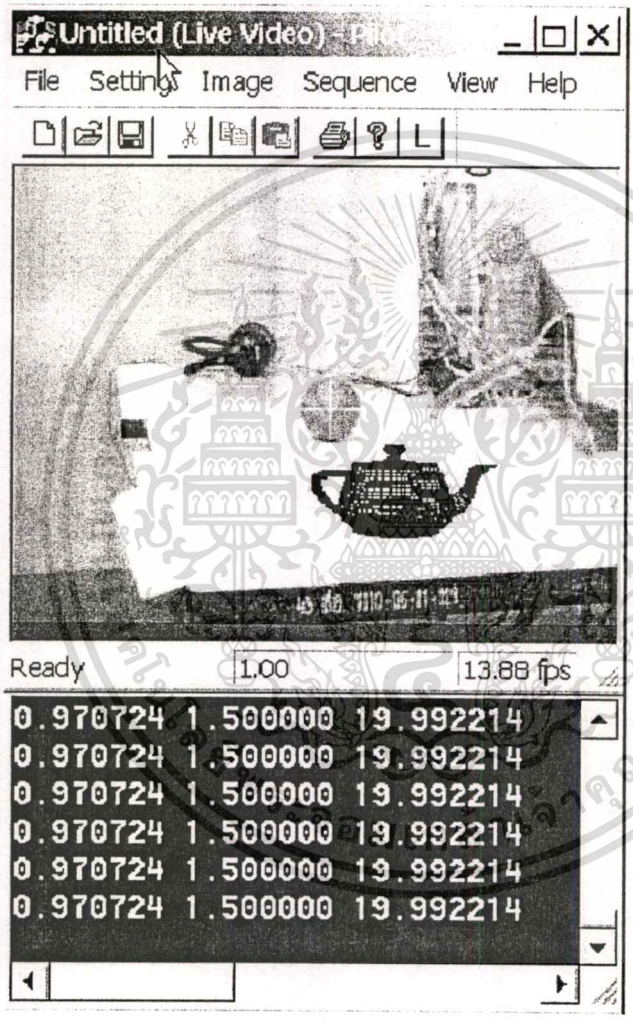
หน้าต่างข้างล่างจะแสดง ค่าต่างๆ เช่น ค่า coordinate ของกล้องวิดีโอ ค่าความเส้นผ่านศูนย์กลางที่หาได้ หรือ ค่าความยาวโฟกัสสัมพัทธ์ที่คำนวณได้ เป็นต้น

4.1 ผลการทดลองเมื่อมีการเปลี่ยนเฉพาะความยาวโฟกัส



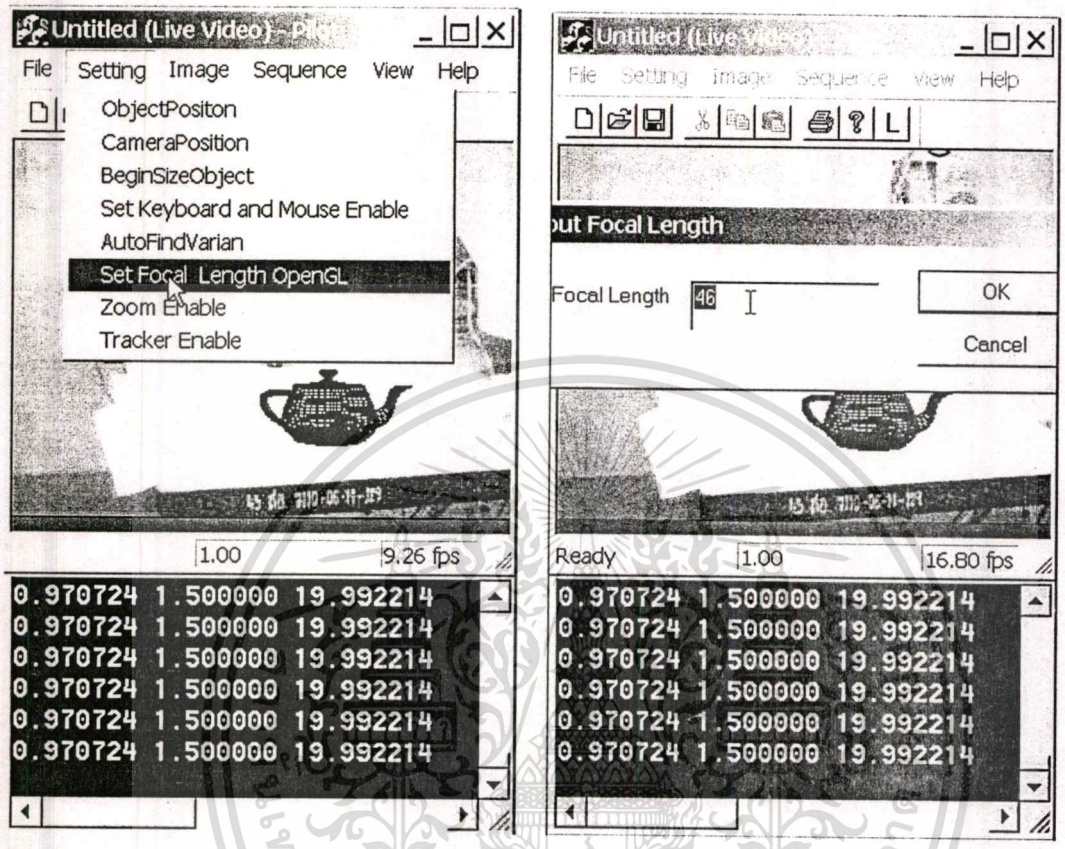
รูปที่ 34 เรียกเมนู Set Keyboard สำหรับการกำหนดขนาด และ ตำแหน่งของวัตถุสังเคราะห์ให้เหมาะสม

เมื่อเริ่มต้นใช้งานต้องมีการกำหนดตำแหน่ง และ ขนาดของวัตถุสังเคราะห์ รูปที่ 34 แสดงการจัดการที่จะกำหนดตำแหน่ง และ ขนาดของวัตถุสังเคราะห์ให้เหมาะสมและ ผลที่ได้ตามรูปที่ 35



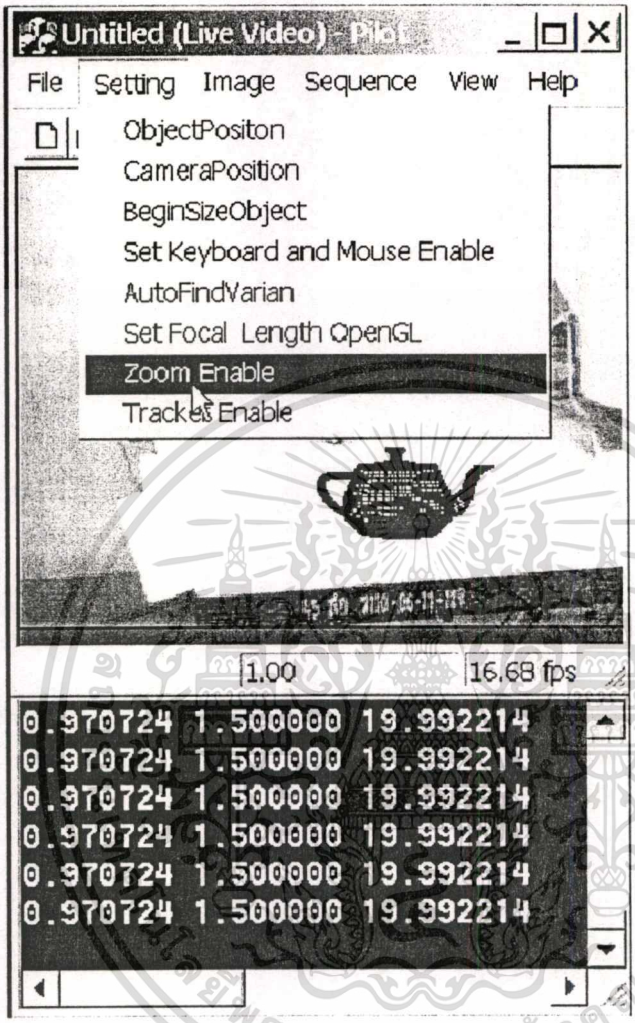
รูปที่ 35 การจัดวางวัตถุสังเคราะห์ลงในภาพ

เมื่อจัดวางวัตถุสังเคราะห์ซึ่งที่ต้องทำต่อไปคือ การกำหนดค่า Field of view (ความยาวโฟกัส) เริ่มต้นให้กับกล้องเสมือนใน OpenGL รูปที่ 36 ซ้ายมือ เลือกเมนูในการกำหนดค่า Field of view เริ่มต้น และ รูปทางขวามือ ป้อนค่าประมาณ 46 องศาซึ่งเป็นค่าปกติของเลนส์ที่มีความยาวโฟกัสขนาด 35 มิลลิเมตร



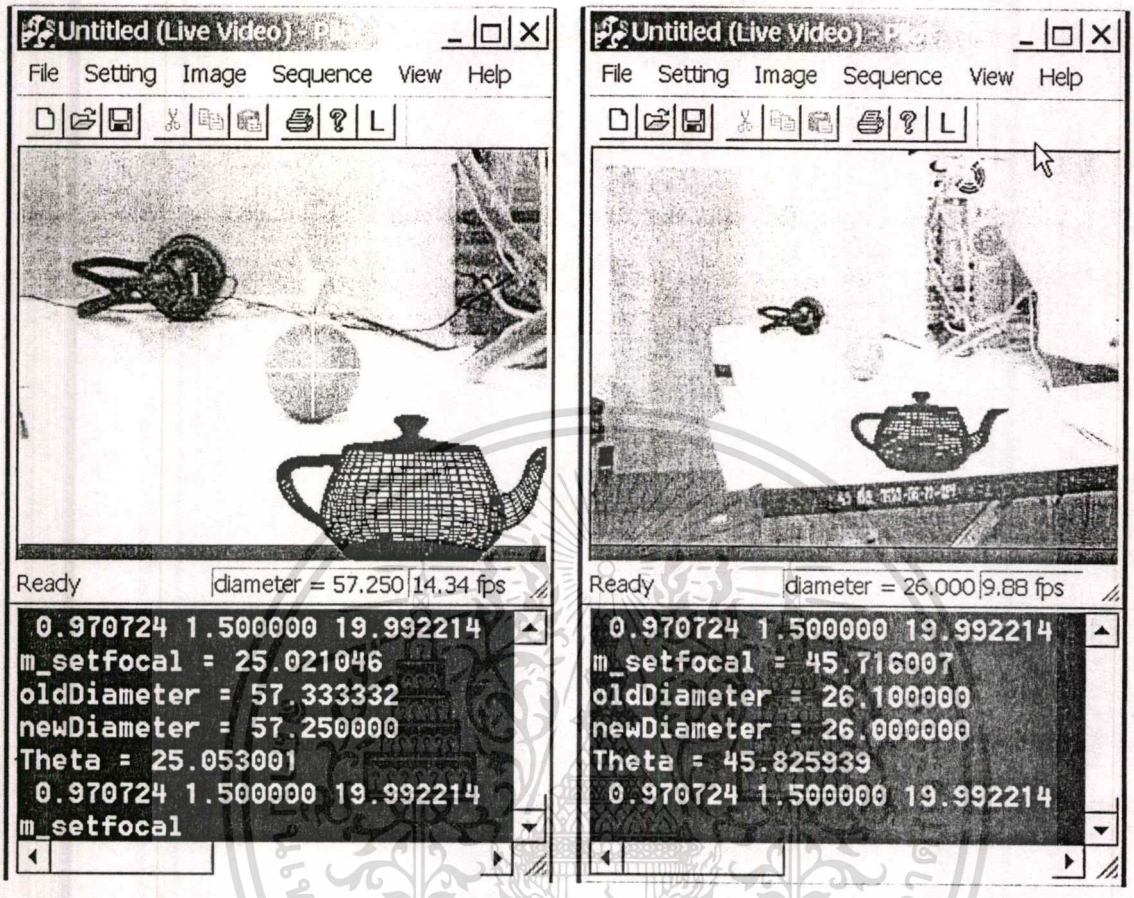
รูปที่ 36 การกำหนดค่าความยาวโฟกัสเริ่มต้น

จากนั้นเริ่มต้นการสั่งให้ระบบสามารถตอบสนองต่อการเปลี่ยนความยาวโฟกัสของกล้องวิดีโอ (ก็คือ การซูมเข้า-ออกนั่นเอง) ตามรูปที่ 37 และ ผลที่ได้ปรากฏในรูปที่ 38 ทางซ้ายมือ คือ ผลที่ได้เมื่อมีการซูมออก และ ทางขวามือ คือ ผลที่ได้จากการซูมเข้า



รูปที่ 37 เลือกเมนูให้ application มีการตอบสนองต่อการเปลี่ยนแปลงความยาวโฟกัส

ผลที่ได้จากการซูมกล้องออก ปรากฏดังรูปที่ 38 ทางซ้าย ภาพกาน้ำมีขนาดใหญ่ขึ้น และ ที่หน้าต่างล่างแสดง ค่าเส้นผ่านศูนย์กลางที่ตรวจพบมีขนาดใหญ่ ซึ่งมีผลทำให้ค่าความยาวโฟกัสมีขนาดมากขึ้น (แต่เนื่องจากค่าที่แสดงเป็นค่า Field of view มีค่าเป็นองศา และ ค่า Field of view นี้แปรผกผันกับค่าความยาวโฟกัส เพราะฉะนั้นจึงปรากฏว่า ค่า Field of view มีขนาดลดลง สามารถศึกษาความสัมพันธ์แบบแปรผกผันได้ในบทที่สอง) ส่วนรูปทางขวามือผลที่ได้ตรงข้ามกัน



รูปที่ 38 ผลที่ได้เมื่อมีการเปลี่ยนแปลงความยาวโฟกัสของกล้องวิดีโอ

จากตารางที่ 4 แสดงผลในเชิงตัวเลขเปรียบเทียบค่าพารามิเตอร์ต่างๆ เมื่อมีการเปลี่ยนความยาวโฟกัสไปมา

ค่า Parameter ต่างๆเมื่อมีการซูมออก	
oldDiameter	57.333332
newDiameter	57.250000
Theta	25.053001
Frame rate	14.34 fps

ค่า Parameter ต่างๆเมื่อมีการซูมเข้า	
oldDiameter	26.100000
newDiameter	26.000000
Theta	45.825939
Frame rate	9.88 fps

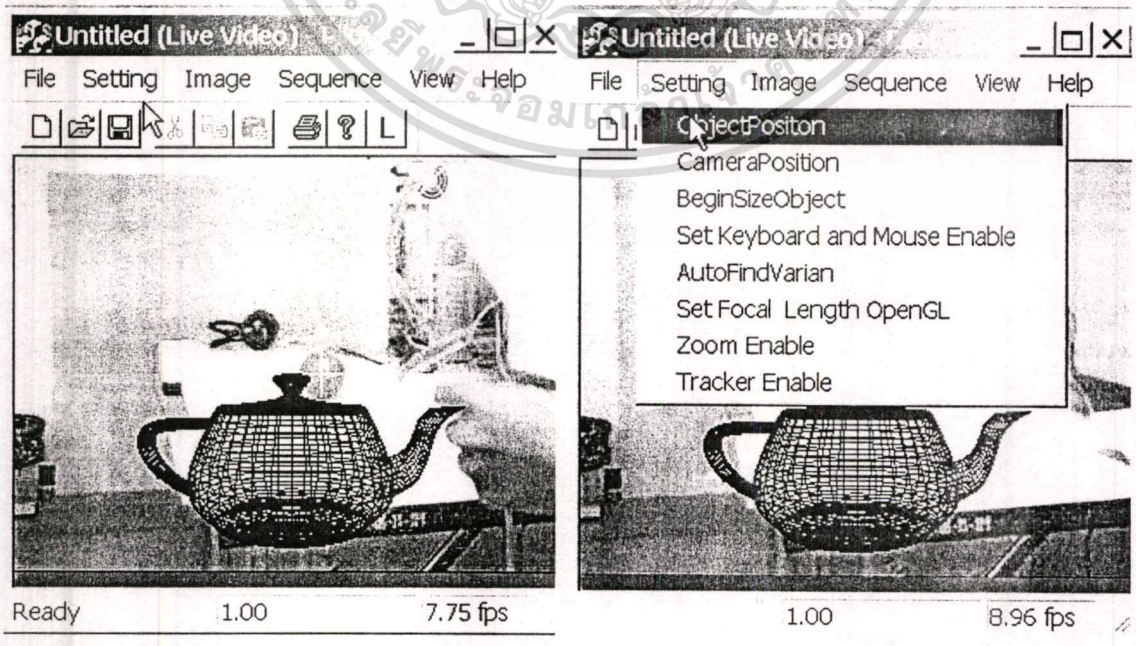
ตารางที่ 4 เปรียบเทียบผลเมื่อมีการเปลี่ยนแปลงความยาวโฟกัส

ค่า oldDiameter คือ ค่าเส้นผ่านศูนย์กลางของเฟรมก่อนๆ และ ค่า newDiameter คือ ค่าเส้นผ่านศูนย์กลางของเฟรมปัจจุบัน เราใช้ buffer ในการแก้ปัญหาเรื่องการไม่นิ่งของค่าที่วัดมาได้ จะเห็นได้ว่าเมื่อมีการเพิ่มขนาดความยาวโฟกัส กล่าวคือ ขนาดของวัตถุจะมีขนาดใหญ่ขึ้น ตามรูป 38 ซ้ายมือ และ ค่าเส้นผ่านศูนย์กลางที่วัดได้มีขนาดใหญ่ด้วยคือ เท่ากับประมาณ 57 pixels ตามตาราง ซ้ายมือ ซึ่งส่งผลให้ค่า Theta (Field of view) ที่คำนวณได้มีค่าลดลง ซึ่งถูกต้องตามทฤษฎีที่กล่าวมา และ ผลที่ได้นี้เปรียบเทียบกับการลดความยาวโฟกัสในตารางด้านขวามือ

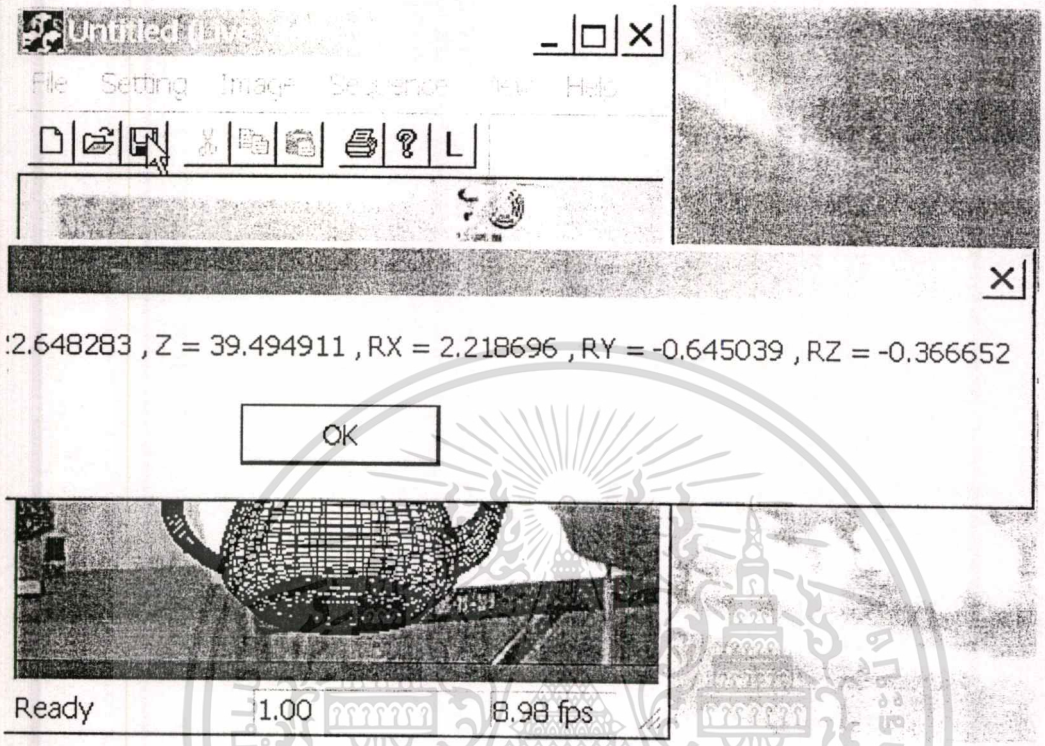
มีผลที่น่าสนใจอีกอย่างคือ ค่า frame rate มีการเปลี่ยนแปลงเมื่อมีการลดขนาดความยาวโฟกัส กล่าวคือ เมื่อมีการลดขนาดความยาวโฟกัสรูปทรงกลมจะมีขนาดเล็กลงทำให้การตรวจหาต้องกระทำหลายรอบมากขึ้นจากอัลกอริทึมในหัวข้อที่ 3.31 frame rate ลดลงจาก 4.34 frame per second ไปเป็น 9.88 frame per second

4.2 แสดงผลการทำงานเมื่อสั่งให้ระบบ tracker ทำงาน

เริ่มต้นนำ receiver ของชุดอุปกรณ์ ISOTRAK®II ไปแตะที่วัตถุทรงกลมที่ใช้เป็นตัว calibrate ตามรูปที่ 39 จุด origin คือ ตำแหน่งที่ transmitter ตั้งอยู่ จากนั้นทำการบันทึกค่าไว้ตามรูปที่ 40 ซึ่งระยะนี้จะคงที่ตลอดเพราะ transmitter และ วัตถุไม่ได้เคลื่อนเลยตลอดการใช้งาน กล้องวิดีโอเป็นอุปกรณ์ที่มีการเคลื่อนที่ตลอดเวลา เพราะฉะนั้นพอเราติดตั้ง receiver ก็จะสามารถอ้างอิงระยะห่างระหว่างวัตถุกับกล้องวิดีโอได้ตลอดเวลา

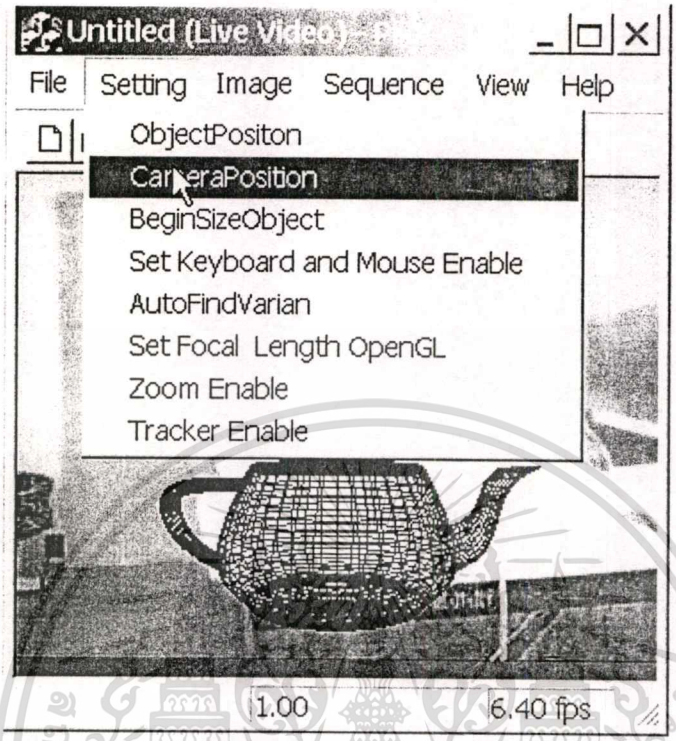


รูปที่ 39 ตรวจหาระยะระหว่างวัตถุที่ใช้ในการ calibrate กับ จุด origin



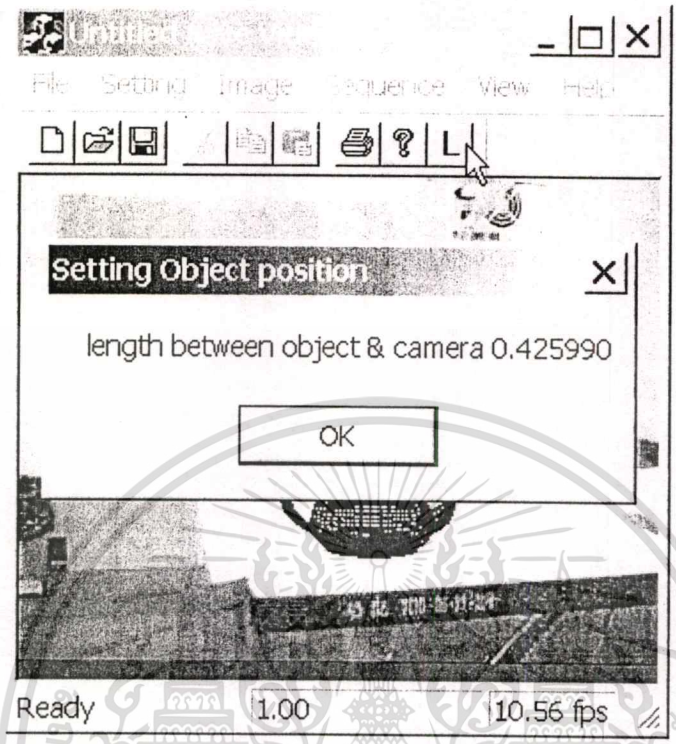
รูปที่ 40 บันทึกระยะห่างระหว่างวัตถุที่ใช้ในการ calibrate กับ จุด origin

จากนั้นนำ receiver ไปติดตั้งที่กล้องวิดีโอแล้วทำการบันทึกตำแหน่งเริ่มต้นซึ่งจะเป็นการบันทึกระยะห่างของกล้องเทียบกับวัตถุ และ บันทึกขนาดเส้นผ่านศูนย์กลางเริ่มต้นไว้ด้วย นั่นคือ ค่า D และ ค่า Dia ที่จะนำไปคำนวณหาความยาวโฟกัสจากสูตรที่กล่าวมาแล้วนั่นเอง

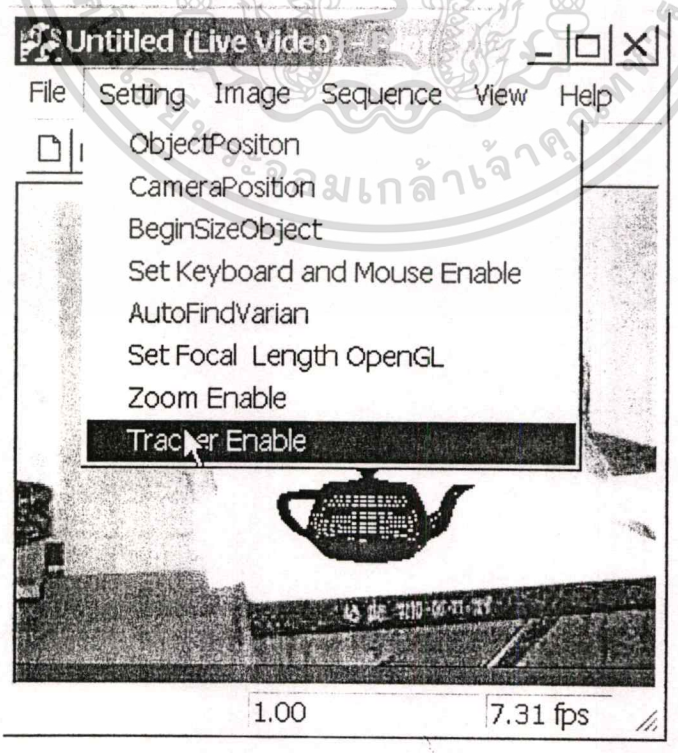


รูปที่ 41 ทำการบันทึกค่าระยะห่างระหว่าง กล้อง และ วัตถุเริ่มต้น

ลองตรวจสอบค่าดูโดยเลือก Tool bar L จะได้ค่าดังรูปที่ 42 จากนั้นทำการวางตำแหน่งของวัตถุสังเคราะห์เหมือนในผลการทดลองที่แล้ว จากนั้นทำการเปิดระบบ Tracker ตามรูปที่ 43

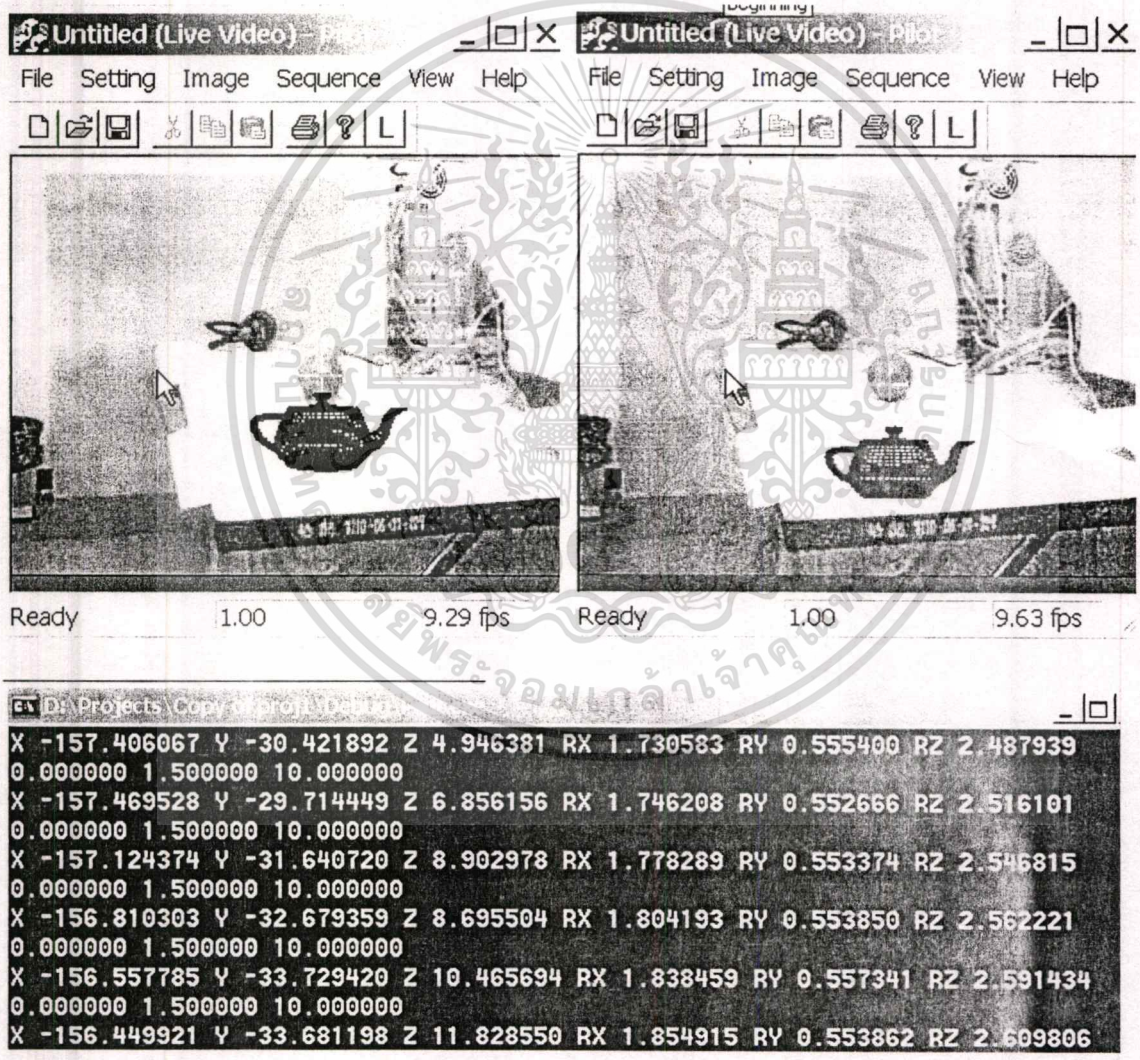


รูปที่ 42 ฟังก์ชั่นตรวจสอบระยะห่าง



รูปที่ 43 ฟังก์ชั่นการ Tracker

ผลที่ได้ตามรูปที่ 44 ตำแหน่งของกาน้ำซึ่งเป็นวัตถุสังเคราะห์มีการเปลี่ยนแปลงตำแหน่งตลอดเวลา ทั้งๆ กล้องถ่ายวิดีโออยู่นิ่งบนขาตั้งกล้อง ซึ่งสาเหตุมาจากการที่อุปกรณ์ ISOTRAK®II ส่งค่าที่มีการเปลี่ยนแปลงตลอดเวลา ดูได้จากหน้าต่างด้านล่างของรูปที่ 44 ค่าที่ได้มีการเปลี่ยนแปลงตลอดเวลาทั้งๆที่ กล้องถ่ายวิดีโออยู่บนขาตั้งกล้องตลอดเวลา เพราะฉะนั้นค่านี้จะถูกส่งไปให้กล้องเสมือนใน OpenGL ซึ่งจะทำให้กล้องเสมือนมีการเคลื่อนที่ตลอดเวลาซึ่งทำให้ภาพต่อเนื่องที่แสดงออกมามีความผิดพลาด



รูปที่ 44 ผลลัพธ์ที่ได้จากการเปิดระบบ Tracker

บทที่ 5

สรุปและแนวทางการพัฒนาในอนาคต

จากผลการทดลองสรุปได้ว่า

1. ค่าที่วัดได้จากการหาเส้นผ่านศูนย์กลางมีความแปรปรวนในระดับหนึ่งจะทำให้ค่าความยาวโฟกัสที่ได้ไม่นิ่งพอ กล่าวแม้กล้องไม่ได้มีการเปลี่ยนแปลงความยาวโฟกัส ภาพของวัตถุสังเคราะห์กับมีการเปลี่ยนแปลงขนาดไปมาได้เอง ทั้งผลน่าจะมาจากเซ็นเซอร์รับแสงของกล้อง รวมทั้งสภาพแสงภายในห้องทดลองมีการเปลี่ยนแปลง ซึ่งในการพัฒนาระบบงานนี้ เราแก้ไขด้วยวิธีการ buffer ผลที่คำนวณได้ก่อนนำค่าไปใช้ ให้ทำการตัดค่าสูงสุด และ ค่าต่ำที่สุด ที่น่าสงสัยออก วิธีการนี้มีประสิทธิพอควร โดยถ้าต้องการความเรียบของภาพต่อเนื่องต้องมีการ buffer ค่าไว้มาก แต่ข้อเสียที่ตามมาคือ frame rate ในการแสดงผลจะตกต่ำลงซึ่งเป็นข้อควรคำนึงที่สำคัญอย่างหนึ่งในงานแบบ real time จากการทดลองพบว่าค่าที่เหมาะสมคือ ประมาณ 5 ค่า
2. ภาพวัตถุสังเคราะห์ที่ได้ยังขาดความสมจริง กล่าว ลักษณะทางการให้แสง และ เงา รวมทั้งความสามารถในการรับรู้สภาพแวดล้อม อย่างเช่น กาน้ำไม่มีความสามารถในการปรับตัวให้เข้ากับพื้นโต๊ะซึ่งทำให้ภาพดูไม่เหมือนจริง
3. ปัญหาที่เกิดจากอุปกรณ์ ISOTRAK®II ให้ค่าไม่คงที่ สาเหตุน่าจะมาจากหลักการทำงานของอุปกรณ์ที่ใช้คลื่นแม่เหล็กไฟฟ้า ดังนั้นการใช้อุปกรณ์หลายๆ อย่างในห้องแล็ปจึงอาจทำให้เกิดการกวนกันของสัญญาณได้

ในส่วนการปรับปรุงแก้ไขระบบงานในอนาคตมีดังนี้

1. ศึกษาทดลองใช้ระบบการหาตำแหน่งระบบอื่นๆ เช่น ระบบที่ใช้แสงพิเศษในการติดตามตำแหน่ง
2. ศึกษาวิธีการหาทิศทางจากภายในภาพโดยตรง วิธีต้องอาศัยขบวนการทาง image processing สูงทำให้ต้องเสียเวลาในการคำนวณ ดังนั้นต้องอาศัยอุปกรณ์ เช่น คอมพิวเตอร์ที่มีประสิทธิภาพสูง เป็นต้น ซึ่งมีข้อเสีย ก็คือทำให้มีค่าใช้จ่ายเพิ่มขึ้นมาก
3. ใช้คอมพิวเตอร์หลายเครื่องมาช่วยในการทำงาน คือ ใช้การประมวลผลแบบกระจาย เช่น แบ่งหน้าที่ของคอมพิวเตอร์กันทำงาน ให้เครื่องหนึ่งทำหน้าที่ capture อีกเครื่อง

ทำหน้าที่คำนวณ อีกเครื่องทำหน้าที่สร้างวัตถุสามมิติ และ เครื่องสุดท้ายทำหน้าที่แสดงผล เป็นต้น

4. เพิ่มความสามารถของวัตถุสามมิติ เช่น ในเรื่องการรับรู้สภาวะแวดล้อม การทับบังกันของวัตถุ การให้แสง และ เงาแก่วัตถุ เป็นต้น
5. ต้องทำให้ application มีความคงทน กล่าวสามารถนำ application ไปใช้ ณ. สภาพแวดล้อมแบบใดก็ได้ไม่จำเป็นต้องใช้ในห้องแล็ป ก็ยังคงมีประสิทธิภาพเท่าเดิม

บรรณานุกรม

Azuma, R. T. 1997. "A Survey of Augmented Reality." **Teleoperators and Virtual Environments.** 6(4): 355-385.

Kutulakos, K. N. and Vallino J. R. 1998. "Calibration-Free Augmented Reality." **IEEE Transaction on visualization and computer graphics.** 4(1):1-20

Shivaram, G. and Seetharaman, G. **A New Technique for Finding the Optical Center of Cameras.** [Online]. Available: http://citeseer.nj.nec.com/345537.html

Vision Technology Group, Microsoft Research. 2000. **The Microsoft Vision SDK Version 1.2, May 2000.** [Online]. Available: http://research.microsoft.com/projects/VisSDK/VisSDK.doc.

Baxes, G. 1994. **Digital Image Processing : Principles and Applications**. New York: John Wiley & Sons.

Kruglinski, D. J. 1997. **Inside Visual C++.** 4th edition. Redmond: Microsoft Press.

ภาคผนวก

ภาคผนวก ก

ทรัพยากรที่ใช้ในการพัฒนาโปรแกรม

ซอฟต์แวร์(Software)

- Microsoft Visual C++ Version 6.0
- Microsoft Vision System Development Kit (MS Vision SDK) Version 1.2
- DirectX Version 8.0
- Microsoft Windows XP professional
- VirtualHand® Software library
- Open GL

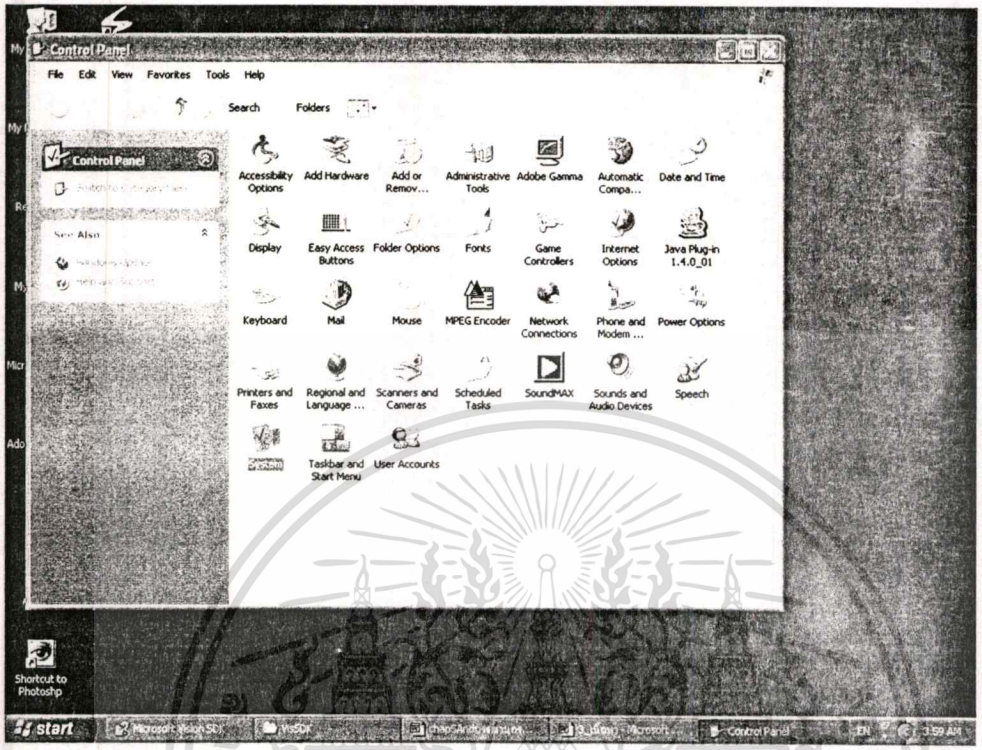
ฮาร์ดแวร์(Hardware)

- CPU Pentium 4 1.8 GHz
- RAM 128 MB
- กล้องวิดีโอ Sony TRV-140E ต่อผ่าน USB port
- สาย USB ขนาด 3.5 เมตร
- ISOTRAK II อุปกรณ์ตรวจหาทิศทาง

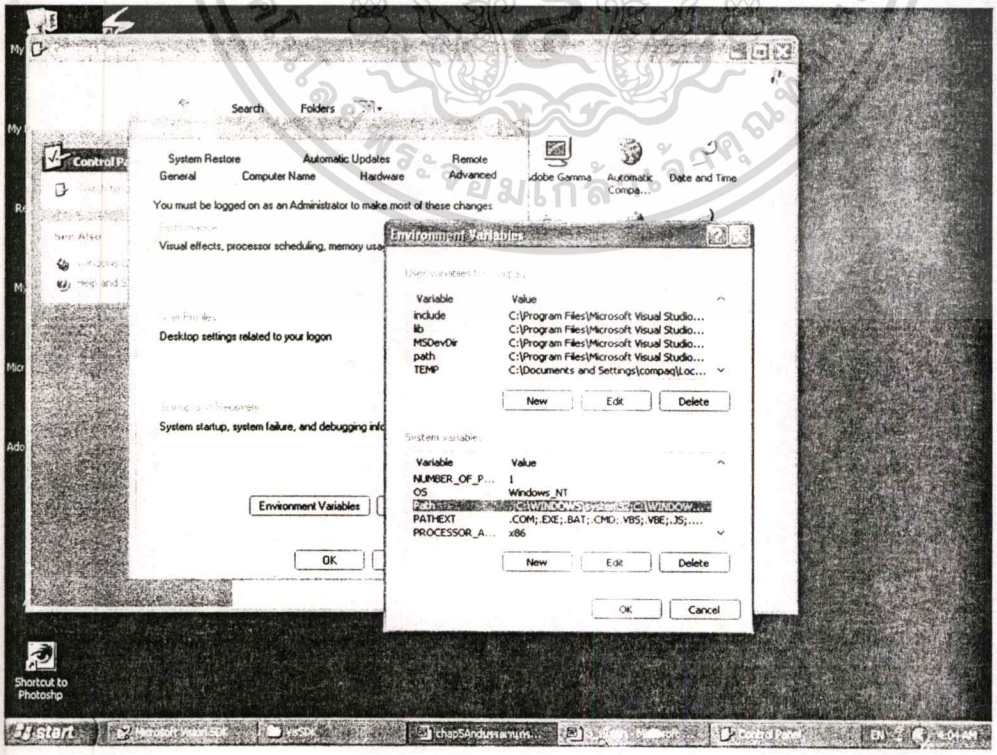
การติดตั้งโปรแกรม MS Vision SDK

MS Vision SDK Version ล่าสุดคือ Version 1.2 สามารถ download ได้ที่ http:/ www.research.microsoft.com research vision จะได้ file VisSDK_full.zip เมื่อทำการ extract zip file ดังกล่าว MS Vision SDK จะทำการสร้าง directory C:\Projects และ subdirectory C:\Projects\VisSDK จากนั้นต้องทำการกำหนดค่าต่างๆดังนี้

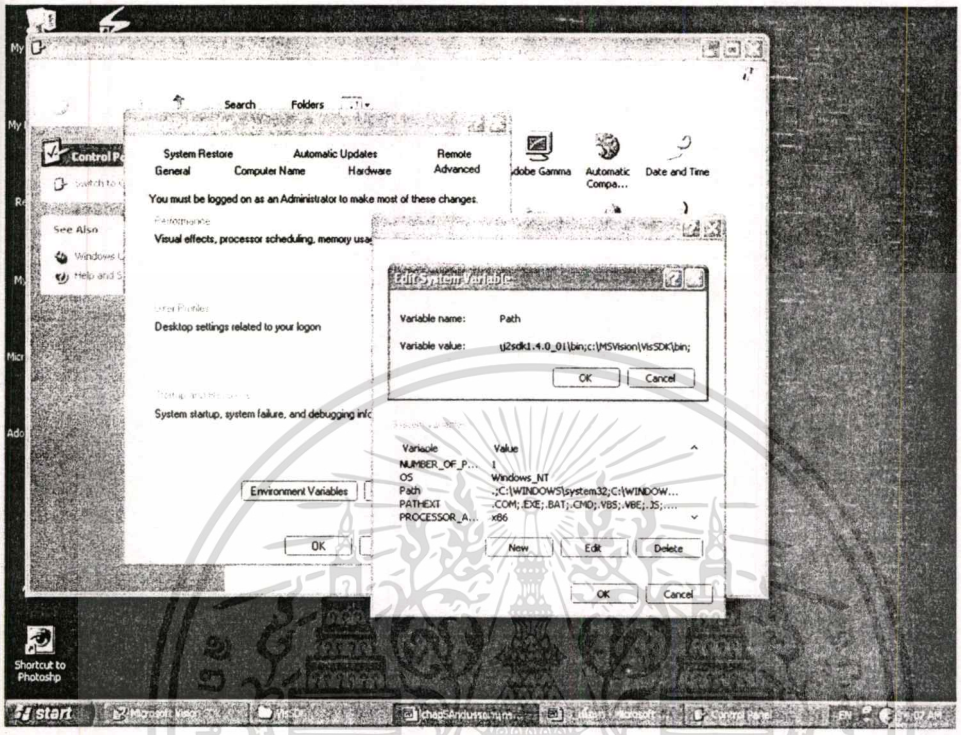
1. เพิ่ม directory bin ของ project VisSDK (เช่น C:\Projects\VisSDK\bin) เข้าไปที่ system path กรณี Windows NT ให้ไปที่ control panel จากนั้นเลือก icon System ที่ System เลือก tab Advanced จากนั้นให้เลือก tab Environment Variables ไปที่ console System Variables เลือก path แล้วทำการเพิ่ม C:\Projects\VisSDK\bin; ต่อท้าย ดังรูป 45-51



รูปที่ 45 เลือก Systems ใน control panel



รูปที่ 46 กำหนด Path โดยเลือกจาก Environment Variables

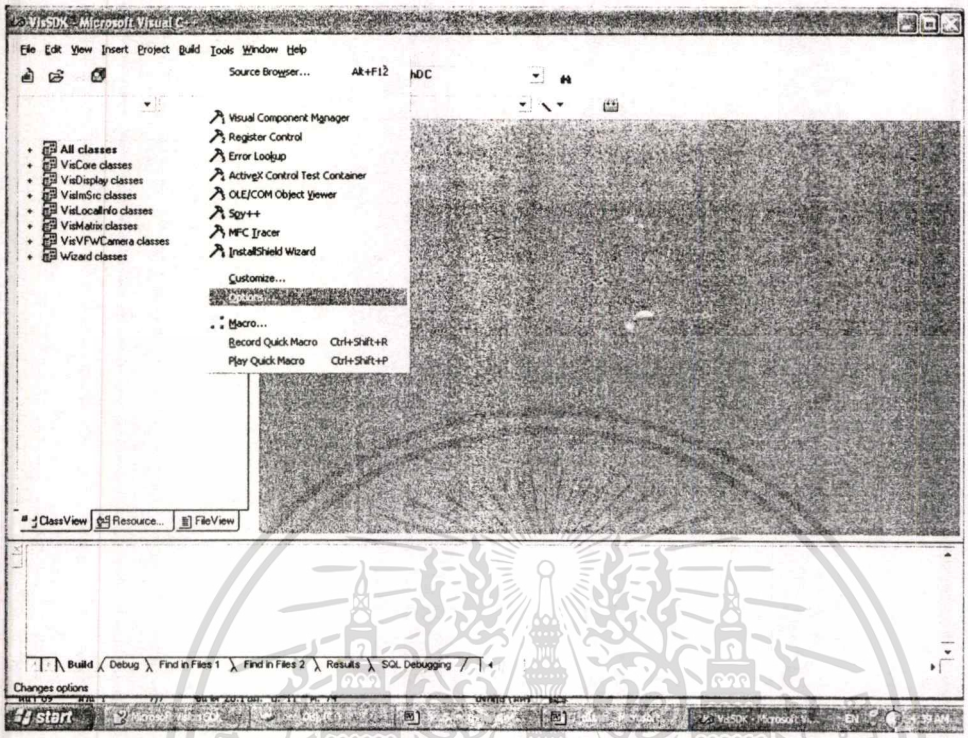


รูปที่ 47 เพิ่ม C:\Projects\VisSDK\bin; เข้าไปที่ path

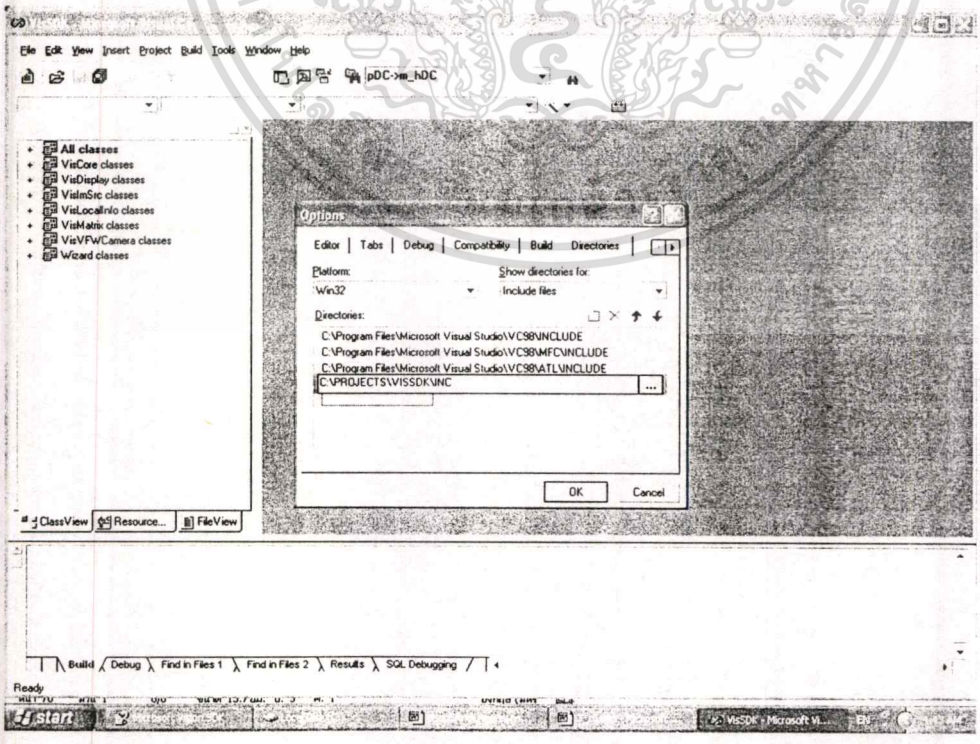
ส่วนกรณี Windows 9x ให้เพิ่ม PATH variable เข้าไปที่ autoexec.bat file

2. เปิด workspace (VisSDK.dsw) ใน Visual C++ (อยู่ที่ C:\Projects\VisSDK\VisSDK.dsw)

3. เลือก tab Options จากเมนู Tools จากนั้นเลือก tab directories จาก dialogue Options เพิ่ม “inc” directory (C:\Projects\VisSDK\inc) ที่ include file และเพิ่ม “lib” directory (C:\Projects\VisSDK\lib) library file ด้วยวิธีเดียวกัน ดังแสดงในรูป xx-yy

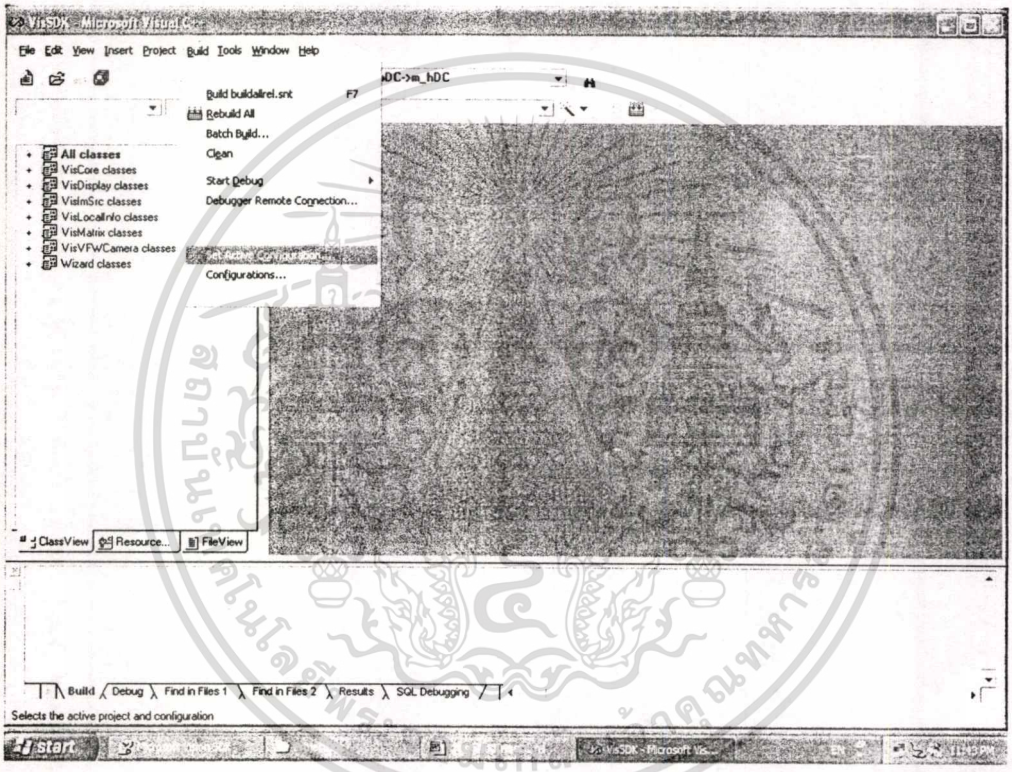


รูปที่ 48 เลือก Options จากเมนู Tools ใน work space ของ VisSDK

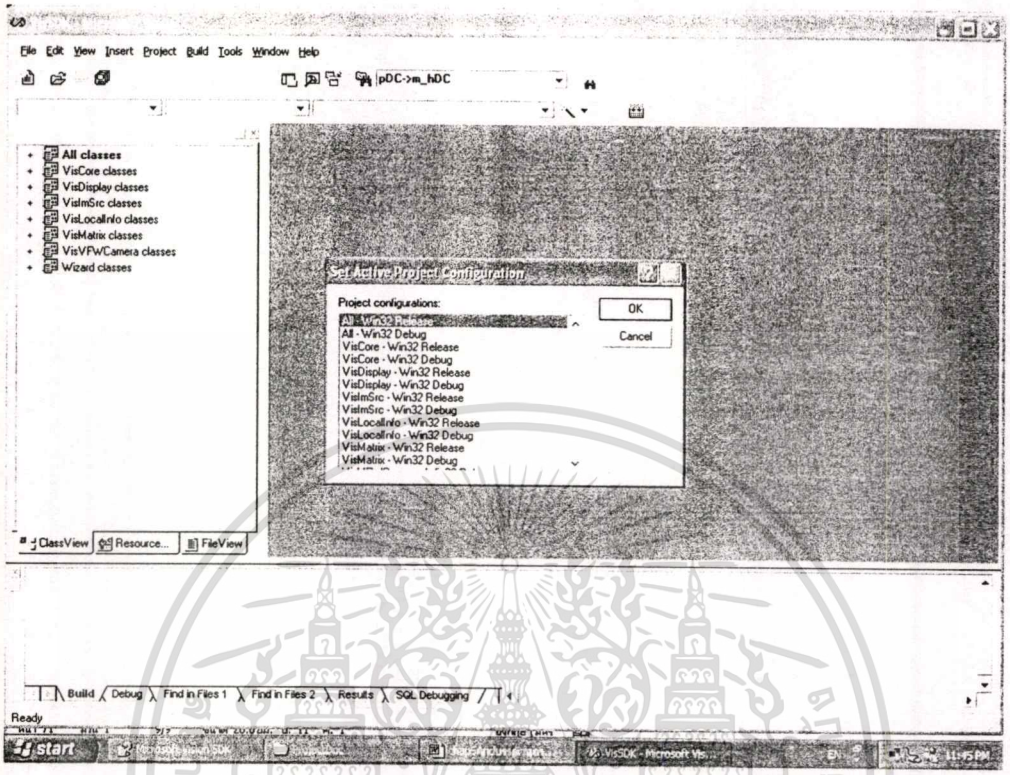


รูปที่ 49 กำหนด “inc” path ที่ include files

4. Build Debug และ Release versions ของ VisSDK ด้วยการเลือก Build ที่เมนูบาร์ จากนั้นเลือก tab Set Active Configuration จะปรากฏ dialogue Set Active Configuration ให้เลือก ALL-Win32 Debug ในช่อง Projects Configuration จากนั้นทำการ Rebuild โดยเลือก tab Rebuild All จากเมนู Build และ ทำวิธีการเดียวกันใน ALL-Win32 Release วิธีการทั้งหมดแสดงในรูป xx-yy



รูปที่ 50 แสดงการ Set Active Configuration



รูปที่ 51 แสดงการ Rebuild ALL-Win32 Debug

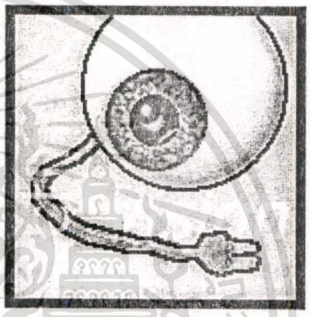
5. ในโครงการพัฒนาระบบงานนี้มีการใช้ digitizer (ในที่นี้คือ กล้องถ่ายวิดีโอ) สำหรับการ capture ภาพสดๆ จึงต้องมีการติดตั้งค่า system registry ให้กับระบบปฏิบัติการ

- Video For Windows (VFW) ดับเบิ้ลคลิ้กที่ file VisVFWCamera.reg ซึ่งอยู่ใน directory VisVFWCamera เพื่อเพิ่มข้อมูล VisVFWCamera DLLs ให้กับ system registry
- DirectShow (อยู่ใน driver DirectX 8.0) ดับเบิ้ลคลิ้กที่ file VisXDSReg.bat ซึ่งอยู่ใน directory VisXDS เพื่อเพิ่มข้อมูลเกี่ยวกับ VisXDS และ VisXRenderFil DLLs ให้กับ system registry
- ถ้ามีการใช้ กล้องแบบอนาล็อก ต่อผ่าน capture card เช่น Metrox® , Meteor® หรือ MeteorII® ต้องมีการ Build Debug และ Release ของ VisMeteor Project จากนั้นให้ดับเบิ้ลคลิ้กที่ file VisMeteor.reg

ภาคผนวก ข

แผนผังคลาสภายใน MS Vision SDK

- Classes & Related Functions
  - VisCore project
    - CVisEnumPixel
      - CVisEnumPixelOf
    - CVisError
      - CVisFileIOError
      - CVisMemoryError
    - CVisFileHandler
      - CVisFileHandlerMagick
      - CVisPPMFileHandler
      - CVisPSFileHandler
    - CVisImageBase
      - CVisImage
    - CVisMemBlock
      - CVisMemBlockOf
    - CVisPump
    - CVisRefCntMemBlock
      - CVisRefCntMemOf
    - CVisRGBA
    - CVisSDStream
    - CVisSequenceBase
      - CVisSequence
    - CVisShape
    - CVisYUVA
    - VisCore functions
      - VisErrorCodeToMessage
      - VisErrorMessageToCode
      - VisIsSigned
      - VisMap functions
      - VisPixFmtGetTPixel
      - VisRangeMax
      - VisRangeMin
  - VisDisplay project
  - VisImSrc project
  - VisMatrix project
  - VisMeteor project
  - VisVFWCamera project
  - VisXImageMagick project



Vision SDK Version 1.2

- Classes & Related Functions
  - + VisCore project
  - − VisDisplay project
    - CVisPane
    - CVisPaneArray
    - − VisDisplay functions
      - VisDisplayImage
      - VisDisplayImages
  - − VisImSrc project
    - CVisDlgListSz
    - CVisImageFromImSrcFrame
    - CVisImagePump
    - CVisImageSource
    - CVisListSz
    - IVisImSrcDevice
    - IVisImSrcProvider
    - IVisImSrcSettings
    - IVisListSz
    - − VisImSrc functions
      - Image Encoding functions
      - Bit-Depth Conversion functions
      - RGB Conversions
      - YUV Conversions
      - Image Source functions
      - Registry Key functions
      - Registry Entry functions
      - Registry Read/Write
      - Inline Registry functions
  - + VisMatrix project
  - + VisMeteor project
  - + VisVFWCamera project
    - VisXImageMagick project
- + Type Definitions
- + Enumerated Constants

- Classes & Related Functions
  - + VisCore project
  - + VisDisplay project
  - + VisImSrc project
  - - VisMatrix project
    - CVisDMatrix
    - CVisDVector
    - CVisFrame
    - CVisFrameSequence
    - CVisTransform4x4
    - CVisTransformChain
    - CVisVector4
    - - VisMatrix functions
      - VisBackInvisiblePixels
      - VisMakePixelsInvisible
      - VisMatrixEQConstrainedLS
      - VisMatrixLeastSquares
      - VisMatrixSolve
      - VisMatrixSolveQR
      - VisMatrixSolveSPD
      - VisMatrixSqrt
      - VisMatrixSqrtInverse
      - VisMatrixSVD
      - VisMaxEigenValue
      - VisMinEigenValue
      - VisMinMaxEigenValue
  - + VisMeteor project
  - + VisVFWCamera project
  - VisXImageMagick project
- + Type Definitions
- + Enumerated Constants

- Classes & Related Functions
  - + VisCore project
  - + VisDisplay project
  - + VisImSrc project
  - + VisMatrix project
  - - VisMeteor project
    - CVisMeteor
    - CVisMeteorProvider
    - - VisMeteor functions
      - VisGetImSrcProvider
  - - VisVFWCamera project
    - CVisVFWCamera
    - CVisVFWProvider
    - - VisVFWCamera functions
      - VisGetImSrcProvider
  - VisXImageMagick project
- + Type Definitions
- + Enumerated Constants

- Classes & Related Functions
  - + VisCore project
  - + VisDisplay project
  - + VisImSrc project
  - + VisMatrix project
  - + VisMeteor project
  - + VisVFWCamera project
  - VisXImageMagick project
- Type Definitions
  - Gray Pixel Types
  - RGBA Pixel Types
  - YUVA Pixel Types
  - Image Types
- Enumerated Constants
  - EVisConvYUV
  - EVisError
  - EVisIColor
  - EVisImEncoding
  - EVisImInf
  - EVisImOpt
  - EVisImSrc
  - EVisKeyList
  - EVisMemBlock
  - EVisMouse
  - EVisNormalize
  - EVisPad
  - EVisPane
  - EVisPaneDisp
  - EVisPixFmt
  - EVisSequence
  - EVisVidDlg

ประวัติผู้เขียน

ชื่อ นามสกุล	นายวิชิต นันทรัตนพงศ์
เกิดวันที่	10 พฤศจิกายน 2510
ที่อยู่	11/7 หมู่ 5 ต.โสนลอย อ.บางบัวทอง จ.นนทบุรี 11110
สถานที่เกิด	จังหวัด นนทบุรี
การศึกษา	เภสัชศาสตร์บัณฑิต มหาวิทยาลัย ศิลปากร
การทำงาน	เภสัชกร ประจำร้าน บ้านยา