บทคัดย่อ

ความสามารถในการหาความลึกจากภาพ มีประโยชน์ในการบ่งชี้ตำแหน่งของวัตถุที่สนใจในภาพ เมื่อทราบถึงตำแหน่งก็จะจัดการกับวัตถุดังกล่าวได้ เช่น ทำการเคลื่อนย้ายวัตถุ เป็นต้น ซึ่งจะเห็นได้ว่าความสามารถในการหาความลึกดังกล่าวสามารถนำไปประยุกต์ได้ทั้งงานอุตสาหกรรมเช่นการเคลื่อนย้ายวัตถุในสายการผลิต ตลอดจนงานวิจัยที่จำเป็นต้องระบุตำแหน่งของวัตถุเช่นเทคโนโลยีหุ่นยนต์ เป็นต้น

การได้มาซึ่งความลึกนั้นอาศัยเทคโนโลยีหลายอย่าง ในงานวิจัยนี้เราจะนำเสนอการหาภาพสามมิติสองภาพจากกล้องสามตัว และนำเสนอวิธีซ้อนทับภาพสามมิติสองภาพดังกล่าวซึ่งอยู่บนแกนสามมิติสองแกนเข้าด้วยกัน โดยใช้ความรู้พื้นฐานของกล้องประเภทสเตอริโอเพื่อหาความสัมพันธ์ของกล้องซ้ายกับกล้องกลางซึ่งพิจารณาบนแกนสามมิติสองแกนที่แตกต่างกัน ได้แก่ การหาค่าคงที่ของกล้อง การแก้ไขความเพี้ยนของกล้อง การปรับตั้งกล้อง และการหาความลึกของภาพจากกล้องมากกว่าหนึ่งตัว ซึ่งผลที่ออกมานั้นจะได้ภาพสามมิติในแกนเดียวกัน

ABSTRACT

Ability to find depth using images is useful in localizing an object in a scene. When the position is known, we can manipulate the object, e.g., move it, etc. One can see that the ability to find depth can be applied to both industry, e.g. in moving objects in manufacturing lines, as well as to researches that are related to localization, e.g. robotic technology, etc. Acquiring depth information relies on many technologies. In this research, we will present a technique that reconstructs two 3D images from 3 webcam cameras. Also, we will show a method that overlay the 3D images in different 3D coordinates. The method is based on theories of stereo cameras for finding relationship of left camera and middle camera which are in the different 3D coordinates. Namely, Camera Constant Determination, Distortion Correction, Calibration, and Depth from Disparity. The result is a set of 3D point cloud in just one coordinate.

บทนำ

1 ความสำคัญและที่มาของโครงงาน

ปัจจุบันนี้การใช้งานกล้องเพื่อการประมวลผลภาพได้เข้ามามีส่วนอย่างมากในการพัฒนาเทคโนโลยีในปัจจุบัน และการพัฒนาการใช้งานกล้องหลายตัวเพื่อสร้างโมเดลสามมิติซึ่งสามารถนำไปพัฒนาใช้ได้กับงานหลายประเภท เช่น ทางการแพทย์ หรือในระบบอุตสาหกรรม เป็นต้น โดยงานวิจัยนี้จะศึกษาวิธีการสร้างโมเดลสามมิติซึ่งได้รับข้อมูลจากกล้องเว็บแคม 3 ตัว เพื่อเป็นความรู้และพื้นฐานในการนำไปประยุกต์ใช้กับงานอื่นได้ต่อไปในอนาคต

2 วัตถุประสงค์ของโครงงาน

โครงงานนี้เป็นการศึกษาและเขียนโปรแกรมเพื่อการใช้งานกล้องเว็บแคมสามตัว เพื่อที่จะหาระยะความลึกระหว่างกล้องกับสิ่งแวดล้อมบนแกนสามมิติ และทำการซ้อนภาพสามมิติที่อยู่บนแกนสามมิติที่ต่างกันเพื่อประกอบเป็นโมเดลสามมิติ ซึ่งได้จากการรับภาพจากกล้องสามตัว

3 ขอบเขตของโครงงาน

โครงงานนี้มีเป้าหมายคือ ทำการซ้อนภาพสามมิติเพื่อประกอบเป็นโมเดลสามมิติ ซึ่งได้จากการคำนวณระยะของสิ่งแวดล้อมถึงกล้องโดยการรับภาพจากกล้องสามตัว

4 ขั้นตอนและวิธีดำเนินการ

หลักการและขั้นตอนสำหรับการใช้เว็บแคมสามตัวในการสร้างโมเดลสามมิติประกอบไปด้วยขั้นตอนต่างๆ ซึ่งสามารถแบ่งแยกย่อยได้ออกเป็นขั้นตอนใหญ่ๆได้ดังนี้

4.1 หาข้อมูลและศึกษาพื้นฐาน

ขั้นตอนนี้จะทำการศึกษาเรื่องที่เกี่ยวข้องกับการแปลงภาพจากกล้องสามตัวให้ฉายลงบนแกนสามมิติโดยใช้พื้นฐานกล้องสเตอริโอ และวิธีการซ้อนทับภาพสามมิติซึ่งอยู่บนแกนสามมิติที่ต่างกันสองวิว รวมถึงเนื้อหาพื้นฐานที่เกี่ยวข้องกับ Computer vision เพื่อเป็นรากฐานในการพัฒนาโปรแกรม

4.2 ศึกษาการเขียนโปรแกรมภาษาซี

ขั้นตอนนี้จะทำการเริ่มต้นเขียนโปรแกรมโดยใช้โอเพนซอร์สที่พัฒนาโดยบริษัท อินเทล ที่ชื่อว่า OpenCV (Open Source Computer Vision) และการจำลองโมเดลสามมิติโดยการแสดงผลลงบนโอเพนซอร์สที่ชื่อว่า OpenGL (Open Graphics Library)

4.3 ทดลองนิยามของพื้นฐานการหาระยะของวัตถุจากภาพและการซ้อนทับสิ่งแวดล้อมที่อยู่บนแกนสามมิติที่แตกต่างกัน

ทำการทดลองหาระยะวัตถุที่สนใจเมื่อเปรียบเทียบกับกล้องที่เราใช้งาน โดยทำการประมาณว่ากล้องมีการเกิดความบิดเบี้ยวของภาพหรือที่เรียกว่า distortion และมีสัญญาณรบกวน รวมถึงกล้องทั้งสามตัวถูกวางอยู่บนระนาบที่แตกต่างกันเล็กน้อย ทำการคำนวณและแปลงภาพที่อยู่บนแกนสองมิติให้ฉายลงบนแกนสามมิติโดยใช้พื้นฐานของกล้องประเภทสเตอริโอ โดยจะได้ภาพสามมิติสองภาพบนแกนสามมิติที่ต่างกันสองแกน และทำการซ้อนภาพสามมิติสองภาพนั้นเข้าด้วยกันโดยใช้แกนอ้างอิงร่วม

5 แผนการดำเนินงาน

ระยะเวลาและขั้นตอนในการทำโครงงาน

ตาราง 1 ตารางแสดงเป้าหมายเวลาการทำโครงงาน

ลำดับ	ขั้นตอน	ระยะเวลา
1	หาข้อมูลและศึกษาพื้นฐานการหาระยะของวัตถุจากภาพ	1 - 2 เดือน
2	ศึกษาการเขียนโปรแกรมภาษาซี	1 เดือน
3	ทดลองนิยามของพื้นฐานการหาระยะของวัตถุจากภาพ	1 เดือน
4	ศึกษาและพัฒนาโปรแกรมเมื่ออยู่ในสภาพการใช้งานใกล้เคียงสภาพจริง	1 เดือน
5	หาข้อมูลและศึกษาพื้นฐานการซ้อนภาพสามมิติสองภาพบนแกนสามมิติที่ต่างกัน	1 - 2 เดือน
6	ศึกษาการเขียนโปรแกรมภาษาซีโดยใช้โอเพนซอร์ส OpenCV และ OpenGL ร่วมกัน	1 เดือน

ลำดับ	ขั้นตอน	ระยะเวลา
7	ทดลองนิยามของพื้นฐานการหาระยะของวัตถุจากภาพ และการซ้อนทับสิ่งแวดล้อมที่อยู่บนแกนสามมิติที่แตกต่างกัน	2 เดือน

6 เครื่องมือที่ใช้ในการทำโครงงาน

ฮาร์ดแวร์ที่ใช้ในการทดลอง

1. กล้องเวปแคม Logitech รุ่น Webcam C600 For HD video จำนวน 3 ตัว
2. คอมพิวเตอร์พีซี
3. ซอฟต์แวร์ที่ใช้ในการทดลอง
4. Microsoft Visual Studio 2008

7 ผลที่ได้รับ

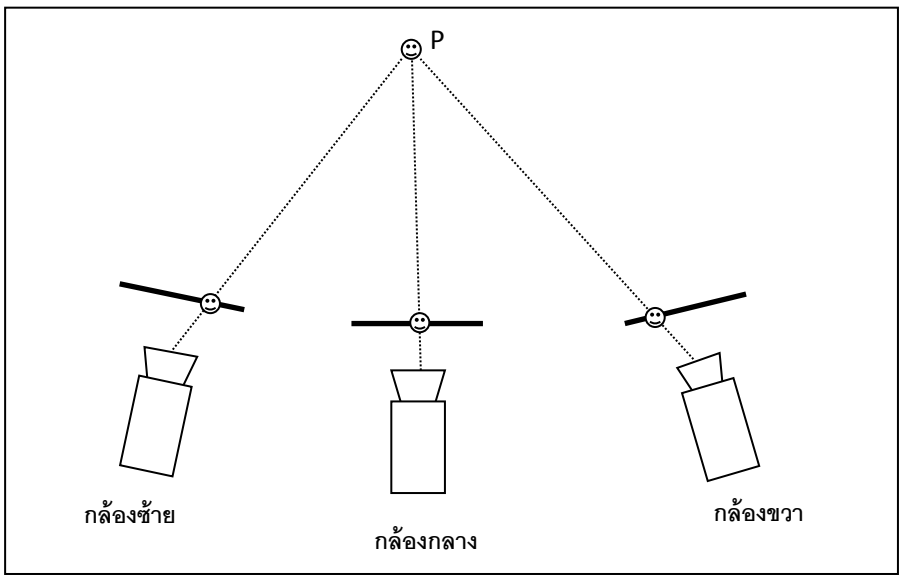
ผลที่ได้จากโครงงานนี้คือโปรแกรมที่ทำการคำนวณระยะทางจากกล้องถึงวัตถุโดยอัตโนมัติ ลงบนแกนสามมิติที่ต่างกันสองแกนจากกล้องสามตัว รวมไปถึงการซ้อนโมเดลสามมิติที่แตกต่างกันสองแกนได้

วิธีวิจัย

1. ทฤษฎีเบื้องต้นและตัวอย่างการใช้โปรแกรม

1.1 การรับภาพจากกล้องสามตัว

ในการสร้างโมเดลสามมิติเพื่อให้ได้รายละเอียดหลายให้มากที่สุดนั้น จำเป็นจะต้องใช้กล้องมากกว่าสองตัวในการเก็บภาพให้ได้หลายมุมมอง ซึ่งในงานวิจัยนี้เราจะใช้กล้องสามตัวในการรับภาพดังแสดงในรูปที่ 1



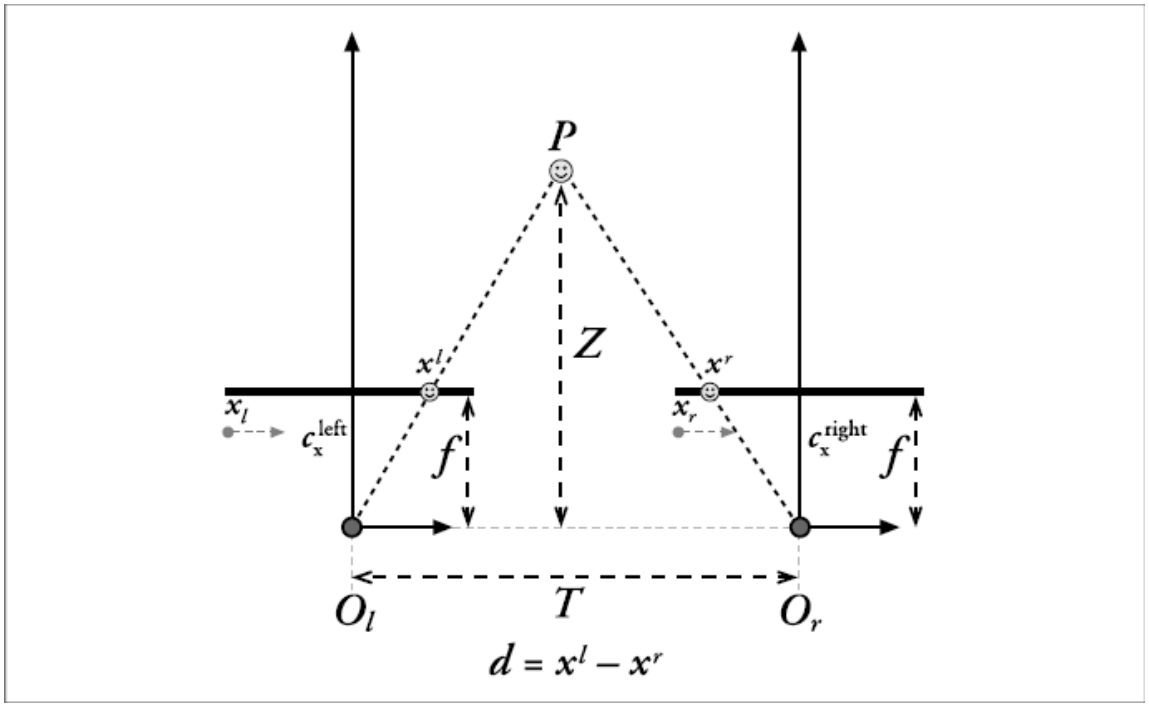
<u>รูปที่ 1</u> แสดงความสัมพันธ์ระหว่างกล้องสามตัว โดยประกอบด้วยกล้องซ้าย กล้องกลาง และกล้องขวา ในการฉายจุดสามมิติ (จุด P) ลงบนระนาบของภาพในแต่ละกล้อง

จากนั้นเราจะใช้พื้นฐานของกล้องประเภทสเตอริโอในการคำนวณหาระยะความลึกจากวัตถุหรือสิ่งแวดล้อมเทียบกับแกนระหว่างกล้อง โดยภาพสามมิติบนแกนสามมิติแกนที่ 1 จะหาได้จากความสัมพันธ์ระหว่างกล้องซ้ายกับกล้องกลาง และภาพสามมิติบนแกนสามมิติแกนที่ 2 จะหาได้จากความสัมพันธ์ระหว่ากล้องกลางกับกล้องขวา ซึ่งการคำนวณหาพิกัดบนภาพสามมิตินั้นจะอธิบายในหัวข้อถัดไป

1.2 การคำนวณหาพิกัดของภาพในแกนสามมิติด้วยการใช้ OpenCV2.1

การหาจุดพิกัดสามมิติบนแกนของกล้องนั้น จำเป็นต้องใช้พื้นฐานของทฤษฎีสามเหลี่ยมคล้ายดังจะเห็นได้ในรูปที่ 2 ซึ่งสมการที่จะคำนวณหาความลึกจากแกนกลางระหว่างกล้องถึงพิกัดของวัตถุสามมิตินั้นคำนวณได้จาก

$$\frac{T-(x^l-x^r)}{Z-f}=\frac{T}{Z}\Rightarrow Z=\frac{fT}{x^l-x^r} \qquad (1)$$

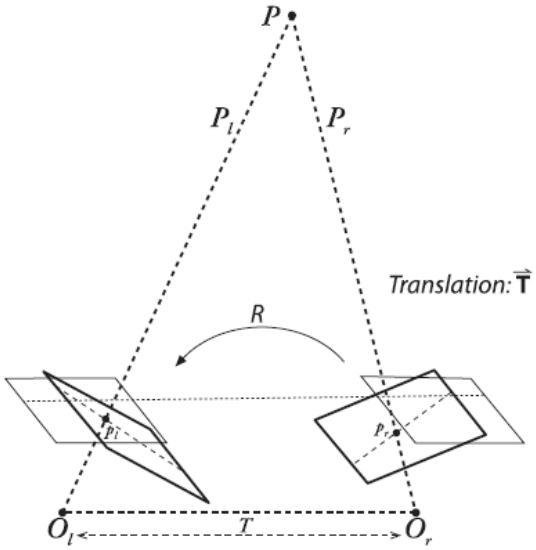


<u>รูปที่</u> 2 จากภาพไม่มีการบิดเบี้ยว และวางตัวในแนวนอนเป็นแนวเดียวกัน การหาความลึก *Z* สามารถทำได้โดยทฤษฎีสามเหลี่ยมคล้าย โดยที่รังสีมูลฐาน *(principal ray)* ของกล้องนั้นเริ่มต้นจากจุดกึ่งกลางการฉาย *(center of projection)* O_l และ O_r และขยายไปจนถึงระนาบของภาพทั้งสอง ณ จุดนั้นเราเรียกว่า *principal point* หรือจุดมูลฐานที่จุด *C*

วิธีการคำนวณหาค่าความลึกจากแกนกลางระหว่างกล้องถึงพิกัดของวัตถุสามมิติมีขั้นตอนทั้งหมด 4 ส่วน ดังนี้

1.2.1 เนื่องจากการที่เราจะหาจุดพิกัดบนสามมิติได้นั้น จำเป็นจะต้องใช้ทฤษฎีสามเหลี่ยมคล้ายดังรูปที่ 2 แต่ในทางปฏิบัตินั้นการตั้งกล้องให้ขนานกัน 100% นั้นทำได้ยาก(รูปที่ 3) ดังนั้นเราจะใช้วิธีการบิดภาพซ้ายและภาพขวาให้มาขนานกันแทน แต่ก่อนที่จะทำการบิดภาพให้มาขนานกันนั้น เราจำเป็นจะต้อง

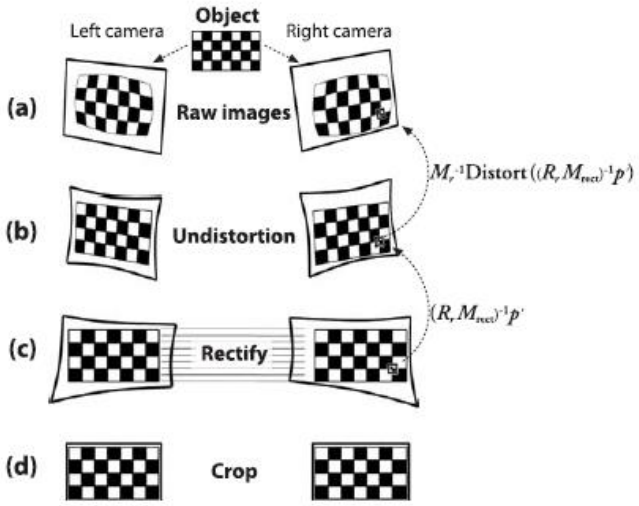
ทราบค่าตัวแปรภายในของกล้อง (Intrinsic Parameter), สัมประสิทธิ์การบิดเบี้ยวของเลนส์กล้อง (distortion_coeffs) และตัวแปรภายนอก (Extrinsic Parameter) โดยตัวแปรภายนอกนั้นจะประกอบไปด้วยค่าเมตริกซ์การหมุน R และเวกเตอร์การเลื่อน T ระหว่างกล้องซ้ายกับกล้องขวา โดยค่าตัวแปรที่ไม่ทราบค่าทั้งหมดนั้นสามารถหาได้จากคำสั่ง cvStereoCalibrate()



<u>รูปที่ 3</u> เป้าหมายในการคำนวณเพื่อทำการบิดภาพจากสองกล้องให้ขนานหรืออยู่ในระนาบเดียวกัน และแกน x ของภาพทั้งสองก็จะถูกจัดเรียงในแนวเดียวกันด้วย

1.2.2 จากนั้นมุมของกล้องซ้ายและกล้องขวาจะถูกปรับให้มาขนานกัน โดยวิธีปฏิบัติคือการปรับภาพซ้ายและภาพขวาให้ขนานกันแทนการปรับกล้องซึ่งจะเรียกวิธีของขั้นตอนนี้ว่า **rectification** ซึ่งเราจะใช้ algorithm ของ Bouguet สามารถหาได้จากคำสั่ง cv::stereoRectify() โดยการป้อนค่าคงที่ที่คำนวณได้จากคำสั่ง cvStereoCalibrate() แต่ความบิดเบี้ยวของเลนส์กล้องนั้นมีผลทำให้ภาพที่ถูกถ่ายออกมาเกิดความบิดเบี้ยวตามไปด้วย ดังนั้นสมการทางคณิตศาสตร์จะถูกนำมาใช้เพื่อแก้ความบิดเบี้ยวของภาพซึ่งจะเรียกวิธีของขั้นตอนนี้ว่า **undistortion** โดยเราสามารถคำนวณการปรับเทียบระนาบของภาพซ้ายและภาพขวาในเบื้องต้นโดยการสร้างตารางปรับเทียบค่า (look-up table) สำหรับภาพทางซ้ายและขวาโดยการเรียกคำสั่ง cv::InitUndistortRectifyMap() แยกกัน ซึ่งผลลัพธ์ที่ได้ออกมาจะเป็นภาพซ้ายและภาพขวาที่ถูกแก้ความบิดเบี้ยวและบิดสองภาพมาให้ขนานกัน โดยใช้คำสั่ง cv::remap() แยกกันระหว่างภาพ

ซ้ายกับภาพขวา โดยขั้นตอนตั้งแต่การรับภาพจนถึงการบิดภาพซ้ายและภาพขวามาให้ขนานกันจะแสดงอยู่ในรูปที่ 4

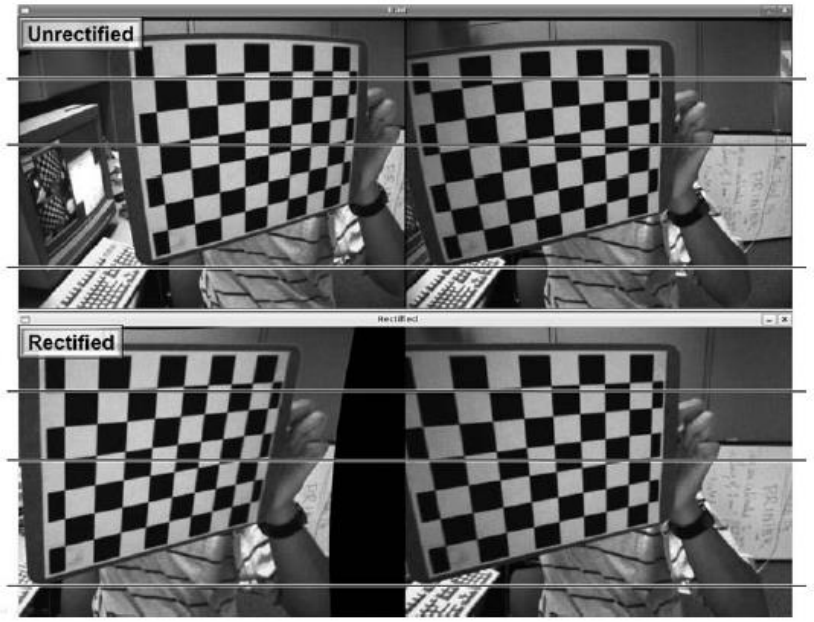


<u>รูปที่ 4</u> ภาพแสดงการปรับแก้กล้องสเตอริโอ จากกล้องซ้ายและกล้องขวา (a) รูปภาพก่อนการปรับแต่ง (b) ภาพทำการแก้ความบิดเบี้ยว (c) ภาพแสดงการทำการปรับระนาบ (d) ทำการตัดส่วนที่ใช้งานออกมา

1.2.3 ทำการหาจุดในแกนสามมิติที่ตกกระทบลงบนภาพซ้ายและภาพขวาซึ่งจะเรียกวิธีของขั้นตอนนี้ว่า **correspondence** ซึ่งผลลัพธ์ที่ได้ออกมาจะเป็นระยะห่างจากจุดในแกนสามมิติที่ตกกระทบลงบนภาพในแนวแกน **X** ของแกนของภาพซ้ายรวมกับภาพขวาซึ่งเราจะเรียกระยะห่างนี้ว่า disparity โดยคำนวณหาได้จาก $d = x^l - x^r$ แต่ในการคำนวณโดยการเขียนโปรแกรมนั้นเราต้องรู้พิกัดของจุด P ที่ฉายลงบนภาพซ้ายและภาพขวา (รูปที่ 2) ซึ่ง OpenCV นั้นได้มีการวิจัยและพัฒนาการหาพิกัดจุดบนโลกสามมิติที่ฉายลงบนภาพซ้ายและภาพขวาโดยเรียกกระบวนการนี้ว่า **“Stereo correspondence”** และยิ่งไปกว่านั้น OpenCV ได้ทำการสร้างคำสั่งที่รวดเร็วและมีประสิทธิภาพในการหาจุดที่สอดคล้องกันทั้งสองภาพโดยการประมวลผลเป็นลำดับซึ่งใช้คำสั่ง CVFindStereoCorrespondenceBM() ซึ่งภายในคำสั่งนี้จะมีการปรับค่าพารามิเตอร์ต่างๆ เพื่อให้ได้ค่า disparity ที่เหมาะสมและใกล้เคียงกับความจริงมากที่สุด โดยเราจะคำนวณได้จากการรับข้อมูลภาพซ้ายกับภาพขวาที่ถูกแก้ความบิดเบี้ยวของภาพและบิดทั้งสองภาพมาให้ขนานกัน ขั้นตอนในการคำนวณจะแบ่งเป็น 3 ขั้นตอนใหญ่ดังนี้

a) **การปรับความสว่างของภาพซ้ายกับภาพขวาให้เท่ากันและการปรับปรุงรายละเอียดของภาพให้ดีขึ้น** โดยการสร้างหน้าต่างเพื่อเลื่อนไปยังส่วนต่างๆของภาพ ซึ่งในฟังก์ชันจะแทนด้วยตัวแปร BMState->prefilterSize มีค่าตั้งแต่ 3 ถึง ขนาดที่ครอบคลุมได้ทั้งภาพ โดยค่าบ่งชี้หรือค่าที่ควรใช้จะอยู่ที่ค่า 21 และค่าตัวแปรนั้นจะต้องแทนด้วยเลขจำนวนคี่เท่านั้น ส่วนการปรับความสว่างทั้งสองภาพนั้นจะคำนวณภายในหน้าต่างที่สร้างขึ้น โดยจุดกึ่งกลางของหน้าต่างจะแทนด้วยตัวแปร I_c คือค่าความสว่างของพิกเซลในจุดนั้น และจะแทนค่าที่ได้จากการคำนวณ $\min[(I_c - \bar{I}, I_{cap}), I_{cap}]$ ซึ่ง $\bar{I}$ คือค่าความสว่างเฉลี่ยภายในหน้าต่าง ณ จุดต่างๆบนภาพ และ c คือ ค่าจำนวนเต็มบวกมีค่าตั้งแต่ 1 ถึง 63 โดยค่าบ่งชี้หรือค่าที่ควรใช้จะอยู่ที่ค่า 30 ในฟังก์ชันจะแทนด้วยตัวแปร BMState->preFilterCap ซึ่งใช้ในการปรับลดความแตกต่างของแสง (contrast) ถ้าในภาพมีค่าความสว่างที่แตกต่างกันมาก (a lot of noise) จะใช้ค่า I_{cap} ต่ำ แต่ถ้าในภาพมีค่าความสว่างที่แตกต่างกันน้อย (less noise) จะใช้ค่า I_{cap} สูง

b) **การค้นหาจุดที่สอดคล้องกันทั้งสองภาพ** คำนวณโดยการสร้างหน้าต่างเพื่อเลื่อนไปต่างส่วนต่างๆของภาพ ในฟังก์ชันจะแทนด้วยตัวแปร BMState->SADWindowSize โดยมีค่าตั้งแต่ 3 ถึงขนาดที่ครอบคลุมได้ทั้งภาพ ในการตั้งขนาดของหน้าต่างนั้นเราจะพิจารณาจากรายละเอียดของภาพเป็นหลัก กล่าวได้ว่าถ้าภาพมีองค์ประกอบที่มีลายระเอียดมากเราจะใช้หน้าต่างที่มีขนาดเล็ก และในทางตรงข้ามถ้าในภาพมีองประกอบที่มีลายระเอียดน้อยเราก็สามารถใช้หน้าต่างที่มีขนาดใหญ่ขึ้นได้ ซึ่งหน้าต่างจะทำการ

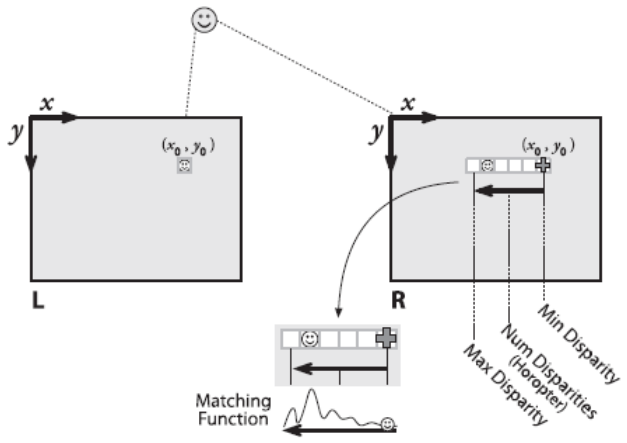


<u>รูปที่ 5</u> ภาพคู่ล่างนั้น แสดงถึงภาพที่ถูกบิดมาให้ขนานกัน และเส้นที่ลากผ่านแต่ละแถวนั้นแสดงให้เห็นถึงการเรียงตัวในแต่ละแถวของแต่ละภาพนั้นมีค่าเท่ากัน ยกตัวอย่างเช่น จุดตัดของกระดานหมากรุกในภาพซ้ายจะเรียงตัวอยู่แถวเดียวกับภาพขวา

ค้นหาจากภาพซ้ายที่ถูกบิดภาพมาขนานแล้ว (rectification image) ดังที่แสดงในรูปที่ 5

เราจะสังเกตได้ว่าจุดตัดของกระดานหมากรุกในภาพที่ซ้ายและภาพที่ขวานั้นจะเรียงตัวอยู่ในแถวเดียวกัน (row-alignment) ดังนั้นค่าเริ่มต้นในการค้นหาจุดที่สอดคล้องกันในฟังก์ชันจะแทนด้วยตัวแปร BMState->mindisparity ซึ่งเท่ากับพิกัด (x_0, y_0) ดังที่แสดงในรูปที่ 6 และถ้ากล้องซ้ายและขวาถูกตั้งได้ขนานกัน ค่า mindisparity จะถูกตั้งให้มีค่าเท่ากับศูนย์ แต่ในทางปฏิบัติจริงนั้นกล้องทั้งสองจะถูกตั้งไม่ขนานกัน 100% ดังนั้นจะส่งผลให้เกิดค่า disparity ที่ติดลบ ดังนั้นเราจึงควรตั้งค่า mindisparity ที่มีค่าติดลบเล็กน้อย จากนั้นโปรแกรมจะเริ่มทำการค้นหาจุดที่สอดคล้องกันโดยเริ่มจากค่า mindisparity และทำการเลื่อน SAD Window ไปทางซ้ายไปสิ้นสุดที่ค่า disparity สูงสุด ซึ่งเราจะกำหนดช่วงของ disparity ได้จากตัวแปร BMState->numberOfDisparities โดยมีเงื่อนไขว่า ค่าที่เรากำหนดนั้นจะต้องเป็นจำนวนเต็มบวกและหารด้วย 16 ลงตัว

c) **การกำจัดค่า disparity ที่ผิด ที่ได้รับจากการค้นหาจุดที่สอดคล้องกันทั้งสองภาพ** เราจะใช้คำสั่ง BMState->uniquenessRatio เป็นตัวแปรที่คำนวณมาจากสมการ (match_val – min_match)/min_match ซึ่งเป็นค่าที่ใช้กำหนดอัตราส่วนระหว่างค่าความแตกต่างของ disparity เทียบกับค่า disparity ที่น้อยที่สุดเพื่อพิจารณาว่าค่าที่ต่ำกว่า uniqunessRatio ที่เราตั้งไว้นั้นให้กำจัดทิ้ง และถ้าเรากำหนดค่า uniquenessRatio น้อยแปลว่าในสิ่งแวดล้อมที่ถูกถ่ายภาพมานั้นมีความเข้มของพื้นผิวใกล้เคียงกัน และค่าบ่งชี้หรือค่าที่ควรใช้จะอยู่ที่ค่า 12 ต่อมาเราจะใช้คำสั่ง BMState->textureThreshold เพื่อกำจัดค่า disparity ที่อยู่ตำแหน่งใกล้เคียงกันแต่มีค่าแตกต่างกันมากเกินความเป็นจริง ซึ่งค่าที่ต่ำกว่าที่กำหนดไว้ก็จะถูกกำจัดทิ้งเช่นกัน และค่าบ่งชี้จะอยู่ที่ค่า 12 และสุดท้ายปัญหา



<u>รูปที่ 6</u> ภาพแสดงการใช้งาน SAD window โดยจะเริ่มที่จุดที่เป็นพิกัดเท่ากัน โดยมีค่า disparity เท่ากับ 0 และเลื่อนไปทางซ้ายจนถึงที่มีค่า disparity สูงสุด โดยค่าที่ได้นั้นแสดงให้เห็นในด้านล่างของภาพ

ของการได้รับค่า disparity ที่ผิดนั้น จะเกิดขึ้นบริเวณเส้นขอบของวัตถุ และเพื่อที่จะปกป้องค่า disparity บริเวณเส้นขอบของวัตถุนั้น เราจะกำหนดหน้าต่างขึ้นมาโดยการกำหนดค่าในฟังก์ชัน BMState->speckleWindowSize โดยหน้าต่างจะมีขนาดตั้งแต่ 5x5 ถึง 21x21 และเราจะพิจารณาค่าสูงสุดและต่ำสุดของ disparity ที่อยู่ในหน้าต่างว่าอยู่ในช่วงของ spackleRange หรือไม่ ถ้าใช่ค่า disparity นั้นจะถูกเก็บค่าไว้ โดยเราสามารถกำหนดค่า spackleRange ได้จาก BMState->speckleRange ซึ่งค่าบ่งชี้จะมีค่าเท่ากับ 4 และค่าในตัวแปร BMState->roi1 และ BMSTate->roi2 เนื่องจากภาพที่ถูกบิดมาให้ขนานกันนั้นจะได้รับมาจากฟังก์ชัน CVStereoRectify ซึ่งค่าในทุกพิกเซลในภาพซ้ายและภาพขวานั้นเป็นค่าที่สมเหตุสมผลและยอมรับได้ แต่ถ้าเราคัดลอกค่าในภาพที่ถูกบิดมาให้ขนานกันเข้าไปในฟังก์ชัน CVStereoBMState แล้วฟังก์ชันนี้จะคำนวณและกำจัดค่า disparity ที่ไม่สมเหตุสมผลทิ้งโดยอัตโนมัติ ดังนั้นตัวแปร roi1 และ roi2 ได้ถูกกำหนดมาไว้เพื่อเก็บค่า disparity ที่ไม่สมเหตุสมผลแต่อาจจะใช้ได้เอาไว้ ส่วนค่าใน BMState->disp12MaxDiff นั้นเป็นค่ามากที่สุดของ disparity ที่ถูกคำนวณจากภาพซ้ายไปภาพขวาและค่า disparity ที่ถูกคำนวณจากภาพขวาไปซ้าย ซึ่งถ้าค่าผลต่างของ disparity มีค่าเกิน threshold ที่ตั้งไว้ ค่า disparity ที่พิกเซลนั้นจะถูกกำจัดทิ้ง

จากที่อธิบายวิธีการหาค่า disparity ที่เหมาะสมมาทั้งหมด 3 ขั้นตอนนั้น จะเป็นการปรับค่าตัวแปรที่อยู่ในฟังก์ชัน cvCreateStereoBMState() แต่ใน OpenCV2.1 นั้น ได้มีการพัฒนาและปรับปรุงการหาค่า disparity ที่ใกล้เคียงกับความเป็นจริงมากที่สุด โดยเพิ่มฟังก์ชันในการปรับค่าตัวแปรต่างๆ ให้ได้ disparity ที่เหมาะสม ซึ่งคำสั่งหรือตัวแปรที่ใช้ปรับหาค่า disparity จะอยู่ในฟังก์ชัน cv::StereoSGBM::SteroSGBM โดยจะประกอบด้วยตัวแปร minDisparity, numDisparities, SADWindowSize, disp12MaxDiff, preFilterCap, uniquenessRatio, speckleWindowSize, speckleRange และจะมีตัวแปรที่เพิ่มขึ้นคือ P1,P2 เป็นตัวแปรที่ควบคุมให้ disparity มีความละเอียดขึ้น ซึ่งคำนวณได้จากสมการดังต่อไปนี้

$$P1 = 8*cn*SADWindowSize*SADWindowSize$$

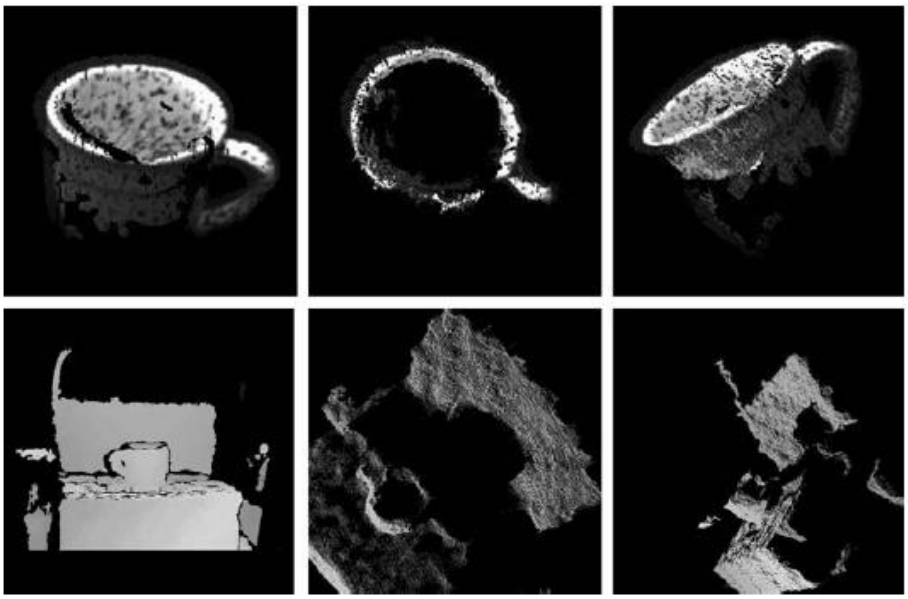
$$P2 = 32*cn*SADWindowSize*SADWindowSize$$

โดยตัวแปร cn จะแทนด้วยจำนวน channel ของภาพที่จะนำมาคำนวณ และตัวแปรสุดท้ายที่เพิ่มมาคือตัวแปร fullDP ด้วยการตั้งค่าเป็น true เพื่อใช้ในการรันโปรแกรม

1.2.4 เมื่อเราได้ค่า disparity ที่เหมาะสมจากที่ได้กล่าวมาในข้อ 1.2.3 แล้ว เพื่อที่จะฉายกลับภาพสองมิติไปยังแกนสามมิตินั้น เราจะใช้คำสั่งคือ cvReprojectImageTo3D() ซึ่งเราจะใช้ค่า disparity และค่าพิกัดสองมิติ (x,y) ในแต่ละพิกเซลมาคำนวณหาพิกัดสามมิติได้ตามสมการต่อไปนี้

$$Q\begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \tag{2}$$

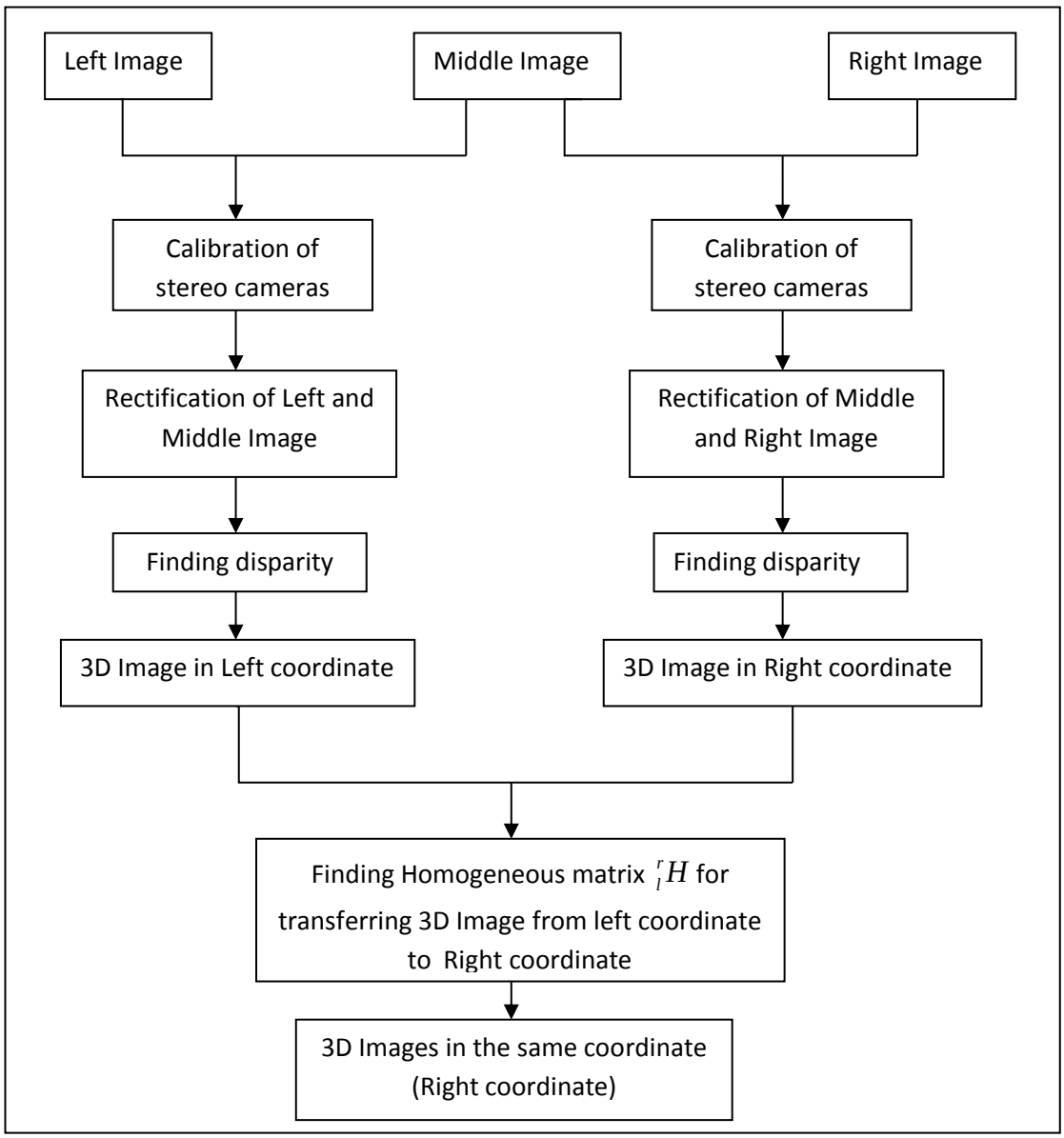
ซึ่งจุดสามมิติจะมีพิกัดเป็น (X/W,Y/W,Z/W) และตัวอย่างภาพสามมิติที่ถูกฉายกลับนั้นจะแสดงผลให้เห็นในรูปที่ 7



<u>รูปที่ 7</u> ตัวอย่างที่ได้จากการฉายภาพสองมิติกลับไปยังแกนสามมิติ คือ ภาพแก้วน้ำและเก้าอี้ โดยใช้คำสั่ง cv::StereoSGBM::StereoSGBM(); ในการหาค่า disparity และใช้คำสั่ง cvReprojectImageTo3D() ในการหาพิกัดจุดสามมิติของภาพ

1.3 การประกอบภาพสามมิติสองวิวโดยหาแกนอ้างอิงร่วม

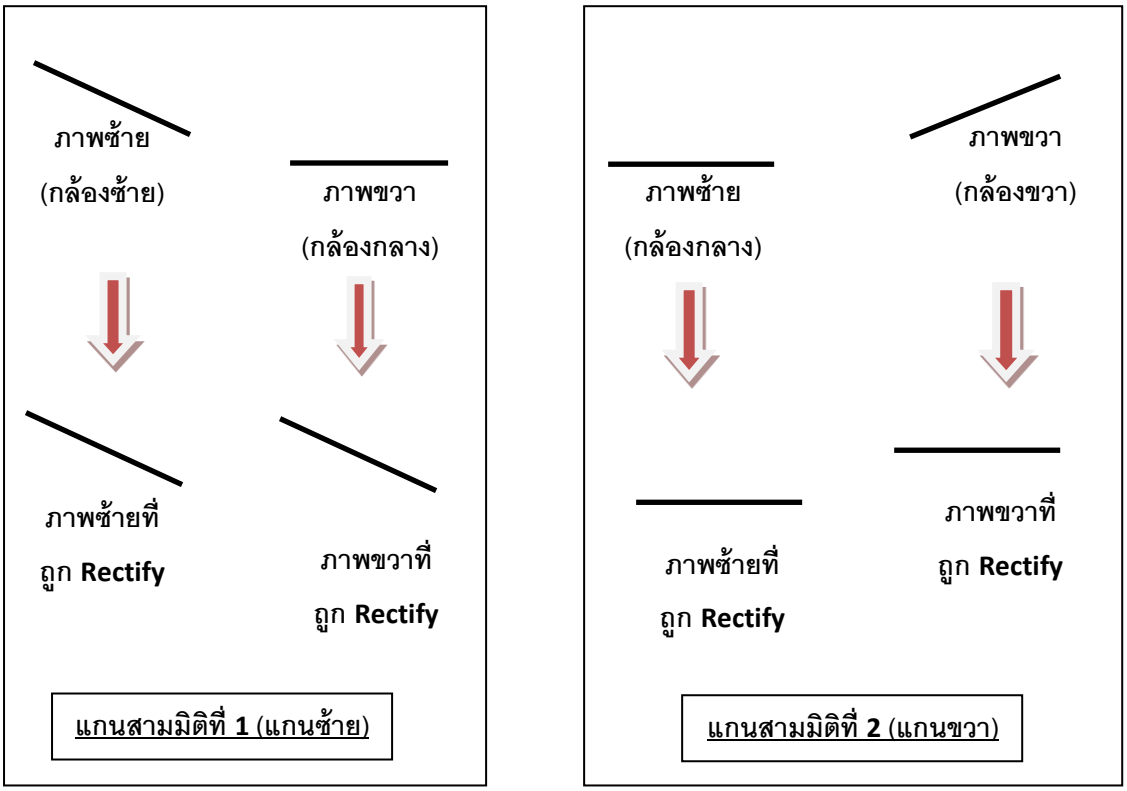
ในการประกอบภาพสามมิติ (ภาพที่แสดงค่าความลึกจากวัตถุที่สนใจถึงกล้องหรือค่า Z โดยกำหนดให้ X กับ Y มีค่าเป็น 0) ทั้งสองวิวเข้าด้วยกันนั่น ในงานวิจัยนี้เราจะใช้กล้องเว็บแคม 3 ตัว โดยภาพสามมิติในแกนสามมิติที่ 1 (แกนซ้าย) นั้นจะหาได้จากกล้องเว็บแคมกล้องซ้ายกับกล้องกลางดังรูปที่ 1 และการหาภาพสามมิติในแกนสามมิติที่ 2 (แกนขวา) จากการคำนวณหาพิกัดในแกนสามมิติในหัวข้อที่ 1.2 นั้นจะหาได้จากกล้องเว็บแคมกล้องกลางกับกล้องขวา และขั้นตอนการหาภาพสามมิติสองวิวโดยการ



<u>รูปที่ 8</u> ขั้นตอนในการประกอบภาพสามมิติโดยการหาแกนอ้างอิงร่วม โดยตั้งแต่การรับภาพจากกล้องเว็บแคมสามตัวจนถึงการประกอบภาพสามมิติสองภาพเข้าด้วยกัน

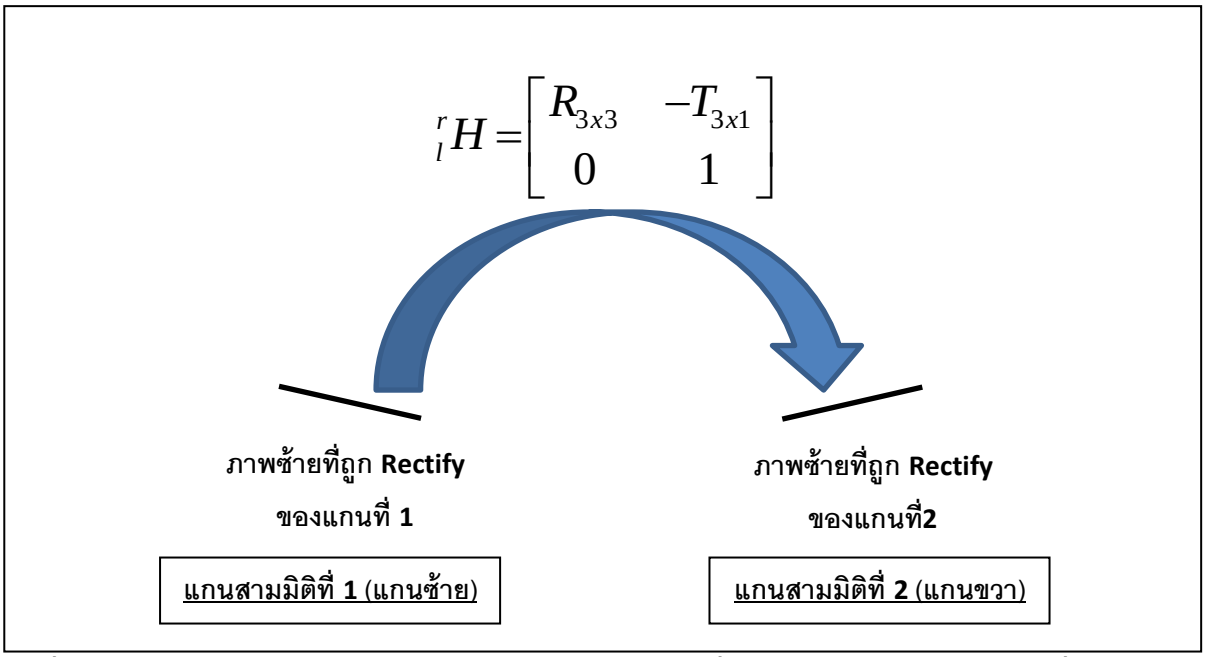
หาแกนอ้างอิงร่วมนั้นจะแสดงให้เห็นได้ในรูปที่ 8

การหาความสัมพันธ์ของแกนสามมิติที่ 1 (แกนซ้าย) และแกนสามมิติที่ 2 (แกนขวา) ในงานวิจัยนี้จะคำนวณจากความสัมพันธ์ระหว่างภาพของกล้องซ้ายบนแกนสามมิติที่ 1 (แกนซ้าย) ที่ถูกบิดมาให้ขนานแล้ว (Rectification) กับภาพของกล้องกลางบนแกนสามิติที่ 2 (แกนขวา) ที่ถูกบิดมาให้ขนานแล้วเช่นกัน ดังที่แสดงในรูปที่ 9



รูปที่ 9 แสดงความสัมพันธ์ระหว่างแกนสามมิติที่ 1 และแกนสามมิติที่ 2 ซึ่งได้รับจากกล้องเว็บแคม 3 ตัว

จากนั้นเราจะนิยามภาพซ้ายที่ถูก Rectify ในแกนสามมิติที่ 1 ให้เป็นภาพซ้าย(RectLeft) ในข้อมูลชุดใหม่ และภาพซ้ายที่ถูก Rectify ในแกนสามมิติที่ 2 เป็นภาพขวา(RectRight) ในข้อมูลชุดใหม่เช่นกัน เนื่องจากเราทราบข้อมูลจากการใช้ฟังก์ชัน cvStereoCalibrate() ว่าจะได้รับค่า เมตริกซ์การหมุน R และเมตริกซ์การเลื่อน T ซึ่งมีความสัมพันธ์จากกล้องขวาไปกล้องซ้าย และจากที่เราเลือกภาพซ้ายที่ถูก Rectify ทั้งสองแกนเพื่อหาค่าในการบิดแกนมาให้ทับกันนั้น เนื่องจากค่า disparity ที่คำนวณได้ในแต่ละแกน จะถูกนำไปเก็บในพิกัดของรูปซ้ายเป็นหลัก เราจึงสันนิษฐานได้ว่า ถ้าเราหาความสัมพันธ์ของเมตริกซ์จากการหมุนและเลื่อนจากภาพซ้ายที่ถูก Rectify ในแกนที่ 2 ไปยังภาพซ้ายที่ถูก Rectify ในแกนที่ 1 ซึ่งถูกนิยามเป็นเมตริกซ์เขียนอยู่ในรูปตัวแปร $'H$ ดังจะเห็นได้จากความสัมพันธ์ในรูปที่ 10



รูปที่ 10 แสดงความสัมพันธ์ระหว่างแกนสามมิติแกนที่ 1 และแกนสามมิติแกนที่ 2 โดยใช้ความสัมพันธ์ของภาพซ้ายที่ Rectify ทั้งสองแกน

ดังนั้นเราจะใช้ภาพทั้งสองที่แสดงในรูปที่ 10 ซึ่งเป็นภาพตารางหมากรุกที่ถูก Rectify แล้วทั้งสองภาพ โดยเก็บข้อมูลไว้หลายคู่ภาพ ป้อนกลับเข้าไปยังฟังก์ชัน cvStereoCalibrate() เพื่อหาค่าเมตริกซ์การหมุน R และเมตริกซ์การเลื่อน -T เพื่อนำกลับเข้าไปแทนในตัวแปรเมตริกซ์ $_l^rH$

จากนั้นเราจะทำการฉายกลับภาพสามมิติจากแกนที่ 1 ไปยังแกนสามมิติแกนที่ 2 โดยการเลื่อนพิกัดสามมิติทุกจุดในภาพสามมิติแกนที่ 1 โดยใช้สมการต่อไปนี้

$$P_{right} = {}_l^rHP_{left}$$

ซึ่ง $P_{right} = \begin{bmatrix} X_{right} & Y_{right} & Z_{right} & 1 \end{bmatrix}^T$ คือ พิกัดของวัตถุบนแกนสามมิติแกนที่ 1

$P_{left} = \begin{bmatrix} X_{left} & Y_{left} & Z_{left} & 1 \end{bmatrix}^T$ คือ พิกัดของวัตถุบนแกนสามมิติแกนที่ 2

$_l^rH = \begin{bmatrix} R_{3x3} & -T_{3x1} \\ 0 & 1 \end{bmatrix}$ คือ เมตริกซ์ที่ประกอบไปด้วย เมตริกซ์การหมุน R และเมตริกซ์การเลื่อน -T

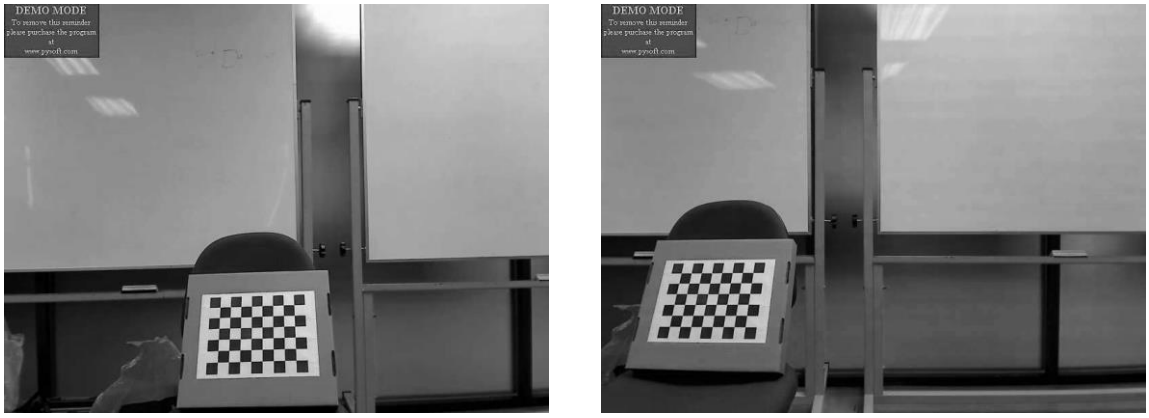
2 การทดลองนิยามเบื้องต้น

2.1 ข้อมูลที่ใช้ในการทดลอง

ในส่วนนี้ได้แสดงการทดลองนิยามเบื้องต้นของกล้องสามตัว โดยใช้พื้นฐานของระบบกล้องสเตอริโอ ซึ่งภาพที่ถูกถ่ายจากกล้องสามตัวนี้จะมีวิวประกอบร่วมกันนั่นคือ เก้าอี้กับกระดานหมากรุกที่ถูกถ่ายภาพในเวลาเดียวกัน และเราจะแบ่งข้อมูลเป็นสองชุดเพื่อใช้ในการหาภาพสามมิติสองภาพ โดยข้อมูลที่ถูกป้อนเข้าในโปรแกรมในชุดที่ 1 ประกอบด้วยภาพถ่าย 15 คู่ภาพ จะถูกแสดงในรูปที่ 11

รูปที่ 11 ภาพที่ถูกถ่ายในข้อมูลชุดที่ 1 ซึ่งถูกถ่ายด้วยกล้องซ้าย (ภาพซ้าย) และกล้องกลาง (ภาพขวา)

และข้อมูลที่ถูกป้องเข้าในโปรแกรมในชุดที่ 2 ประกอบด้วยภาพถ่าย 15 คู่ภาพเช่นกัน (รูปที่ 12)



รูปที่ 12 ภาพที่ถูกถ่ายในข้อมูลชุดที่ 2 ซึ่งถูกถ่ายด้วยกล้องกลาง (ภาพซ้าย) และกล้องขวา (ภาพขวา)

2.2 อุปกรณ์และโปรแกรมที่ใช้ในการทดลอง

อุปกรณ์และโปรแกรมในการทดลองการซ้อนทับภาพสามมิติสองภาพโดยการหาแกนอ้างอิงร่วม

- Computer notebook หน่วยประมวลผล Core2 Duo 2.00 GHz RAM 4 GB
- กล้อง Webcam Logitech C600 for HD video
- โปรแกรม Active WebCam
- โปรแกรม Microsoft Visual studio 2008
- OpenCV2.1 Library
- OpenGL Library

ผลและวิจารณ์

1. ผลการทดลองการหาภาพสามมิติในแกนสามมิติที่ 1 (แกนซ้าย)

1.1 ผลการทดลองค่าตัวแปรภายในของกล้องซ้ายกับกล้องกลาง

จากการป้อนภาพถ่ายข้อมูลชุดที่ 1 ที่ประกอบด้วยภาพตารางหมากรุกเพื่อทำการ calibrate หาค่าตัวแปรภายในกล้องซ้ายกับกล้องกลาง ด้วยคำสั่ง cvStereoCalibrate() ซึ่งเราต้องป้อนค่าจุดตัดของตารางหมากรุกในแนวแกน x (nx = 8) , แนวแกน y (ny = 6) และขนาดของตารางหมากรุกแต่ละช่องในโลกจริงซึ่งมีขนาด 27.5 mm. จะได้ค่าตัวแปรภายในของกล้องซ้ายกับกล้องกลางในแกนสามมิติที่ 1 (แกนซ้าย) ซึ่งมีผลดังนี้

$$M11=\begin{bmatrix} 6.0894848490869640e+002 & 0 & 3.9895365288157797e+002 \\ 0 & 6.0894848490869640e+002 & 3.2355396122145362e+002 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D11=\begin{bmatrix} 2.2852408173600763e\text{-}002 & -1.0603165095754015e\text{-}001 & 0 & 0 & 0 \end{bmatrix}$$

$$M21=\begin{bmatrix} 6.0894848490869640e+002 & 0 & 3.8897847705076464e+002 \\ 0 & 6.0894848490869640e+002 & 3.2931113149619961e+002 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D21=\begin{bmatrix} 4.1391712816110919e\text{-}002 & -1.0023986330058776e\text{-}001 & 0 & 0 & 0 \end{bmatrix}$$

โดย M11 คือ ค่าตัวแปรภายในของกล้องซ้ายในแกนสามมิติที่ 1 (intrinsic matrix of left camera)

D11 คือ ค่าสัมประสิทธิ์การบิดเบี้ยวของเลนส์กล้องซ้ายในแกนสามมิติที่ 1 (distortion matrix of left camera)

M21 คือ ค่าตัวแปรภายในของกล้องกลางในแกนสามมิติที่ 1 (intrinsic matrix of middle camera)

D21 คือ ค่าสัมประสิทธิ์การบิดเบี้ยวของเลนส์กล้องกลางในแกนสามมิติที่ 1 (distortion matrix of middle camera)

1.2 ผลการทดลองในการบิดภาพซ้ายกับภาพขวามาขนานกันในแกนสามมิติแกนที่ 1 (แกนซ้าย) และแก้ความบิดเบี้ยวทั้งสองภาพ

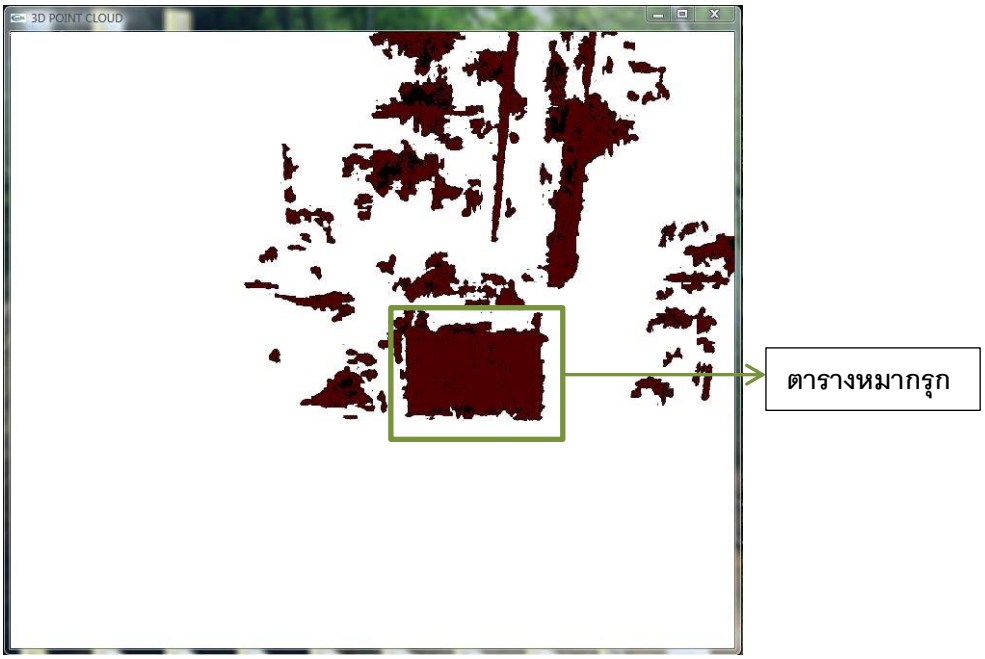
จากนั้นเราจะนำค่าตัวแปรภายในกล้องทั้งสองจากการทดลองที่ 1.1 มาใช้ในกระบวนการแก้ความบิดเบี้ยวของภาพ (undistortion) และบิดภาพให้ขนานกัน (rectification) ซึ่งในผลการทดลองจะแสดงให้เห็นถึงความแตกต่างจากคู่ภาพต้นฉบับ (คู่ภาพด้านบนในรูปที่ 1.2.1) จากภาพซ้ายที่ 1 (Left01) กับภาพ

รูปที่ 1.2.1 ภาพคู่บนแสดงให้เห็นถึงภาพซ้ายและขวาที่ถูกถ่ายมาจากกล้องเว็บแคมโดยตรง ส่วนภาพคู่ล่างเป็นภาพซ้ายและขวาที่ถูกบิดมาให้ขนานกันเทียบกับแกนสามมิติแกนที่ 1 (แกนซ้าย)

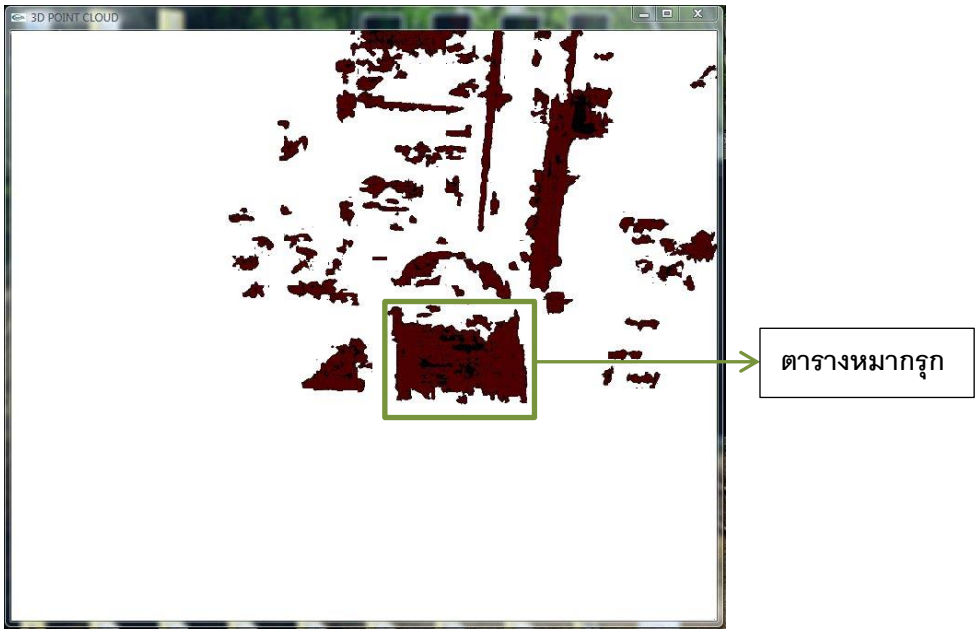
ขวาที่ 1 (mid01) และภาพที่ถูกบิดมาให้ขนานกัน (rectification) ดังที่แสดงให้เห็นในรูปที่ 1.2.1 โดยเส้นสีแดงที่ลากผ่านภาพทั้งสองนั้นคือ epipolar line จากผลการทดลองจะเห็นว่าภาพคู่ล่างที่ถูกบิดมาให้ขนานกันนั้น จะถูกเลื่อนให้แถวทั้งสองข้างเสมอกันด้วย กล่าวคือ เราสามารถสังเกตเส้น epipolar line ที่ลากผ่านภาพกระดานหมากรุกของภาพคู่ล่างจะเห็นได้ว่า ภาพนั้นได้ถูกเลื่อนมาให้อยู่แถวเดียวกัน ซึ่งแตกต่างอย่างเห็นได้ชัด ถ้าเทียบกับคู่ภาพบน หลังจากนั้นเราจะนำผลการทดลองที่ได้(หลังจากทำการ rectify แล้ว) ไปหาค่า disparity ซึ่งได้กล่าวไว้ในกระบวนกการ stereo correspondence และนำไปหาพิกัดในแกนสามมิติซึ่งจะกล่าวไว้ในหัวข้อถัดไป

1.3 ผลการทดลองของการหาภาพสามมิติ

จากขั้นตอนการหาพิกัดในแกนสามมิติ เราได้ทำการทดลองโดยนำคู่ภาพที่ถูกบิดมาให้ขนานกัน (rectification) มาหาค่า disparity โดยทำการปรับค่าตัวแปรจากการยึดภาพกระดานหมากรุกเป็นหลัก



(ก) ภาพสามมิติจากคู่ภาพลำดับที่ 1



(ข) ภาพสามมิติจากคู่ภาพลำดับที่ 3

รูปที่ 1.3.1 ภาพ (ก) และภาพ (ข) นั้นเป็นภาพสามมิติบนแกนสามมิติแกนที่ 1 (แกนซ้าย)

จากนั้นใช้คำสั่ง cvReprojectImageTo3D() ซึ่งใช้ค่าเมทริกซ์ Q ที่ได้จากขั้นตอน rectification เพื่อคำนวณหาพิกัดของวัตถุที่ปรากฏบนภาพในแกนสามมิติ (พิกัดสามมิติบนแกนของกล้องซ้าย) ซึ่งเราจะแสดงผลการทดลองในรูปแบบของภาพสามมิติ โดยใช้ค่าสีแดงแสดงค่าความลึกจากวัตถุไปยังแกนของกล้องซ้าย (แกนสามมิติที่ 1) จากการกำหนดค่า X และ Y มีค่าเป็น 0 ซึ่งเราจะใช้ผลที่ได้จากคู่ภาพลำดับที่ 1 และ 3 จากภาพถ่ายในข้อมูลชุดที่ 1 ดังแสดงในรูปที่ 1.3.1

2. ผลการทดลองการหาภาพสามมิติในแกนสามมิติที่ 2 (แกนขวา)

2.1 ผลการทดลองค่าตัวแปรภายในของกล้องกลางกับกล้องขวา

จากการป้อนภาพถ่ายข้อมูลชุดที่ 1 ที่ประกอบด้วยภาพตารางหมากรุกเพื่อทำการ calibrate หาค่าตัวแปรภายในกล้องกลางกับกล้องขวา ด้วยคำสั่ง cvStereoCalibrate() ซึ่งเราต้องป้อนค่าจุดตัดของตารางหมากรุกในแนวแกน x (nx = 8) , แนวแกน y (ny = 6) และขนาดของตารางหมากรุกแต่ละช่องในโลกจริงซึ่งมีขนาด 27.5 mm. จะได้ค่าตัวแปรภายในของกล้องกลางกับกล้องขวาในแกนสามมิติที่ 2 (แกนขวา) ซึ่งมีผลดังนี้

$$M12=\begin{bmatrix} 5.7907379475653056e+002 & 0 & 4.0168834725319812e+002 \\ 0 & 5.7907379475653056e+002 & 3.2832313537702532e+002 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D12=\begin{bmatrix} -2.3001033155336120e-002 & -5.3546659276288523e-002 & 0 & 0 & 0 \end{bmatrix}$$

$$M22=\begin{bmatrix} 5.7907379475653056e+002 & 0 & 4.1054446002528715e+002 \\ 0 & 5.7907379475653056e+002 & 3.2438396341160922e+002 \\ 0 & 0 & 1 \end{bmatrix}$$

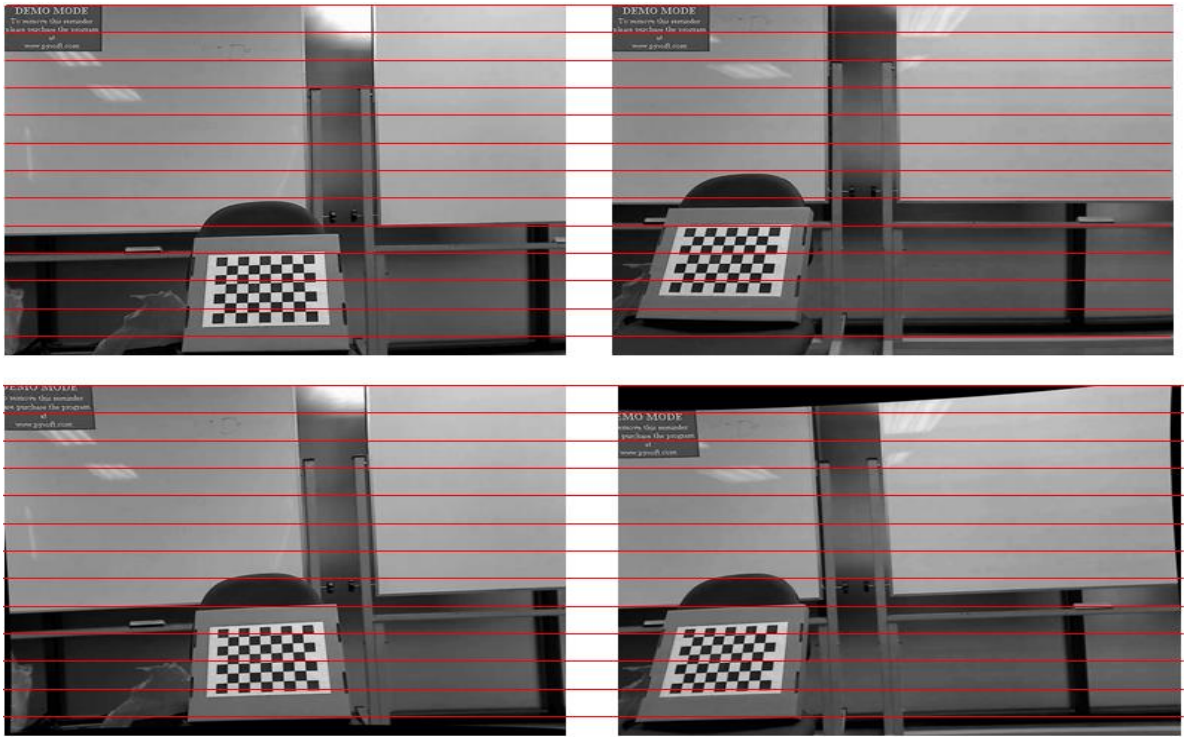
$$D22=\begin{bmatrix} -1.4388130396015775e-002 & -6.2506368498548237e-002 & 0 & 0 & 0 \end{bmatrix}$$

โดย M12 คือ ค่าตัวแปรภายในของกล้องกลางในแกนสามมิติที่ 2 (intrinsic matrix of middle camera)

D12 คือ ค่าสัมประสิทธิ์การบิดเบี้ยวของเลนส์กล้องกลางในแกนสามมิติที่ 2 1 (distortion matrix of middle camera)

M22 คือ ค่าตัวแปรภายในของกล้องขวาในแกนสามมิติที่ 2 (intrinsic matrix of right camera)

D22 คือ ค่าสัมประสิทธิ์การบิดเบี้ยวของเลนส์กล้องขวาในแกนสามมิติที่ 2 (distortion matrix of right camera)



รูปที่ 2.2.1 ภาพคู่บนแสดงให้เห็นถึงภาพที่ถูกถ่ายมาจากกล้องเว็บแคมโดยตรง ส่วนภาพคู่ล่างเป็นภาพที่ถูกบิดมาให้ขนานกันเทียบกับแกนสามมิติแกนที่ 2 (แกนขวา)

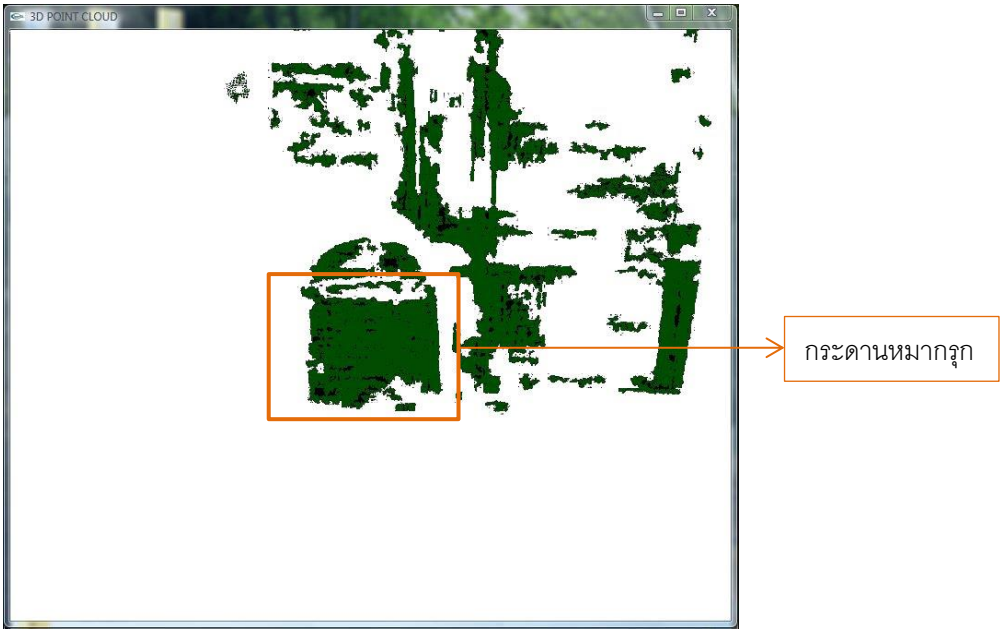
2.2 ผลการทดลองในการบิดภาพซ้ายกับภาพขวามาขนานกันในแกนสามมิติแกนที่ 2 (แกนขวา) และแก้ความบิดเบี้ยวทั้งสองภาพ

จากนั้นเราจะนำค่าตัวแปรภายในกล้องทั้งสองจากการทดลองที่ 2.1 มาใช้ในกระบวนการแก้ความบิดเบี้ยวของภาพ (undistortion) และบิดภาพให้ขนานกัน (rectification) ซึ่งในผลการทดลองจะแสดงให้เห็นถึงความแตกต่างจากคู่ภาพต้นฉบับ (คู่ภาพด้านบนในรูปที่ 2.2.1) จากภาพซ้ายที่ 1 (mid01) กับภาพขวาที่ 1 (right01) และภาพที่ถูกบิดมาให้ขนานกัน (rectification) ดังที่แสดงให้เห็นในรูปที่ 2.2.1 โดยเส้นสีแดงที่ลากผ่านภาพทั้งสองนั้นคือ epipolar line จากผลการทดลองจะเห็นว่าภาพคู่ล่างที่ถูกบิดมาให้ขนานกันนั้น จะถูกเลื่อนให้แถวทั้งสองข้างเสมอกันด้วย กล่าวคือ เราสามารถสังเกตเส้น epipolar line ที่ลากผ่านภาพกระดานหมากรุกของภาพคู่ล่างจะเห็นได้ว่า ภาพนั้นได้ถูกเลื่อนมาให้อยู่แถวเดียวกัน ซึ่งแตกต่างอย่างเห็นได้ชัด ถ้าเทียบกับคู่ภาพบน หลังจากนั้นเราจะนำผลการทดลองที่ได้(หลังจากทำการ rectify แล้ว)

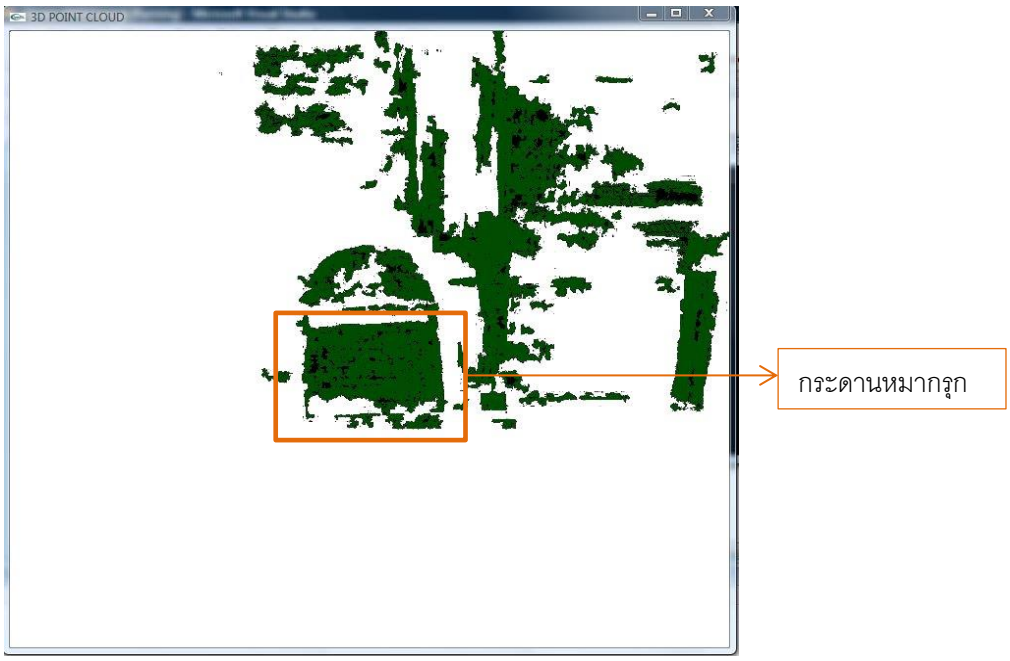
ไปหาค่า disparity ซึ่งได้กล่าวไว้ในกระบวนกการ stereo correspondence และนำไปหาพิกัดในแกนสามมิติซึ่งจะกล่าวไว้ในหัวข้อถัดไป

2.3 ผลการทดลองของการหาภาพสามมิติ

จากขั้นตอนการหาพิกัดในแกนสามมิติ เราได้ทำการทดลองโดยนำคู่ภาพที่ถูกบิดมาให้ขนานกัน (rectification) มาหาค่า disparity โดยทำการปรับค่าตัวแปรจากการยึดภาพกระดานหมากรุกเป็นหลัก จากนั้นใช้คำสั่ง cvReprojectImageTo3D() ซึ่งใช้ค่าเมทริกซ์ Q ที่ได้จากขั้นตอน rectification เพื่อคำนวณหาพิกัดของวัตถุที่ปรากฏบนภาพในแกนสามมิติ (พิกัดสามมิติบนแกนของกล้องซ้าย) ซึ่งเราจะแสดงผลการทดลองในรูปแบบของภาพสามมิติ โดยใช้ค่าสีแดงแสดงค่าความลึกจากวัตถุไปยังแกนของกล้องซ้าย (แกนสามมิติที่ 1) จากการกำหนดค่า X และ Y มีค่าเป็น 0 ซึ่งเราจะใช้ผลที่ได้จากคู่ภาพลำดับที่ 1 และ 3 จากภาพถ่ายในข้อมูลชุดที่ 1 ดังแสดงในรูปที่ 2.3.1



(ก) ภาพสามมิติจากคู่ภาพลำดับที่ 1

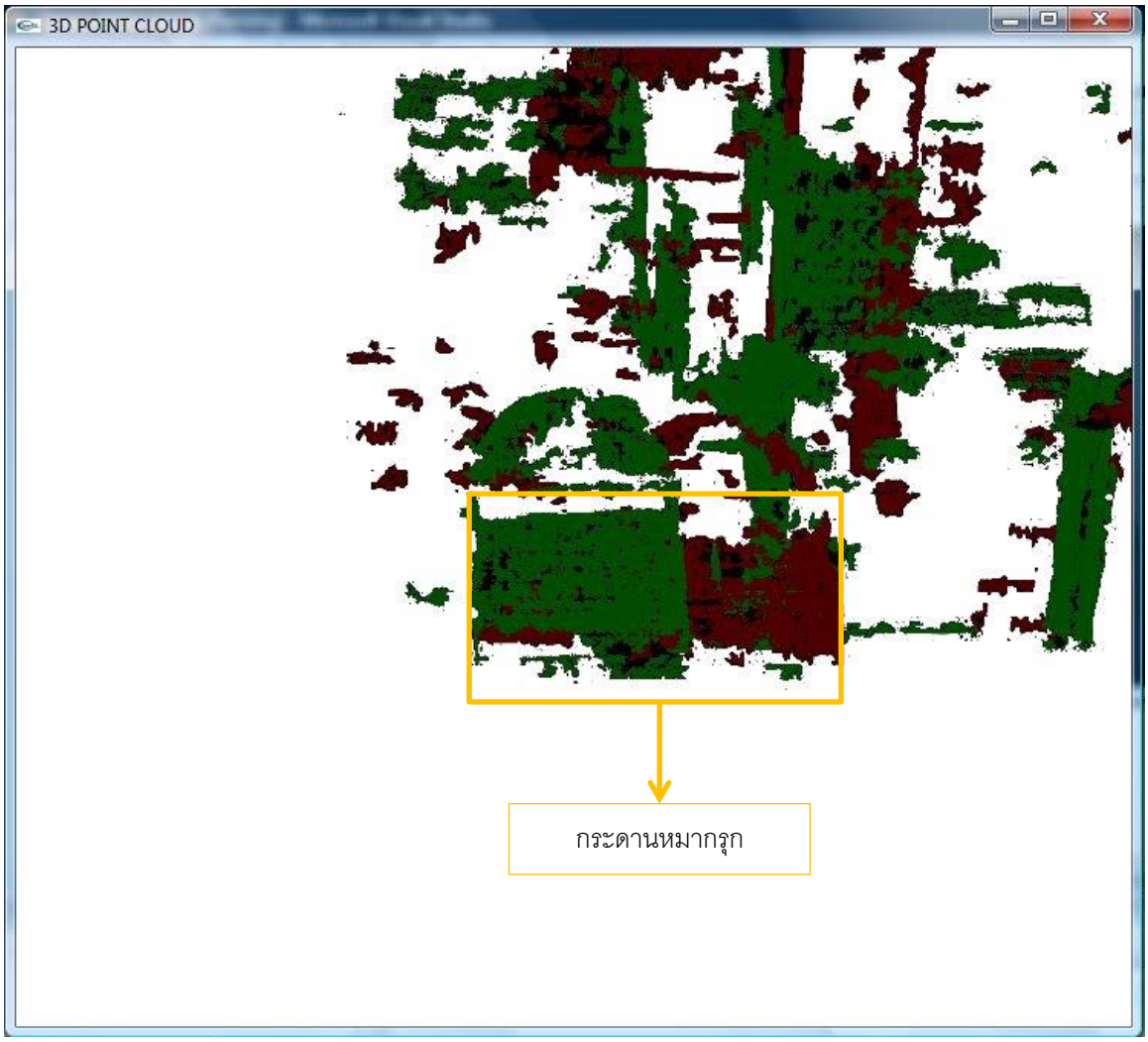


(ข) ภาพสามมิติจากคู่ภาพลำดับที่ 3

รูปที่ 2.3.1 **ภาพ (ก) และภาพ (ข) นั้นเป็นภาพสามมิติบนแกนสามมิติแกนที่** 2 **(แกนขวา)**

3. ผลการทดลองการประกอบภาพสามมิติสองวิวโดยหาแกนอ้างอิงร่วม

จากผลการทดลองในหัวข้อที่ 1 และหัวข้อที่ 2 เราจะได้ภาพสามมิติสองภาพบนแกนสามมิติที่แตกต่างกัน ดังที่แสดงในรูปที่ 3.1 เราจะเห็นได้ว่าภาพสามมิติจากแกนสามมิติที่ 1 (แกนซ้าย) ซึ่งเป็นภาพสิ่งแวดล้อมภายในภาพจะแสดงให้เห็นเป็นสีแดง และภาพสามมิติจากแกนสามมิติแกนที่ 2 (แกนขวา) ซึ่งสิ่งแวดล้อมภายในภาพจะแสดงให้เป็นเป็นสีเขียว เราจะเห็นได้ว่าภาพสามมิติทั้งสองภาพนั้นไม่ซ้อนทับกันพอดี เนื่องจากว่าทั้งสองภาพตั้งอยู่บนแกนสามมิติคนละแกนกัน



รูปที่ 3.1 (ข) ภาพสามมิติจากแกนสามมิติแกนที่ 1 และแกนที่ 2 จากคู่ภาพลำดับที่ 3

รูปที่ 3.1 ภาพสามมิติ (ก) และ (ข) จากแกนสามมิติแกนที่ 1 (รูปสีแดง) และแกนที่ 2 (รูปสีเขียว) โดยที่แกนสามมิติทั้งสองแกนนั้นอยู่คนละตำแหน่งกัน

จากนั้นเราจะใช้ภาพที่ถูก rectify คือภาพซ้ายในแกนสามมิติแกนที่ 1 (ภาพซ้าย) และภาพที่ถูก rectify ของภาพซ้ายในแกนสามมิติแกนที่ 2 (ภาพขวา) ทั้งหมด 15 คู่ภาพใส่ลงในข้อมูลชุดที่ 3 แล้วป้อนกลับเข้าไปยังฟังก์ชัน cvStereoCalibrate() เพื่อหาเมตริกซ์การหมุน R และเมตริกซ์การเลื่อน T ซึ่งทั้งสองเมตริกซ์มีความสัมพันธ์จากแกนสามมิติแกนที่ 2 (แกนขวา) ไปแกนสามมิติแกนที่ 1 (แกนซ้าย) ซึ่งได้ค่าผลการทดลองดังต่อไปนี้

$$ {}^{r}_{l}R = \begin{bmatrix} 9.9593135173304215e\text{-}001 & 8.4713005465263072e\text{-}002 & 3.0731894511041392e\text{-}002 \\ -8.2733703286932705e\text{-}002 & 9.9471573736973362e\text{-}001 & -6.0792566728318120e\text{-}002 \\ -3.5719420146822785e\text{-}002 & 5.8002659715133215e\text{-}002 & 9.9767720956747585e\text{-}001 \end{bmatrix} $$

$$ {}^{r}_{l}T = \begin{bmatrix} -3.5382598753630589e+002 & 3.2185325790274092e+001 & 3.3924682360131285e+001 \end{bmatrix}^{T} $$

โดย ${}^{r}_{l}R$ คือ เมตริกซ์การหมุนจากแกนสามมิติแกนที่ 2(แกนขวา) ไปแกนสามมิติแกนที่ 1 (แกนซ้าย)

${}^{r}_{l}T$ คือ เมตริกซ์การเลื่อนจากแกนสามมิติแกนที่ 2(แกนขวา) ไปแกนสามมิติแกนที่ 1 (แกนซ้าย)

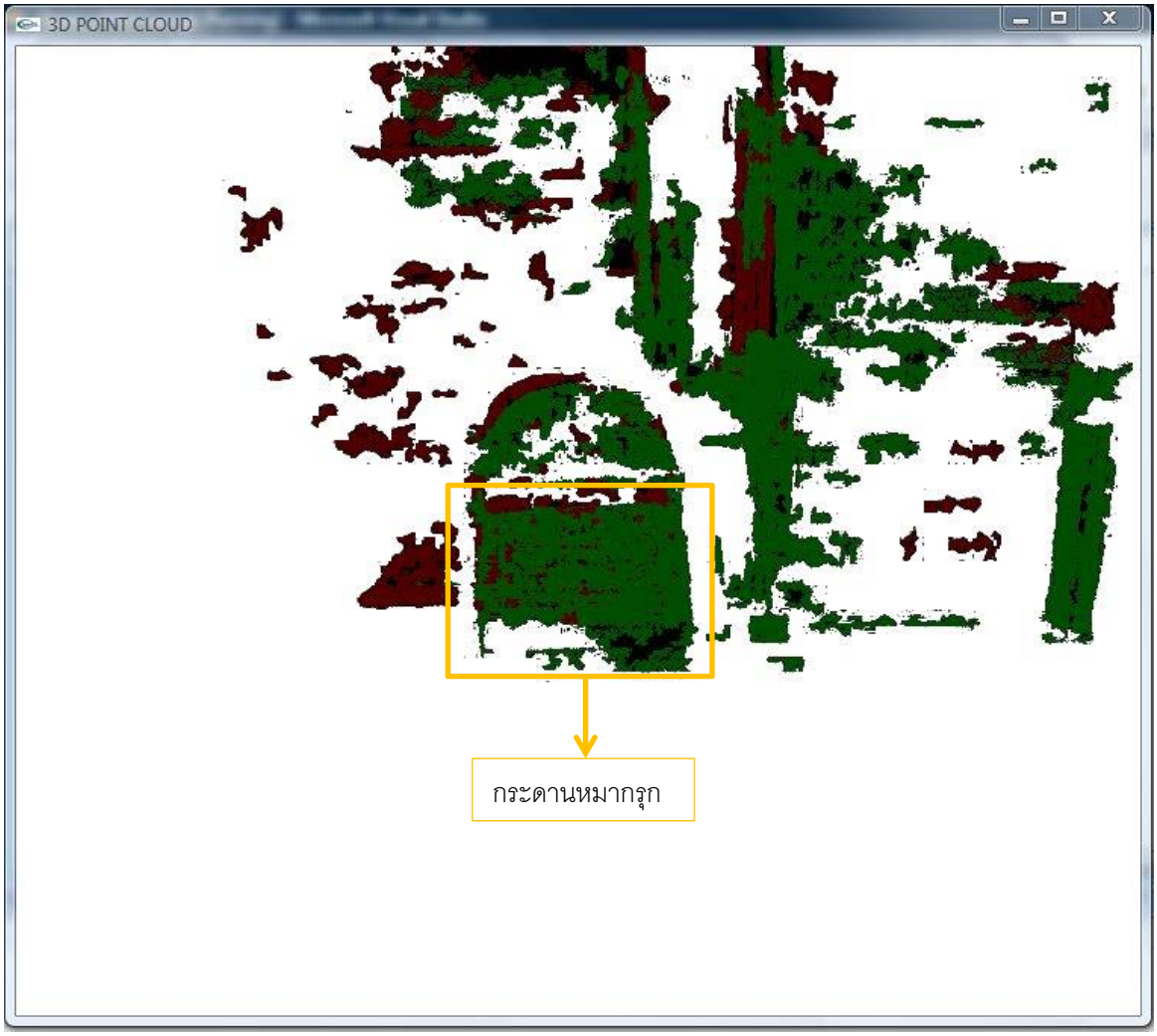
จากนั้นเราได้นำค่าเมตริกซ์การหมุน R และเมตริกซ์การเลื่อน T นำไปแทนในเมตริกซ์ ${}^{r}_{l}H$ ซึ่งได้ผลดังนี้

$$ {}^{r}_{l}H = \begin{bmatrix} 9.9593135173304215e\text{-}001 & 8.4713005465263072e\text{-}002 & 3.0731894511041392e\text{-}002 & (3.5382598753630589e+002)/16 \\ -8.2733703286932705e\text{-}002 & 9.9471573736973362e\text{-}001 & -6.0792566728318120e\text{-}002 & (-3.2185325790274092e+001)/16 \\ -3.5719420146822785e\text{-}002 & 5.8002659715133215e\text{-}002 & 9.9767720956747585e\text{-}001 & (-3.3924682360131285e+001)/16 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

และเราจะหาแกนอ้างอิงร่วมโดยยึดแกนสามมิติแกนที่ 2 (แกนขวา) เป็นหลักแล้วทำการฉายกลับภาพสามมิติจากแกนที่ 1 ไปยังแกนสามมิติแกนที่ 2 โดยการเลื่อนพิกัดสามมิติทุกจุดจากภาพสามมิติแกนที่ 1 โดยใช้สมการ $P_{right} = {}^{r}_{l}HP_{left}$ โดยได้ถูกอธิบายรายละเอียดไว้ในหัวข้อ 1.3

ซึ่งจะได้ผลการทดลองเป็นภาพสามมิติสองภาพซ้อนทับกันบนแกนสามมิติแกนที่ 2 ดังที่แสดงในรูป 3.2

รูปที่ 3.2 (ก) ภาพสามมิติจากแกนสามมิติแกนที่ **1** และแกนที่ **2** จากคู่ภาพลำดับที่ **1**



รูปที่ 3.2 (ข) ภาพสามมิติจากแกนสามมิติแกนที่ **1** และแกนที่ **2** จากคู่ภาพลำดับที่ **3**

รูปที่ 3.1 **ภาพสามมิติ** (ก) **และ** (ข) **จากแกนสามมิติแกนที่** 1 (**รูปสีแดง**) **และแกนที่** 2 (**รูปสีเขียว**) **โดยที่รูปสามมิติบนแกนสามมิติที่** 1 (**แกนซ้าย**) **ได้ถูกฉายกลับไปยังแกนสามมิติแกนที่** 2 (**แกนขวา**) **ซึ่งรูปสามมิติสองรูปนี้จะอยู่ตำแหน่งเดียวกันพอดี**

สรุปและเสนอแนะ

จากการทดลองสามารถสรุปได้ว่า เราสามารถคำนวณหาภาพสามมิติได้จากกล้องเวปแคมสามตัว โดยใช้พื้นฐานของกล้องประเภทเสตอริโอได้ และสามารถซ้อนทับภาพสามมิติสองภาพซึ่งอยู่บนแกนสามมิติสองแกนที่แตกต่างกันได้ โดยใช้ความสัมพันธ์ระหว่างภาพจากกล้องซ้ายที่บิดไปขนานกับแกนสามมิติแกนที่ 1 และภาพจากกล้องกลางที่บิดไปขนานกับแกนสามมิติแกนที่ 2 อย่างไรก็ตามเราต้องปรับค่าเมตริกซ์ที่จะนำไปคูณกับพิกัดในภาพสามมิติภาพซ้ายในแกนสามมิติแกนที่ 1 เพื่อย้ายไปยังแกนสามมิติแกนที่ 2 โดยหารด้วย 16 บนพิกัดของภาพบนแกนสามมิติ เนื่องจากในขั้นตอนการหา disparity ได้มีการคูณด้วยค่า 16 เข้าไปเพื่อให้ได้รับค่า disparity ที่ใกล้เคียงความจริงเพิ่มขึ้น

บรรณานุกรม

[1] Ramesh Jain, Rangachar Kasturi and Brian G.Schunck, “Machine Vision”, McGraw-Hill,1995

[2] Rafael C. Gonzalez ,Richard E. Woods, “Digital Image Processing”, Prentice-Hall, 2002

[3] Carsten Sterger, Markus Ulrich, Christian Wiedeman, “Machine Vision Algorithms and Applications”, Wiley-VCH, 2008

[4] Gary Bradski, Adrian Kaehler, “Learning OpenCV”, O’REILLY, 2008