รายงานการวิจัยฉบับสมบูรณ์

การควบคุมการมองเห็นของหุ่นยนต์ ด้วยการตรวจจับการเคลื่อนไหวของดวงตา

ROBOT VISION CONTROL BY EYE MOVEMENT DETECTION

สรยุทธ กลมกล่อม

พัทธพล กุลถวายพร

ภูมิพัทธ์ รัตนสุวรรณ

เจริญ วงษ์ชุ่มเย็น

ได้รับทุนสนับสนุนงานวิจัยจากแหล่งเงินรายได้ ประจำปีงบประมาณ 2556

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

รายงานการวิจัยฉบับสมบูรณ์

การควบคุมการมองเห็นของหุ่นยนต์ ด้วยการตรวจจับการเคลื่อนไหวของดวงตา

ROBOT VISION CONTROL BY EYE MOVEMENT DETECTION

สรยุทธ กลมกล่อม

พัทธพล กุลถวายพร

ภูมิพัทธ์ รัตนสุวรรณ

เจริญ วงษ์ชุ่มเย็น

ได้รับทุนสนับสนุนงานวิจัยจากแหล่งเงินรายได้ ประจำปีงบประมาณ 2556

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อโครงการ (ภาษาไทย) การควบคุมการมองเห็นของหุ่นยนต์ด้วยการตรวจจับการเคลื่อนไหวของดวงตา

แหล่งเงิน แหล่งเงินรายได้

ประจำปีงบประมาณ 2556 จำนวนเงินที่ได้รับการสนับสนุน 80,000 บาท

ระยะเวลาทำการวิจัย 1 ปี ตั้งแต่ ตุลาคม 2555 ถึง กันยายน 2556

ชื่อ-สกุล หัวหน้าโครงการ และผู้ร่วมโครงการวิจัย

อาจารย์ สรยุทธ กลมกล่อม

นาย พัทธพล กุลถวายพร

นาย ภูมิพัทธ์ รัตนสุวรรณ

ผศ. เจริญ วงษ์ชุ่มเย็น

บทคัดย่อ

การควบคุมอุปกรณ์อิเล็กทรอนิกส์โดยใช้ส่วนหนึ่งของร่างกายในการควบคุม จะทำให้ผู้ใช้งานสามารถควบคุมได้ตรงกับความต้องการมากที่สุด ทางคณะผู้วิจัยจึงเลือกทำการควบคุมการเคลื่อนไหวของกล้องจับภาพด้วยการเคลื่อนของดวงตา ซึ่งกล้องจับภาพสามารถเคลื่อนไหวได้ตามมุมองศาตามที่ดวงตาเคลื่อนที่ไป โดยการพัฒนาระบบการเคลื่อนไหวกล้องด้วยการเคลื่อนไหวของดวงตาจะประกอบด้วยส่วนของอุปกรณ์สวมใส่บนศีรษะติดตั้งกล้องจับภาพดวงตาซึ่งจะจับภาพดวงตาของผู้ใช้งาน ส่วนต่อมาคือส่วนแอพพลิเคชั่นเพื่อประมวลผลภาพการเคลื่อนที่ของดวงตาที่รับมาจากส่วนอุปกรณ์จับภาพดวงตาเพื่อคำนวณองศาการหมุนของกล้องจับภาพของหุ่นยนต์ และส่วนสุดท้ายตัวหุ่นยนต์ติดกล้องที่สามารถเคลื่อนไหวได้ครึ่งทรงกลม จะส่งภาพที่จับได้มาแสดงผลที่ส่วนแอพพลิเคชั่นเพื่อแสดงแก่ผู้ใช้งาน ซึ่งทางผู้วิจัยได้ศึกษาและทดลองการประมวลผลภาพดวงตาในรูปแบบต่างๆและสรุปผลเพื่อให้ได้แอพพลิเคชั่นที่มีประสิทธิภาพสูงที่สุด และสร้างหุ่นยนต์ติดตั้งกล้องจับภาพที่สามารถเคลื่อนที่กล้องจับภาพในลักษณะที่ใกล้เคียงกับการเคลื่อนไหวของดวงตามนุษย์มากที่สุด

คำสำคัญ : การตรวจจับดวงตา, การเคลื่อนไหวของดวงตา, การมองเห็นของหุ่นยนต์

Research Title: Robot Vision Control by Eye Movement Detection

Researcher:

Mr. Sorayut Glomglome

Mr. Patthapol Kulthawaiporn

Mr. PoomPat Rattanasuwan

Asst. Prof. Charoen Vongchumyen

Faculty: Engineering **Department:** Computer Engineering

ABSTRACT

If electronic devices are controlled by human's perception, it would be match to user's need. So, researchers have developed the robot vision control system by eye movement detection to make a moveable camera on the robot move like human's eyes. This system consist of 3 parts which is application for image processing, camera for detect eyes, and the robot with moveable camera. The robot also can move. An eyes detection part will send data to application on computer to process eyes image and send the result to control the robot and its camera. Also, the robot's camera will send frame to show on the application. For the best result, researchers have studied and test the image processing in many ways and designed the movement of the robot's camera to be the most similar to the movement of human's eye.

Keywords : Eye Tracking, Eye Movement, Robot Vision

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญภาพ

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการควบคุมอุปกรณ์อิเล็กทรอนิกส์ส่วนมากจะควบคุมผ่านสวิตช์หรือปุ่มกดหรือซึ่งทำให้ควบคุมได้ไม่สะดวก ซึ่งหากสามารถควบคุมอุปกรณ์อิเล็กทรอนิกส์ด้วยประสาทสัมผัสอื่นๆของมนุษย์นอกจากมือได้ คล้ายกับว่าอุปกรณ์อิเล็กทรอนิกส์ใดๆเป็นส่วนหนึ่งของร่างกายมนุษย์สามารถเคลื่อนไหวได้อย่างอิสระและตรงกับความต้องการของผู้ใช้งานมากขึ้น ดังนั้นทางคณะผู้วิจัยมีความสนใจในการพัฒนาการควบคุมอุปกรณ์อิเล็กทรอนิกส์ด้วยประสาทสัมผัสของมนุษย์คือ ดวงตา โดยได้ทดลองควบคุมการหมุนของกล้องให้หมุนได้ตามที่องศาที่ดวงตาหมุนจริง

การพัฒนาระบบการควบคุมกล้องจับภาพด้วยดวงตาจะต้องทำการประมวลผลภาพเพื่อหาตำแหน่งของดวงตาซึ่งมีหลายงานวิจัยที่ได้ทำการประมวลผลภาพเพื่อหาอวัยวะบนใบหน้ามาก่อนหน้านี้ เช่น การประมวลผลภาพใบหน้าเพื่อระบุอารมณ์ของผู้ใช้, การประมวลผลภาพดูการเคลื่อนไหวของดวงตาเพื่อสังเกตพฤติกรรมของมนุษย์ เป็นต้น ซึ่งทางคณะผู้วิจัยได้นำการประมวลผลภาพใบหน้าและดวงตาเพื่อนำมาพัฒนาในแง่มุมของการควบคุมอุปกรณ์อิเล็กทรอนิกส์ โดยการพัฒนาระบบการควบคุมกล้องจับภาพด้วยดวงตาจะติดตั้งกล้องที่ต้องการควบคุมบนตัวหุ่นยนต์เพื่อให้สามารถเคลื่อนที่ได้ในรูปแบบเดียวกับการเคลื่อนที่ของดวงตา และการส่งข้อมูลไปยังตัวหุ่นยนต์จะส่งผ่านระบบเครือข่ายไร้สาย ทำให้เกิดความสะดวกยิ่งขึ้นเนื่องจากผู้ใช้งานไม่จำเป็นต้องอยู่ใกล้กับกล้องที่ต้องการควบคุมและสามารถนำไปใช้งานจริงได้กว้างขวางมากขึ้น

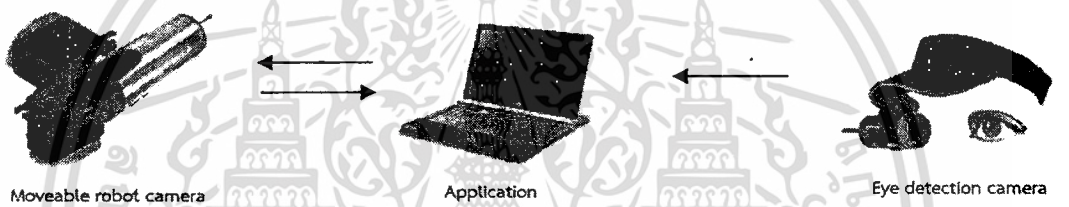
1.2 วัตถุประสงค์ของการวิจัย

1) การควบคุมอุปกรณ์อิเล็กทรอนิกส์ด้วยการหมุนของดวงตาเพื่อนำมาควบคุมแทนประสาทสัมผัสอื่นๆ
2) เพื่อนำความรู้ในหลายๆด้าน ได้แก่ การประมวลผลภาพ, การรับส่งข้อมูลผ่านระบบเครือข่ายไร้สาย, การพัฒนาบน Micro Controller เป็นต้น มาพัฒนาเพื่อต่อยอดเทคโนโลยีในปัจจุบัน
3) ลดค่าใช้จ่ายในการสร้างเทคโนโลยี

1.3 ขอบเขตของการวิจัย

ระบบการควบคุมกล้องจับภาพบนตัวหุ่นยนต์ เป็นระบบที่สามารถกำหนดการเคลื่อนไหวของกล้องจับภาพผ่านคอมพิวเตอร์ที่สั่งการด้วยการเคลื่อนไหวของดวงตา โดยกล้องจับภาพจะติดอยู่บนหุ่นยนต์ที่สามารถเคลื่อนที่ได้ในลักษณะของการหมุนแบบครึ่งทรงกลม การส่งข้อมูลจะมีการทำงานแบบไร้สายแบบ Infrastructure Mode ซึ่งสามารถเคลื่อนย้ายอุปกรณ์ได้อย่างอิสระ

การพัฒนาโครงงานวิจัยนี้ เพื่อให้หุ่นยนต์สามารถเคลื่อนกล้องจับภาพซึ่งสามารถหมุนได้ตามการเคลื่อนไหวของดวงตา โดยใช้กล้อง 1 ตัวจับภาพการเคลื่อนไหวของตาดำ 1 ข้างแล้วส่งข้อมูลไปยัง Application บนคอมพิวเตอร์ จากนั้น Application จะประมวลผลภาพการเคลื่อนไหวของตาและส่งข้อมูลการเคลื่อนไหวของกล้องผ่าน Internet ไปควบคุมกล้องที่ติดอยู่บนหุ่นยนต์ และในขณะเดียวกันกล้องที่ตัวหุ่นยนต์ก็สามารถส่งภาพกลับมาที่จอภาพอย่างไร้สายได้เช่นกัน



ภาพที่ 1.1 การส่งข้อมูลระหว่างกล้องจับภาพดวงตา แอพพลิเคชั่น และกล้องบนหุ่นยนต์

การพัฒนาระบบการควบคุมกล้องหุ่นยนต์ด้วยการเคลื่อนที่ของดวงตานั้นจะพัฒนาเป็น 3 ส่วนหลัก ได้แก่

1) อุปกรณ์จับภาพดวงตาซึ่งจะส่งภาพไปยัง Application บนคอมพิวเตอร์โดยสามารถจับภาพของดวงตา แล้วส่งไปให้ Application บนคอมพิวเตอร์ทำการประมวลผลต่อไป
2) Application บนคอมพิวเตอร์ทำหน้าที่ในการประมวลผลภาพและส่งข้อมูลผลลัพธ์ดังนี้
   a) ประมวลผลภาพดวงตาที่ได้จากอุปกรณ์จับภาพดวงตาเพื่อคำนวณตำแหน่งของดวงตา องศาของดวงตา
   b) ส่งข้อมูลผลลัพธ์คือ องศาของดวงตาเทียบกับจุดอ้างอิงไปยัง Micro controller บนตัวหุ่นยนต์ ด้วย Infrastructure Mode
   c) รับข้อมูลภาพจากกล้องบนตัวหุ่นยนต์ได้และแสดงผลบนหน้าจอ โดยการรับข้อมูลจะผ่านระบบเครือข่ายไร้สาย Infrastructure Mode

3) หุ่นยนต์ติดกล้องจับภาพ จะควบคุมโดย Micro controller โดยติดกล้องบนตัวหุ่นยนต์ที่สามารถหมุนด้วย Servo Motor

a) กล้องบนตัวหุ่นยนต์สามารถหมุนในแนวแกน X 120 องศา แนวแกน Y ได้ 120 องศา

b) หุ่นยนต์สามารถประมวลผลการคำนวณตำแหน่งของการเคลื่อนไหวกล้องเองได้

c) สามารถรับคำสั่งจาก Application บนคอมพิวเตอร์ด้วยการส่งข้อมูลแบบ Infrastructure Mode

d) ส่งข้อมูลภาพจากกล้องกลับไปยัง Application บนคอมพิวเตอร์ด้วยการส่งข้อมูลแบบไร้สาย

การออกแบบ Protocol การส่งข้อมูลเฉพาะของ Application

1) ออกแบบการส่งข้อมูลด้วย Protocol TCP ในชั้น Transport Layer ทั้งหมด

2) ออกแบบ Sequence การส่งข้อมูล

3) ไม่ได้ออกแบบให้มีความปลอดภัยในการส่งข้อมูลผ่าน Internet

1.4 วิธีดำเนินการวิจัย

การพัฒนาระบบการควบคุมกล้องจับภาพบนตัวหุ่นยนต์ด้วยการหมุนของดวงตาจะแบ่งเป็น 4 ส่วนใหญ่ๆได้แก่

1.4.1 การพัฒนา Application บนคอมพิวเตอร์ ได้แก่

1) ศึกษาพื้นฐานเกี่ยวกับภาพและการประมวลผลภาพ พร้อมทั้งศึกษาการเขียนโปรแกรมติดต่อกับกล้อง WebCam ด้วยภาษา C#

2) ศึกษางานวิจัยการประมวลผลภาพที่เกี่ยวข้องกับการระบุอวัยวะบนใบหน้ามนุษย์

3) ค้นหาหรือคิด Algorithm ในการระบุใบหน้าจากภาพที่สนใจ โดยศึกษาจาก EmguOpenCV

4) นำ Algorithm ที่พบมาเขียนโปรแกรมเพื่อระบุพิกัดของดวงตาได้

5) ศึกษาการเขียนโปรแกรมติดต่อผ่านระบบเครือข่าย

6) เขียนโปรแกรมติดต่อผ่านระบบเครือข่ายและทดลองการส่งข้อมูลระหว่าง PC กับ PC

7) คิดค้น Protocol ในการส่งข้อมูลเพื่อให้สามารถส่งภาพหรือคำสั่งไปยังตัวหุ่นยนต์ได้

8) ออกแบบ User Interface ของ Application บนคอมพิวเตอร์

1.4.2 การพัฒนาตัวหุ่นยนต์ติดกล้อง ได้แก่

1) ศึกษาการเขียนโปรแกรมติดต่อกับ Micro Controller
2) ทดลองเขียนโปรแกรมติดต่อ Micro Controller สั่งงาน Port ต่างๆ
3) ศึกษาการใช้งาน WiFi Module ด้วย Micro Controller
4) ออกแบบการติดตั้งกล้องบนตัวหุ่นยนต์
5) ออกแบบการสร้างตัวหุ่นยนต์และจัดหาวัสดุอุปกรณ์ที่จำเป็นต้องใช้ในการสร้างสร้างตัวหุ่นยนต์
6) ติดตั้งกล้องบนตัวหุ่นยนต์และทดลองการหมุนของ Servo Motor
7) ปรับระบบการหมุนของ Servo Motor ให้กล้องหมุนได้ตามต้องการ
8) ทดลองเขียนโปรแกรมติดต่อการส่งข้อมูลผ่านระบบเครือข่ายของ PC กับหุ่นยนต์

1.4.3 ทำอุปกรณ์ติดตั้งกล้องเพื่อจับภาพดวงตา

1) ออกแบบอุปกรณ์ติดตั้งกล้อง
2) จัดหาวัสดุอุปกรณ์ที่จำเป็นต้องใช้ในการสร้าง
3) สร้างอุปกรณ์ติดตั้งกล้อง
4) ปรับปรุงอุปกรณ์ที่สร้างขึ้นให้สามารถใช้งานได้ง่ายและเข้ากับสรีระ

1.4.4 การปรับปรุงระบบให้ทำงานได้ดีขึ้นและจัดทำเอกสารรวมทั้งการนำเสนอโครงงาน

1) นำระบบทั้ง 3 ส่วนมาใช้งานร่วมกัน
2) ปรับปรุงคุณภาพของระบบโดยรวม
3) จัดทำเอกสารทั้งหมด

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) สามารถพัฒนาและสร้างระบบที่ใช้การควบคุมอุปกรณ์อิเล็กทรอนิกส์ด้วยดวงตาได้ ส่งผลให้นำไปต่อยอดการควบคุมอุปกรณ์อื่นๆได้มากมาย
2) ได้รับความรู้เกี่ยวกับการประมวลผลภาพและสามารถนำความรู้ไปต่อยอดพัฒนาอย่างอื่นได้ในอนาคต
3) ได้รับความรู้เกี่ยวกับการพัฒนาหุ่นยนต์ด้วย Micro Controller ซึ่งส่งผลต่อการนำไปต่อยอดได้ในอนาคต
4) สามารถลดค่าใช้จ่ายในการสร้างเทคโนโลยีใหม่ๆได้
5) รู้จักการแก้ปัญหาเพื่อให้โครงงานสำเร็จด้วยดี
6) เรียนรู้การทำเอกสารซึ่งเป็นส่วนหนึ่งของการพัฒนา Application ซึ่งสามารถนำไปใช้ในอนาคตการทำงานได้

บทที่ 2
แนวคิดและทฤษฎีที่เกี่ยวข้อง

2.1 การประมวลผลภาพ

การประมวลผลภาพ คือการเอาภาพมาประมวลผล โดยคำนวณด้วยคอมพิวเตอร์ เพื่อให้คอมพิวเตอร์รู้ว่าภาพนั้นคือภาพอะไรใช่สิ่งที่เราต้องการหรือไม่

การมองเห็นของมนุษย์เป็นสิ่งที่สำคัญและเป็นกลไกการรับภาพที่ซับซ้อนอย่างหนึ่ง ซึ่งจะให้ข้อมูลที่มีความจำเป็นสำหรับใช้ในงานง่าย ๆ (ตัวอย่างเช่น การจดจำวัตถุ)และสำหรับงานที่มีความซับซ้อน(ได้แก่ การวางแผน การตัดสินใจ การค้นคว้าทางวิทยาศาสตร์ การพัฒนาทางด้านความคิด) รูปภาพมีบทบาทมากสำหรับองค์กรต่าง ๆ เช่น หนังสือพิมพ์ โทรทัศน์ ภาพยนตร์ซึ่งได้ใช้ภาพ (ภาพนิ่ง ภาพเคลื่อนที่)เป็นสื่อนำเสนอข้อมูลข่าวสารต่าง ๆ สิ่งที่น่าสนใจของข้อมูลที่เกี่ยวกับการมองเห็นหรือข้อมูลภาพนั้นก็คือกระบวนการประมวลผลภาพ (Image Processing) โดยใช้ดิจิตอลคอมพิวเตอร์

Digital image processing จะเกี่ยวกับการแปลงข้อมูลภาพให้อยู่ในรูปแบบข้อมูลดิจิตอล (Digital format) ซึ่งสามารถที่จะนำเอาข้อมูลนี้จัดผ่านกระบวนการต่าง ๆ ด้วยดิจิตอลคอมพิวเตอร์ได้ ในระบบของดิจิตอล อินพุตและเอาต์พุตของระบบจะอยู่ในรูปแบบดิจิตอลเท่านั้น

Digital image analysis จะเกี่ยวกับวิธีการอธิบายและการจดจำข้อมูลภาพดิจิตอล ซึ่งอินพุตของระบบจะเป็นข้อมูลภาพดิจิตอลและเอาพุตจะเป็นเครื่องหมายที่ใช้แทนข้อมูลภาพดิจิตอลเหล่านั้น ในการวิเคราะห์ภาพมีอยู่หลายวิธีด้วยกันที่ได้นำมาจากการทำงานของตามนุษย์(human vision)นั่นก็คืองานทางด้าน Computer Vision เป็นลักษณะเดียวกับ Digital image analysis นั่นเอง การมองเห็นของมนุษย์นับว่าเป็นกระบวนการที่ซับซ้อนซึ่งลักษณะเทคนิคโดยทั่ว ๆ ไปในกระบวนการ Digital image analysis และ Computer Vision จะค่อนข้างซับซ้อนเช่นกัน

การคำนวณนั้นมีหลายวิธี เช่นการนำเอาจุดสีแต่ละ Pixel มาคำนวณหรือการนำหลาย ๆ จุดมารวมกันเป็นบริเวณกว้างขึ้น เช่น การดูPattern ของวัตถุ การวิเคราะห์หารูปร่างของวัตถุ เพื่อหาค่าอะไรบางอย่างที่ช่วยให้เข้าใจได้ว่า ภาพนั้นมีลักษณะอย่างไร

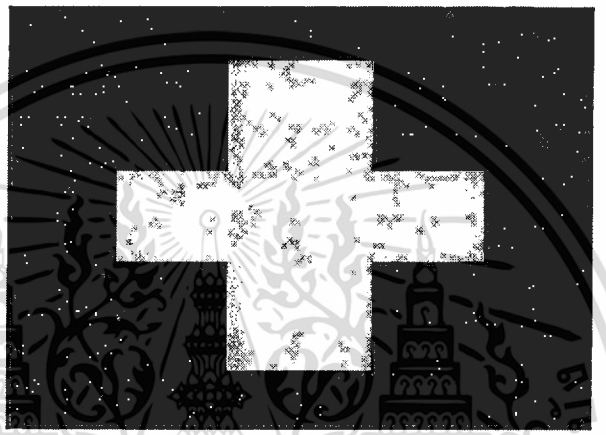
2.1.1 Pixel

Pixel คือ พื้นที่เล็กๆจุดหนึ่งในภาพ โดยในแต่ละจุดนั้นจะมีค่าตัวเลขกำกับ ซึ่งตัวเลข เหล่านี้จะมาจากค่าของแม่สีสามสี R (สีแดง) G (สีเขียว) B (สีฟ้า) ใช้บอกระดับความเข้มของแต่ละ เฉดสี หากมี Pixel หลายๆจุดมาต่อกันจะกลายเป็นภาพซึ่งมีขนาด จำนวน Pixel ด้านกว้าง X จำนวน Pixel ด้านยาว ยกตัวอย่าง เช่น รูปภาพขนาด 800 x 600 pixels หมายความว่ารูปภาพนี้มีความกว้าง 800 pixels และมีความยาว 600 pixels เป็นต้น

2.1.2 ระดับเทา (Gray level)

ระดับเทา (Gray Level) เป็นค่าซึ่งระบุความสว่างหรือความเข้ม ซึ่งมีค่าตั้งแต่ 0-255 (0 คือ ระดับเข้ม 255 คือระดับสว่าง) รวมทั้งพิกัดแนวนอนและแนวตั้ง ซึ่งใช้ระบุตำแหน่งในแถวลำดับ ภาพ (Image Array) เช่นจากรูปตัวอย่างที่ 2.1 และ 2.2 จุดภาพแถวนอนที่ 3 และแนวตั้งที่ 2 ในภาพที่ 2.2 ซึ่งมีค่าระดับเทา 40 วิธีการหาค่าระดับเทา (Gray Level) สามารถทำได้ดังสมการ 2.1

$$\text{Gray Level} = \frac{R+G+B}{3} \tag{2.1}$$



ภาพที่ 2.1 ภาพที่ใช้ในการทดสอบระดับเทา

40	40	40	40	40	40	40	40	40	40
40	40	40	40	200	200	40	40	40	40
40	40	40	40	200	200	40	40	40	40
40	40	200	200	200	200	200	200	40	40
40	40	40	40	200	200	40	40	40	40
40	40	40	40	200	200	40	40	40	40
40	40	40	40	40	40	40	40	40	40

ภาพที่ 2.2 ระดับเทาของ pixel ในแถวแนวนอนที่ 3 และแถวแนวตั้งที่ 2

2.1.3 การแปลงภาพสีเป็นภาพขาว-ดำ

การแปลงภาพสีให้เป็นภาพขาว-ดำ (Thresholding) เป็นกระบวนการแปลงภาพสีให้มีการ แสดงผลได้แค่ 2 ระดับ คือ ขาว และดำ โดยจะแปลงข้อมูลภาพให้เป็นภาพ binary (Binary Image) มีกระบวนการแปลงภาพที่มีความเข้มหลายระดับ (Multilevel Image) ให้เป็นภาพที่มีความเข้มเพียง 2 ระดับ หรือ 1 บิต (bit) คือ 0 และ 1 โดย 0 แทนด้วยจุดที่มีภาพสีขาว และ 1 แทนด้วยจุดที่มีภาพสี ดำ Thresholding Technique คือการพิจารณาจุด pixel ในภาพว่าจุดใดควรจะเป็นจุดขาวหรือจุดใด ควรจะเป็นจุดที่มีค่าเท่ากับ 1 (จุดดำ) โดยจะทำการเปรียบเทียบค่าของแต่ละpixel (f(x,y)) กับค่าคงที่ ที่เรียกว่า Threshold (Threshold Value) เทคนิคนี้นิยมใช้กันมากในกรณีที่ความแตกต่างระหว่าง วัตถุ (Object) และพื้นหลัง (Background) ค่า pixel ในภาพที่มีค่าน้อยกว่าค่า Threshold จะถูก กำหนดเป็น 1 (จุดดำ) และถ้าค่าของ pixel ใด ๆ ในภาพมีค่ามากกว่าหรือเท่ากับค่า Threshold จะถูก กำหนดให้เป็น 0 (จุดขาว) ในการทำภาพ Binary โดยการทำ Thresholding ให้ได้ภาพดีและคมชัด ต้องเกิดจากการเลือกค่า Threshold ที่ถูกต้องและเหมาะสม ถ้าเลือกค่า Threshold ไม่เหมาะสม เช่น ค่า Threshold ที่มากหรือน้อยจนเกินไป ภาพที่ได้จะขาดความคมชัดหรืออาจทำให้รายละเอียดของ ภาพขาดหายไป หรือภาพที่ได้อาจจะมืดเกินไป หรือสว่างเกินไป หรืออาจจะเป็นภาพที่มีสิ่งรบกวน (Noise) เกิดขึ้น ทำให้ภาพผลลัพธ์ที่ได้ไม่ชัดเจน

2.1.4 ระบบสี RGB

ระบบสี RGB เป็นระบบสีที่เกิดจากการรวมกันของแสงสีแดง เขียวและน้ำเงินโดยมีการรวมกันแบบ Additive ซึ่งโดยปกติจะนำ ไปใช้ในจอภาพแบบ CRT (Cathode ray tube) ในการใช้งานระบบสี RGB ยังมีการสร้างมาตรฐานที่แตกต่างกันออกไปที่นิยมใช้งานได้แต่ RGBCIE และ RGBNTSC

2.1.4.1 ระบบสีแบบ RGB ของ CIE

เป็นระบบสีที่พัฒนาขึ้นโดย CIE (Commission International l 'Eclairage) ซึ่งอ้างอิงสีด้วยสีแดงที่ 700nm สีเขียวเท่ากับ 546.1 nm และสีน้ำเงิน 435.8 nm

2.1.4.2 ระบบสีแบบ RGB ของ NTSC

เป็นระบบที่พัฒนาโดย NTSC (National Television System Committee) เพื่อใช้สำหรับการแสดงภาพของจอภาพแบบ CRT เป็นมาตรฐานสำหรับผู้ผลิตแบบ CRT ให้มีลักษณะเดียวกัน

2.1.5 ระบบสี HSV

ระบบสี HSV (Hue Saturation Value) เป็นการพิจารณาสี โดยใช้ Hue Saturation และ Valueซึ่ง Hue คือค่าสีของสีหลัก (แดง เขียวและน้ำเงิน) ในทางปฏิบัติจะอยู่ระหว่าง 0 และ 255 ซึ่งถ้า Hue มีค่าเท่ากับ 0 จะแทนสีแดงและเมื่อ Hue มีค่าเพิ่มขึ้นเรื่อย ๆ สีก็จะเปลี่ยนแปลงไปตามสเปกตรัมของสีจนถึง256 จึงจะกลับมาเป็นสีแดงอีกครั้ง ซึ่งสามารถแทนให้อยู่ในรูปขององศาได้

ดังนี้คือ สีแดง = 0 องศา สีเขียวเท่ากับ 120 องศา สีน้ำเงินเท่ากับ 240 องศา Hue สามารถคำนวณได้จากระบบสี RGB ได้ดังนี้

$$red_h = red - min(red,green,blue) \quad (2.2)$$

$$green_h = green - min(red,green,blue) \quad (2.3)$$

$$blue_h = blue - min(red,green,blue) \quad (2.4)$$

จากลักษณะโมเดลของระบบ Hue พบว่าจะมีค่าอย่างน้อยหนึ่งค่าที่จะเท่ากับ 0 แต่ถ้ามีสองค่าเท่ากับ 0 แล้ว hue จะเป็นมุมของสี(ค่าสี)มีค่าเป็นไปตามสี่ที่สามและถ้าทั้งสามสีมีค่าเท่ากับ 0 แล้วจะทำให้ไม่มีค่าของ Hue หรือสีที่ได้จะมีค่าเท่ากับสีขาวนั่นเอง ตัวอย่างเช่น จอภาพขาว-ดำ ถ้าเกิดมีสีใดสีหนึ่งมีค่าเท่ากับ 0 จะทำให้ค่าสีที่ได้เป็นไปตามสีที่เหลือ การให้น้ำหนักในการพิจารณาเมื่อสีแดงมีค่าเท่ากับ 0

$$\frac{(240\times blue_h)+(120\times green_h)}{blue_h+green_h} \quad (2.5)$$

Saturation คือความบริสุทธิ์ของสีซึ่งถ้า Saturation มีค่าเท่ากับ 0 แล้วสีที่ได้จะไม่มี Hue ซึ่งจะเป็นสีขาวล้วน แต่ถ้า Saturation มีค่าเท่ากับ 255 แสดงว่าจะไม่มีแสงสีขาวผสมอยู่เลย Saturation สามารถคำนวณได้ดังนี้

$$Saturation = \frac{max(red,green,blue)-min(red,green,blue)}{max(red,green,blue)} \quad (2.6)$$

Value คือความสว่างของสี ซึ่งสามารถวัดได้โดยค่าความเข้มของความสว่างของแต่ละสีที่ประกอบกันสามารถคำนวณได้จาก

$$value = max(red,green,blue) \quad (2.7)$$

2.1.6 การตรวจหาตำแหน่งดวงตา

การตรวจหาตำแหน่งบริเวณดวงตาจะแสดงดังภาพที่ 2.3 ซึ่งได้แบ่งออกเป็นสองขั้นตอน ขั้นตอนแรกจะทำการหาบริเวณดวงตาจากค่าข้อมูลสี (Chrominance) และขั้นตอนที่สองเป็นการตรวจหาตำแหน่งบริเวณดวงตาจากค่าความสว่าง (Luminance) จากภาพสี YCbCr โดยตำแหน่ง

บริเวณดวงตาจากค่าข้อมูลสี (Chrominance) ได้จากการสังเกตค่าสูงสุดของค่าข้อมูลสี Cb และค่าต่ำสุด Cr ซึ่งหาได้จากสมการ ดังนี้

$$E_{MC} = \frac{1}{3}\{(C_b^2) + (C_r^{\sim 2}) + (C_b/C_r)\} \quad (2.8)$$

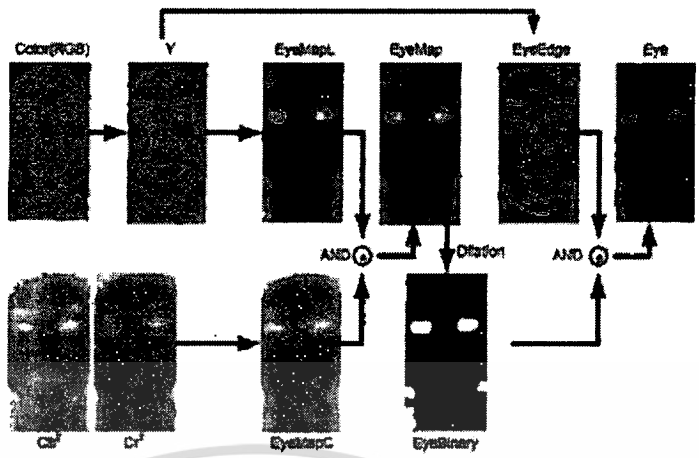
สมการ 2.8 คือการหา EyeMapC ดังภาพที่ 2.3 โดย (C2b),(C~r 2) และ (Cb/Cr) โดยทำการปรับค่าทั้งหมดให้อยู่ ในช่วง (0-255)และ C~r คือ ค่าลบของ (255- Cr) เนื่องจาก ตำแหน่งบริเวณดวงตา จะมีลักษณะของจุดภาพที่สว่างและมืด อยู่ในค่าความสว่าง (Luminance) ซึ่งเป็นคุณสมบัติของตัว ดำเนินการมอร์โฟโลยีค่าระดับเทาโดยการไดเลชันและอีโรชัน ซึ่งในงานวิจัยจะใช้ การไดเลชัน และ อีโรชันค่าระดับเทาร่วมกับ 8 ส่วนประกอบโครงร่างแบบ hemispheric ซึ่งตำแหน่งบริเวณดวงตาที่ตรวจพบจากค่าความสว่าง หาได้จากสมการดังนี้

$$E_{ML} = \frac{Y(x,y) \oplus g\sigma(x,y)}{Y(x,y) \Theta g\sigma(x,y)+1} \quad (2.9)$$

สมการ 2.9 คือการหา EyeMapL ดังภาพที่ 2.3 เมื่อตัวดำเนินการไดเลชันและอีโรชัน ค่าระดับเทา อยู่บน ฟังก์ชันพื้นฐาน f:F $\subset$ R2 -> Rโดยใช้ส่วนประกอบโครงร่างซึ่งสามารถเขียนเป็นฟังก์ชันได้ดังนี้ g:G $\subset$ R2 -> R เพื่อทำให้ภาพตำแหน่งบริเวณดวงตาที่ตรวจพบจากค่าข้อมูลสีมีคุณภาพดีขึ้น จะนำภาพตำแหน่งบริเวณดวงตาที่ตรวจพบ จาก ค่าข้อมูลสีไปทำการประมวลผลภาพด้วยฮิสโตรแกรมอีควอไลเซชัน เพื่อให้ระดับความเข้มของแสงภาพมีการกระจายอย่างสม่ำเสมอ แล้วนำภาพที่ได้จากการประมวลผลไป AND เข้ากับ ภาพตำแหน่งบริเวณดวงตาที่ตรวจพบจากค่าความสว่าง ซึ่งหาได้จากสมการ ดังนี้

$$EMAP = (EMC)AND(EML) \quad (2.10)$$

สมการ 2.10 คือกระบวนการทำ EyeMap ดังภาพที่ 2.3 ซึ่งภาพผลลัพธ์ที่ได้จากการตรวจหาตำแหน่งบริเวณดวงตา เรายังสามารถทำการปรับค่าไดเลชัน เพื่อเพิ่มความสว่างให้กับ ตำแหน่งบริเวณดวงตา รวมถึงการปรับปรุงภาพโดยใช้เทคนิค การกำหนดค่าเทรสโชล และค่าไบนารี ที่เหมาะสมให้กับภาพ จะทำให้ภาพผลลัพธ์มีประสิทธิภาพมากยิ่งขึ้นเมื่อทำกระบวนการดังกล่าวแล้ว จะได้ผลลัพธ์คือภาพ EyeBinary ดังภาพที่ 2.3 จากนั้นนำภาพ EyeEdge และ EyeBinary มาทำกระบวนการ AND ซึ่งจะได้ผลลัพธ์คือขอบของดวงตาเท่านั้น ซึ่งกระบวนการตรวจหาตำแหน่งดวงตาจะแสดงดังภาพที่ 2.3



ภาพที่ 2.3 การตรวจหาดวงตา

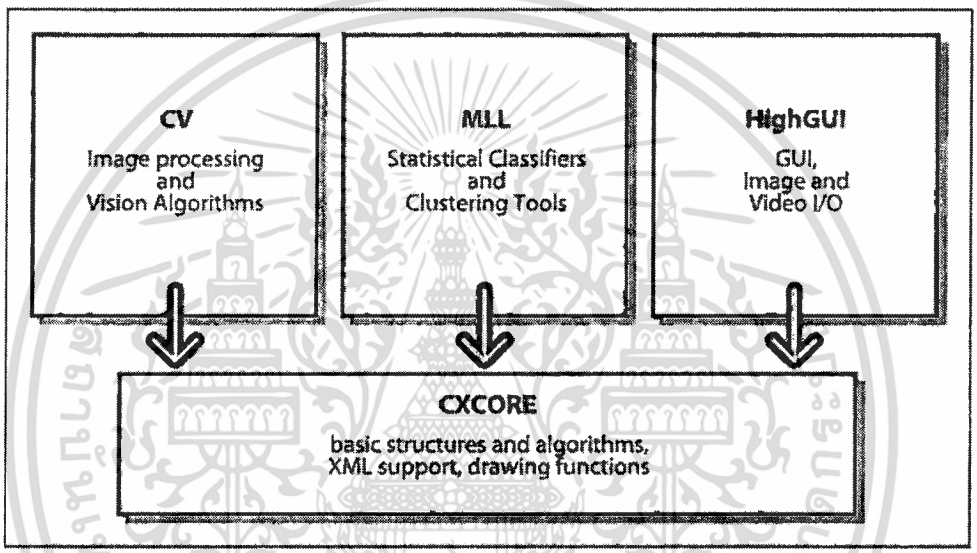
2.2 OpenCV

OpenCV ย่อมาจาก Open Source Computer Vision ซึ่ง OpenCV เป็นไลบรารี่แบบ ครอสแพลตฟอร์ม (Cross-platform) โดยสามารถรันบนระบบปฏิบัติการลินุกซ์ (Linux), วินโดวส์ (Windows) และแมคโอเอสเอ็ก (Mac OS X) OpenCV นั้นถูกออกแบบมา สำหรับการคำนวณที่มีประสิทธิภาพอย่างยิ่งสำหรับแอพพลิเคชั่นที่ทำงานแบบเรียลไทม์ OpenCV นี้ถูกเขียนขึ้นโดยภาษาซีที่ถูกออฟติไมซ์แล้วและสามารถใช้ข้อได้เปรียบจากโปรเซสเซอร์แบบมัลติคอร์

จุดประสงค์หนึ่งของ OpenCV ก็คือการทำให้โครงสร้างพื้นฐานของคอมพิวเตอร์วิชั่น นั้นง่ายต่อการใช้งาน ซึ่งช่วยให้ผู้ใช้สามารถสร้างวิชั่นแอพพลิเคชั่น (Vision application) ที่ค่อนข้างซับซ้อนได้อย่างรวดเร็ว OpenCV ไลบรารี่นั้นประกอบไปด้วยฟังก์ชันมากกว่า 500 ฟังก์ชัน ซึ่งครอบคลุมหลายๆหัวข้อของวิชั่น รวมถึงการตรวจสอบสินค้าจากโรงงาน การประมวลผลภาพในทางการแพทย์ ระบบรักษาความปลอดภัย ส่วนติดต่อกับผู้ใช้ การปรับเทียบกล้องสเตอริโอวิชั่น และการใช้งานกับหุ่นยนต์ และเนื่องจากคอมพิวเตอร์วิชั่น และแมตชีนเลินนิ่ง (Machine Learning) นั้นมักจะต้องทางานไปด้วยกันบ่อยครั้ง OpenCV จึงมีแมตชีนเลินนิ่งไลบรารี่ รวมอยู่ด้วย ซึ่งไลบรารี่ย่อยนี้ เน้นไปที่การทำแพทเทินเรคคอกนิชั่น (Pattern recognition) แบบสถิติ และการทำคลัสเตอริ่ง (Clustering) ซึ่งแมตชีนเลินนิ่งไลบรารี่นั้นสามารถใช้ได้เป็นอย่างดีกับงานที่เกี่ยวกับวิชั่น ซึ่งสิ่งนี้คือหัวใจหลักของ OpenCV และโดยทั่วไปแล้วเพียงพอที่จะใช้ในปัญหาของแมตชีนเลินนิ่งต่างๆได้

2.2.1 โครงสร้างของ OpenCV และองค์ประกอบ

OpenCV นั้นมีโครงสร้างอยู่ห้าส่วนประกอบหลัก สี่ในห้าแสดงดังรูปต่อไปนี้โดยส่วนซีวี (CV) ประกอบไปด้วยการประมวลผลภาพพื้นฐาน และคอมพิวเตอร์วิชั่นอัลกอริทึมระดับสูงกว่า และแอ็มแอล (ML) คือแมตชีนเลินนิ่งไลบรารี่ซึ่งรวมถึงสแตทิสทิคอลคลาสสิไฟลเออร์ (Statistical classifiers) และคลัสเตอริ่งทูลในส่วนของไฮจียูไอ (HighGUI) จะประกอบไปด้วยการทางานกับอินพุต เอาท์พุทและฟังก์ชันสาหรับการจัดเก็บและโหลดวีดีโอและภาพและ ซีเอ็กคอร์ (CXCore) ประกอบไปด้วยโครงสร้างข้อมูล (Data structures) และคอนเท็น (Content) พื้นฐาน ภาพที่ 2.4 แสดงโครงสร้างของ OpenCV



ภาพที่ 2.4 โครงสร้างของ OpenCV

2.2.2 ระบบปฏิบัติการและสถาปัตยกรรมที่รองรับ

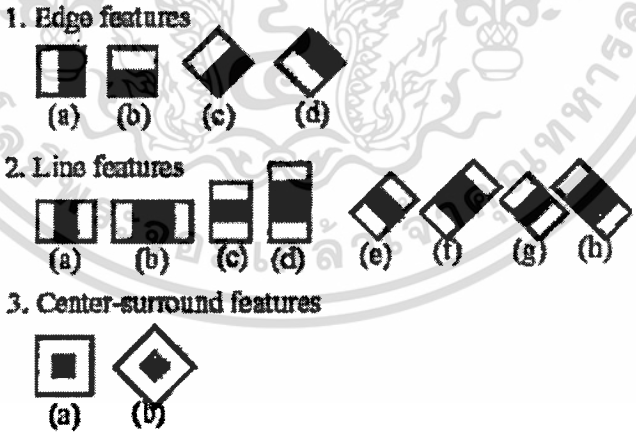
OpenCV นั้นถูกออกแบบมาให้สามารถใช้งานได้ง่าย เดิมทีนั้น OpenCV ถูกเขียนขึ้นและคอมไพล์ผ่าน Borland C++, Microsoft Visual C++ และอินเทลคอมไพล์เลอร์ซึ่งการใช้ภาษาC และC++ ในการโค้ดเป็นมาตรฐานทาให้ OpenCV สามารถสนับสนุนการครอสแพลตฟอร์มได้ง่ายขึ้น รูปต่อไปนี้แสดงถึงแฟลตฟอร์มที่ OpenCV สามารถทำงานได้ เริ่มจากระบบ IA32 บนวินโดวส์ตามด้วยลินุกซ์ บนสถาปัตยกรรมเดียวกัน บนแมคโอเอสเอ็ก นั้นมีความสำคัญตั้งแต่แอปเปิล ได้เริ่มใช้โปรเซสเซอร์ของอินเทลตามด้วยสถาปัตยกรรมแบบ 64 บิต และบนซันฮาร์ดแวร์และบนระบบปฏิบัติการอื่นๆ โดยภาพที่ 2.5 แสดงถึงระบบปฏิบัติการและสถาปัตยกรรมที่ OpenCV สนับสนุน

	IA32	EM64T	IA64	Other (PPC, Sparc)
Windows	✓ (w. IPP; MSVC6, .NET2005+OMP, ICC, GCC, BCC)	✓ (w. IPP; MSVC6+PSDK.NET2005+OMP, PSDK)	± (w. IPP; PSDK, some tests fail)	N/A
Linux	✓ (w. IPP; GCC, BCC)	✓ (w. IPP; GCC, BCC)	✓ (GCC, ICC)	✗
MacOSX	✓ (w. IPP, GCC, native APIs)	? (not tested)	N/A	✓ (iMac G5, GCC, native APIs)
Others (BSD, Solaris...)	✗	✗	✗	Reported to build on UltraSparc Solaris

ภาพที่ 2.5 ระบบปฏิบัติการและสถาปัตยกรรมที่ OpenCV สนับสนุน

2.3 Haar like-feature

Haar like-Features เป็นวิธีการตรวจจับและตีความวัตถุภายในภาพ โดยใช้การสร้างรูปเหลี่ยม (Feature) โดยที่ภาพนี้แสดงถึงผลต่างระหว่างพื้นที่ส่วนสีขาว และส่วนที่เป็นสีดำ ซึ่งรูปเหลี่ยมที่สร้างขึ้นสามารถเปลี่ยนแปลงขนาด และตำแหน่งได้ ใช้สำหรับการตรวจจับลักษณะบนภาพแบบต่าง เช่น เส้นตรง, วงกลม เป็นต้น ภาพที่ 2.6 แสดงรูปแบบของรูปเหลี่ยมสำหรับการตรวจจับลักษณะแบบต่างๆ

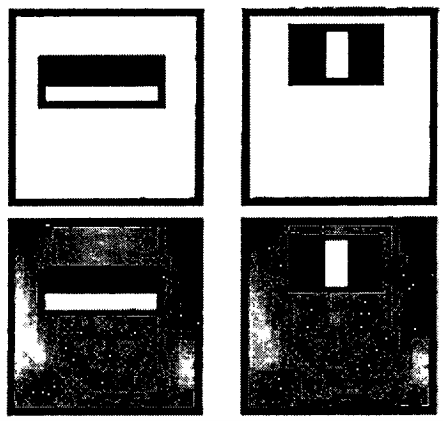


ภาพที่ 2.6 รูปแบบของรูปเหลี่ยมสำหรับการตรวจจับลักษณะแบบต่างๆ

(1) ความสามารถของขอบ

(2) ความสามารถของเส้น

(3) ความสามารถของบริเวณที่ล้อมรอบจุดตรงกลาง



ภาพที่ 2.7 ตัวอย่างการใช้รูปเหลี่ยมตรวจจับลักษณะต่างๆ

การคำนวณค่าของรูปเหลี่ยม (feature) นั้น ใช้หลักการคำนวณแบบ Integral image คือ ผลรวมของค่าในทุกๆ จุดภาพ ที่ตำแหน่ง (x, y) ใดๆ ดังสมการที่ 2.11 ภาพที่ 2.8 แสดงการคำนวณแบบ Integral image

$$P(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \qquad (2.11)$$

ภาพที่ 2.8 การคำนวณแบบ Integral image

ในการทำ Haar like-feature นั้น จำเป็นต้องมีภาพตัวอย่างจำนวนมาก ซึ่งใช้ในการคัดเลือกลักษณะของรูปต้องการตรวจจับและตีความหมาย ซึ่งมีสองลักษณะคือ Positive Image หรือรูปมีวัตถุนั้นๆประกอบอยู่ภายในภาพ และ Negative Image หรือภาพใดๆที่ไม่มีวัตถุที่เราต้องการอยู่ภายในภาพ จากนั้นใช้ขั้นตอนวิธีของ AdaBoost (Adaptive Boost) ซึ่งเป็นกระบวนการหารูปเหลี่ยมที่มีลักษณะใกล้เคียง และแตกต่างกับภาพนำเข้า สำหรับการจัดประเภทของภาพ โดยการถ่วง

น้ำหนักให้ส่วนต่างๆภายในภาพ บนภาพ Positive และภาพ Negative เพื่อใช้หาลักษณะของวัตถุที่ "ใช่" และ "ไม่ใช่" ในลักษณะต่างๆ

2.4 Region of interest (ROI)

Region of interest (ROI) คือบริเวณที่เราสนใจซึ่งอาจจะเป็นบริเวณใดภายในภาพก็ได้ โดยการตีกรอบล้อมรอบบริเวณที่สนใจ ด้วยวงกลม กรอบสี่เหลี่ยม หรือกรอบรูปเหลี่ยมใดๆ เพื่อนำภาพเฉพาะส่วนดังกล่าวมาประมวลผลหรือเปลี่ยนแปลงภาพตามต้องการ โดยไม่มีผลกระทบกับส่วนอื่นๆซึ่งภายในหนึ่งภาพ สามารถกำหนดได้หลายๆ บริเวณที่สนใจ การตัดเฉพาะส่วนของภาพเพื่อนำมาประมวลผลทำให้การประมวลผลภาพเร็วขึ้นเนื่องจาก Resolution ของภาพที่นำมาประมวลผลมีขนาดเล็กลง ซึ่งในการประมวลผลภาพขนาดใหญ่จะทำให้เห็นความแตกต่างของประสิทธิภาพในการประมวลผลมาก ภาพที่ 2.9 แสดงตัวอย่างการกำหนดบริเวณที่สนใจ

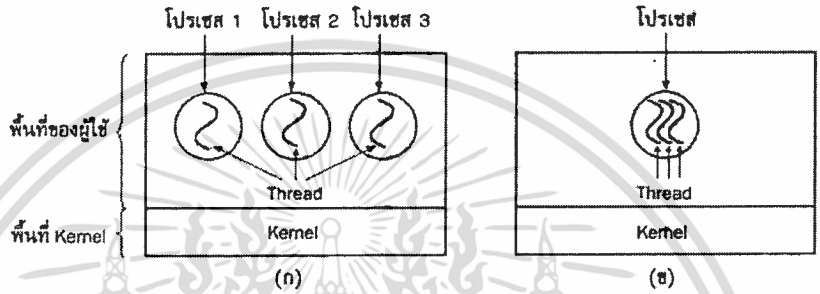
ภาพที่ 2.9 ตัวอย่างการกำหนดบริเวณที่สนใจ

2.5 Circle Detection

Circle Detection คือการตรวจจับวงกลมในรูปภาพเพื่อใช้ในการตรวจหาตาดำ โดยจะใช้ภาพ Grey Scale เพื่อใช้ในการตรวจจับวงกลมโดยการตรวจจับวงกลมจะดูความลึกของสีซึ่งลักษณะของนัยน์ตาอาจมีสีดำหรือสีฟ้าซึ่งเมื่อแปลงเป็น Grey Scale แล้วจะทำให้มีระดับความลึกของสีอยู่ในระดับที่แตกต่างกันกับของตาขาวทำให้สามารถใช้การตรวจหาวงกลมเพื่อหานัยน์ตาได้ โดย EmguCV มี Library ที่สำเร็จรูปในการตรวจหาวงกลมซึ่งสามารถกำหนด Parameter 4 ค่าคือ ค่า Threshold ที่ใช้ในการทำ Canny Edge Detection, ค่า Threshold ที่ใช้ในการทำ Accumulator, ค่าอัตราส่วนของภาพที่จะทำการตรวจหาวงกลม, ค่าระยะของรัศมีในการตรวจหามากสุดและน้อยสุดของวงกลม, และค่าระยะห่างของวงกลมที่ตรวจสอบพบกับวงกลมถัดไป ซึ่งในการตรวจสอบนัยน์ตาจะต้องกำหนดค่าของรัศมีวงกลมให้พอเหมาะจึงจะสามารถตรวจหานัยน์ตาได้อย่างแม่นยำ

2.6 Multithread Processing

Process ดั้งเดิมที่เรียกว่า "Heavy Weight" ที่มีการควบคุมเพียง 1 Thread หมายถึงภายใน 1 Process จะควบคุมงานเพียง 1 งานเท่านั้น แต่ในระบบปฏิบัติการสมัยใหม่ในแต่ละ Processสามารถมีได้หลายThread อาจจะกล่าวได้ว่า Thread คือส่วนประกอบย่อยของ Process นั่นเอง ซึ่งอาจเรียก Thread ว่า "Lightweight Process" ในภาพที่ 2.10 (ก) แสดงโปรเซส 3 Process แต่ละ Process จะมี Address เป็นของตนเอง และควบคุมเพียง 1 งานเท่านั้น แต่ละภาพที่ 2.10 (ข) จะเห็นว่าในแต่ละ Process จะควบคุม 3 Thread โดยใช้ Address เดียวกันอยู่

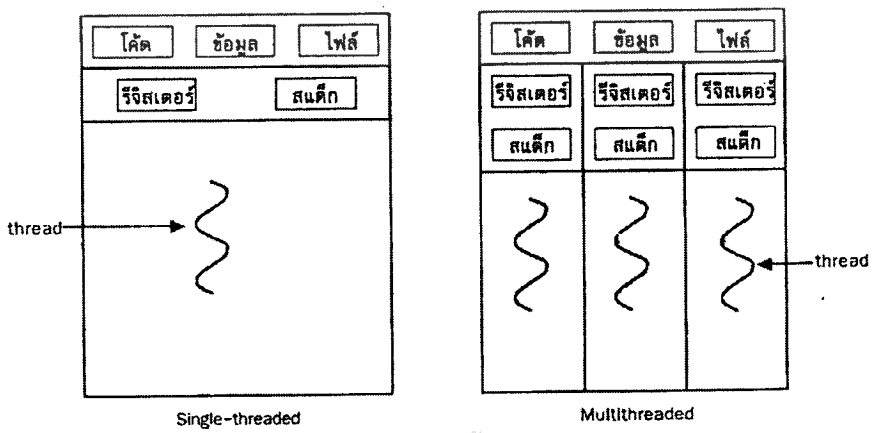


ภาพที่ 2.10 ตัวอย่างการทำงานแบบ Multithread

(ก) การทำงาน 3 Process

(ข) การทำงาน 1 Process

Thread เป็นหน่วยพื้นฐานของการจัดสรรการใช้ประโยชน์ของของซีพียู Process จะประกอบด้วย Thread จะมีการแชร์โค้ด, ข้อมูล และ Resource เช่น ไฟล์, อุปกรณ์ต่างๆเป็นต้น แต่ถ้า Process มีหลาย Thread หรือ Multithread จะทำงานได้หลายงานในเวลาเดียวกัน Process ที่เป็น Single-Threaded และ Multithread ภาพที่ 2.11 แสดงโครงสร้างของ Single-Threaded และ Multithreaded



ภาพที่ 2.11 โครงสร้างของ Single-Threaded และ Multithreaded

Software ปัจจุบันที่ทำงานกับระบบปฏิบัติการสมัยใหม่มีการออกแบบให้รองรับ Multithread โดยแยกออกเป็น Process ที่ควบคุมหลายๆ Thread เช่น Web browser ที่มี Thread หนึ่งในการแสดงรูปภาพหรือเขียนข้อความในขณะที่อีก Thread หนึ่งกำลังดึงข้อมูลจาก Network หรืออย่างในโปรแกรมWord Processing ที่มีหลาย Thread โดยที่ Thread หนึ่งกำลังแสดงภาพกราฟฟิก ในขณะที่ Thread ที่สองกำลังรอรับคำสั่งจากคีย์บอร์ดจากผู้ใช้ ในขณะที่ Threadที่สามกำลังตรวจสอบคำสะกดและไวยกรณ์ในลักษณะ Background เป็นต้น

การที่ระบบปฏิบัติการสนับสนุนระบบ Multithread ทำให้มีข้อได้เปรียบ 4 ข้อหลักๆ ดังนี้

1) การตอบสนองในระบบ Multithread ที่โต้ตอบ Application จะยินยอมให้ Program ยังคงดำเนินต่อไปถึงแม้ว่าจะมีบางส่วนรอการทำงานอยู่ เช่น มีการปฏิบัติที่ยาวนานเนื่องจากการเพิ่มการ โต้ตอบกับผู้ใช้นั่นเอง ยกตัวอย่างเช่น โปรแกรมเว็บบราวเซอร์ยังคงโต้ตอบกับผู้ใช้ได้ ในขณะที่มีหนึ่งที่กำลังโหลดรูปภาพอยู่
2) การแชร์ทรัพยากรของ Multithread จะแชร์หน่วยความจำ และทรัพยากรของ Process อยู่แล้ว ข้อได้เปรียบ ของการแชร์ Code จะทำให้ Application สามารถมีกิจกรรมของ Thread ได้หลายๆกิจกรรมภายใน Address เดียวกัน
3) ความประหยัดของ Multithread การจัดสรรหน่วยความจำและทรัพยากรสำหรับการสร้าง Process มีค่าใช้จ่ายมาก ในทางตรงข้าม เนื่องจาก Thread แชร์ทรัพยากรของ Process ที่มันอาศัยอยู่แล้ว ทำให้เกิดการประหยัดในการสร้าง Thread และทำการ Context switching ของ Thread เป็นการยากที่จะวัดความแตกต่างระหว่างการสร้างและดงสภาพ Process ที่มีมากกว่าThread ได้ แต่โดยปกติแล้วจะใช้เวลาในการสร้างและดงสภาพ Process สูงกว่าเวลาที่ใช้กับ Thread ใน Solaris2 การสร้าง Process จะช้ากว่าการสร้าง thread 30 เท่าและ Context switching Process จะช้ากว่า Thread 5 เท่า
4) การเอื้อประโยชน์ของสถาปัตยกรรม Multiprocessing ช่วยเสริมสถาปัตยกรรม Multiprocessing ให้สูงขึ้น ในขณะที่แต่ละ Thread สามารถประมวลผลขนานกันไปใน Process อื่นได้ใน Process ที่มี Thread เดียวสามารถประมวลผลเพียงซีพียูเดียวเท่านั้น ไม่ว่าจะมีกี่ซีพียูก็ตามระบบ Multithread ในเครื่องที่มีหลายซีพียูจะเพิ่มประสิทธิภาพในการทำงานพร้อม ๆ กันได้มากขึ้น ในสถาปัตยกรรมที่มีซีพียูเดียว ซีพียูจะย้ายแต่ละ Thread ให้เร็วมากขึ้นเพื่อให้ทำงานเสมือนว่าขนานกันอยู่ แต่ในความเป็นจริงจะรันเพียง thread เดียวเท่านั้นในเวลานั้น ๆ

2.7 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP, ทีซีพี) เป็นหนึ่งในโพรโทคอลหลักในเครือข่ายอินเทอร์เน็ต หน้าที่หลักของทีซีพี คือ ควบคุมการรับส่งข้อมูลระหว่าง Host ถึง Host ในเครือข่ายเพื่อใช้แลกเปลี่ยนข้อมูลระหว่างกัน โดยตัวโพรโทคอลจะรับประกันความถูกต้อง และลำดับของข้อมูลที่ส่งผ่านระบบเครือข่าย นอกจากนั้นทีซีพียังช่วยจำแนกข้อมูลให้ส่งผ่านไปยังแอปพลิเคชัน ที่ทำงานอยู่บนโฮสเดียวกันให้ถูกต้องด้วย

งานหลักที่สำคัญของทีซีพีอีกงานหนึ่งคือ เป็นโพรโทคอลที่ขั้นกลางระหว่างแอปพลิเคชันและเครือข่ายไอพี ทำให้แอปพลิเคชันจากโฮสหนึ่ง สามารถส่งข้อมูลออกยังอีกโฮสหนึ่งผ่านเครือข่ายเปรียบเสมือนมีท่อส่งข้อมูลระหว่างกัน

TCP มีลักษณะดังต่อไปนี้

2.7.1 Connection-Oriented

ก่อนส่งข้อมูล จะเกิดสองกระบวนการนี้ใน Application Layer ต้องมีการติดต่อก่อนเชื่อมต่อ การใช้กระบวนการการสร้างการเชื่อมต่อ TCP และการปิดการเชื่อมต่ออย่างเป็นทางการ สำหรับคำแนะนำ TCP มากกว่าเกี่ยวกับกระบวนการการเชื่อมต่อ TCP ดูในบทที่11 โปรโตคอลหน่วยควบคุม “Transmission (TCP) Connections”

2.7.2 การส่งแบบสองทาง (Full duplex)

สำหรับ TCP แต่ละชั้น การเชื่อมต่อ TCP ประกอบด้วยการเชื่อมต่อทางลอจิก ข้อมูลปลายทางและข้อมูลที่เข้ามา จะใช้เทคโนโลยีที่ส่งข้อมูลได้สองทางพร้อมกัน ก็คือข้อมูลสามารถไหลออกมาจากปลายทางและเข้าไปสายส่งข้อมูลที่เข้ามาพร้อมกัน TCP header บรรจุหมายเลขลำดับข้อมูลต้นทางและปลายทางและ acknowledgment ของข้อมูลที่เข้ามา

2.7.3 ความน่าเชื่อถือ(Reliable)

ข้อมูลที่เชื่อมต่อใน TCP จะต้องถูกตามลำดับและมีสัญญาณ positive acknowledgment จากเครื่องรับ ถ้าไม่ใช่ positive acknowledgment TCP ก็จะทำการส่งข้อมูลอีกครั้ง ที่เครื่องรับ จะทำสำเนาส่วนถูกทิ้งและที่มาถึงออกมาจากข้อมูลตามลำดับและเก็บข้อมูลเรียงลำดับไห้ถูกต้อง และมี TCP checksum ใช้ในการตรวจสอบความถูกต้องของข้อมูล

2.7.4 ความต่อเนื่องของข้อมูล(Byte stream)

ข้อมูลที่ส่งออก และรับเข้ามาผ่านสายส่งสัญญาณมีความต่อเนื่องของข้อมูลแต่ละ Byte มาก หมายเลขลำดับและหมายเลข Acknowledgment ในTCP header ที่ถูกกำหนดในรอยต่อของแต่ละ byte

TCP ไม่มีบันทึกหรือข้อความบอกความต่อเนื่องของข้อมูล โปรโตคอล Application Layer ต้องแยกตามความเหมาะสมของข้อมูลขาเข้าแต่ละ byte

2.7.5 ผู้ส่ง-และผู้รับ (Sender- and receiver-side flow control)

เพื่อหลีกเลี่ยงการส่งข้อมูลที่เยอะจนทำให้เกิดการแออัดภายใน เราเตอร์(Router) ของเครือข่าย TCP ดำเนินการส่งควบคุมการไหลของข้อมูลที่ค่อยทำการส่งข้อมูลในแต่ละครั้ง เพื่อหลีกเลี่ยงผู้ส่งส่งข้อมูลที่ไม่ได้รับ บัฟเฟอร์ TCP จะทำกระบวนการ flow control ทางผู้รับแสดงจำนวนไบต์ที่รับจะสามารถรับข้อมูลได้

2.7.6 การแบ่งส่วนของแอพลิเคชั่นเลเยอร์ดาต้า (Segmentation of Application Layer data)

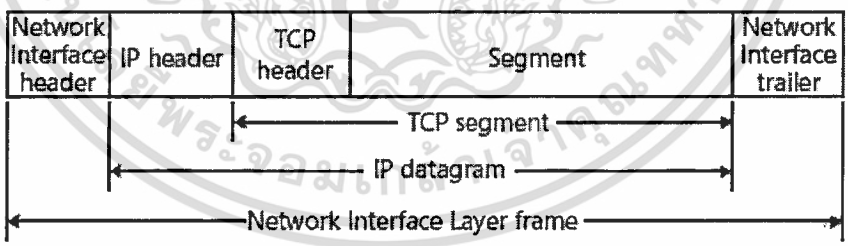
TCP มีการแบ่งข้อมูลที่ได้รับจากชั้น Application คือการแบ่งข้อมูลแต่ละ layer ให้เหมาะสมจนถึงการส่ง IP datagram ภายในเครือข่าย ภายในชั้น TCP แสดงถึงขนาดที่มากที่สุดของแต่ละ segment และในการรับข้อมูลและยังมีการปรับขนาดที่ดีที่สุดเพื่อจะได้ค้นพบเส้นทางที่เหมาะสมในการส่ง

2.7.7 การส่งแบบหนึ่งต่อหนึ่ง (One-to-one delivery)

TCP มีการเชื่อมต่อทางลอจิกแบบจุดต่อจุดระหว่าง โปรโตคอลในชั้น applications layer TCP จะไม่มีการจัดส่งข้อมูลแบบ จุด ต่อ หลายจุด TCP ใช้เมื่อ Applications layer ต้องการส่งข้อมูลที่มีความเชื่อถือได้

2.7.8 การแบ่งย่อยของข้อมูล (The TCP Segment)

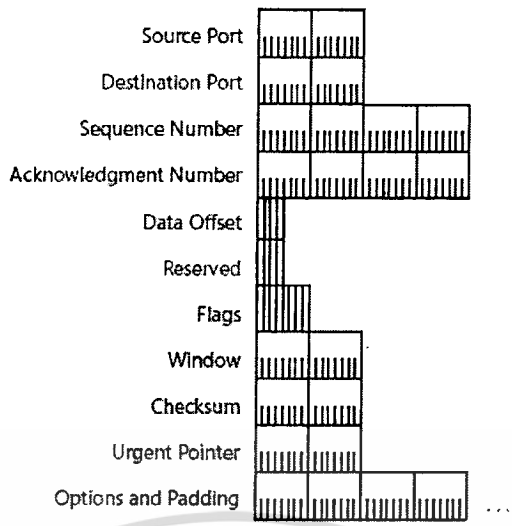
ประกอบด้วย TCP header และการเลือก payload แต่ละ segment ซึ่งมีการระบุไว้ใน IPV6 TCP Segment สามารถส่งข้อมูลได้มากที่สุด 65,495 bytes IP headerขนาดเล็กที่สุด 20 bytes ซึ่งส่งผลให้เกิดการ Encapsulated ที่เหมาะสม ในชั้น Network Interface ส่วนหัวและส่วนท้ายข้องเฟรม แสดงในรูปภาพ ภาพที่ 2.12 แสดงส่วนประกอบของ TCP header



ภาพที่ 2.12 ส่วนประกอบของ TCP header

2.7.9 The TCP Header

ขนาดของ TCP header ประกอบด้วย ส่วนต่างๆที่แสดง ในภาพที่ 2.13 เมื่อ TCP ปัจจุบันของ TCP header มีความยาว 20 ไบท์ ภาพที่ 2.13 แสดงโครงสร้างของ TCP header



ภาพที่ 2.13 โครงสร้างของ TCP header

2.7.9.1 Source Port

ขนาด 2 byte เป็นส่วนที่บอกถึงพอร์ต address ของผู้ส่งซึ่งมีไม่ซ้ำกัน

2.7.9.2 Destination Port

ขนาด 2 byte เป็นส่วนที่ โพรโตคอลชั้น application แสดงถึง ปลายทางในการส่งข้อมูล ซึ่งรวม IP address ใน IP header และ destination port ใน TCP header ซึ่งจะบอกถึงต้นทาง *socket*—a ซึ่งเป็นตำแหน่งที่ส่งจะมีเพียงแอดเดรสเดียวในโลก

2.7.9.3 Sequence Number

4 byte ระบุถึงลำดับที่ทำการส่งไบต์แรกของ segment Sequence Number จะถูต้องค่าไว้เสมอ แม้แต่เวลาไม่มีการส่งข้อมูลใน Segment ในกรณีนี้ Sequence Number จะถูกตั้งไว้ในการส่งข้อมูล byte ต่อไป เมื่อมีการเชื่อมต่อ TCP segments และ Synchronization ค่าแฟลกที่ 1 จะถูกเซทเป็นค่าเริ่มต้นของ Sequence Number ซึ่งแสดงว่าเป็น byte แรกในการส่งข้อมูล

2.7.9.4 Acknowledgment Number

ขนาด 4-byte ซึ่งระบุหมายเลขลำดับของหมายเลขต่อไปที่จะทำการส่งไปให้ผู้รับ หมายเลข acknowledgment จะแจ้ง positive acknowledgment กลับมา แต่ไม่รวมถึงตัวเลข acknowledgment ที่ถูกรับ ตัวเลขacknowledgment มีความสำคัญใน TCP segments ทั้งหมดขึ้นอยู่กับการตั้งค่า flagของacknowledgment

2.7.9.5 Data Offset

ขนาด 4 bit เป็นพื้นที่บอกถึงจุดเริ่มต้นของ TCP แต่ละ segment และบอกถึงขนาดของ TCP Headerและ IP header ว่ามี Header length, Data Offset มีขนาด 32 บิต สำหรับ TCP Header ที่มีขนาดเล็ก (no options) Data Offset เซตไว้ที่ 5 (0x5), เป็นการชี้บอกข้อมูลเริ่มต้น

ใน byte ที่ 20 จากการเริ่มต้นของ TCP Segment กับส่วนของ Data Offset ตั้งค่ามากที่สุดไว้เป็น 15 (0xF), มากกว่า TCP Header รวมถึงในออฟชั่นต่าง ในTCP สามารถยาวได้ 60 ไบต์

2.7.9.6 Reserved

ขนาด 4 bit สงวนไว้ใช้สำหรับอนาคต ผู้ใช้ตั้งค่าบิตนี้ไว้ เป็น 0

2.7.9.7 Flags

ขนาด 8 บิต เป็นพื้นที่บอกถึงแฟลก ทั้ง8 ใน TCP ที่นิยามใน RFCs 793 และ 3168 ในแฟลกทั้ง 8 เรียกว่า CWR (Congestion Window Reduced), ECE (Explicit Congestion Notification [ECN]-Echo), URG (Urgent), ACK, PSH (Push), RST (Reset),SYN and FIN (Finish) ซึ่งจะกล่าวถึงในรายละเอียดในเรื่อง TCP flag

2.7.9.8 Window

ขนาด 2 byte เป็นส่วนที่บอกจำนวนไบต์ที่ส่งเข้ามาในแต่ละพื้นที่โดยการดูwindow size ทุก segment ผู้รับบอกผู้ส่งถึงจำนวนข้อมูลที่สามารถส่งได้และข้อมูลที่รับเสร็จแล้ว ผู้ส่งจะไม่ส่งข้อมูลที่มากกว่าผู้รับสามารถรับข้อมูลได้ ถ้าผู้รับไม่สามารถรับข้อมูลได้มากกว่านี้ก็จะบอกผู้ส่งว่า Window size มีขนาดเท่ากับ 0 ผู้ส่งจะไม่สามารถส่งข้อมูลได้จนกว่าขนาดของ Window size จะมีค่าไม่เท่ากับ 0 byte เครื่องมือที่สำคัญของการบอกว่า window size เป็น0 ก็คือ flow control

2.7.9.9 Checksum

ขนาด 2 byte เป็นบิตที่ให้บริการตรวจสอบข้อผิดพลาดของแต่ละเซกเม็นต์ (TCP header and segment) ค่าของ Checksum ถูกคำนวณเหมือนกับ IP header checksum ทั้งหมดมีขนาด 16 บิต ใน TCP pseudo header, TCP header, TCP segment และถ้าต้องการ padding เพิ่มจะได้ค่าเป็น 0x00 ไบต์ padding ใช้เฉพาะความยาวของ segment ที่มีย่อยมาก ค่าของ Checksum จะถูกตั้งไว้เป็น 0 ขณะที่มีการคำนวณ

2.7.9.10 Urgent Pointer

ขนาด 2 byte เป็นพื้นที่บอกว่าข้อมูลเร่งด่วนที่จะทำการส่งข้อมูล

2.7.9.11 Options

หรือเรียกอีกอย่างว่า TCP Options สามารถเพิ่ม TCP Header แต่ไม่เกิน 4 byte เพื่อบอกว่าขนาดของ TCP Header มีส่วนที่เป็น Data Offset

2.7.9.12 TCP Ports

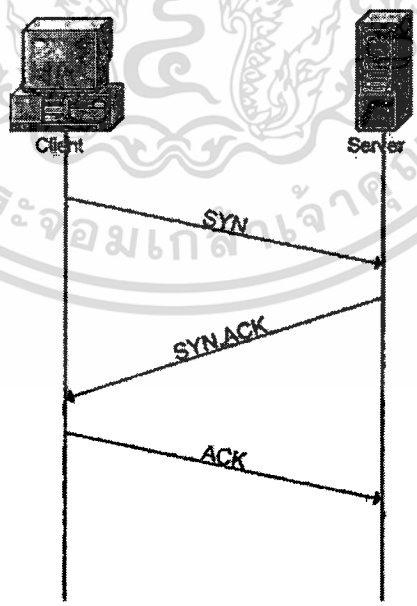
TCP port เป็นพอร์ตที่ให้บริการเชื่อมต่อในการส่งข้อมูล รวมถึงแต่ละพอร์ตของ TCP Segment คือพอร์ตที่ Application Layer ใช้ในการทำกระบวนการต่างจาก segment ที่ทำการส่ง และ พอร์ตเป้าหมายที่ Application Layerใช้ในกระบวนการส่งข้อมูล มีตัวเลขพอร์ตที่

ถูกกำหนดโดย Internet Assigned Numbers Authority (IANA) ถึงโพรโตคอลในชั้น Application Layer

2.7.10 3-Way hand shake

ก่อนที่จะเริ่มต้นการสื่อสาร จะต้องมีการส่งสัญญาณเพื่อบอกโฮสต์อีกฝั่งหนึ่งให้เตรียมตัวติดต่อ ซึ่งกระบวนการที่ใช้มีชื่อเรียกว่า 3-Way Hand Shake ดังแสดงในภาพที่ 2.14 มีขั้นตอนคือ

1) เครื่องไคลเอนต์จะทำการส่งเซกเมนต์ โดยเปิด SYN Flag ระบุหมายเลขพอร์ตที่ต้องการติดต่อบนเซิร์ฟเวอร์และระบุหมายเลข ลำดับของข้อมูล (ISN - Initial Sequence Number)
2) เครื่องเซิร์ฟเวอร์เมื่อได้รับข้อมูลเซกเมนต์จากข้อ 1 ก็จะตอบกลับด้วยการเพิ่มค่า ISN ที่ได้รับขึ้นอีก 1 พร้อมทั้งระบุหมายเลขลำดับ (ISN) ของตนเอง และเปิด SYN กับ ACK Flag
3) ไคล์เอนต์เมื่อได้รับการตอบกลับจากเซิร์ฟเวอร์ตามข้อ 2 ก็จะทำการตอบรับกลับไป โดยการเพิ่มค่า ISN ของเซิร์ฟเวอร์ขึ้นอีก 1 และเปิด ACK Flag เมื่อผ่านการสร้าง connection ทั้ง 3 ขั้นตอนแล้ว ตอนนี้ทั้งไคล์เอนต์ และเซิร์ฟเวอร์เปรียบเสมือนมีการเชื่อมต่อถึงกันแล้ว สถานะของการเชื่อมต่อในขณะนี้เรียกว่า Established



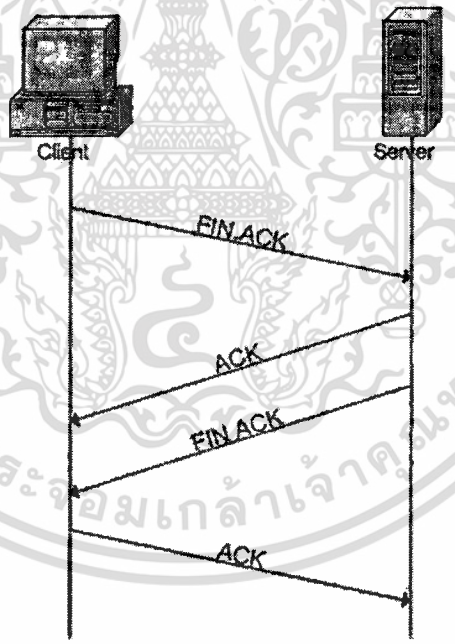
ภาพที่ 2.14 แบบจำลอง 3-Way hand shake

2.7.11 Connection Termination

เมื่อการสื่อสารของทั้งสองฝั่งจบลง และไม่ต้องการรับส่งข้อมูลอีกต่อไป จะต้องทำตามขั้นตอนการยุติการสื่อสารเพื่อให้การสื่อสารจบลงอย่างสมบูรณ์ ซึ่งมีอยู่ 4 ขั้นตอนคือ

1) ไคลเอนต์ทำการส่ง ISN พร้อมกับ FIN ACK Flag ไปยังเซิร์ฟเวอร์
2) เซิร์ฟเวอร์ทำการตอบรับ ISN และบวกค่า ISN อีก 1 พร้อม ACK Flag
3) เซิร์ฟเวอร์ทำการส่ง ISN พร้อมกับ FIN ACK Flag ไปยังไคลเอนต์
4) ไคลเอนต์ทำการตอบรับ ISN และบวกค่า ISN อีก 1 พร้อม ACK Flag

การยุติการเชื่อมต่อ โดยส่ง FIN ACK ออกไปมีความหมายคือ โฮสต์ที่ส่งไม่มีข้อมูลจะส่งไปอีก มิใช่ต้องการปิดการสื่อสารทั้งหมดในทันที ดังนั้นจึงต้องทำทั้งสองทาง การสื่อสารจึงจะยุติลงอย่างสมบูรณ์ ในการใช้งานจริง อาจมีการยุติการสื่อสารเพียงด้านเดียว คือหยุดส่งข้อมูล แต่ยังคงเปิดพอร์ตไว้รอรับข้อมูลจากอีกด้านหนึ่ง ทั้งนี้ขึ้นอยู่กับลักษณะการใช้งาน การปิดพอร์ตสื่อสารเพียงด้านเดียวเช่นนี้ เรียกว่า Half-Close ภาพที่ 2.15 แสดงแบบจำลองการปิดการสื่อสาร



ภาพที่ 2.15 แบบจำลองการปิดการสื่อสาร

2.7.12 Listening

ในการเขียนโปรแกรมเพื่อติดต่อแบบ Socket Programming แบบ TCP นั้น ในส่วนของ TCP Server จะต้องเปิด Port รอรับการเชื่อมต่อเพื่อสร้าง Connection กับ Client โดยการ

เปิด Port เพื่อรอรับการเชื่อมต่อนี้เรียกว่า Listening Port ซึ่ง Listening Port นี้จะต้องเป็นหมายเลข Port ที่ไม่เปลี่ยนแปลงและใช้เพื่อรอรับการเชื่อมต่อจาก Client เท่านั้น

2.7.13 Accept Socket

เมื่อมี Client ติดต่อเข้ามายัง Listening Port เพื่อขอสร้างการเชื่อมต่อใน TCP แล้ว ทาง Server จะทำการ Accept Socket โดยการสร้าง Socket ใหม่เพื่อมารองรับ Connection ที่จะสร้างขึ้นเพื่อเชื่อมต่อกับ Client โดย Accept Socket ที่สร้างขึ้นจะใช้หมายเลข Port ที่ Server generate ขึ้นมาให้

2.8 Microcontroller

Microcontroller คือ อุปกรณ์ควบคุมขนาดเล็ก ซึ่งบรรจุความสามารถที่คล้ายคลึงกับระบบคอมพิวเตอร์ โดยในไมโครคอนโทรลเลอร์ได้รวมเอาซีพียู, หน่วยความจำและพอร์ต ซึ่งเป็นส่วนประกอบหลักสำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกัน โดยทำการบรรจุเข้าไว้ในตัวถังเดียวกัน

โครงสร้างโดยทั่วไป ของไมโครคอนโทรลเลอร์นั้น สามารถแบ่งออกมาได้เป็น 5 ส่วนใหญ่ๆ ดังต่อไปนี้

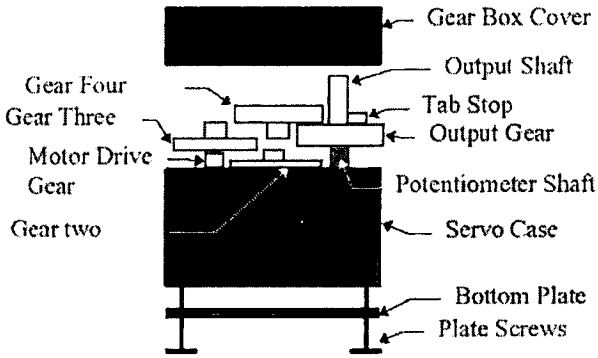
1) หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)
2) หน่วยความจำ (Memory) สามารถแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำที่มีไว้สำหรับเก็บโปรแกรมหลัก (Program Memory) เปรียบเสมือนฮาร์ดดิสก์ของเครื่องคอมพิวเตอร์ตั้งโต๊ะ คือข้อมูลใดๆ ที่ถูกเก็บไว้ในนี้จะไม่สูญหายไปแม้ไม่มีไฟเลี้ยง อีกส่วนหนึ่งคือหน่วยความจำข้อมูล (Data Memory) ใช้เป็นเหมือนกกระดาษทดในการคำนวณของซีพียู และเป็นที่พักข้อมูลชั่วคราวขณะทำงาน แต่หากไม่มีไฟเลี้ยง ข้อมูลก็จะหายไปคล้ายกับหน่วยความแรม (RAM) ในเครื่องคอมพิวเตอร์ทั่วๆ ไป แต่สำหรับไมโครคอนโทรลเลอร์สมัยใหม่ หน่วยความจำข้อมูลจะมีทั้งที่เป็นหน่วยความจำแรม ซึ่งข้อมูลจะหายไปเมื่อไม่มีไฟเลี้ยง และเป็นอีอีพรอม (EEPROM : Erasable Electrically Read-Only Memory) ซึ่งสามารถเก็บข้อมูลได้แม้ไม่มีไฟเลี้ยง
3) ส่วนติดต่อกับอุปกรณ์ภายนอก หรือพอร์ต (Port) มี 2 ลักษณะคือ พอร์ตอินพุต (Input Port) และพอร์ตส่งสัญญาณหรือพอร์ตเอาต์พุต (Output Port) ส่วนนี้จะใช้ในการเชื่อมต่อกับอุปกรณ์ภายนอก ถือว่าเป็นส่วนที่สำคัญมาก ใช้ร่วมกันระหว่างพอร์ตอินพุตเพื่อรับสัญญาณ อาจจะด้วยการกดสวิตช์ เพื่อนำไปประมวลผลและส่งไปพอร์ตเอาต์พุตเพื่อแสดงผลเช่น การติดสว่างของหลอดไฟ เป็นต้น

4) ช่องทางเดินของสัญญาณ หรือบัส (BUS) คือเส้นทางการแลกเปลี่ยนสัญญาณข้อมูลระหว่าง ซีพียู หน่วยความจำและพอร์ต เป็นลักษณะของสายสัญญาณ จำนวนมากอยู่ภายในตัวไมโครคอนโทรลเลอร์ โดยแบ่งเป็นบัสข้อมูล (Data Bus), บัสแอดเดรส (Address Bus) และบัสควบคุม (Control Bus)

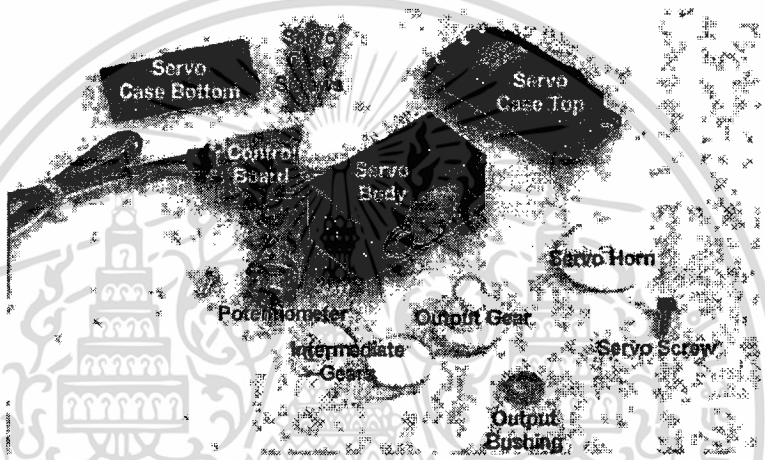
5) วงจรกำเนิดสัญญาณนาฬิกา นับเป็นส่วนประกอบที่สำคัญมากอีกส่วนหนึ่ง เนื่องจากการทำงานที่เกิดขึ้นในตัวไมโครคอนโทรลเลอร์ จะขึ้นอยู่กับการกำหนดจังหวะ หากสัญญาณนาฬิกามีความถี่สูง จังหวะการทำงานก็จะสามารถทำได้ถี่ขึ้นส่งผลให้ไมโครคอนโทรลเลอร์ตัวนั้น มีความเร็วในการประมวลผลสูงตามไปด้วย

2.9 Servo Motor

Servo motor คือ มอเตอร์ไฟฟ้ากระแสตรง (DC motor) ที่ถูกประกอบรวมกับชุดเกียร์และส่วนควบคุมต่างๆไว้ในโมดูลเดียวกัน หรือภายในกล่องพลาสติกเดียวกัน โดยมอเตอร์ชนิดนี้จะมีสายต่อใช้งานงานเพียง 3 เส้นเท่านั้น คือ VCC, GND และ สายสัญญาณควบคุม(Control Line) ซึ่งสามารถควบคุมให้มอเตอร์หมุนซ้าย หรือ ขวาได้จากสายสัญญาณเพียงเส้นเดียว โดยสัญญาณที่ใช้ควบคุมนี้จะเป็นสัญญาณ พัลส์วิดมอด (PWM) แบบ TTL Level ระดับแรงดันที่จ่ายให้มอเตอร์นี้จะอยู่ในช่วงประมาณ 4 ถึง 6 โวลต์ ขึ้นอยู่กับคุณสมบัติของมอเตอร์แต่ละตัว ข้อดีของมอเตอร์ชนิดนี้ก็คือ จะมีขนาดเล็กน้ำหนักเบา, ให้แรงบิดสูง, กินพลังงานน้อยและสามารถควบคุม ด้วยแรงดันลอจิกที่เป็น TTL ได้โดยตรงไม่จำเป็นต้องต่อวงจรขับ(Driver) อื่นๆ เพราะ มอเตอร์ชนิดนี้จะมีวงจรควบคุมบรรจุไว้ภายในอยู่แล้ว ซึ่งมอเตอร์ชนิดนี้สามารถควบคุมให้หมุนไปในตำแหน่ง หรือ ทิศทางองศาที่ต้องการได้ โดยอาศัยสัญญาณความกว้างพัลส์ ที่ป้อนให้มอเตอร์ แต่เซอร์โวมอเตอร์นี้จะหมุนได้แค่เพียงในช่วงประมาณ 180° หรือ ครึ่งรอบเท่านั้น หรือ บางรุ่นอาจหมุนได้ถึง 210° แต่จะไม่สามารถหมุนเป็นวงรอบได้เนื่องจากโครงสร้างภายในจะประกอบด้วย ตัวต้านทานชนิดปรับค่าได้ (VR) ที่ทำหน้าที่ตรวจสอบตำแหน่งการหมุนของมอเตอร์ และ ตัวต้านทานนี้จะถูกยึดติดกับแกนหมุนของมอเตอร์ ซึ่งจากการที่ตัวต้านทานปรับค่านี้ไม่สามารถหมุนเป็นวงรอบได้ ดังนั้น เซอร์โวมอเตอร์จึงถูกออกแบบให้หมุนได้เพียงแค่ประมาณ 180 องศา หรือ ครึ่งรอบเท่านั้น เพื่อป้องกันความเสียหายที่จะเกิดกับตัวต้านทานปรับค่าได้ แต่ถ้าหากเราต้องการให้มอเตอร์หมุนเป็นวงรอบ (360°) นั้นก็สามารถทำได้ โดยจะต้องทำการปรับแต่ง (Modify) ดัดแปลงชิ้นส่วนบางอย่างของมอเตอร์ ภาพที่ 2.16 แสดงองค์ประกอบของ Servo Motor และภาพที่ 2.17 ส่วนประกอบของ Servo Motor



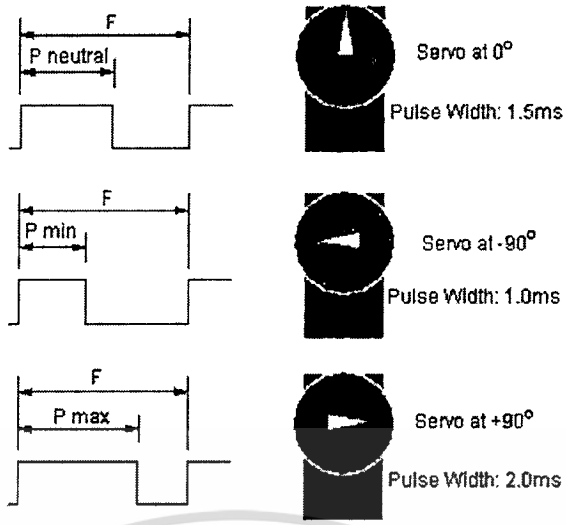
ภาพที่ 2.16 องค์ประกอบของ Servo Motor



ภาพที่ 2.17 ส่วนประกอบของ Servo Motor

การควบคุมการทำงานของ เซอร์โวมอเตอร์ ทำได้โดย การป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้จะขึ้นอยู่กับขนาดของความกว้างของพัลส์นั้นๆ โดยทั่วไปแล้วความกว้างของสัญญาณพัลส์จะมีจุดให้อ้างอิง 3 จุด ดังภาพที่ 2.18 คือ

- สัญญาณความกว้างพัลส์ขนาด 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือ จุดกึ่งกลางของมอเตอร์
- สัญญาณความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม - 90 องศา หรือในทิศทางทวนเข็มนาฬิกา
- สัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม + 90 องศา หรือในทิศทางตามเข็มนาฬิกา



ภาพที่ 2.18 สัญญาณพัลส์กับตำแหน่งการหมุน

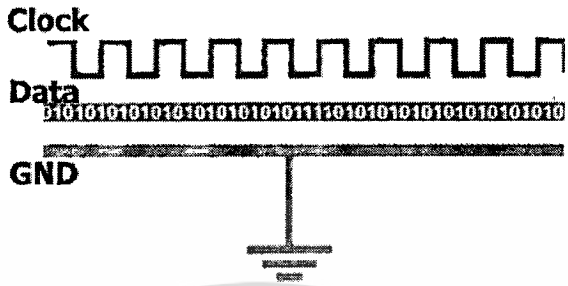
ส่วนการที่จะควบคุมให้มอเตอร์หมุนเป็นมุมอื่นๆ นั้นก็สามารถทำได้โดยการป้อนสัญญาณพัลส์เป็นระดับความกว้างต่างๆ โดยอ้างอิงจากจุด ทั้ง 3 จุดที่กล่าวมานี้ ตัวอย่างเช่น ถ้าต้องการให้มอเตอร์หมุนไปที่มุม - 45 องศา เราก็จะต้องป้อนสัญญาณพัลส์ที่มีความกว้าง 1.25 ms เป็นต้น และสัญญาณพัลส์นี้จะต้องจ่ายให้มอเตอร์ทุกๆ 20 ms (Period) เพื่อรักษาสภาพตำแหน่งของมอเตอร์ไว้ โดยหลักการก็คือ จะอาศัยการเปรียบเทียบช่วงเวลาของความกว้างพัลส์ที่จ่ายให้กับมอเตอร์ทางขาสัญญาณควบคุมกับค่าเวลาของวงจร RC ภายในบอร์ดควบคุมในตัวของมอเตอร์ ซึ่งค่าเวลาของวงจร RC นี้จะมีการเปลี่ยนแปลงตามการหมุนของมอเตอร์ เนื่องจากตัวต้านทานปรับค่าจะถูกยึดติดอยู่กับแกนหมุนของมอเตอร์ ซึ่งการหมุนของมอเตอร์จะทำให้ค่าความต้านทานของตัวต้านทานปรับค่า (VR) เปลี่ยนแปลงไป เป็นผลทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงตามไปด้วย โดยในขณะที่เราป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ทางขาสัญญาณควบคุม สัญญาณนี้จะถูกนำไปเปรียบเทียบกับค่าเวลาของวงจร RC หากค่าทั้ง 2 ไม่เท่ากันมอเตอร์ก็จะหมุนทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงจนกระทั่งค่าเวลาความกว้างพัลส์ของ วงจร RC เปลี่ยนแปลงจนเท่ากับสัญญาณพัลส์ทางขาควบคุม (Control line) มอเตอร์จึงจะหยุดหมุน

2.10 UART Serial Communication

การสื่อสารแบบอนุกรมจะแบ่งเป็น 2 แบบ คือ

1) การสื่อสารอนุกรมแบบ Synchronous เป็นรูปแบบที่ใช้วิธีส่งข้อมูล โดยใช้สัญญาณ Clock มาเป็นตัวกำหนดจังหวะ การรับส่งข้อมูล การส่งข้อมูลแบบนี้เป็นการรับส่งที่ค่อนค่างมีคุณภาพ และส่งได้ที่ความเร็วสูง มีโอกาสที่ข้อมูลจะสูญหายระหว่างการส่งน้อย ตัวอย่างการส่งข้อมูลลักษณะนี้เช่น I2C, I2S, SPI

ข้อเสียของการส่งข้อมูลแบบนี้คือ ต้องใช้สายสัญญาณมาก เพราะว่าต้องส่ง Clock ไปด้วย โดยภาพที่ 2.19 การสื่อสารอนุกรมแบบ Synchronous



ภาพที่ 2.19 การสื่อสารอนุกรมแบบ Synchronous

2) การสื่อสารอนุกรมแบบ Asynchronous เป็นการส่งข้อมูลที่ไม่ต้องใช้สัญญาณ Clock มาเป็นตัวกำหนดจังหวะการรับส่งข้อมูลแต่ ใช้วิธีกำหนด รูปแบบ Format การรับส่งข้อมูลขึ้นมาแทน และ อาศัยการกำหนด ความเร็วของการรับ และ ส่ง ที่เท่ากันทั้งฝั่งรับและฝั่งส่ง ข้อดีของการใช้ Asynchronous คือสามารถสื่อสารแบบ Full Duplex รับ และ ส่งได้ในเวลาเดียวกัน แต่ Asynchronous มีโอกาสที่ข้อมูลจะสูญหายขณะรับส่งข้อมูล หรือรับส่งข้อมูลผิดพลาดได้มากกว่าแบบ Synchronous

UART ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter หมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส (Asynchronous) ซึ่งเป็นส่วนหนึ่งในการสื่อสารอนุกรมแบบ Asynchronous

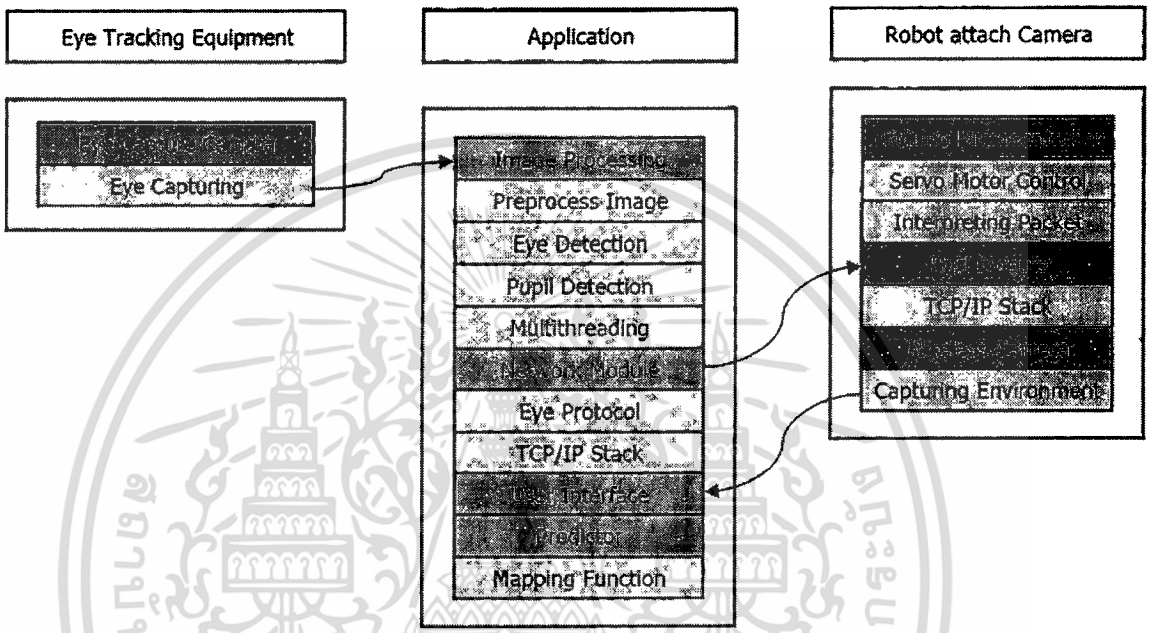
UART (Universal Asynchronous Receiver Transmitter) จะมีรูปแบบการส่งข้อมูล ที่ถูกกำหนดขึ้นมาเพื่อใช้รับส่งข้อมูลแบบ Asynchronous โดยมีรูปแบบดังภาพที่ 2.20

ภาพที่ 2.20 รูปแบบการส่งข้อมูลของ UART

เริ่มต้นจาก Start Bit เป็น Logic 0 จากนั้นจะตามด้วย Data ที่เราส่ง แล้วจะถูกปิดด้วย STOP Bit เป็น Logic 1

บทที่ 3
การออกแบบและพัฒนา

การออกแบบและพัฒนาระบบการควบคุมกล้องจับภาพบนตัวหุ่นยนต์ด้วยการหมุนของดวงตาแบ่งเป็น 3 ส่วนหลัก ดังแสดงในภาพที่ 3.1



ภาพที่ 3.1 Block diagram ของระบบ

3.1 อุปกรณ์ติดตั้งกล้องจับภาพดวงตา

3.1.1 ออกแบบอุปกรณ์ติดตั้ง

การออกแบบอุปกรณ์ติดตั้งจะออกแบบลักษณะคล้ายหมวกให้สวมใส่ได้ง่ายโดยมีกล้องติดอยู่ข้างหน้าจับการเคลื่อนไหวของดวงตาข้างซ้าย โดยกล้องที่ติดกับหมวกเป็นกล้อง Microsoft LifeCam ซึ่งจะส่งภาพของดวงตาไปยัง Application ด้วย USB 2.0 โดยการออกแบบส่วนนี้ทำขึ้นเพื่อป้องกันปัญหาที่อาจจะเกิดขึ้นกับการจับภาพดวงตา ซึ่งทำให้กล้องสามารถจับภาพดวงตาได้อย่างแม่นยำตลอดเวลาแม้จะเกิดการเคลื่อนไหวของศีรษะก็ตาม ภาพที่ 3.2 แสดงการออกแบบอุปกรณ์ติดตั้งกล้องจับภาพดวงตา และภาพที่ 3.3 แสดงอุปกรณ์ติดตั้งกล้องจับภาพดวงตาที่ได้ออกแบบ

ภาพที่ 3.2 การออกแบบอุปกรณ์ติดตั้งกล้องจับภาพดวงตา

ภาพที่ 3.3 อุปกรณ์ติดตั้งกล้องจับภาพดวงตาที่ได้ออกแบบ

3.1.2 กล้องจับภาพดวงตา

กล้องจับภาพดวงตาจะเลือกใช้กล้อง Microsoft LifeCam Studio ซึ่งสามารถถ่ายภาพได้ความละเอียดสูงสุดที่ 1920 x 1080 pixel ซึ่ง Microsoft LifeCam Studio มีรายละเอียดดังนี้

- 1080p HD Sensor ซึ่งให้คุณภาพของภาพและความคมชัดสูง
- 720p HD video chat
- ใช้วัสดุแก้วคุณภาพสูง lens จึงให้คุณภาพของภาพที่ดี

- TrueColor Technology ช่วยในการควบคุมความสว่างและสีสันโดยอัตโนมัติ-รองรับการใช้งาน Skype ในระดับ HD
- Wideband microphone สำหรับการอัดเสียงอย่างมีคุณภาพและให้เสียงที่เป็นธรรมชาติ
- หมุนได้ 360 องศาทั้ง 2 ทิศทางเพื่อมุมมองรอบด้าน
- Auto focus ตั้งแต่ระยะ 4 นิ้วจนถึงระยะอนันต์
- Wide-angle lens สามารถจับภาพรอบทิศทางมากขึ้น

โดยการที่ผู้พัฒนาเลือกใช้กล้องจับภาพชนิดนี้ เพราะเป็นกล้องที่มีความละเอียดของภาพสูงซึ่งจำเป็นอย่างยิ่ง เนื่องจากการเคลื่อนไหวของตาดำนั้นมองเห็นได้กว้างประมาณ ๑๐๐ องศา ทางด้านข้าง ๖๐ องศา ทางด้านบนและด้านใกล้จมูก และ ๗๕ องศา ทางด้านล่าง ในขณะที่การเคลื่อนที่ของตาดำในกล้องที่มีความละเอียดระดับ 1920 x 1080 pixel นั้นจะสามารถเคลื่อนไปได้ประมาณ 450 pixel ทางด้านข้าง 250 pixel ทางด้านบนและด้านใกล้จมูก และ 300 pixel ทางด้านล่าง ซึ่งเมื่อนำการเปลี่ยนแปลงของ pixel มาใช้ในการขับเคลื่อนแขนหุ่นยนต์จะสามารถเคลื่อนแขนหุ่นยนต์ได้อย่างละเอียดและและมีความส่ายของแขนที่น้อย นอกจากนี้ กล้อง Microsoft Lifecam Studio ยังมีระยะโฟกัสที่น้อยที่สุดที่ห่างพอดีกับระยะห่างระหว่างกล้องกับดวงตาที่เหมาะสมอีกด้วย

3.2 Application

Application ทำงานบนสภาพแวดล้อมของระบบปฏิบัติการ Window 7 โดย Application จะทำหน้าที่หลัก 2 หน้าที่คือประมวลผลภาพจากกล้องจับภาพดวงตาเพื่อตรวจหาดวงตาและทำนายทิศทางของดวงตาดำ อีกหนึ่งหน้าที่คือการรับส่งข้อมูลด้วยระบบ Network บน TCP เพื่อส่งข้อมูลไปยังตัวหุ่นยนต์และรับข้อมูลภาพและคำสั่งจากตัวหุ่นยนต์ได้ ซึ่งการพัฒนา Application จะประกอบด้วยการเลือก Software Development Tool และแนวคิดของการประมวลผลภาพดังนี้

3.2.1 Software Development Tool and Development Language

การพัฒนา Application จะพัฒนาบนสภาพแวดล้อมของระบบปฏิบัติการ Window 7 ซึ่งประกอบด้วย Software Tool เครื่องมือที่ใช้ในการพัฒนาระบบขึ้นดังนี้

1) Microsoft Visual Studio 2008 เป็น Software Tool หลักที่ใช้ในการพัฒนา Window Application Form ซึ่งทาง Microsoft ผู้ออกแบบ Software Tool นี้เพื่อให้พัฒนา Application เพื่อทำงานบนระบบปฏิบัติการของ Window Microsoft Visual Studio 2008 สามารถสร้าง Window Form ได้อย่างง่ายดายด้วย User Graphic Interface และสามารถเขียนโปรแกรมเพื่อควบคุมการทำงานและ Event ที่เข้าใจง่าย นอกจากนี้มี Text Editor ช่วยทำให้ง่ายต่อการพัฒนาและ

ลดความผิดพลาดในการเขียนได้ Microsoft Visual Studio 2008 มี User Interface ที่ง่ายต่อการ Debug โปรแกรมด้วยการประมวลผลคำสั่งแบบ Break Point ได้และสามารถดูค่าตัวแปรและ Address ของตัวแปรได้ง่าย ดังนั้นทางกลุ่มผู้พัฒนาจึงเลือก Software Tool นี้เพื่อใช้ในการพัฒนา Application

2) ภาษา C# เป็นภาษาเชิง Object Oriented Programming มี Syntax คล้ายกับภาษา Visual Basic สามารถเข้าใจได้ง่าย มีความปลอดภัยในการเข้าถึงตัวแปรด้วย Scope ของ Object สามารถพัฒนา Application ได้โดยใช้เวลาไม่นาน ซึ่งทางกลุ่มจะเลือกใช้ภาษา C# ในการพัฒนา Application บน Microsoft Visual Studio 2008

3) EmguCV 2.3 Library คือ Opensource Library เกี่ยวกับการประมวลผลภาพของคอมพิวเตอร์ซึ่งมีการเปิดเผย SourceCode ผู้พัฒนาสามารถแก้ไขหรือปรับเปลี่ยน Algorithm ให้เหมาะสมตามที่ต้องการได้ EmguCV จะใช้พัฒนาร่วมกับภาษา C# ซึ่งประกอบด้วย Function Call ที่สำเร็จรูปมากมายเกี่ยวกับการประมวลผลภาพ ซึ่งใน Library นี้มีการรวมการประมวลผลภาพเพื่อตรวจหาอวัยวะในส่วนใบหน้าด้วย Haar Algorithm ซึ่งมีความแม่นยำอยู่ในระดับที่ดี นอกจากนี้ยังสามารถหาแหล่งอ้างอิงของข้อมูลได้ง่ายเนื่องจาก EmguCV 2.3 Library ค่อนข้างได้รับความนิยมในการเขียนโปรแกรมเพื่อพัฒนา Application ประมวลผลภาพ

4) MATLab เป็น Software Tool เพื่อใช้ในการประมวลผล Matrix ที่ได้รับความนิยมในทางวิศวกรรม ในการประมวลผลภาพ MATLab ได้รับความนิยมเนื่องจากภาพคือ Matrix ของสีในแต่ละ Pixel ดังนั้น MATLab สามารถใช้ในการประมวลผลภาพด้วยการทำ Operation บน Matrix ซึ่ง MATLab ใช้ภาษา MATLab C ในการพัฒนา MATLab สามารถพัฒนาได้ด้วยระยะเวลาไม่นาน และมี Function ที่เกี่ยวกับการจัดการ Matrix ของภาพค่อนข้างหลากหลาย เช่น การแยก Channel ของภาพ, การตรวจหาขอบของรูปภาพ, และการแปลงภาพเป็น Greyscale เป็นต้น เนื่องจาก MATLab สามารถประมวลผลข้อมูลใน Matrix ในแต่ละ Offset ได้ทำให้สามารถประมวลผลด้วย Algorithm ที่สามารถออกแบบได้เอง ซึ่งทางกลุ่มพัฒนาได้นำ MATLab มาเพื่อเขียน Function ในหารตรวจหาใบหน้าและดวงตา จากนั้น Export Function ในการประมวลผลภาพที่ได้เขียนไว้มาเป็นนามสกุล dll ได้ซึ่งสามารถนำไป Interface กับภาษา C# ได้

3.2.2 Graphic User Interface

การออกแบบหน้าจอของ Graphic User Interface ของ Application จะแบ่งเป็น 2 ส่วนหลักดังนี้

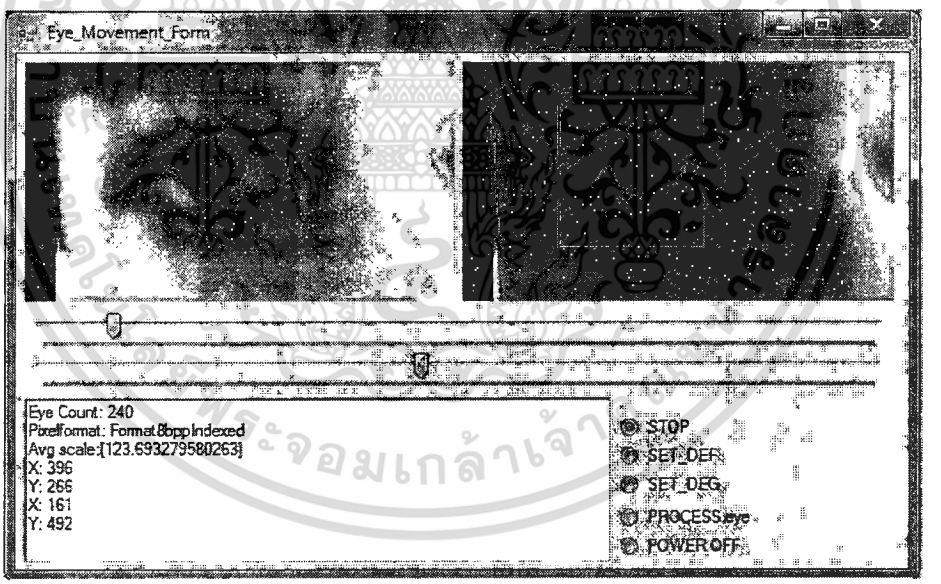
3.2.2.1 การออกแบบ User Interface เพื่อใช้ในการพัฒนา

การออกแบบ User Interface เพื่อใช้ในการพัฒนาหรือทดสอบการประมวลผลภาพจะต้องออกแบบให้ง่ายต่อการดูผลลัพธ์ของการประมวลผลภาพและง่ายต่อการหาข้อผิดพลาดการทำงานของ Application ซึ่งหน้าจอ Application จะประกอบด้วยการแสดงผลภาพหลาย Picture Box และประกอบด้วย Textbox เพื่อใช้ในการ Input ค่าและแสดงผล Output ค่าแบบ Text ในการตรวจหาข้อผิดพลาด ซึ่งการออกแบบหน้าจอ User Interface ในการพัฒนานั้นจะเปลี่ยนแปลงไปตามความเหมาะสมของการพัฒนา เช่น ในช่วงแรกของการพัฒนาจะเน้นในการประมวลผลภาพจึงออกแบบหน้าจอให้แสดงผลภาพด้วยขนาดที่เหมาะสม ในภายหลังเมื่อสามารถประมวลผลภาพได้ตามต้องการแล้วได้ออกแบบให้มีการแสดงข้อความเพื่อดูผลลัพธ์ตำแหน่งที่ตรวจจับได้เพื่อตรวจสอบความถูกต้อง เป็นต้น ภาพที่ 3.4 แสดง Graphic User Interface สำหรับการพัฒนาเพื่อแสดงผลภาพการตรวจจับใบหน้า

ภาพที่ 3.4 Graphic User Interface สำหรับการพัฒนาเพื่อแสดงผลภาพการตรวจจับใบหน้า

โดยภาพที่ 3.4 แสดงถึง Graphic User Interface ที่ออกแบบเพื่อในใช้การตรวจหาใบหน้าด้วยกล้อง Web Camera บน Notebook Computer เท่านั้น รายละเอียดของ Graphic User Interface แสดงดังภาพที่ 3.5 มีดังนี้

1) Pre processing image คือภาพ raw image จากกล้องจับดวงตาที่ถูกนำมาประมวลผลก่อนในครั้งแรกเพื่อนำไปประมวลผลภาพตรวจจับใบหน้าต่อไป
2) Post processing image คือภาพที่นำมาประมวลผลในการตรวจหาใบหน้าโดยการวาดกรอบสี่เหลี่ยมให้เห็นว่าตรวจจับใบหน้าที่ตำแหน่งใดและมีขนาดของสี่เหลี่ยมเป็นเท่าใดและตำแหน่งที่คาดว่าจะตรวจจับดวงตาคือตำแหน่งใบของใบหน้า
3) Left eye image และ Right eye image จะแสดงภาพดวงตาซ้ายและดวงตาขวาที่ Crop มาจากภาพ Pre processing image เพื่อแสดงให้เห็นว่าการทำ Pre processing image ส่งผลให้เห็นดวงตาดำชัดเท่าใด
4) Information Textbox จะแสดงข้อมูลที่ใช้ในการพัฒนาในการประมวลผลภาพ เช่น พิกัดของใบหน้าและความกว้างของกรอบใบหน้า, พิกัดของดวงตาซ้ายและขวา, และจำนวนภาพที่ถูก process ติดต่อกัน เป็นต้น



ภาพที่ 3.5 Graphic User Interface สำหรับการพัฒนาเพื่อแสดงผลภาพการตรวจจับดวงตา

Graphic User Interface นี้ออกแบบเพื่อในใช้การตรวจหาดวงตาด้วยกล้อง TP Web Camera M490 ที่มีความละเอียด 640 x 480 รายละเอียดของ Graphic User Interface ดังภาพที่ 3.5 มีดังนี้

1) Pre processing image คือภาพ raw image จากกล้องจับดวงตาที่ถูกนำมาประมวลผลก่อนในครั้งแรกเพื่อนำไปประมวลผลภาพตรวจจับใบหน้าต่อไป
2) Post processing image คือภาพที่นำมาประมวลผลในการตรวจหาดวงตาโดยการวาดกรอบสี่เหลี่ยมให้เห็นว่าตรวจจับดวงตาที่ตำแหน่งใดและมีขนาดของสี่เหลี่ยมเป็นเท่าใด
3) Information Textbox จะแสดงข้อมูลที่ใช้ในการพัฒนาในการประมวลผลภาพ เช่น พิกัดของดวงตา, ขนาดความกว้างและความยาวของกรอบดวงตา, จำนวนภาพที่ประมวลผลได้ในหนึ่งวินาที, และจำนวนภาพที่ถูก process ติดต่อกัน เป็นต้น

ภาพที่ 3.6 Graphic User Interface สำหรับการพัฒนาเพื่อแสดงผลภาพการตรวจจับรูม่านตาหรือดวงตาดำ

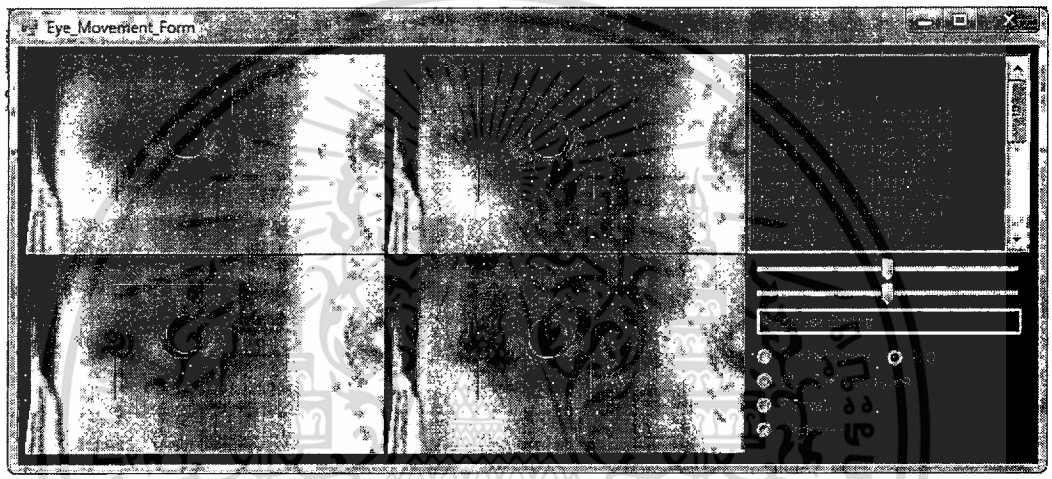
Graphic User Interface นี้ออกแบบเพื่อในใช้การตรวจหาดวงตาด้วยกล้อง Microsoft LifeCam ที่มีความละเอียด 1920*1080 ดังภาพที่ 3.6

รายละเอียดของ Graphic User Interface มีดังนี้

1) Pre processing image คือภาพ raw image จากกล้องจับดวงตาที่ถูกนำมาประมวลผลก่อนในครั้งแรกเพื่อนำไปประมวลผลภาพตรวจจับใบหน้าต่อไป

2) Post processing image คือภาพที่นำมาประมวลผลในการตรวจหาดวงตาโดยการวาดกรอบสี่เหลี่ยมให้เห็นว่าตรวจจับดวงตาที่ตำแหน่งใดมีขนาดของสี่เหลี่ยมเป็นเท่าใด และแสดงการตรวจหารูม่านตาด้วยการแสดงวงกลมภายในกรอบของดวงตาที่ตรวจพบ

3) Information Textbox จะแสดงข้อมูลที่ใช้ในการพัฒนาในการประมวลผลภาพ เช่น พิกัดของดวงตา, ขนาดความกว้างและความยาวของกรอบดวงตา, จำนวนภาพที่ประมวลผลได้ในหนึ่งวินาที, และจำนวนภาพที่ถูก process ติดต่อกัน เป็นต้น



ภาพที่ 3.7 Graphic User Interface สำหรับการพัฒนาเพื่อแสดงผลภาพด้วย Multithread Processing

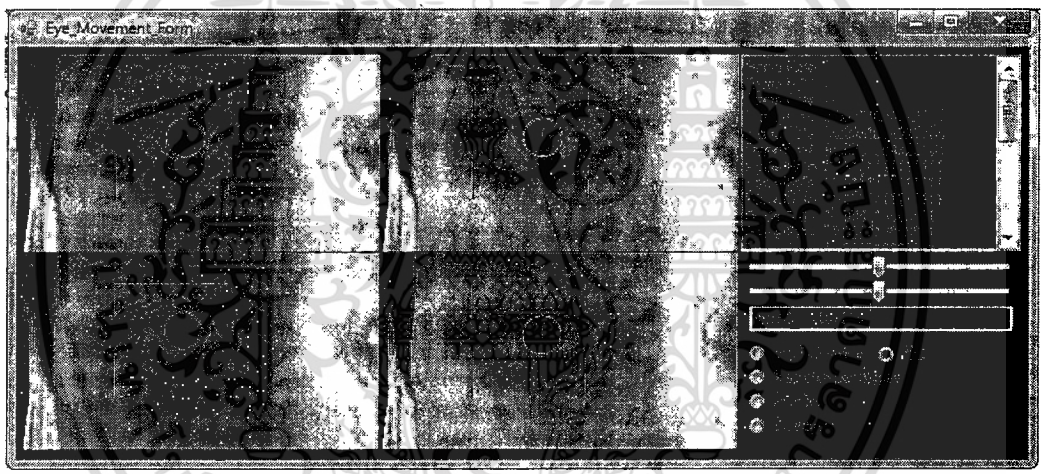
Graphic User Interface นี้ออกแบบเพื่อในใช้การตรวจหาดวงตาด้วยกล้อง Microsoft LifeCam ที่มีความละเอียด 1920*1080 และทำการประมวลผลด้วย 4 Threads เพื่อประมวลผลตรวจหาดวงตาดำและรูม่านตาดังภาพที่ 3.7

รายละเอียดของ Graphic User Interface มีดังนี้

1) Post processing image จำนวน 4 Picture Box คือภาพที่นำมาประมวลผลในการตรวจหาดวงตาโดยการวาดกรอบสี่เหลี่ยมให้เห็นว่าตรวจจับดวงตาที่ตำแหน่งใดมีขนาดของสี่เหลี่ยมเป็นเท่าใด และแสดงการตรวจหารูม่านตาด้วยการแสดงวงกลมภายในกรอบของดวงตาที่ตรวจพบ การแสดงผลจะเป็น 4 Picture Box ซึ่งแสดงผลให้เห็นในการประมวลผลภาพของแต่ละ Thread

2) Track bar จะแสดงผลให้เห็นตำแหน่งของรูม่านตาที่ตรวจจับได้ในแกน X และแกน Y เพื่อให้ง่ายต่อการดูผลลัพธ์มากกว่าการใช้ Textbox

3) Information Textbox จะแสดงข้อมูลที่ใช้ในการพัฒนาในการประมวลผลภาพด้วย 4 Thread เนื่องจาก Thread มีการทำงานด้วยการ Schedule ของระบบปฏิบัติการดังนั้นการ Debug ด้วย Break Point จึงทำได้ยากแต่ใช้การ Debug ด้วยการแสดงผลเป็น Text ออกทาง Textbox แทนซึ่งแสดงผล เช่น พิกัดของดวงตา, ขนาดความกว้างและความยาวของกรอบดวงตา, จำนวนภาพที่ประมวลผลได้ในหนึ่งวินาที, จำนวนภาพที่ถูก process ติดต่อกัน, ลำดับการเข้าทำงานของ Thread, และลำดับการประมวลผลเสร็จของ Thread เป็นต้น



ภาพที่ 3.8 Graphic User Interface สำหรับการพัฒนาเพื่อหาค่าที่เหมาะสมในการตรวจจับรูม่านตา

Graphic User Interface นี้ออกแบบเพื่อในใช้การตรวจหาดวงตาด้วยกล้อง Microsoft LifeCam ที่มีความละเอียด 1920*1080 และทำการกำหนดค่าที่เหมาะสมในการใช้ตรวจหาตำแหน่งของรูม่านตาด้วยการกำหนดค่าตัวแปรต่างๆ ดังภาพที่ 3.8

รายละเอียดของ Graphic User Interface มีดังนี้

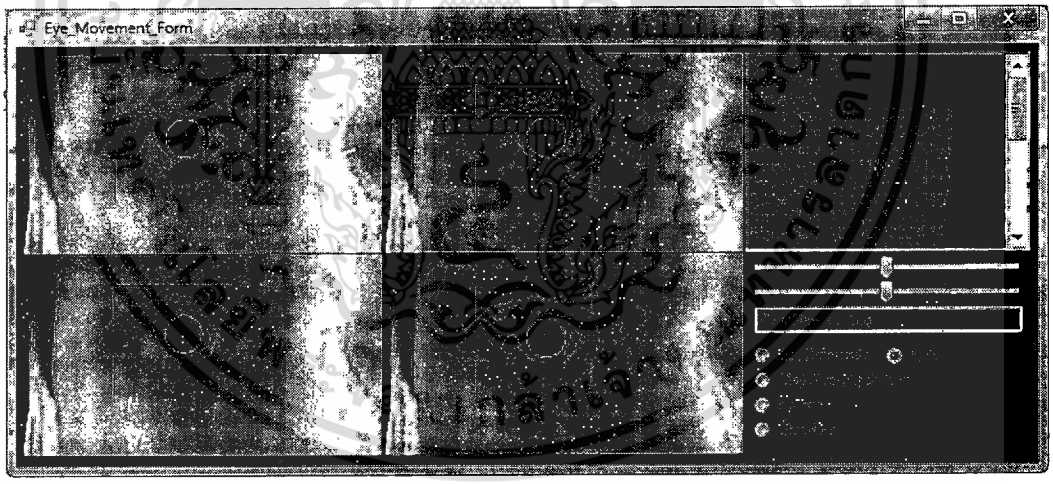
1) Post processing image จำนวน 4 Picture Box คือภาพที่นำมาประมวลผลในการตรวจหาดวงตาโดยการวาดกรอบสี่เหลี่ยมให้เห็นว่าตรวจจับดวงตาที่ตำแหน่งใดมีขนาดของสี่เหลี่ยมเป็นเท่าใด และแสดงการตรวจหารูม่านตาด้วยการแสดงวงกลมภายในกรอบของ

ดวงตาที่ตรวจพบ การแสดงผลจะเป็น 4 Picture Box ซึ่งแสดงผลให้เห็นในการประมวลผลภาพของแต่ละ Thread

2) Track bar จะแสดงผลให้เห็นตำแหน่งของรูม่านตาที่ตรวจจับได้ในแกน X และแกน Y เพื่อให้ง่ายต่อการดูผลลัพธ์มากกว่าการใช้ Textbox

3) Input Textbox ใช้เพื่อการรับค่าเพื่อนำไปใช้ในการกำหนดค่า Parameter ในการตรวจหาตำแหน่งของรูม่านตาให้เหมาะสมโดยจะประกอบด้วย 6 Input Textbox

4) Information Textbox จะแสดงข้อมูลที่ใช้ในการพัฒนาในการประมวลผลภาพด้วย 4 Thread เนื่องจาก Thread มีการทำงานด้วยการ Schedule ของระบบปฏิบัติการดังนั้นการ Debug ด้วย Break Point จึงทำได้ยากแต่ใช้การ Debug ด้วยการแสดงผลเป็น Text ออกทาง Textbox แทนซึ่งแสดงผล เช่น พิกัดของดวงตา, ขนาดความกว้างและความยาวของกรอบดวงตา, จำนวนภาพที่ประมวลผลได้ในหนึ่งวินาที, และจำนวนภาพที่ถูก process ติดต่อกัน, เป็นต้น



ภาพที่ 3.9 Graphic User Interface สำหรับการพัฒนาเพื่อทดสอบการส่งข้อมูลผ่านระบบ Network ด้วย Protocol ที่คิดขึ้นมา

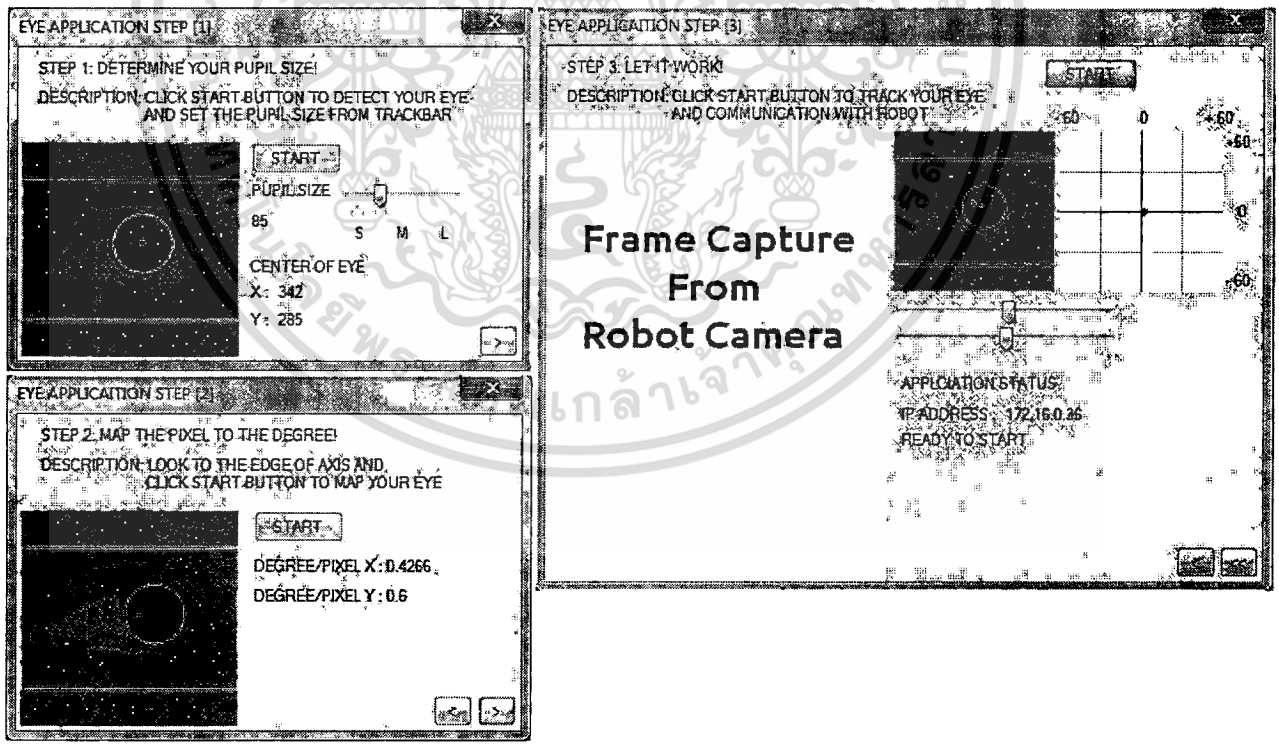
Graphic User Interface นี้ออกแบบเพื่อในใช้การทดสอบการส่งข้อมูลด้วย Protocol ที่ผู้จัดทำคิดขึ้นมา ดังภาพที่ 3.9

รายละเอียดของ Graphic User Interface มีดังนี้

1) Post processing image จำนวน 4 Picture Box คือภาพที่นำมาประมวลผลในการตรวจหาดวงตาโดยการวาดกรอบสี่เหลี่ยมให้เห็นว่าตรวจจับดวงตาที่ตำแหน่งใดมีขนาดของสี่เหลี่ยมเป็นเท่าใด และแสดงการตรวจหารูม่านตาด้วยการแสดงวงกลมภายในกรอบของดวงตาที่ตรวจพบ การแสดงผลจะเป็น 4 Picture Box ซึ่งแสดงผลให้เห็นในการประมวลผลภาพของแต่ละ Thread
2) Track bar จะแสดงผลให้เห็นตำแหน่งของรูม่านตาที่ตรวจจับได้ในแกน X และแกน Y เพื่อให้ง่ายต่อการดูผลลัพธ์
3) Input Textbox ใช้เพื่อการรับค่าเพื่อนำไปใช้ในการกำหนดค่า Parameter ในการตรวจหาตำแหน่งของรูม่านตาให้เหมาะสม
4) Information Textbox จะแสดงข้อมูลที่ใช้ในการส่งและรับข้อมูลผ่านระบบ Network ที่จะติดต่อกับ Wi -Fi Module บนตัวหุ่นยนต์

3.2.2.2 การออกแบบ User Interface เพื่อการใช้งานของ User

การออกแบบ Graphic User Interface เพื่อนำไปใช้งานกับ User จะออกแบบให้มีลักษณะการใช้งานที่ง่ายที่สุดและเน้นความสวยงามของ Application ดังภาพที่ 3.10



ภาพที่ 3.10 User Interface สำหรับใช้การใช้งานของ User

การออกแบบ User Interface จะแบ่งเป็น 3 หน้าต่าง ได้แก่ หน้าต่างการตั้งค่าจุดกึ่งกลางของดวงตา, หน้าต่างการตั้งค่าการแปลง Pixel เป็นองศา, และหน้าต่างการประมวลผลดวงตา

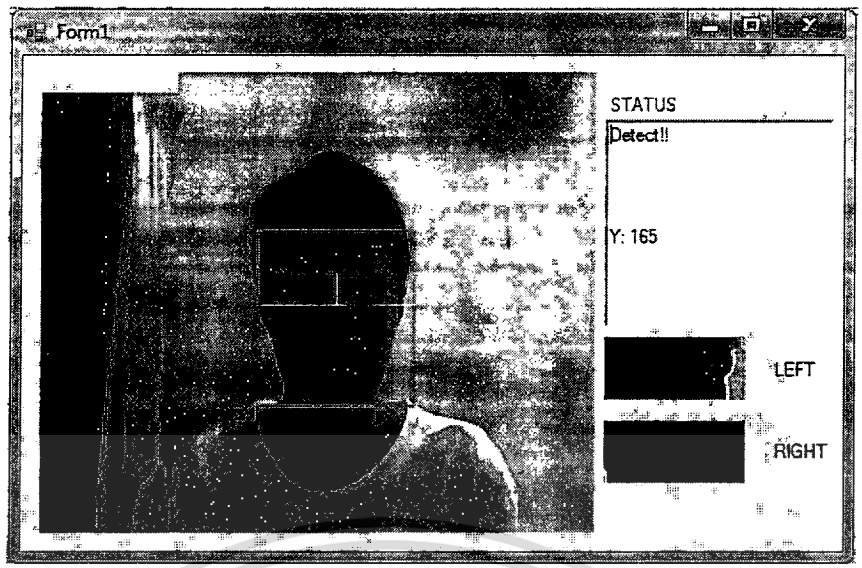
รายละเอียดของ Graphic User Interface แต่ละหน้าต่าง มีดังนี้

1) Eye Application Step [1] ใช้ในการตั้งค่าจุดกึ่งกลางของดวงตา ประกอบด้วยปุ่ม Start เพื่อเริ่มการทำงาน นอกจากนี้สามารถปรับขนาดของรูม่านตาให้เหมาะสมด้วย Trackbar ได้
2) Eye Application Step [2] ใช้ในการตั้งค่าการแปลง Pixel เป็นองศา ประกอบด้วยปุ่ม Start เพื่อเริ่มการทำงาน นอกจากนี้มีปุ่ม <, > ใช้ในการเปลี่ยนหน้าต่างไปยังหน้าต่างก่อนหน้าและหน้าต่างถัดไป
3) Eye Application Step [3] ใช้ในการเริ่มการทำงานของระบบ ประกอบด้วยปุ่ม Start เพื่อเริ่มการทำงาน, Picturebox ด้านซ้ายแสดงผลลัพธ์ภาพที่ถ่ายจากตัวหุ่นยนต์, Picturebox ด้านขวาแสดงถึงภาพดวงตาที่ประมวลผลแล้ว, กราฟแสดงถึงตำแหน่งการมองของดวงตา ประกอบด้วยแกน X, Y ซึ่งจะแสดงด้วยค่าองศา, Trackbar จะแสดงค่าองศาในแนวแกน X, Y คล้ายกับกราฟ, Text Label จะแสดง IP Address ของ Application และแสดงสถานะการทำงานของระบบ

3.2.3 การประมวลผลภาพเพื่อตรวจหาใบหน้า

การประมวลผลภาพเพื่อตรวจหาตำแหน่งของใบหน้าจะใช้ EmguCV Library ซึ่ง EmguCV มี Function สำเร็จรูปในการตรวจหารูปร่างของใบหน้า เรียกว่า Haar Detection ซึ่งเป็น Algorithm ในการหาวัตถุในภาพโดยการสร้างสี่เหลี่ยมขึ้นมาจำนวนมากและใช้ค่าถ่วงน้ำหนักในการตัดสินใจว่าเป็นวัตถุที่ต้องการหรือไม่ Haar Detection ใบ EmguCV สามารถตรวจสอบใบหน้าได้โดยอ้างอิงจาก Face.xml ซึ่งเป็น File ที่เก็บข้อมูลสถิติเพื่อใช้ในการตัดสินใจวัตถุในรูปภาพ

การตรวจสอบใบหน้าทำได้โดยการนำภาพที่ถ่ายจากกล้อง Web Camera ของ Notebook ขนาด 640 x 480 มาทำการ Preprocess ดังภาพที่ 3.11 โดยการแปลงภาพเป็น Grey Scale และทำ Equalizehist เพื่อทำให้ระดับสีเทาของภาพมีความเข้มขึ้น จากนั้นจึงนำภาพที่ทำ Preprocess แล้วมาเป็นภาพ Input ของกระบวนการ Haar Face Detection ซึ่งสามารถเลือกการตรวจหาภาพได้หลายแบบ เช่น Canny Pruning และ Biggest Object ซึ่งจะตรวจหาวัตถุที่ใหญ่ที่สุดผลลัพธ์ที่ได้จะเป็นวัตถุเพียงรูปเดียวเท่านั้น เป็นต้น เมื่อทำ Haar Face Detection แล้วจะได้ตำแหน่งพิกัดมุมซ้ายบนของใบหน้าที่ตรวจสอบพบและขนาดของสี่เหลี่ยมที่ตรวจสอบพบ

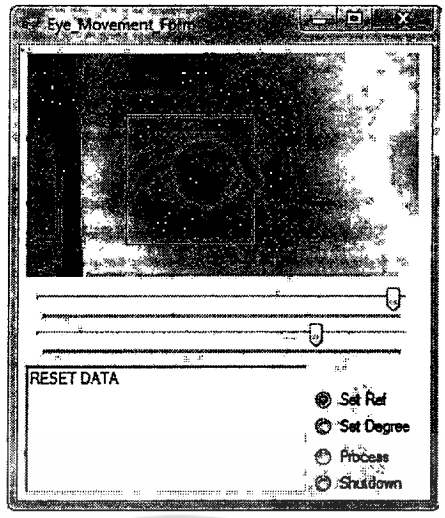


ภาพที่ 3.11 การประมวลผลภาพเพื่อตรวจหาใบหน้า

3.2.4 การประมวลผลภาพเพื่อตรวจหาดวงตา

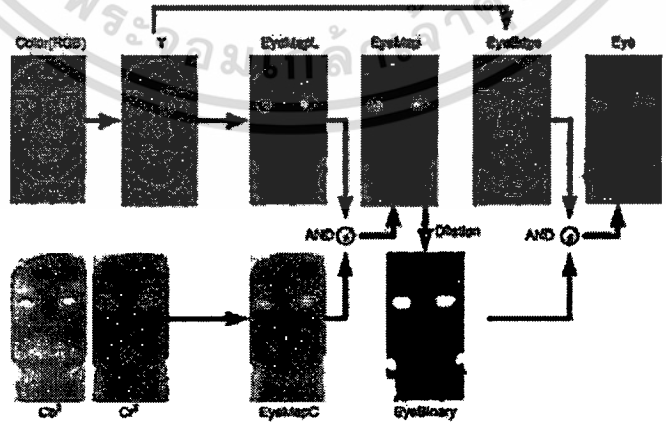
การประมวลผลภาพเพื่อตรวจหาตำแหน่งของดวงตาในรูปภาพ ทางกลุ่มพัฒนาได้ทำการทดลอง 2 วิธีในการตรวจหาตำแหน่งของดวงตาดังนี้

1) กระบวนการตรวจหาดวงตาโดยใช้ EmguCV Library โดยใช้ Haar Eye Detection ซึ่งใน EmguCV มี Function ที่สำเร็จรูปในการหาตำแหน่งของดวงตา โดยการหาตำแหน่งดวงตาเริ่มจากการนำภาพที่ถ่ายจากกล้อง Microsoft LifeCam ที่มีขนาด 1920 x 1080 มาทำการ Preprocess โดยการแปลงภาพเป็น Grey Scale และทำ Equalizehist เพื่อทำให้ระดับสีเทาของภาพมีความเข้มขึ้น จากนั้นจึงนำภาพที่ทำ Preprocess แล้วมาเป็นภาพ Input ของกระบวนการ Haar Eye Detection ซึ่งสามารถเลือกการตรวจหาภาพได้หลายแบบ เช่น Canny Pruning และ Biggest Object ซึ่งจะตรวจหาวัตถุที่ใหญ่ที่สุดผลลัพธ์ที่ได้จะเป็นวัตถุเพียงรูปเดียวเท่านั้น เป็นต้น เมื่อทำ Haar Eye Detection แล้วจะได้ตำแหน่งพิกัดมุมซ้ายบนของดวงตาที่ตรวจสอบพบและขนาดของสี่เหลี่ยมที่ตรวจสอบพบ ภาพที่ 3.12 แสดงการประมวลผลภาพเพื่อตรวจหาดวงตาโดยใช้ EmguCV



ภาพที่ 3.12 การประมวลผลภาพเพื่อตรวจหาดวงตาโดยใช้ EmguCV

2) การตรวจหาดวงตาด้วยการประมวลผลด้วย MATLab ซึ่งการตรวจหาดวงตาจะทำโดย เริ่มจากการนำภาพที่ถ่ายจากกล้อง Microsoft LifeCam ที่มีขนาด 1920 x 1080 มาทำการแปลง Channel ของสีเป็น YCbCr และนำ Channel ของ Y ไปทำกระบวนการ EyeMapL ส่วน Channel Cb, Cr ให้นำมาทำกระบวนการ EyeMapC และให้นำภาพผลลัพธ์ของ EyeMapL และ EyeMapC มาทำ Operation And กัน ผลลัพธ์ของการทำ And เรียกว่า EyeMap ให้นำภาพผลลัพธ์นี้ไปทำกระบวนการ Dilation จะได้เป็นภาพ Binary ที่มีบริเวณสีขาวเป็นบริเวณดวงตา ให้นำภาพ Binary ที่ได้ไปทำกระบวนการ And กับภาพ Y ที่ได้ทำ Edge Detection ทำให้ได้ภาพเฉพาะส่วนดวงตาตามที่ต้องการ ดังภาพที่ 3.13



ภาพที่ 3.13 การประมวลผลภาพเพื่อตรวจหาดวงตาโดยใช้ Matlab

ซึ่งกระบวนการทั้งหมดทำโดยการเขียน function ใน MATLab และ Export ออกมาเป็น Dll extension เพื่อนำไปเชื่อมโยงกับภาษา C# ได้

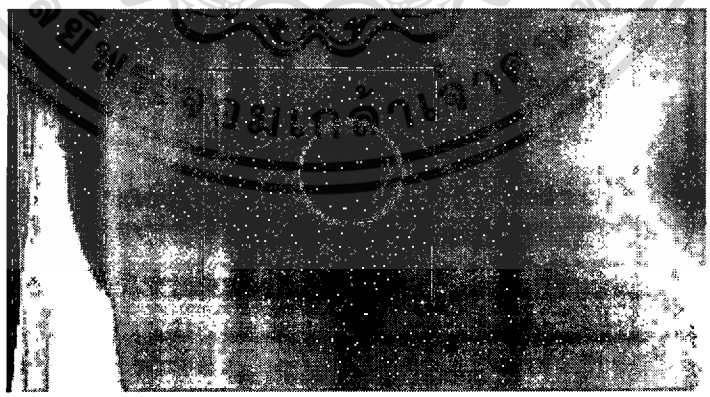
เปรียบเทียบกระบวนการหาตำแหน่งของดวงตาจาก 2 วิธีได้ดังนี้

1) การหาดวงตาโดยใช้ Haar Eye Detection มีความแม่นยำในการตรวจหาตำแหน่งดวงตามากกว่ากระบวนการ EyeMap ซึ่งเขียนใน MATLab
2) ประสิทธิภาพของความเร็วในการประมวลผล EmguCV สามารถทำได้รวดเร็วกว่าการประมวลผลด้วย Function ที่ Export มาจาก MATLab มากจนเห็นได้ชัด

ดังนั้นในการประมวลผลภาพดวงตาทางกลุ่มผู้พัฒนาจึงเลือกใช้ EmguCV Library ด้วย Haar Eye Detection เพื่อนำมาตรวจหาตำแหน่งของดวงตา

3.2.5 การประมวลผลภาพเพื่อตรวจหารูม่านตา

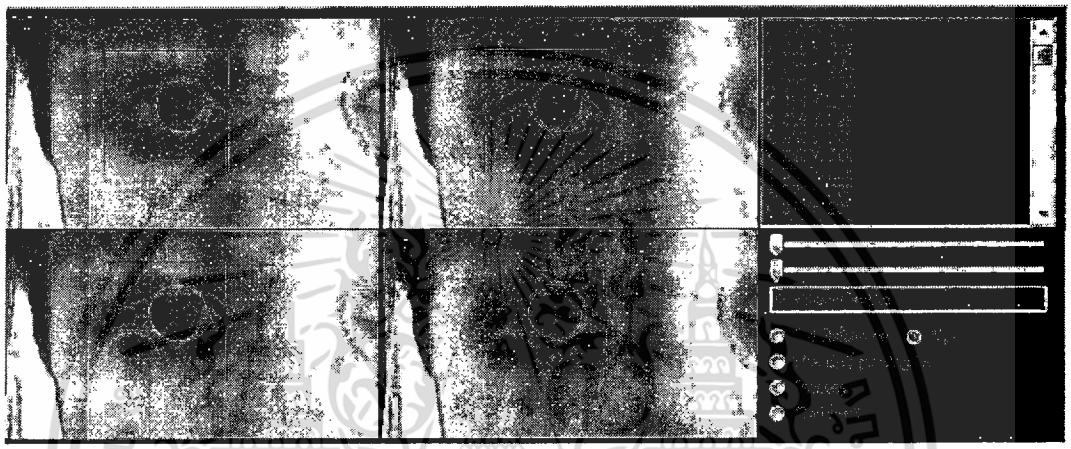
การประมวลผลภาพเพื่อตรวจหาตำแหน่งรูม่านตาทำโดยการนำภาพที่ได้แปลงเป็น Grey Scale และ Equalizehist นำมาทำ ROI เพื่อด้วยตำแหน่งของดวงตาที่ตรวจหาพบจากการบวนการ Eye Detection จากนั้นนำภาพ ROI ไปทำการหาวงกลมในภาพโดยใช้ EmguCV Library ด้วย Circle Detection ในการหาวงกลมจะต้องกำหนดค่า Threshold เพื่อแยกแยะระดับ Grey Scale ของวงกลมที่ต้องการหาและต้องกำหนดขนาดของวงกลมที่ต้องการหาได้แก่ ค่าระดับสี Threshold ของการทำ Canny Edge Detection, ค่าระดับสี Threshold ของการทำ Accumulator, ค่ารัศมีของวงกลมน้อยที่สุดและมากที่สุดที่ต้องการหา, และค่าระยะมากที่สุดและน้อยที่สุดของวงกลมวงถัดไปที่ต้องการหาอ้างอิงจากวงกลมวงที่ตรวจสอบพบ เป็นต้น การหาตำแหน่งของรูม่านตาจะได้พิกัดกึ่งกลางของวงกลมและขนาดของรัศมี โดยภาพที่ 3.14 แสดงการประมวลผลภาพเพื่อตรวจหารูม่านตา



ภาพที่ 3.14 การประมวลผลภาพเพื่อตรวจหารูม่านตา

3.2.6 การหาตำแหน่งอ้างอิงของรูม่านตา

การหาตำแหน่งอ้างอิงของรูม่านตาเพื่อใช้เป็นจุดอ้างอิงกับรูม่านตาเมื่อขยับไปในทิศทางอื่นๆ ซึ่งในการหาตำแหน่งอ้างอิงของรูม่านตาทำได้โดยการให้ผู้ทดสอบมองตรงไปข้างหน้าเพื่อให้รูม่านตาอยู่กึ่งกลางของดวงตานั่นเอง จากนั้นโปรแกรมจำทำการตรวจหาดวงตาและหาพิกัดกึ่งกลางของรูม่านตา โดยทำการประมวลผลภาพ 35 ภาพติดต่อกันและหาค่าเฉลี่ยของพิกัดกึ่งกลางของรูม่านตาแกน X, Y ที่ตรวจสอบพบ จากนั้นจะนำค่าที่ได้ไปทำการอ้างอิงเมื่อมีการเคลื่อนไหวของดวงตา ดังภาพที่ 3.15



ภาพที่ 3.15 การประมวลผลภาพเพื่อตรวจหาตำแหน่งอ้างอิงของรูม่านตา

3.2.7 การแปลง Pixel เป็นองศา

เมื่อทำการหาตำแหน่งอ้างอิงของรูม่านตาแล้วจะต้องทำการหาว่าดวงตาเคลื่อนที่ไปเพื่อมองในทิศทางใด วิธีในการคำนวณว่าดวงตาจะมองในทิศทางใดจะทำการหาตำแหน่งของรูม่านตาปัจจุบันและนำมาเทียบกับตำแหน่งของรูม่านตาอ้างอิงที่ได้ทำไว้ก่อนแล้ว

กระบวนหาทิศทางเริ่มจากการตรวจสอบตำแหน่งพิกัดของรูม่านตานำมาหาระยะห่างที่สัมพันธ์กับตำแหน่งพิกัดอ้างอิงของรูม่านตา โดยการนำพิกัด X ของรูม่านตาปัจจุบันมาลบกับพิกัด X ของรูม่านตาอ้างอิงและทำในการ Y เช่นกัน เมื่อนำมาลบกันแล้วจะได้ทิศทางการเคลื่อนที่ของรูม่านตาดังนี้

1) ค่าระยะห่างของพิกัดแกน X เทียบกับจุดอ้างอิงเป็นค่าติดลบ = ดวงตาเคลื่อนที่ในแนวแกน X ไปทางซ้าย
2) ค่าระยะห่างของพิกัดแกน X เทียบกับจุดอ้างอิงเป็นค่าติดบวก = ดวงตาเคลื่อนที่ในแนวแกน X ไปทางขวา

3) ค่าระยะห่างของพิกัดแกน Y เทียบกับจุดอ้างอิงเป็นค่าติดลบ = ดวงตาเคลื่อนที่ในแนวแกน Y ไปทางด้านบน

4) ค่าระยะห่างของพิกัดแกน Y เทียบกับจุดอ้างอิงเป็นค่าติดบวก = ดวงตาเคลื่อนที่ในแนวแกน Y ไปทางด้านล่าง

กระบวนการที่กล่าวด้านบนจะช่วยให้หาทิศทางของการเคลื่อนที่ของรูม่านตาได้ แต่ในการ Map จาก Pixel ของภาพเป็นองศาจะต้องทำตามขั้นตอนดังนี้

1) ทดสอบด้วยการกำหนดจุดอ้างอิงด้านขวาบน เช่น การมองตำแหน่งขอบจอขวาบนของ Notebook เป็นต้น และในการทดสอบจะต้องรู้ระยะห่างของผู้ทดสอบกับวัตถุที่มอง

2) ทำการมองจุดขวาบนที่กำหนดไว้แล้วให้โปรแกรมทำการตรวจหาพิกัดของรูม่านตาด้วย 35 ภาพติดต่อกันเพื่อหาค่าเฉลี่ยจากนั้นนำค่าพิกัด X, Y ที่ได้มาหาระยะห่าง Pixel กับจุดพิกัดอ้างอิงของการมองตรงไปข้างหน้า

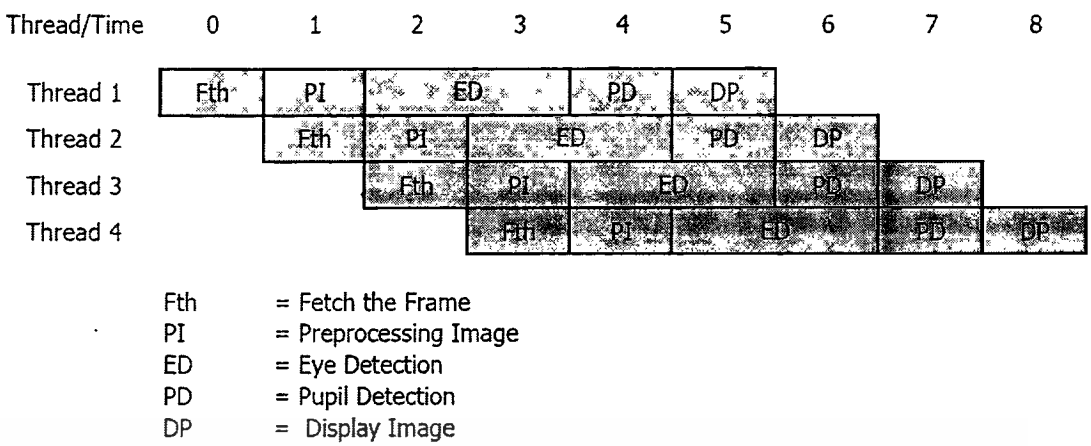
3) นำระยะห่าง Pixel ที่ได้ของแกน X, Y มาทำการคำนวณมุมจากสามเหลี่ยม Pythagoras ด้วยการใช้ Arccos เพื่อหามุมของสามเหลี่ยมในแนวแกน X และมุมของสามเหลี่ยมในแนวแกน Y

4) เมื่อได้ค่ามุมของแนวแกน X, Y แล้วให้นำมุมที่ได้มาหารด้วยระยะห่างของพิกัดกึ่งกลางและพิกัดขวาบนทำให้ได้ค่า Pixel/องศา

ค่า Pixel/องศา จะใช้ในการทำนายมุมของทิศทางการมองของดวงตาได้แต่เป็นการเทียบความสัมพันธ์แบบ linear ซึ่งอาจจะไม่ตรงหรือมีความผิดพลาดเกิดขึ้นเล็กน้อยแต่สามารถใช้ในการหามุมในการมองได้

3.2.8 การประมวลผลภาพด้วย Multithreading

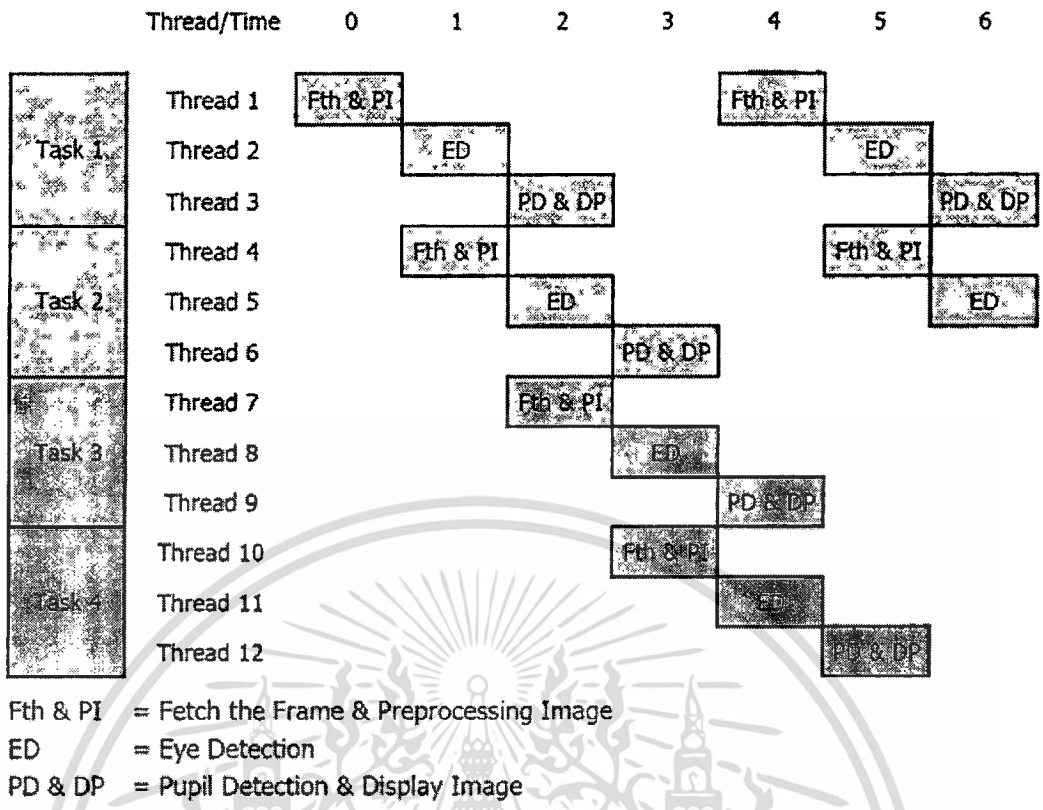
การประมวลผลภาพขนาด 1920 x 1080 ซึ่งเป็นขนาดที่มีความละเอียดสูง และการประมวลผลภาพประกอบด้วยกระบวนการหลายขั้นตอนทำให้การประมวลผลภาพค่อนข้างช้า โดยสามารถประมวลผลได้ประมาณ 5.77 ภาพต่อวินาที ดังนั้นทางกลุ่มผู้พัฒนาจึงนำวิธีการประมวลผลแบบ Parallel มาใช้ในการประมวลผลภาพ ซึ่งแนวความคิดในการประมวลผลนี้คือการใช้ Multithreading ซึ่งแบ่งการประมวลผลภาพเป็นหลาย Thread ทำงานขนานกันคล้ายกับ Pipeline



ภาพที่ 3.16 Pipeline ของการทำงาน Multithreading 4 threads

การแบ่งการประมวลผลภาพแบบ 4 Threads

ภาพที่ 3.16 แสดงการทำงาน Multithreading 4 threads ในการทำงานของ 4 Threads จะมีขั้นตอนการทำงานเหมือนกันทั้ง 4 Threads คือการดึงภาพจากกล้อง, การทำ Preprocessing image, การตรวจหาตำแหน่งของดวงตา, การตรวจหาตำแหน่งของรูม่านตา, และการแสดงผลภาพหลังการประมวลผล ดังนั้นการทำงานของ Thread จะทำงานแบบ Pipeline ซึ่งในบางขั้นตอน Thread จะต้องรอการทำงานกัน ได้แก่ ดึงภาพจากกล้องเพียง 1 ตัวจะต้องรอการเข้าถึงตัวแปรของกล้องและการเข้าถึง Picturebox ใน Graphic User Interface จะต้องรอการเข้าถึง Picturebox เพียง 1 Thread ในเวลาใดๆเท่านั้น



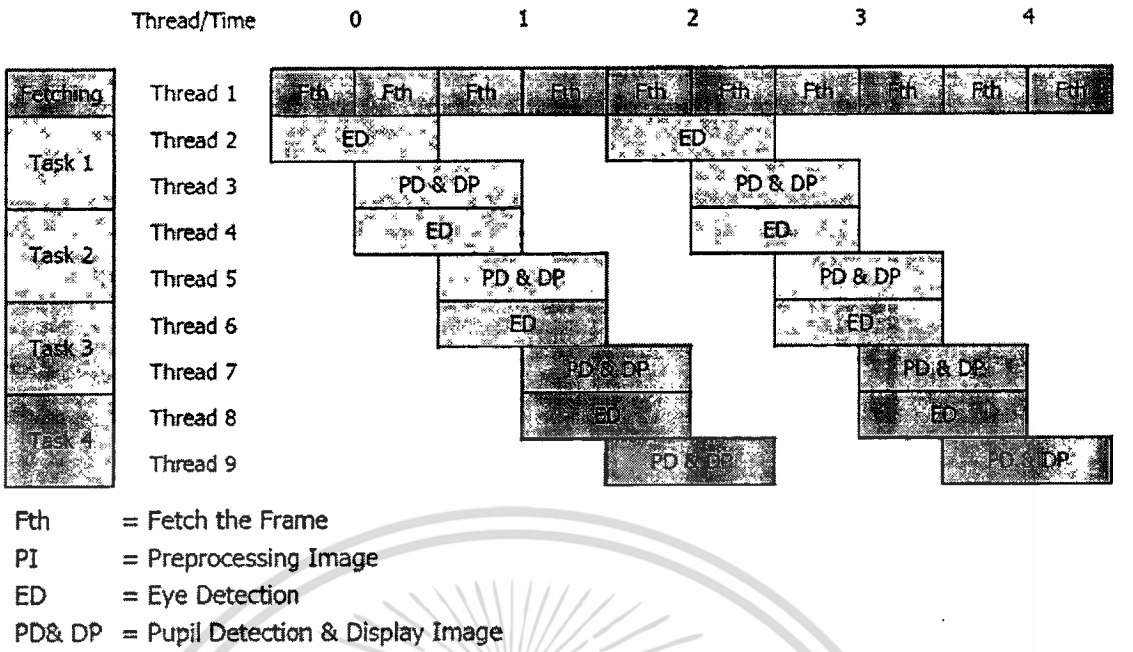
ภาพที่ 3.17 Pipeline ของการทำงาน Multithreading 12 threads

การแบ่งการประมวลผลภาพแบบ 12 Threads

ภาพที่ 3.17 แสดงการทำงาน 12 Threads การแบ่งการทำงานการประมวลผลภาพเป็น 12 Threads แต่ยังคงประมวลผลแบบ 4 Frame พร้อมๆกัน โดยการแบ่งให้เกิดงานในการประมวลผล 3 งาน ซึ่งแต่ละงานประกอบด้วย 3 Threads การแบ่งงานในการประมวลผลภาพ 1 Frame ออกเป็นงานย่อย 3 งานคือ

1) การดึงภาพจากกล้องและ Preprocessing image
2) การตรวจหาตำแหน่งของดวงตา
3) การตรวจหาตำแหน่งของรูม่านตาและการแสดงผลลัพธ์บน Picturebox

ดังนั้นการประมวลผล 4 Frame พร้อมๆกันจะให้ 12 Thread ทำงานร่วมกัน ซึ่งข้อจำกัดของการประมวลผลแบบ 12 Thread คือ การรอการเข้าถึงตัวแปรของกล้องเพียง 1 ตัว, การ Thread ก่อนหน้าที่ขึ้นต่อกันทำงานเสร็จ, และการรอการเข้าถึง Picturebox ใน Graphic User Interface



ภาพที่ 3.18 Pipeline ของการทำงาน Multithreading 9 threads

การแบ่งการประมวลผลภาพแบบ 9 Threads

การแบ่งการทำงานการประมวลผลภาพเป็น 9 Threads ดังภาพที่ 3.18 แต่ยังคงประมวลผลแบบ 4 Frame พร้อมๆกันโดยการแบ่งให้มีเพียง 1 Thread เท่านั้นที่ทำการดึงภาพจากกล้องจับดวงตา และอีก 8 Threads ให้ทำงานโดยเหมือนมีการแบ่งงานเป็น 4 งานซึ่งแต่ละงานมี 2 Thread ช่วยกันประมวลผล โดยมีการแบ่งงานเป็น 2 Thread ดังนี้

1) การตรวจหาตำแหน่งของดวงตา

2) การตรวจหาตำแหน่งของรูม่านตาและการแสดงผลลัพธ์บน Picturebox

ดังนั้นการประมวลผล 4 Frame พร้อมๆกันจะให้ 9 Thread ทำงานร่วมกัน ซึ่งข้อจำกัดของการประมวลผลแบบ 9 Thread คือ การรอการเข้าใช้งานภาพที่ดึงได้จากกล้องจับดวงตาซึ่ง Thread ที่เข้าถึงต้องมีเพียง 1 Thread ดังนั้น Thread ที่เหลือจะต้องรอคอยการเข้าถึง, การรอ Thread ก่อนหน้าที่ขึ้นต่อกันทำงานเสร็จ, และการรอการเข้าถึง Picturebox ใน Graphic User Interface

ประสิทธิภาพของการประมวลผล Multithreading แบบ 4 Thread, 12 Threads, และ 9 Threads

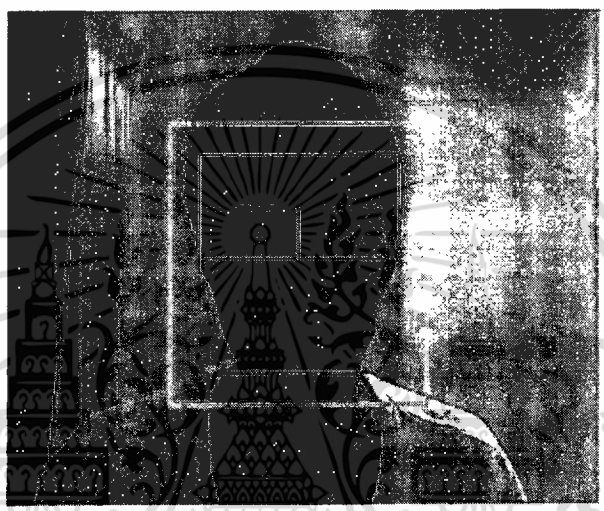
ในการประมวลผลแบบ 4 Threads พบว่าความเร็วในการประมวลผลภาพเพิ่มขึ้นสูงสุด 2.96 เท่า โดยสามารถประมวลผลได้ที่ความเร็ว 13 - 17 ภาพต่อวินาที

ในการประมวลผลแบบ 12 Threads พบว่าความเร็วในการประมวลผลภาพยังคงมีความเร็วเท่ากับการประมวลผลแบบ 4 Threads ซึ่งอาจเกิดจากการที่ทรัพยากรของเครื่องคอมพิวเตอร์ที่ใช้ประมวลผลภาพมีความแตกต่างเมื่อเทียบกับการ Fork Thread

ในการประมวลผลแบบ 9 Threads พบว่าความเร็วในการประมวลผลภาพยังคงมีความเร็วใกล้เคียงกับการประมวลผลแบบ 4 Threads

3.2.9 การ Optimization ในการประมวลผลภาพด้วย Region of Interest

การประมวลผลภาพขนาด 1920 x 1080 ซึ่งมีขนาดใหญ่ การประมวลผลภาพทั้งภาพทำให้การประมวลผลช้า การทำ Optimization จะเป็นการทำให้ขนาดของภาพที่ใช้ในการประมวลผลเฉพาะพื้นที่ที่สนใจเท่านั้น เมื่อขนาดของภาพเล็กลงทำให้สามารถประมวลผลภาพได้เร็วขึ้น ดังภาพที่ 3.19



ภาพที่ 3.19 การทำ Optimization ด้วย Region of interest

การทำ Optimization ทำตามขั้นตอนดังนี้

1) นำภาพมาประมวลผลเพื่อหาตำแหน่งของดวงตา
2) เก็บค่าตำแหน่งและขนาดของสี่เหลี่ยมที่ตรวจดวงตาพบ
3) ในการตรวจสอบดวงตาครั้งถัดไปให้ทำการนำค่าพิกัดและขนาดสี่เหลี่ยมของดวงตาที่ตรวจพบครั้งแรกมาขยายแกน X, Y เพิ่มอีก 10% ซึ่งแสดงในกรอบสีครีม
4) ทำ ROI กับภาพ raw image ในขนาดสี่เหลี่ยมที่ขยายขึ้นจากนั้นนำภาพ ROI ที่ได้มาประมวลผล preprocessing และหาตำแหน่งดวงตาต่อไป
5) หากตรวจหาตำแหน่งของดวงตาในภาพ ROI ไม่พบให้ทำในขั้นตอนที่ 1 ใหม่เพื่อหาขนาดของสี่เหลี่ยมและพิกัดอ้างอิงใหม่
6) หากตรวจหาตำแหน่งของดวงตาในภาพ ROI พบให้ทำการอ้างอิงในการขยายสี่เหลี่ยมจากตำแหน่งที่พบล่าสุด

ในการขยายของขนาดสี่เหลี่ยมขึ้นเพื่อทำนายว่าดวงตาจะยังคงอยู่ภายในกรอบที่ขยายขึ้น ซึ่งหากทำนายถูกจะทำให้ลดเวลาการประมวลผลภาพได้

ข้อจำกัดในการทำการ Optimization คือการใช้ Multithread ในการประมวลผล ซึ่ง Multithread จะประมวลผลโดยไม่เรียงลำดับและระบบปฏิบัติการจะทำการ Scheduling ให้ Thread แต่ละตัวประมวลผลเอง ดังนั้นในการจัดการเพื่อหาพิกัดและขนาดของสี่เหลี่ยมที่ตรวจพบล่าสุดทำได้ยากและการเข้าถึงตัวแปรเพื่อเขียนค่าร่วมกันจะต้องรอการเข้าถึงทีละ Thread

ทางกลุ่มผู้พัฒนาได้ทดลองใช้กระบวนการ Optimization ในการประมวลผลภาพแบบ 1 Thread พบว่าสามารถเพิ่มประสิทธิภาพในการประมวลผลให้ดีขึ้นได้ แต่ในการประมวลผลแบบ Multithread ทางกลุ่มผู้พัฒนาจึงตัดวิธีการ Optimization ออกไป

3.2.10 การติดต่อผ่านทางระบบ Network ด้วย Socket Programming

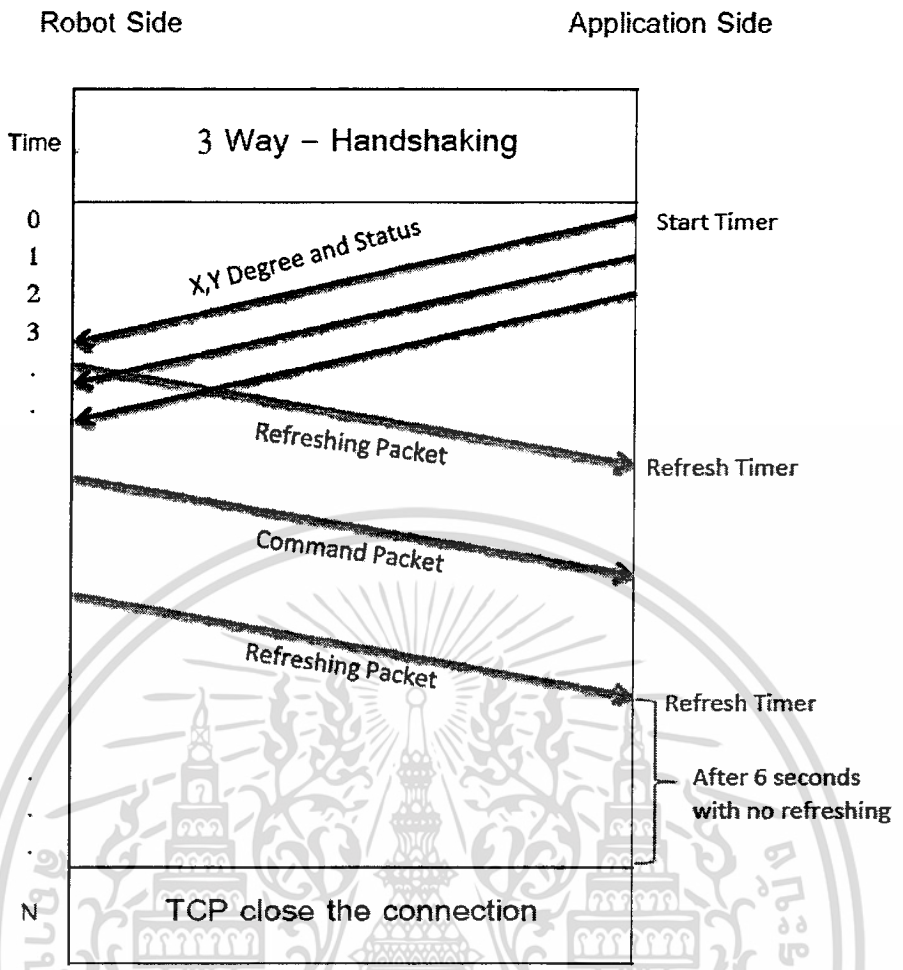
การติดต่อทาง Network ระหว่าง Application และตัวหุ่นยนต์ จะติดต่อด้วย Protocol TCP บน Transport Layer และติดต่อด้วย Protocol ที่ทางกลุ่มผู้พัฒนาคิดขึ้นมาบน Application Layer

3.2.10.1 TCP socket Programming

การติดต่อผ่าน Network พัฒนาด้วย Socket Programming ด้วยภาษา C# การพัฒนาจะประกอบด้วยการเปิด Listening IP และ Port เพื่อรอรับการติดต่อจากตัวหุ่นยนต์ ซึ่งได้ออกแบบให้สามารถรองรับการเชื่อมต่อได้มากที่สุด 16 การเชื่อมต่อหรือ 16 Socket นั่นเอง

3.2.10.2 Protocol เฉพาะบน Application Layer

ทางกลุ่มผู้พัฒนาได้คิดค้น Protocol ในการรับส่งข้อมูลและคำสั่งระหว่าง Application และตัวหุ่นยนต์ให้สามารถทำงานแบบอัตโนมัติได้ การออกแบบ Protocol ในการรับส่งข้อมูลจะเป็นดังภาพที่ 3.20



ภาพที่ 3.20 Protocol การรับส่งข้อมูลบนชั้น Application layer

การทำงานเริ่มหลังจากการเชื่อมต่อด้วย 3 Way Handshaking ดังนี้

Application จะทำการตั้งค่าเวลาเพื่อใช้ในการยกเลิกเชื่อมต่อเมื่อไม่ได้รับการติดต่อตามเวลาที่กำหนด โดยจะตั้งเวลา 6000 มิลลิวินาที และ Application จะทำการส่งข้อมูลองศาของแกน X, Y ไปยังตัวหุ่นยนต์ ในขณะเดียวกัน Application จะรอรับคำสั่งจากตัวหุ่นยนต์ด้วย Header Format ที่ออกแบบขึ้น เช่น การขอยกเลิกการเชื่อมต่อ, การ Refresh การตั้งเวลาการยกเลิกการเชื่อมต่อ, และการกำหนดความเร็วในการส่งข้อมูลองศาของ Application 2 ระดับคือ ระดับเร็วและช้า เป็นต้น

ตัวหุ่นยนต์เมื่อทำการเชื่อมต่อกับ Application แล้วจะทำการรอรับข้อมูลองศาและเมื่อได้รับข้อมูลองศาแล้วจะทำการส่งข้อมูลเพื่อไป Refresh Session ที่ Application

การออกแบบ Header Format เพื่อใช้ในการติดต่อมีดังนี้

1) การติดต่อจาก Application ไปยังตัวหุ่นยนต์จะใช้รูปแบบคือ

$x$$y$$z$

x คือข้อมูลองศาในแกน X ซึ่งที่มีขนาดความยาว 1 ตัวอักษร

y คือข้อมูลองศาในแกน Y ซึ่งที่มีขนาดความยาว 1 ตัวอักษร

z คือข้อมูลคำสั่งจาก Application ไปยังตัวหุ่นยนต์

z ประกอบด้วย

0 คือการสั่งให้หุ่นยนต์หมุนตามองศาที่กำหนด

1 คือการสั่งให้หุ่นยนต์อยู่ใน Idle process หรือไม่ขยับตามองศาที่ส่งมา ซึ่ง Application ส่งมาเพื่อให้หุ่นยนต์รับรู้ว่ายังเชื่อมต่ออยู่

2) การติดต่อจากตัวหุ่นยนต์ไปยัง Application จะมีรูปแบบการส่งดังนี้

@01@ หมายถึงการขอปิดการเชื่อมต่อ

@02@ หมายถึงการขอ Refresh Session Time เพื่อบอก Application รู้ว่ายังติดต่ออยู่

@03@ หมายถึงการขอเปลี่ยนโหมดการส่งข้อมูลให้ Application ส่งข้อมูลช้าลง

@04@ หมายถึงการขอเปลี่ยนโหมดการส่งข้อมูลให้ Application ส่งข้อมูลเร็วขึ้น

3.2.11 การรับภาพจากกล้องบนตัวหุ่นยนต์

การรับภาพจากกล้องบนตัวหุ่นยนต์จะรับภาพด้วยการส่งข้อมูลแบบไร้สายจาก Wireless Camera ที่ติดอยู่บนตัวหุ่นยนต์ ซึ่ง Wireless Camera จะส่งด้วยคลื่นความถี่ 2.4 GHz ด้วยรูปแบบ AV มาที่อุปกรณ์รับข้อมูลและอุปกรณ์รับข้อมูลจะแปลงเป็นการเชื่อมต่อแบบ USB เข้าที่คอมพิวเตอร์ ดังนั้นในการเขียนโปรแกรมเพื่อดึงภาพจากกล้อง Wireless ยังคงสามารถใช้คำสั่งเดียวกับการดึงภาพจากกล้อง Web Camera แบบมีสายได้

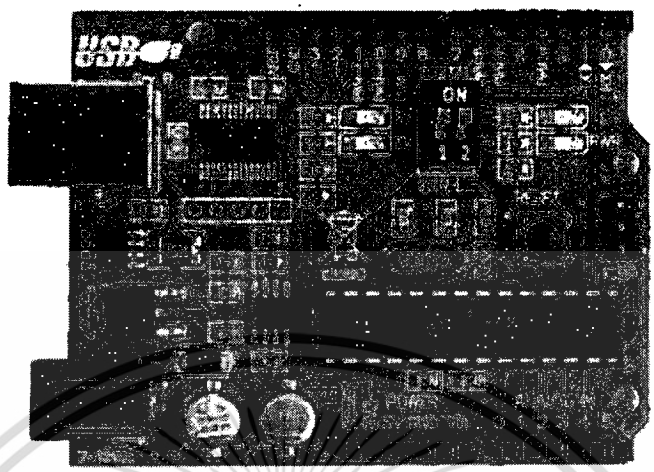
3.3 ตัวหุ่นยนต์ติดกล้องถ่ายสภาพแวดล้อม

3.3.1 Micro Controller

การพัฒนาตัวหุ่นยนต์จะใช้ Microcontroller ตระกูลATmega โดยทางกลุ่มผู้พัฒนาเลือกใช้ Micro Controller ATmega328 ซึ่งทำงานร่วมกับ Arduino Microcontroller Board ซึ่งทางกลุ่มผู้พัฒนามีเหตุผลในการเลือกใช้ดังนี้

1) เนื่องจาก Arduino ใช้ภาษา C ในการพัฒนาและมีรูปแบบของคำสั่งที่ง่ายต่อการเข้าใจและพัฒนา
2) Compiler ที่ใช้ในการแปลภาษาสามารถหาได้ง่ายและมีการปรับปรุงเสมอ
3) Arduino มี Library ในการควบคุม Servo Motor และการติดต่อ Serial Communication ทำให้ง่ายต่อการพัฒนาในภาษาระดับสูง
4) Website ของ Arduino มีข้อมูลเกี่ยวข้องกับการเขียนโปรแกรม, คำสั่ง, และ Library ที่ครบถ้วนและมีคนใช้งาน Arduino Board กันมาก

ทางกลุ่มผู้พัฒนาเลือกใช้ Arduino รุ่น Fino (Arduino Compatible) Board - Duemilanove ATmega328 ดังภาพที่ 3.21



ภาพที่ 3.21 Arduino รุ่น Fino (Arduino Compatible) Board - Duemilanove ATmega328

โดย Fino (Arduino Compatible) Board มีคุณสมบัติดังนี้

1) ATmega328 28pin DIP Microcontroller พร้อมด้วย Duemilanove bootloader
2) ATmega328 ประกอบด้วย 32KB Memory, 1KB EEPROM, 2KB SRAM
3) Quartz crystal 16 MHz
4) Standard ICSP 6pins ซึ่งพัฒนาด้วย AVR ISP
5) เชื่อมต่อกับคอมพิวเตอร์และอัพโหลดโปรแกรมผ่าน USB port
6) ATmega328 Microcontroller ทำงานในระดับความต่างศักย์ 5V
7) Fino (Arduino Compatible) Board ทำงานในระดับความต่าง 7-12V
8) สามารถจ่ายพลังงานให้ Fino (Arduino Compatible) Board ได้โดยการเชื่อมต่อ USB หรือ Power supply ภายนอกซึ่งแหล่งพลังงานจะถูกเลือกโดยอัตโนมัติ
9) Fino (Arduino Compatible) Board มี resettable polyfuse ซึ่งสามารถปกป้อง USB port ของคอมพิวเตอร์จากการช็อตหรือการจ่ายกระแสเกิน
10) Digital Input / Output are 14 pins (PWM 6 channels)
11) Analog Input is 6pins
12) DIP Switch SPST ใช้ในการปิด auto reset และ yellow LED (D13)
13) FT232RL ขับเคลื่อนการเชื่อมต่อสัญญาณกับคอมพิวเตอร์
14) ประกอบด้วย Standard Arduino Pin Header - 8 pins & 6 pins (Female) ซึ่งสามารถทับซ้อนกับอุปกรณ์ภายนอกหรือ Arduino board อื่นๆ

15) ประกอบด้วย 3.1VDC output จาก FT232RLสำหรับการพัฒนาในรูปแบบอื่นๆ

16) ขนาดของ Board 53.1mm (ความกว้าง) * 68.6mm (ความยาว) * 15.0mm (ความสูง)

3.3.2 กล้องบนตัวหุ่นยนต์

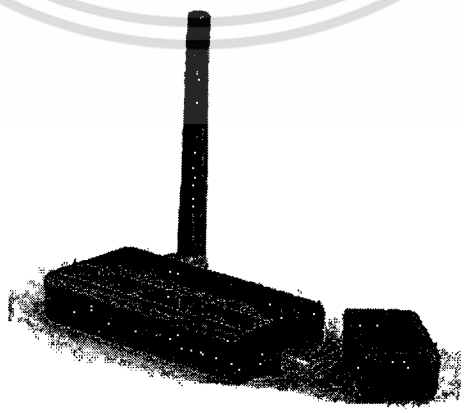
กล้องบนตัวหุ่นยนต์จะเลือกใช้กล้อง Wireless Camera ที่ส่งข้อมูลด้วยคลื่นความถี่ 2.4 GHz ในรูปแบบ AV มาที่ Adaptor ตัวรับข้อมูลแบบ 2.4GHz เพื่อแปลงเป็นการเชื่อมต่อแบบ USB เข้ากับคอมพิวเตอร์ได้

การส่งข้อมูลของกล้อง Wireless จะสามารถส่งได้ 4 Channel ของคลื่นความถี่ 2.4 GHz ซึ่งตัวรับจะต้องรับที่ Channel ของการส่งที่ถูกต้อง

Wireless Camera มีแบตเตอรี่ในตัวสามารถส่งข้อมูลได้ติดต่อกันเป็นเวลา 2 ชั่วโมง ซึ่งทางกลุ่มผู้พัฒนาคิดว่าเหมาะสมในการนำมาใช้งานเนื่องจากสามารถส่งข้อมูลได้เป็นเวลานานพอ ดังภาพที่ 3.22

USB Adaptor คืออุปกรณ์ที่ใช้ในการรับข้อมูลแบบไร้สายที่ความถี่ 2.4 GHz จากนั้นจะแปลงรูปแบบข้อมูลให้สามารถติดต่อได้แบบ USB ดังภาพที่ 3.23

ภาพที่ 3.22 กล้อง Wireless camera

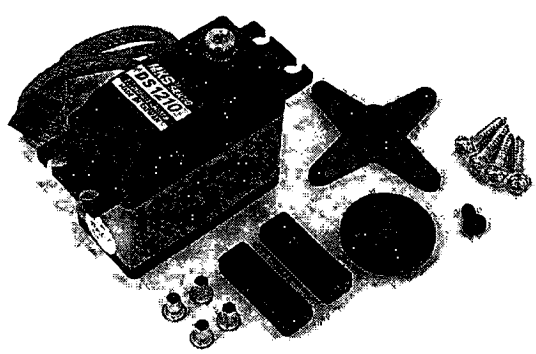


ภาพที่ 3.23 USB Adapter

3.3.3 Servo Motor

Servo motor ดังภาพที่ 3.24 ถูกนำมาใช้ในการเคลื่อนไหวของหุ่นยนต์เพื่อทำให้กล้องจับภาพสามารถขยับได้ในลักษณะครึ่งทรงกลม ซึ่งทางกลุ่มผู้พัฒนาได้เลือกใช้ MKS DS1210 Titanium Gear Standard Digital Servo ซึ่งเป็น Servo motor ที่มีความแข็งแรง ทนทาน เหมาะสำหรับนำมาใช้ทำการพัฒนา โดย MKS DS1210 Servo motor นั้นมีคุณสมบัติดังนี้

- Dead band: 0.002ms (Default)
- Control System: +Pulse Width Control
- Working frequence: 120Hz
- (RX) Required Pulse: 3.0~5.0 Volt Peak to Peak Square Wave
- Operating Voltage: 4.8~6.0 V DC Volts
- Operating Temperature Range: -10 to + 60 Degree C
- Operating Speed (4.8V): 0.15 sec/60° degrees at no load
- Operating Speed (6V): 0.12 sec/60° dagrees at no load
- Stall Torque (4.8V): 8.05 kg-cm (111.79 oz/in)
- Stall Torque (6V): 10 kg-cm (138.87 oz/in)
- 360° Modifiable: NO
- Motor Type: DC Motor
- Potentiometer Drive: Direct Drive
- Driver Type: FET
- Bearing Type: Dual Ball Bearings
- Gear Type: metal gear
- Connector Wire Length: 15.0 cm (5.9 in)
- Dimensions: 40X20X40.30 mm (1.5748X0.7874X1.58 in)
- Weight: 56 g (1.97 oz)

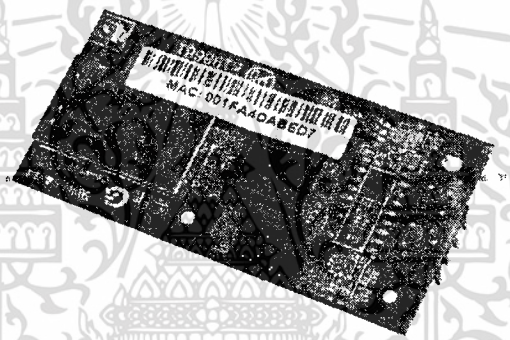


ภาพที่ 3.24 MKS DS1210 Titanium Gear Standard Digital Servo

3.3.4 WiFi Module

WiFi Module ใช้ในการติดต่อการส่งข้อมูลผ่านระบบเครือข่ายไร้สายแบบ Infrastructure Mode โดย WiFi Module จะต้องติดตั้งอยู่บนตัวหุ่นยนต์เพื่อให้หุ่นยนต์สามารถส่งรับข้อมูลกับ Application

WiFi Module ที่ทางกลุ่มผู้พัฒนาเลือกใช้คือ M03 - LVTTL UART WiFi Module ดังภาพที่ 3.25 ซึ่งจะติดต่อกับ Microcontroller ด้วย UART communication ประกอบด้วยขา Rx, Tx, Vcc,และ Gnd ซึ่งการส่งข้อมูลแบบ Serial จะมีความเร็วช้าแต่เนื่องจากการส่งข้อมูลขนาดเล็กคือคำสั่งจากตัวหุ่นยนต์ไปยัง Application ดังนั้นจึงเลือกใช้ WiFi Module แบบ UART โดยรายละเอียดของ UART WiFi Module มีดังนี้



ภาพที่ 3.25 M03 - LVTTL UART WiFi Module

M03 - LVTTL UART to Wi-Fi คือ โมดูลที่ใช้สำหรับแปลงการรับส่งข้อมูลในรูปแบบ UART เป็นการรับส่งข้อมูลในรูปแบบของ Wireless LAN หรือ Wi-Fi (IEEE 802.11b/g) ซึ่งภายในโมดูลมี Software TPC/IP Stack อยู่ทำให้ใช้งานได้ง่าย สะดวก และรวดเร็ว เหมาะสำหรับนำมาใช้กับระบบประมวลผลหรือไมโครคอนโทรลเลอร์ที่มีความเร็วในการประมวลผลต่ำ ใช้หน่วยความจำน้อย และลดเวลาในการพัฒนาลง โดยโมดูลมีการจัดสรร Software ที่เป็นส่วนของ TCP/IP Stack และการควบคุม Hardware ที่ใช้รับส่งข้อมูลผ่านทาง Wireless ไว้แล้ว ดังนั้นเพียงแค่ตั้งค่าตัวโมดูลให้สามารถเชื่อมต่อกับระบบ Network ผ่านทาง Software ที่ผู้ผลิตได้จัดเตรียมไว้ให้ก็สามารถใช้สื่อสารข้อมูลได้เลย นอกจากนี้ยังสามารถสั่งงานหรือเปลี่ยนแปลงค่าต่างๆ ผ่านทาง AT Command ด้วยการเขียนโปรแกรมจากไมโครคอนโทรลเลอร์ได้อีกด้วย

- interface
  - 2x4 Pin of interface
  - สามารถรับส่งข้อมูลผ่านทาง UART ที่ความเร็ว 1200 – 115200 bps

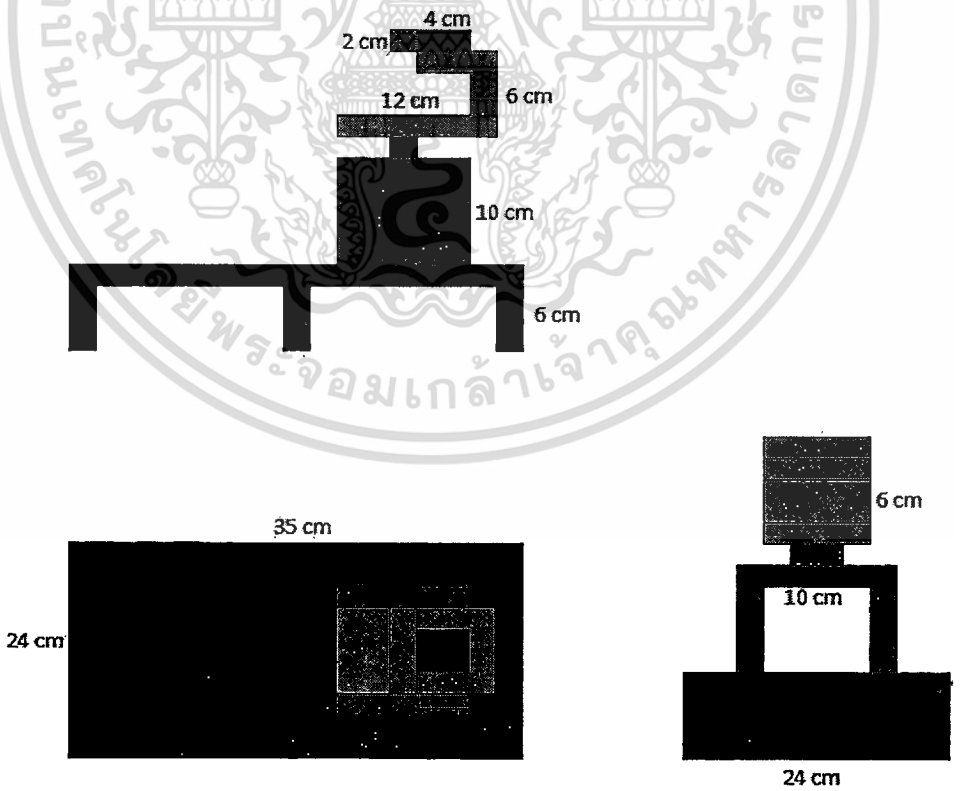
- รองรับ Hardware Flow Control RTS/CTS
- ใช้กระแสไฟฟ้า 3.1V DC

• Wireless

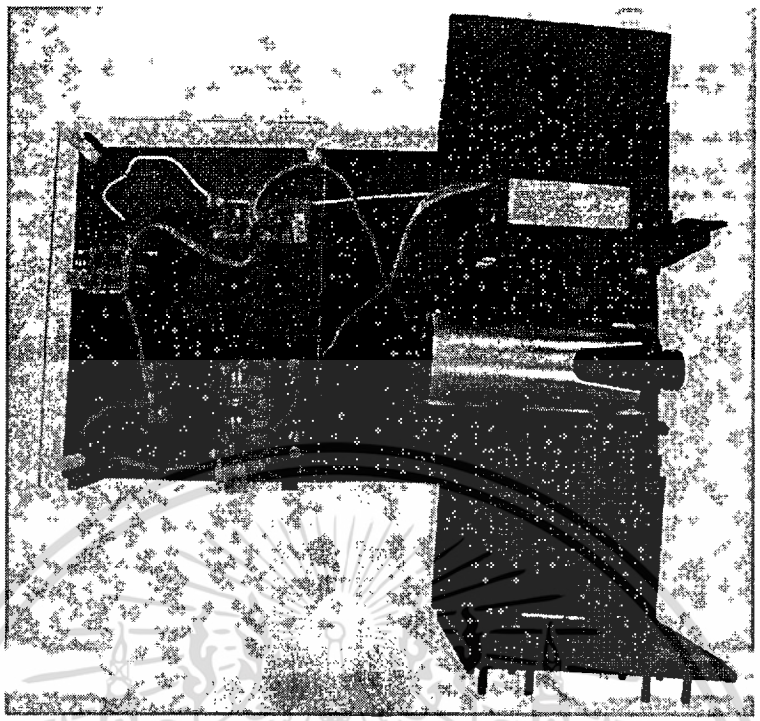
- รองรับการรับส่งข้อมูลผ่าน Wireless ตามมาตรฐาน IEEE 802.11 b/g
- ใช้ช่วงความถี่ 2.412 – 2.484 GHz
- รองรับการใช้งานแบบ Ad hoc และ Infrastructure
- รองรับมาตรฐานความปลอดภัย WEP64/ WEP128/ TKIP/ CCMP(AES)/ WEP/ WPA-PSK/ WPA2-PSK
- รองรับ Network Protocol แบบ TCP/ UDP/ ICMP/ DHCP/ DNS/ HTTP

3.3.5 การออกแบบตัวหุ่นยนต์

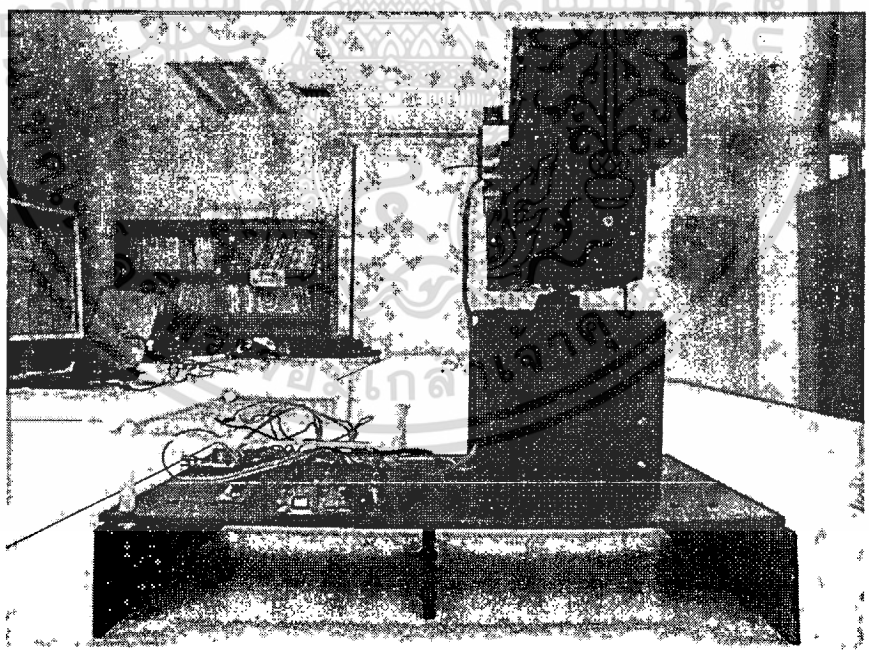
ในการออกแบบตัวหุ่นยนต์ ทางผู้พัฒนาได้ออกแบบให้ตัวหุ่นยนต์มีลักษณะเป็นแขนที่ยกสูงขึ้นจากพื้น โดยมีฐานเป็นแผ่นระนาบขนาดใหญ่ ซึ่งในส่วนของแขนหุ่นยนต์จะใช้พื้นที่ประมาณ ½ ของฐาน โดยพื้นที่ของฐานที่เหลือจะนำมาใช้ในการวาง Micro controller และ WiFi module โดยขนาดของฐานถูกออกแบบมาให้มีขนาดใหญ่เพียงพอในการรองรับน้ำหนักของแขนหุ่นยนต์ และยังสามารถรองรับการสั่นสะเทือนและแรงเหวี่ยงจากการหมุนของแขนหุ่นยนต์เพื่อให้ไม่มีการเคลื่อนที่ส่วนเกินได้ ภาพที่ 3.26 – 3.29 แสดงการออกแบบตัวหุ่นยนต์



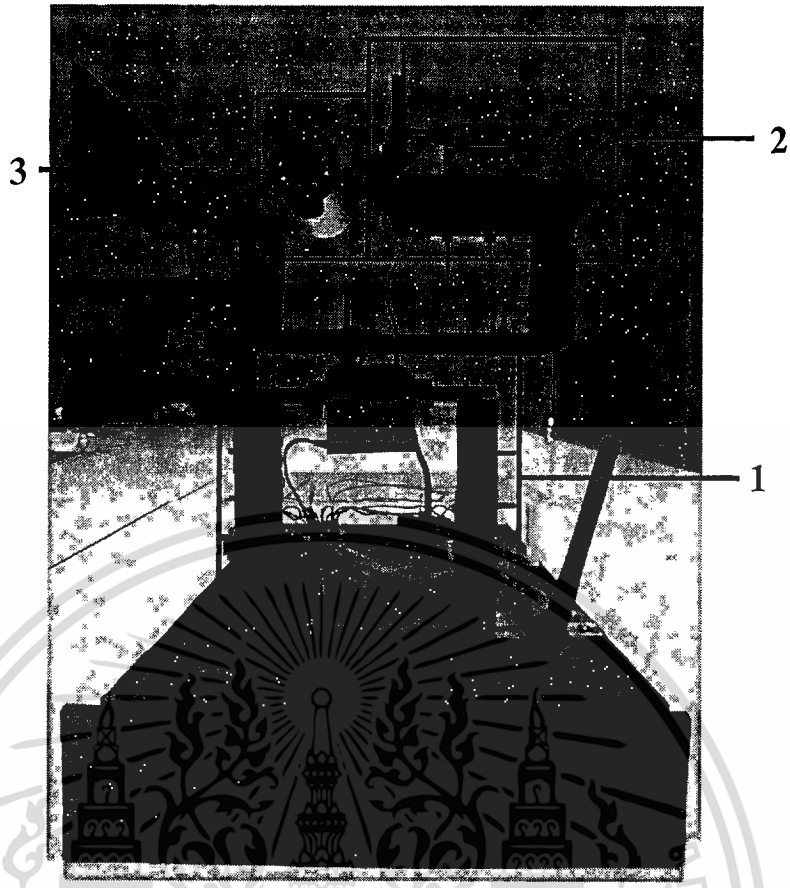
ภาพที่ 3.26 โครงสร้างของหุ่นยนต์



ภาพที่ 3.27 โครงสร้างของหุ่นยนต์ที่ออกแบบมุมมอง Top View

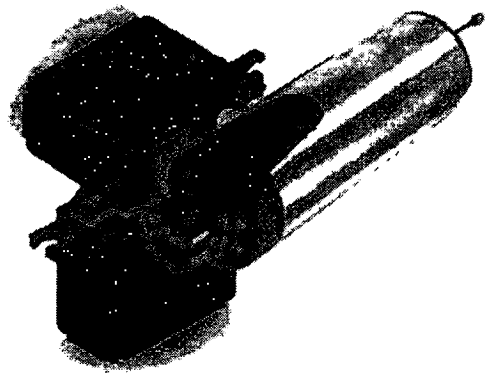


ภาพที่ 3.28 โครงสร้างของหุ่นยนต์ที่ออกแบบมุมมอง Side View



ภาพที่ 3.29 โครงสร้างของหุ่นยนต์ที่ออกแบบมุมมอง Front View

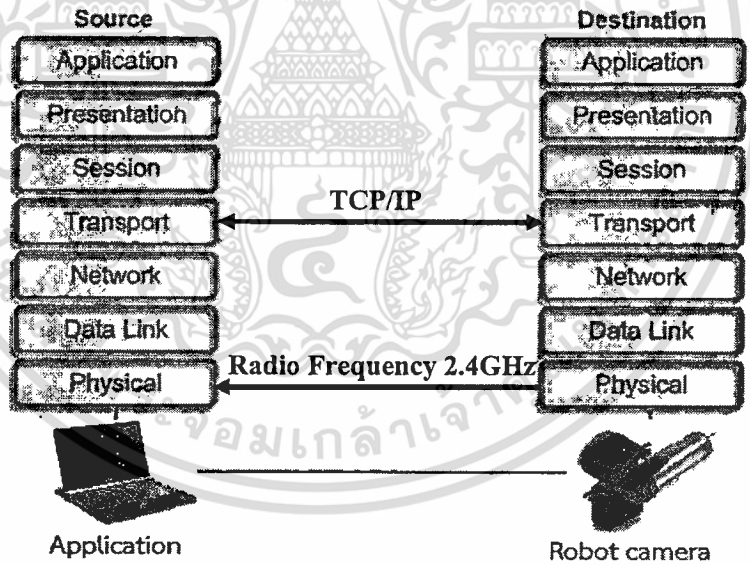
ในส่วนของการออกแบบแขนหุ่นยนต์ ทางผู้พัฒนาได้นำ Servo motor จำนวน 2 ตัว มาใช้ในการหมุนกล้อง โดยมี Servo Motor จำนวน 1 ตัวใช้ในการเคลื่อนที่แนวแกนนอนจะติดตั้งกับฐานด้านล่างซึ่งแสดงในส่วนที่ 1 ดังภาพที่ 3.29 แล Servo motor จำนวน 1 ตัวใช้ในการเคลื่อนที่ในแนวแกนตั้งจะติดตั้งกับฐานด้านบนในส่วนที่ 2 ดังภาพที่ 3.29 ซึ่งทางผู้พัฒนาได้ออกแบบให้แกนการหมุนของมอเตอร์มีตำแหน่งที่ใกล้ชิดกันที่สุด เพื่อให้การหมุนของมอเตอร์มีความคล้ายคลึงกับการเคลื่อนที่ของตามนุษย์มากที่สุด กล้อง Wireless ที่นำมาติดกับตัวหุ่นยนต์จะติดตั้งกับ Servo Motor ตัวบนเพื่อใช้หมุนในแนวแกนตั้งแสดงในส่วนที่ 3 ดังภาพที่ 3.29 และภาพที่ 3.30 แสดงการติดกล้องเข้ากับตัว Servo Motor การออกแบบตัวฐานจะออกแบบให้ยกสูงขึ้นจากพื้นเพื่อลดการสั่นสะเทือนโดยยกสูงจากพื้นประมาณ 6 เซนติเมตร



ภาพที่ 3.30 การออกแบบการหมุนของกล้องจับภาพ

3.4 การติดต่อระหว่าง Application และตัวหุ่นยนต์

การติดต่อสื่อสารเพื่อรับส่งข้อมูลองศาและคำสั่งผ่านระบบ Network จะพัฒนาด้วย WiFi Infrastructure Mode และในการส่งภาพจากตัวหุ่นยนต์จะส่งด้วย Wireless กลับมายัง Application ลักษณะของการสื่อสารข้อมูลจะเป็นดังภาพที่ 3.31



ภาพที่ 3.31 ลักษณะการสื่อสารของข้อมูลในแต่ละส่วน

ในการส่งข้อมูลระหว่าง Application กับตัวหุ่นยนต์จะเป็นการสื่อสารแบบ 2 ทิศทาง แต่การสื่อสารระหว่าง Application กับ อุปกรณ์จับภาพดวงตาจะสื่อสารแบบทิศทางเดียวจากอุปกรณ์จับภาพดวงตาไปยัง Application โดยการสื่อสารจะเป็นดังนี้

1) การติดต่อระหว่าง Application และตัวหุ่นยนต์ จะเป็นการสื่อสาร 2 Channel ซึ่ง Channel แรกจะเป็นการติดต่อด้วย WiFi Infrastructure Mode เพื่อส่งข้อมูลองศาและคำสั่งผ่าน Protocol ที่ได้คิดขึ้นซึ่งทำงานบน TCP และ Channel ที่สองจะใช้ในการส่งภาพจากตัวหุ่นยนต์กลับมายัง Application ด้วย Wireless Communication Radio Frequency 2.4 GHz แบบ AV
2) การติดต่อระหว่างกล้องจับภาพดวงตาและ Application จะติดต่อผ่านสาย USB 2.0

ในด้านของความเร็วในการสื่อสาร ความเร็วในการส่งข้อมูลของ M03 WiFi Module ซึ่งรองรับมาตรฐาน 802.11b ซึ่ง M03 WiFi Module มีความล่าช้าในการสื่อสารที่การส่งข้อมูลแบบ Serial Communication ซึ่ง WiFi Module รองรับที่ความเร็ว 1200 – 115200 bps ซึ่งการส่งข้อมูลด้วย Protocol ที่ทางกลุ่มผู้พัฒนาคิดขึ้นนั้นมีขนาดเล็กโดยมีขนาดไม่ถึง 100 Bytes ดังนั้นการส่งข้อมูลแบบ Serial Communication จึงเหมาะสมที่จะใช้ในการพัฒนา

3.5 การทำงานของระบบ

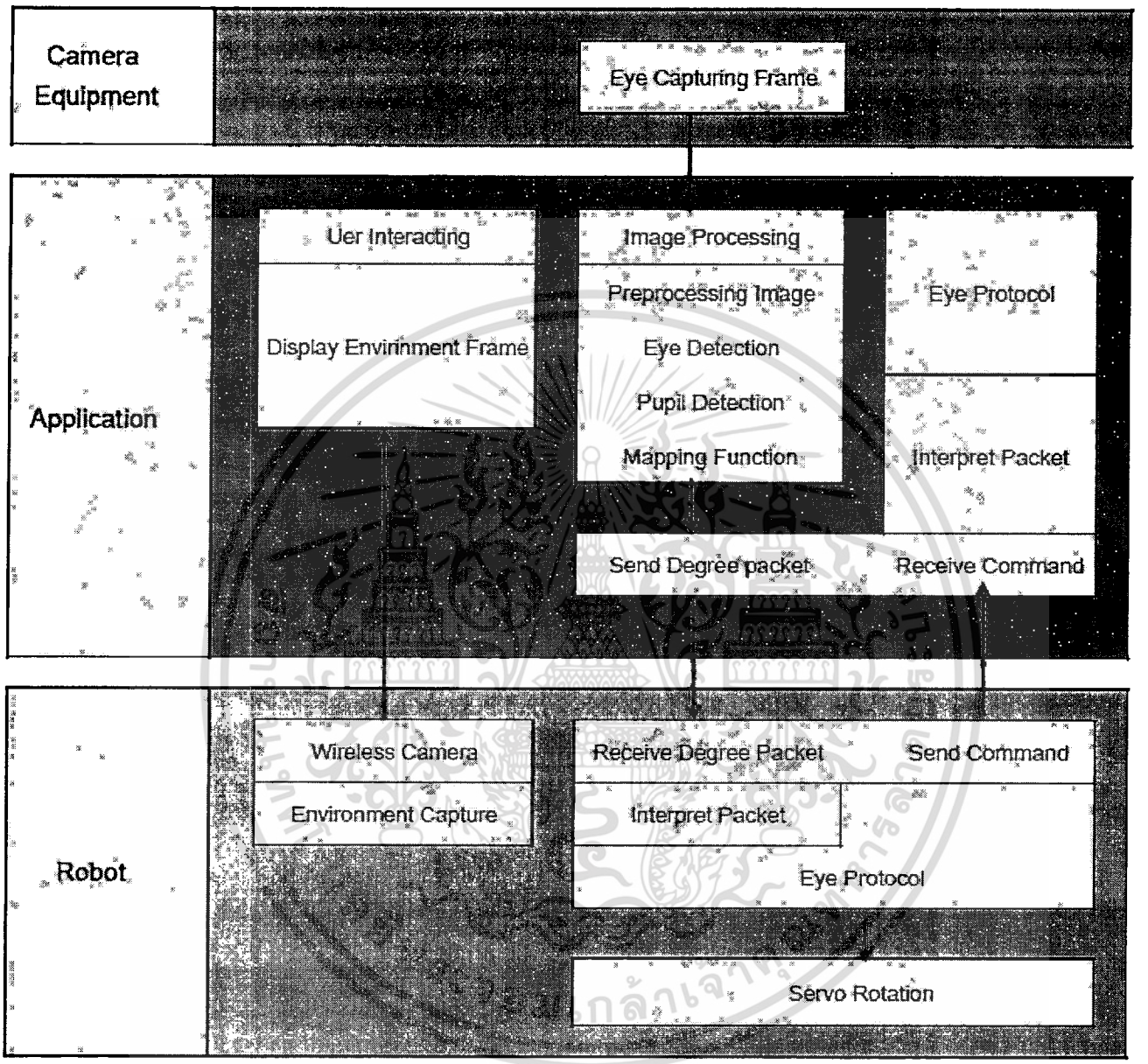
การทำงานของระบบจะแบ่งเป็น 3 ส่วนหลักคือ

1) Application บนคอมพิวเตอร์
2) ตัวหุ่นยนต์ติดตั้งกล้องถ่ายสภาพแวดล้อม
3) หมวกติดตั้งกล้องจับภาพดวงตา

โดยมีระบบการทำงานเป็นดังภาพที่ 3.32 มีการทำงานดังนี้

1) Application ทำงานหลักๆคือการประมวลผลภาพเพื่อหาตำแหน่งของดวงตาและรูม่านตา, ทำนายมุมการมองของดวงตา, รับส่งข้อมูลกับตัวหุ่นยนต์เพื่อควบคุมการทำงาน โดยมีขั้นตอนการทำงานดังนี้
   - ดึงภาพจากหมวกติดตั้งกล้องจับภาพดวงตา จากนั้นจะทำการประมวลผลภาพเพื่อหาตำแหน่งของดวงตาและรูม่านตา
   - ทำการคำนวณเพื่อหาทิศทางการมองของดวงตา
   - ติดต่อการใช้งานกับ User และแสดงผลลัพธ์ด้วย Graphic User Interface
   - ติดต่อการรับส่งข้อมูลกับตัวหุ่นยนต์ผ่านระบบ Network
2) ตัวหุ่นยนต์
   - ติดต่อรับส่งข้อมูลองศาจาก Application ผ่านระบบ Network
   - ควบคุมการหมุนของ Servo Motor ให้หมุนไปยังมุมที่ต้องการได้ในแกน X, Y
   - กล้องถ่ายสภาพแวดล้อมและส่งภาพกลับไปยัง Application ผ่าน Wireless Communication 2.4 GHz

3) หมวกติดตั้งกล้องจับภาพดวงตาจะทำการ ดึงภาพดวงตาของผู้ใช้งานและทำการส่งภาพไปยัง Application ผ่านสาย USB 2.0



ภาพที่ 3.32 การทำงานของระบบ

บทที่ 4
การทดลองและผลการทดลอง

การทดลองระบบการควบคุมกล้องจับภาพด้วยการจับการเคลื่อนที่ของดวงตาจะแบ่งการทดลองออกเป็น 4 ส่วนคือ การประมวลผลภาพบน Application, การทดลองบนตัวหุ่นยนต์, การสื่อสารระหว่างตัวหุ่นยนต์และ Application, และอุปกรณ์ติดกล้องจับภาพดวงตา

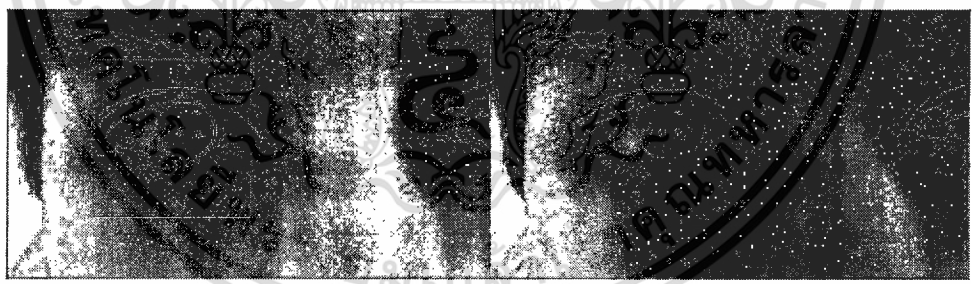
4.1 การประมวลผลภาพของ Application

4.1.1 การทำ Preprocessing Image

การทำ Preprocessing Image คือการทำให้ภาพมีความเหมาะสมในการนำไปประมวลผลเพื่อหาตำแหน่งของดวงตา ซึ่งการทำ Preprocessing Image จะส่งผลต่อความแม่นยำในการตรวจหาตำแหน่งดวงตา ซึ่งการทำ Preprocessing Image จะทำดังนี้

1) การนำภาพมาทำ Equalizehist เพื่อปรับให้ภาพมีความสว่างที่เหมาะสมและทำให้ภาพมีความคมชัดมากขึ้น
2) การนำภาพจากการทำ Equalizehist มาทำ Grammar Correct เพื่อทำให้ภาพมีความเข้มมากขึ้น

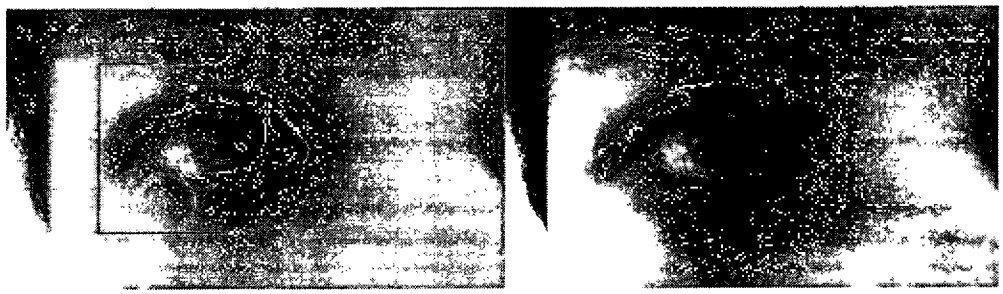
ซึ่งการทดลองจะทดลองโดยการนำภาพ Raw Image เพื่อแปลงเป็น Grey Scale เพื่อทำการตรวจจับดวงตาดังนี้



ภาพที่ 4.1 การนำภาพ Raw image มาแปลงเป็น Grey Scale

จากการทดลองนำภาพ Raw Image แปลงเป็น Grey Scale พบว่าสามารถตรวจสอบตำแหน่งของดวงตาได้ โดยแสดงในภาพที่ 4.1

ทดลองโดยการนำภาพ Grey Scale มาทำ Equalizehist และนำไปหาตำแหน่งของดวงตา โดยแสดงในภาพที่ 4.2

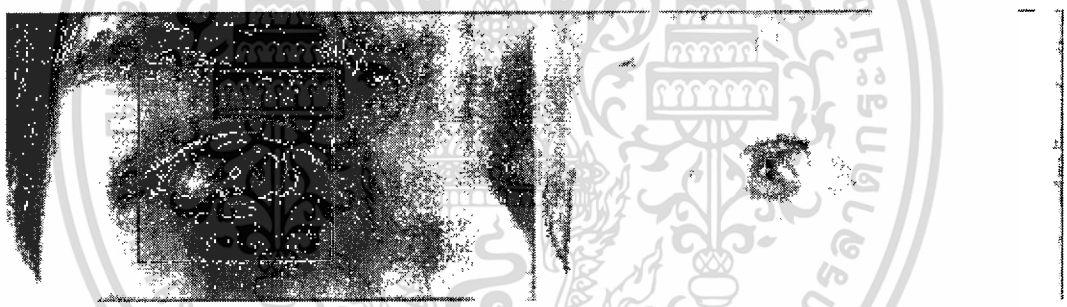


ภาพที่ 4.2 การนำภาพ Grey scale มาทำ Equalizehist

จากการทดลองนำภาพ Grey Scale มาทำ Equalizehist พบว่าทำให้ภาพมีความเข้มขึ้นมากทำให้ตรวจหาตำแหน่งของดวงตามีความแม่นยำน้อยลง

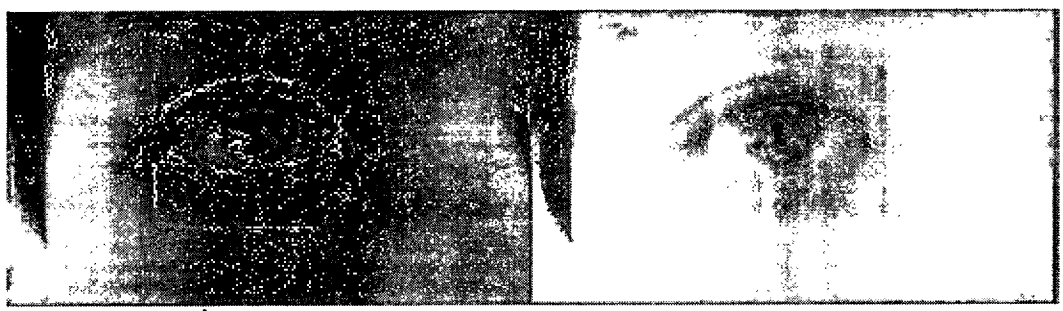
ทดลองโดยการนำภาพ Equalizehist มาทำการปรับ Gamma Correct และนำไปหาตำแหน่งของดวงตา ภาพที่ 4.3 – 4.12 แสดงการประมวลผลด้วย Gamma Correct ที่แตกต่างกัน

- Gamma Correct = 0.1d พบว่าได้ภาพที่มีระดับความเข้มน้อยเห็นดวงตาชัดเจนทำให้หาตำแหน่งดวงตาได้ดี



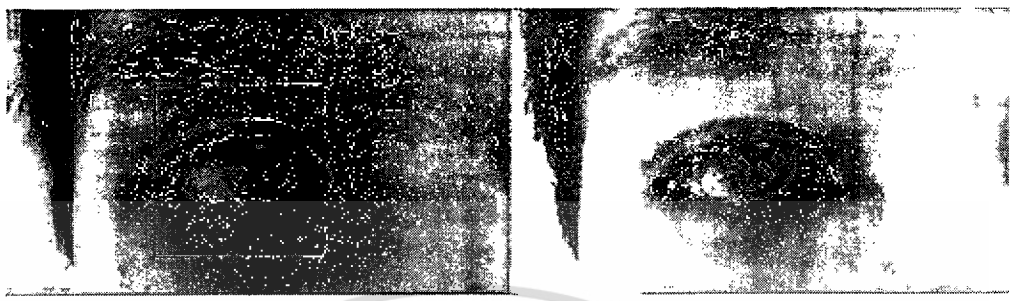
ภาพที่ 4.3 การนำภาพ Eqalizehist มาปรับให้ Gamma correct = 0.1d

- Gamma Correct = 0.2d พบว่าได้ภาพที่มีความเข้มน้อยเห็นดวงตาชัดเจนทำให้หาตำแหน่งดวงตาได้ดี



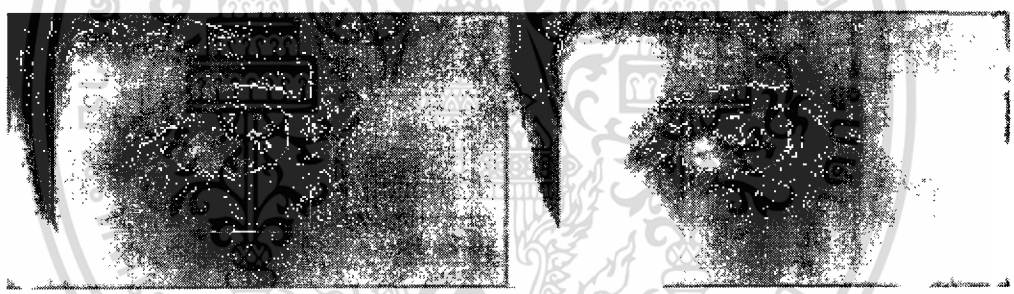
ภาพที่ 4.4 การนำภาพ Eqalizehist มาปรับให้ Gamma correct = 0.2d

- Gamma Correct = 0.3d พบว่าได้ภาพที่มีระดับความเข้มน้อยเห็นดวงตาชัดเจนทำให้หาตำแหน่งดวงตาได้ดี



ภาพที่ 4.5 การนำภาพ Eqalizehist มาปรับให้ Gamma correct = 0.3d

- Gamma Correct = 0.4d พบว่าได้ภาพที่มีระดับความเข้มปานกลางเห็นดวงตาชัดเจนทำให้หาตำแหน่งดวงตาได้ดี

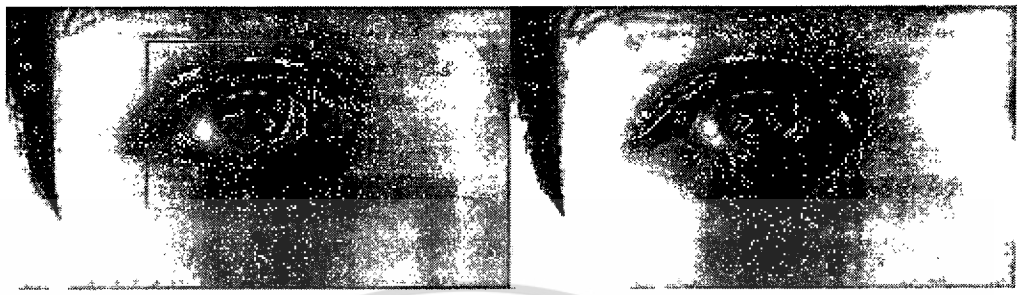


ภาพที่ 4.6 การนำภาพ Eqalizehist มาปรับให้ Gamma correct = 0.4d

- Gamma Correct = 0.5d พบว่าได้ภาพที่มีระดับความเข้มปานกลางเห็นดวงตาชัดเจนทำให้หาตำแหน่งดวงตาได้ดี

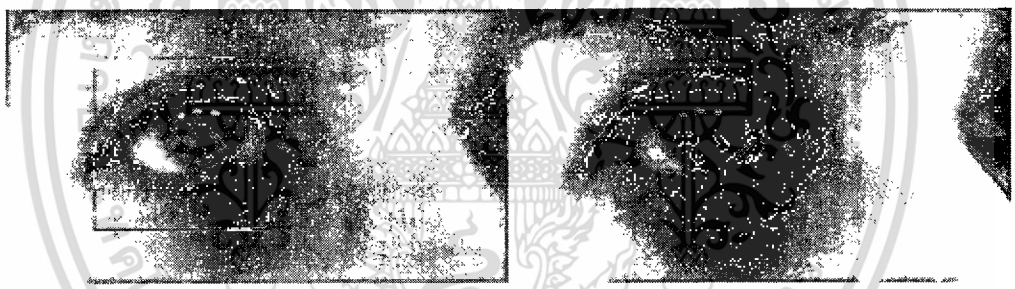
ภาพที่ 4.7 การนำภาพ Eqalizehist มาปรับให้ Gamma correct = 0.5d

- Gamma Correct = 0.6d พบว่าได้ภาพที่มีระดับความเข้มค่อนข้างมากเห็นดวงตามีระดับความลึกใกล้เคียงกับตำแหน่งอื่นๆทำให้หาตำแหน่งดวงตาได้ค่อนข้างยาก



ภาพที่ 4.8 การนำภาพ Eqalizehist มาปรับให้ Gamma correct = 0.6d

- Gamma Correct = 0.7d พบว่าได้ภาพที่มีระดับความเข้มค่อนมากเห็นดวงตามีระดับความลึกใกล้เคียงกับตำแหน่งอื่นๆทำให้หาตำแหน่งดวงตาได้ค่อนข้างยาก

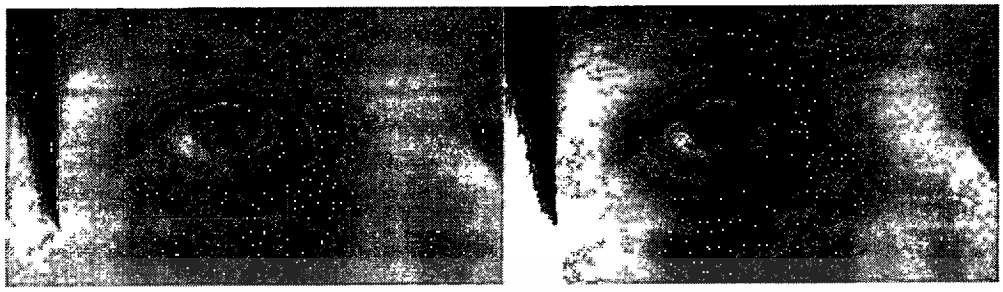


ภาพที่ 4.9 การนำภาพ Eqalizehist มาปรับให้ Gamma correct = 0.7d

- Gamma Correct = 0.8d พบว่าได้ภาพที่มีระดับความเข้มมากเห็นดวงตามีระดับความลึกใกล้เคียงกับตำแหน่งอื่นๆทำให้หาตำแหน่งดวงตาได้ยาก

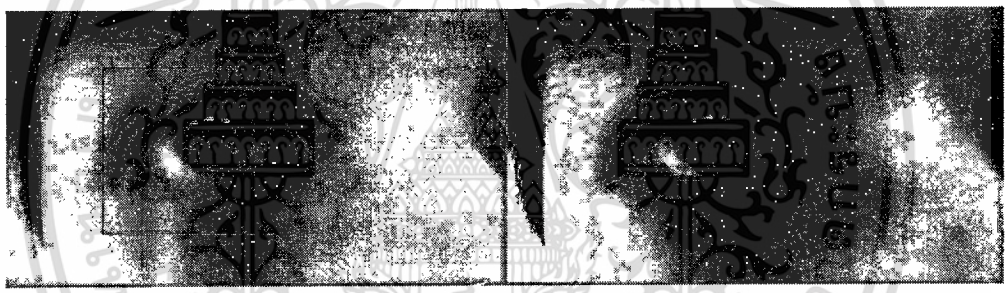
ภาพที่ 4.10 การนำภาพ Eqalizehist มาปรับให้ Gamma correct = 0.8d

- Gamma Correct = 0.9d พบว่าได้ภาพที่มีระดับความเข้มมากเห็นดวงตามีระดับความลึกใกล้เคียงกับตำแหน่งอื่นๆทำให้หาตำแหน่งดวงตาได้ยาก



ภาพที่ 4.11 การนำภาพ Eqalizehist มาปรับให้ Gamma correct = 0.9d

- Gamma Correct = 1.0d พบว่าได้ภาพที่มีระดับความเข้มมากเห็นดวงตามีระดับความลึกใกล้เคียงกับตำแหน่งอื่นๆทำให้หาตำแหน่งดวงตาได้ยาก



ภาพที่ 4.12 การนำภาพ Eqalizehist มาปรับให้ Gamma correct = 1.0d

จากการนำภาพ Equalizehist มาทำ Gamma Correct ด้วยระดับต่างๆทำให้ผลลัพธ์แตกต่างกันสรุปได้ดังนี้

- Gamma Correct ตั้งแต่ 0.1d – 0.5d ให้ผลลัพธ์ในการตรวจสอบดวงตาที่ดี มีความแม่นยำสูง
- Gamma Correct ตั้งแต่ 0.6d - 1.0d ให้ผลลัพธ์ในการตรวจสอบดวงตาที่มีความแม่นยำน้อยลงเนื่องจากมีความเข้มของภาพเกินความเหมาะสมในการตรวจหาดวงตา

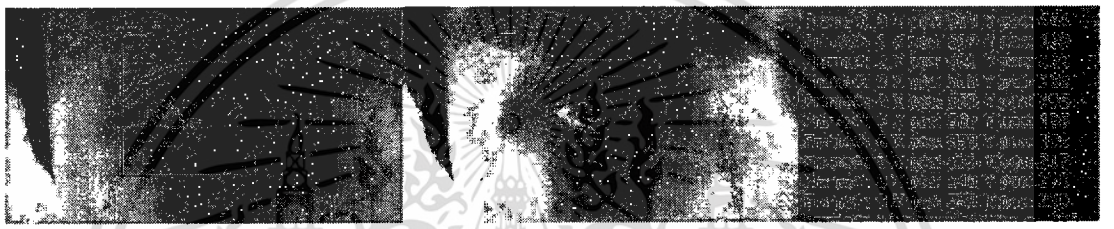
จากการทดลองพบว่าค่า Gamma Correct = 0.19d เป็นค่าที่เหมาะสมในการนำมาใช้งาน

4.1.2 การตรวจจับตำแหน่งของดวงตา

การตรวจจับตำแหน่งของดวงตาโดยการใช้ Haar Face Detection โดยผลลัพธ์ที่ได้คือตำแหน่งซ้ายบนของดวงตาที่พบและขนาดของดวงตาที่พบ

การตรวจจับตำแหน่งของดวงตาจะทำโดยการใช้วิธีหา Biggest Object ซึ่งจะตรวจจับขนาดของวัตถุที่เราสนใจขนาดใหญ่ที่สุดในภาพและให้ผลลัพธ์ของพิกัดและขนาดเพียงวัตถุเดียว

ทดลองโดยการมองตรงไปข้างหน้าที่ตำแหน่ง X, Y ที่ (0, 0) และตรวจหาตำแหน่งของดวงตาจำนวน 35 ภาพซึ่งได้ผลลัพธ์ดังภาพที่ 4.13 ซึ่งผลลัพธ์ดังกล่าวคือตำแหน่ง Pixel ของภาพ



ภาพที่ 4.13 การทดลองมองตรงไปข้างหน้าและตรวจหาตำแหน่งของดวงตาจำนวน 35 ภาพ

ตัวอย่าง 4.1 ข้อมูลพิกัด X และ Y จากการตรวจจับเป็นจำนวน 35 ภาพ

Frame#: 1 X pos: 619 Y pos: 169

Frame#: 2 X pos: 621 Y pos: 168

Frame#: 3 X pos: 618 Y pos: 168

Frame#: 4 X pos: 589 Y pos: 195

Frame#: 5 X pos: 562 Y pos: 195

Frame#: 6 X pos: 559 Y pos: 199

Frame#: 7 X pos: 533 Y pos: 217

Frame#: 8 X pos: 548 Y pos: 220

Frame#: 9 X pos: 559 Y pos: 230

Frame#: 10 X pos: 540 Y pos: 225

Frame#: 11 X pos: 542 Y pos: 227

Frame#: 12 X pos: 538 Y pos: 225

Frame#: 13 X pos: 560 Y pos: 218

Frame#: 14 X pos: 538 Y pos: 217

ตัวอย่าง 4.1 ข้อมูลพิกัด X และ Y จากการตรวจจับเป็นจำนวน 35 ภาพ (ต่อ)

Frame#: 15 X pos: 557 Y pos: 214

Frame#: 16 X pos: 547 Y pos: 212

Frame#: 17 X pos: 557 Y pos: 212

Frame#: 18 X pos: 560 Y pos: 205

Frame#: 19 X pos: 559 Y pos: 203

Frame#: 20 X pos: 556 Y pos: 199

Frame#: 21 X pos: 562 Y pos: 199

Frame#: 22 X pos: 561 Y pos: 205

Frame#: 23 X pos: 568 Y pos: 200

Frame#: 24 X pos: 567 Y pos: 196

Frame#: 25 X pos: 568 Y pos: 182

Frame#: 26 X pos: 579 Y pos: 181

Frame#: 27 X pos: 593 Y pos: 158

Frame#: 28 X pos: 582 Y pos: 163

Frame#: 29 X pos: 585 Y pos: 161

Frame#: 30 X pos: 575 Y pos: 158

Frame#: 31 X pos: 566 Y pos: 169

Frame#: 32 X pos: 563 Y pos: 182

Frame#: 33 X pos: 563 Y pos: 172

Frame#: 34 X pos: 562 Y pos: 175

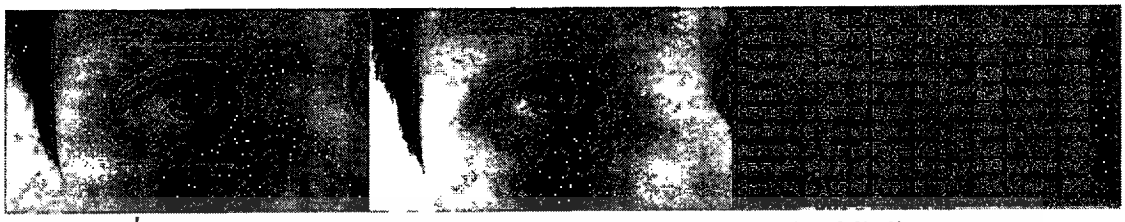
Frame#: 35 X pos: 585 Y pos: 163

จากการทดลองตรวจหาตำแหน่งดวงตาด้วย Gamma Correct = 0.5d พบว่าสามารถตรวจหาตำแหน่งของดวงตาในรูปภาพมีค่าเฉลี่ยอยู่ที่ประมาณของพิกัดแกน X = 550, พิกัดแกน Y =180 ซึ่งมีความแตกต่างของพิกัดในแกน X ประมาณ 100 จุด และมีความแตกต่างในแกน Y ประมาณ 50 ซึ่งอาจเกิดจากการที่กล้องสั่นไม่คงที่ หากกล้องอยู่นิ่งจะทำให้การตรวจสอบมีความผิดพลาดน้อยลง

4.1.3 ขนาดของดวงตาที่ตรวจพบ

การตรวจจับดวงตาด้วย Biggest Object จะให้ผลลัพธ์คือขนาดของดวงตาที่ตรวจพบ ซึ่งในการตรวจพบแต่ละครั้งอาจมีขนาดที่แตกต่างกัน

ทดลองโดยการตรวจจับดวงตาจำนวน 35 ภาพโดยมองตรงไปข้างหน้าที่ตำแหน่ง X, Y ที่ (0, 0) ดังแสดงในภาพที่ 4.14 โดยให้กล้องจับดวงตาอยู่ในระยะเดียวกันเพื่อดูขนาดของดวงตาที่ตรวจจับได้



ภาพที่ 4.14 การทดลองตรวจจับดวงตาจำนวน 35 ภาพติดต่อกันโดยให้กล้องจับดวงตาอยู่ในระยะเดียวกัน

ตัวอย่าง 4.2 ข้อมูลขนาดความสูงและความกว้างของดวงตาจากการตรวจจับจำนวน 35 ภาพ

Frame#: 1 Size of Eye:: Height: 629 Width: 629

Frame#: 2 Size of Eye:: Height: 629 Width: 629

Frame#: 3 Size of Eye:: Height: 640 Width: 640

Frame#: 4 Size of Eye:: Height: 628 Width: 628

Frame#: 5 Size of Eye:: Height: 634 Width: 634

Frame#: 6 Size of Eye:: Height: 629 Width: 629

Frame#: 7 Size of Eye:: Height: 633 Width: 633

Frame#: 8 Size of Eye:: Height: 623 Width: 623

Frame#: 9 Size of Eye:: Height: 628 Width: 628

Frame#: 10 Size of Eye:: Height: 635 Width: 635

Frame#: 11 Size of Eye:: Height: 628 Width: 628

Frame#: 12 Size of Eye:: Height: 641 Width: 641

Frame#: 13 Size of Eye:: Height: 637 Width: 637

Frame#: 14 Size of Eye:: Height: 638 Width: 638

Frame#: 15 Size of Eye:: Height: 644 Width: 644

Frame#: 16 Size of Eye:: Height: 635 Width: 635

Frame#: 17 Size of Eye:: Height: 638 Width: 638

Frame#: 18 Size of Eye:: Height: 635 Width: 635

Frame#: 19 Size of Eye:: Height: 643 Width: 643

Frame#: 20 Size of Eye:: Height: 631 Width: 631

ตัวอย่าง 4.2 ข้อมูลขนาดความสูงและความกว้างของดวงตาจากการตรวจจับจำนวน 35 ภาพ (ต่อ)

Frame#: 21 Size of Eye:: Height: 630 Width: 630

Frame#: 22 Size of Eye:: Height: 629 Width: 629

Frame#: 23 Size of Eye:: Height: 631 Width: 631

Frame#: 24 Size of Eye:: Height: 629 Width: 629

Frame#: 25 Size of Eye:: Height: 635 Width: 635

Frame#: 26 Size of Eye:: Height: 632 Width: 632

Frame#: 27 Size of Eye:: Height: 632 Width: 632

Frame#: 28 Size of Eye:: Height: 639 Width: 639

Frame#: 29 Size of Eye:: Height: 639 Width: 639

Frame#: 30 Size of Eye:: Height: 628 Width: 628

Frame#: 31 Size of Eye:: Height: 633 Width: 633

Frame#: 32 Size of Eye:: Height: 630 Width: 630

Frame#: 33 Size of Eye:: Height: 630 Width: 630

Frame#: 34 Size of Eye:: Height: 631 Width: 631

Frame#: 35 Size of Eye:: Height: 639 Width: 639

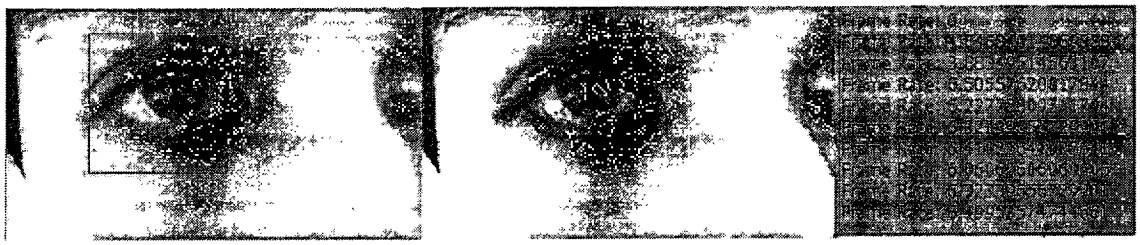
จากการทดสอบเพื่อหาขนาดของดวงตาที่ตรวจพบจำนวน 35 ภาพซึ่งได้ขนาดของดวงตาที่ได้มีความยาวและความกว้างเท่ากันเสมอ และมีค่าเฉลี่ยประมาณ 625 x 625 ดังนั้นทางกลุ่มผู้พัฒนาจะปรับให้ขนาดของดวงตาที่ตรวจพบมีขนาดประมาณ 600 x 600 ซึ่งใช้ในการอ้างอิงตำแหน่งกับตำแหน่งอ้างอิงของดวงตาได้

4.1.4 ความเร็วในการตรวจจับดวงตา

การตรวจจับดวงตาซึ่งมีภาพขนาดแตกต่างกันเป็น Input จะส่งผลต่อความละเอียดในการตรวจดังนั้นการตรวจหาตำแหน่งของดวงตาในขนาดภาพที่แตกต่างกันจะส่งผลต่อความเร็วในการตรวจหาดวงตา

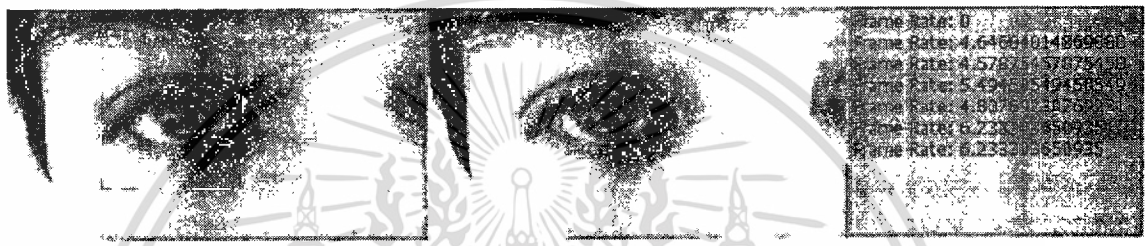
ทดลองโดยการนำภาพที่มีขนาดแตกต่างกันและจับเวลาในการใช้ในการประมวลผลภาพดังนี้ ซึ่งภาพที่ 4.15 – 4.18 จะแสดงรูปประมวลผลด้วยขนาดแตกต่างกัน

- ภาพขนาด 640 x 480



ภาพที่ 4.15 การทดลองความเร็วในการตรวจจับดวงตาโดยใช้ภาพขนาด 640 x 480

- ภาพขนาด 1080 x 720



ภาพที่ 4.16 การทดลองความเร็วในการตรวจจับดวงตาโดยใช้ภาพขนาด 1080 x 720

- ภาพขนาด 1280 x 960

ภาพที่ 4.17 การทดลองความเร็วในการตรวจจับดวงตาโดยใช้ภาพขนาด 1280 x 960

- ภาพขนาด 1920 x 1080

ภาพที่ 4.18 การทดลองความเร็วในการตรวจจับดวงตาโดยใช้ภาพขนาด 1920 x 1080

จากการทดลองการดึงภาพจากกล้องที่มีขนาดแตกต่างกันเพื่อนำมาประมวลผล สรุปดังนี้

- ภาพที่มีขนาด 640 x 480 และ 1080 x 720 มีความเร็วในการประมวลผลที่แตกต่างกันเล็กน้อยซึ่งมีความเร็วอยู่ที่ประมาณ 6.4 Frame/s และ 6.2 Frame/s
- ภาพที่มีขนาด 1280 x 960 มีความเร็วในการประมวลผลที่ประมาณ 5.3 Frame/s ซึ่งมีความเล็กลดลงจากการดึงภาพด้วยขนาด 640 x 480 และ 1080 x 720 อย่างเห็นได้ชัด
- ภาพที่มีขนาด 1920 x1080 มีความเร็วในการประมวลผลที่ 5 Frame/s ซึ่งถือว่ามีความเร็วในการประมวลผลที่ช้าและสามารถพัฒนาได้โดยการใช้ Multithread ช่วยในการประมวลผล

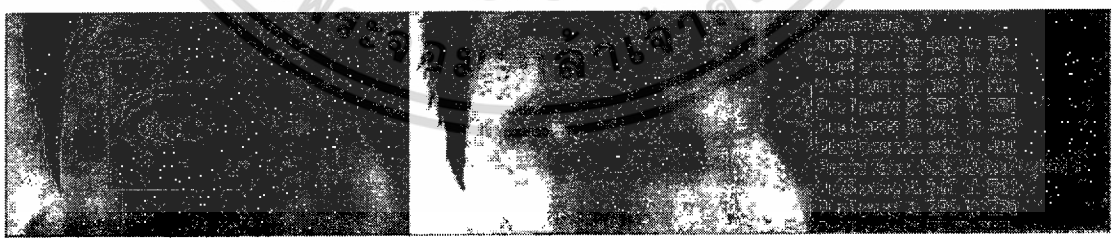
ในการใช้งานทางกลุ่มผู้พัฒนาเลือกการดึงภาพขนาด 1920 x 1080 ซึ่งสามารถทำให้ขนาดของดวงตาที่ตรวจพบมีขนาดประมาณ 600 x 600 ซึ่งมีความละเอียดของ Pixel สูงทำให้สามารถตรวจมุมได้ละเอียดยิ่งขึ้น

4.1.5 การตรวจจับรูม่านตาด้วย HoughCircle

การตรวจจับรูม่านตาสามารถทำได้โดยการนำภาพที่พบดวงตานำไปตรวจหาวงกลมภายในภาพด้วย HoughCircle ซึ่งเป็นการตรวจขอบของวัตถุที่ต้องการหาและใช้ค่า Threshold ของสีในการตัดสินใจซึ่ง HoughCircle ประกอบด้วย Parameter ที่สำคัญคือ Canny Threshold, Accumulator Threshold, Minimum Distance, Minimum Radius, และMaximum Radius

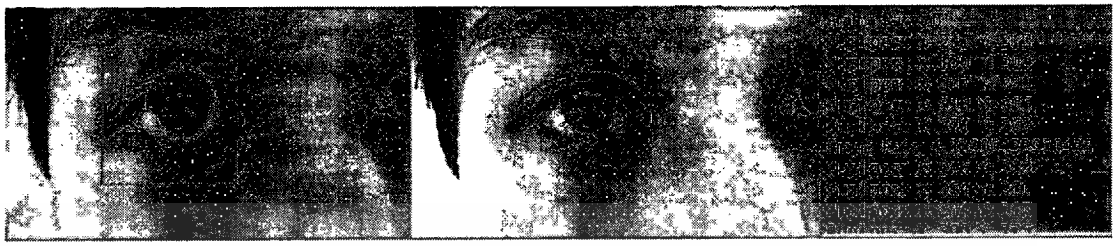
ทดลองโดยการปรับค่าของ Canny Threshold, Accumulator Threshold, Minimum Radius, และMaximum Radius ดังภาพที่ 4.19 – 4.24 ดังนี้

- ทดลองปรับค่า Minimum Radius และ Maximum Radius



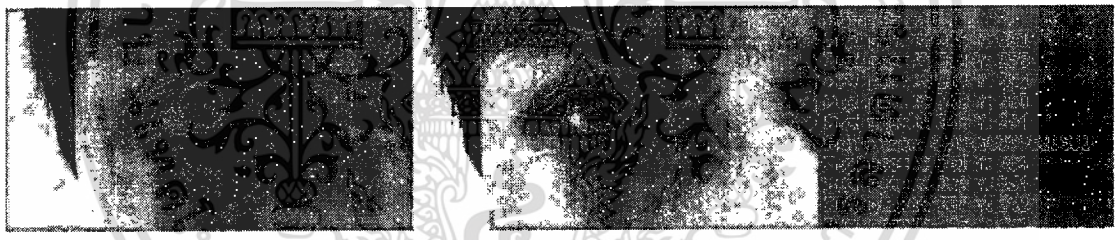
ภาพที่ 4.19 การทดลองการตรวจจับรูม่านตาโดยการปรับค่า Minimum Radius และ Maximum Radius มีค่าตั้งแต่ 40 – 150 pixel

จากการทดลองปรับค่า Minimum Radius และ Maximum Radius มีค่าตั้งแต่ 40 – 150 Pixel ทำให้มีความผิดพลาดในการหารูม่านตาเนื่องจากวงกลมวงเล็กสามารถตรวจพบได้ซึ่งวงกลมวงเล็กจะปรากฏอยู่ในรูม่านตาเป็นจำนวนมาก



ภาพที่ 4.20 การทดลองการตรวจจับรูม่านตาโดยการปรับค่า Minimum Radius และ Maximum Radius มีค่าตั้งแต่ 120-180 pixel

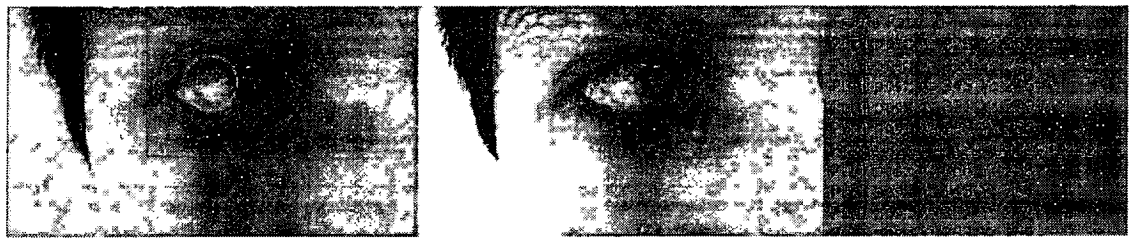
จากการทดลองปรับค่า Minimum Radius และ Maximum Radius มีค่าตั้งแต่ 120 – 180 Pixel ทำให้มีโอกาสเกิดความผิดพลาดในการหารูม่านตาจากการตรวจพบวงกลมวงใหญ่เกินไปซึ่งทำให้ตำแหน่งกึ่งกลางของวงกลมที่ตรวจพบไม่เท่ากับกึ่งกลางของรูม่านตาจริง



ภาพที่ 4.21 การทดลองการตรวจจับรูม่านตาโดยการปรับค่า Minimum Radius และ Maximum Radius มีค่าตั้งแต่ 110-140 pixel

จากการทดลองปรับค่า Minimum Radius และ Maximum Radius มีค่าตั้งแต่ 110 – 140 Pixel ทำให้มีความผิดพลาดในการหารูม่านตาน้อยที่สุดเนื่องจากมีความใกล้เคียงกับ Radius ของรูม่านตาในภาพ ดังนั้นการตรวจหารูม่านตาควรจะกำหนดค่า Radius ของรูม่านตาให้เหมาะสม ซึ่งค่าจะอยู่ในช่วงประมาณ 110 – 140 และหากมีการเปลี่ยนแปลงให้กล้องอยู่ใกล้หรือไกลขึ้นจะต้องทำการปรับค่า Minimum Radius และ Maximum Radius ให้เหมาะสมใหม่

- ทดลองปรับค่า Canny Threshold และ Accumulator ซึ่งจะใช้ในการตรวจหาขอบของวงกลมจากระดับ Grey Scale ที่กำหนดThreshold

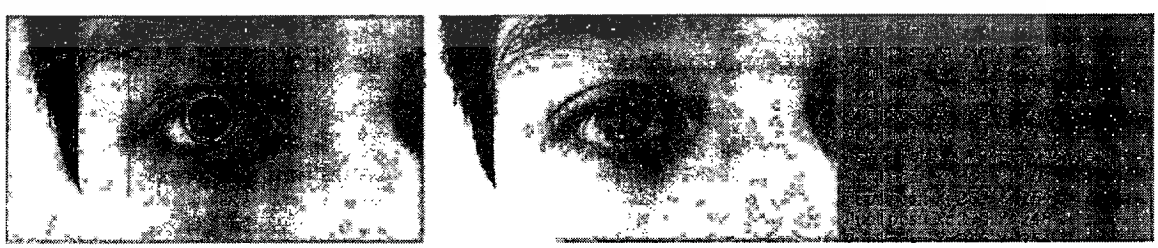


ภาพที่ 4.22 การทดลองการตรวจจับรูม่านตาโดยการปรับค่า Canny Threshold และ Accumulator Threshold โดยใช้ Threshold = 70

จากการปรับค่า Canny Threshold และ Accumulator Threshold ให้มีระดับ Grey Scale ที่เกินค่า Grey Scale ของรูม่านตา โดยจากการทดลองในภาพจะใช้ Threshold = 70 ทำให้มีความผิดพลาดในการตรวจหารูม่านตาได้เนื่องจากระดับ Grey Scale ของรูม่านตามีความเข้มกว่า 70

ภาพที่ 4.23 การทดลองการตรวจจับรูม่านตาโดยการปรับค่า Canny Threshold และ Accumulator Threshold โดยใช้ Threshold = 10

จากการปรับค่า Canny Threshold และ Accumulator Threshold ให้มีระดับ Grey Scale ที่ต่ำค่า Grey Scale ของรูม่านตามาก โดยจากการทดลองในภาพจะใช้ Threshold = 10 ทำให้ไม่สามารถตรวจหารูม่านตาพบเนื่องจาก ระดับ Grey Scale ของรูม่านตามีความเข้มน้อยกว่าค่า Threshold ที่กำหนด



ภาพที่ 4.24 การทดลองการตรวจจับรูม่านตาโดยการปรับค่า Canny Threshold และ Accumulator Threshold โดยใช้ Threshold = 20

จากการปรับค่า Canny Threshold และ Accumulator Threshold ให้มีระดับ Grey Scale ที่มีระดับใกล้เคียงกับระดับ Grey Scale ของรูม่านตา โดยจากการทดลองในภาพจะใช้ Threshold = 20 ทำให้สามารถตรวจหารูม่านตาได้และมีความแม่นยำสูง ดังนั้นควรตั้งค่า Canny Threshold = 20 และ Accumulator Threshold = 30

4.1.6 ความเร็วในการตรวจจับรูม่านตา

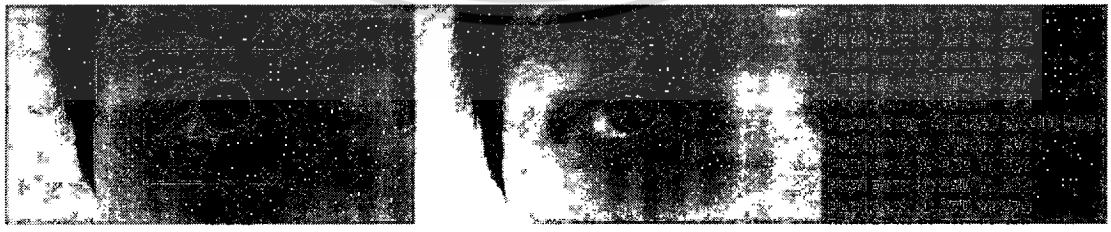
การทดลองโดยการตรวจจับรูม่านตาด้วย HoughCircle โดยทำการจับเวลาในการตรวจจับภาพทุกๆ 1 วินาที ดังภาพที่ 4.25 ดังนี้

ภาพที่ 4.25 การทดลองความเร็วในการตรวจจับรูม่านตาโดยทำการจับเวลาในการตรวจจับภาพทุกๆ 1 วินาที

จากการทดลองเพื่อตรวจหาตำแหน่งของรูม่านตาเพื่อหาความเร็วในการประมวลผลภาพพบว่าสามารถประมวลผลได้ประมาณ 4.8 Frame/s ซึ่งถือว่าอยู่ในระดับที่ช้า

4.1.7 การหาตำแหน่งอ้างอิงของรูม่านตา

การหาตำแหน่งอ้างอิงของรูม่านตาทำโดยการมองตรงไปข้างหน้าโดยสมมติว่ารูม่านตาอยู่กึ่งกลางของดวงตาและทำการหาตำแหน่งของรูม่านตาจำนวน 35 ภาพติดต่อกัน ดังภาพที่ 4.26 ดังนี้



ภาพที่ 4.26 การหาตำแหน่งอ้างอิงของรูม่านตาโดยการมองตรงไปข้างหน้าและทำการหาตำแหน่งของรูม่านตาจำนวน 35 ภาพติดต่อกัน

ตัวอย่าง 4.3 ตำแหน่งของรูม่านตาจากการตรวจจับเป็นจำนวน 35 ภาพ

Pupil pos:: X: 355 Y: 272

Pupil pos:: X: 362 Y: 274

Pupil pos:: X: 304 Y: 227

Pupil pos:: X: 359 Y: 273

Pupil pos:: X: 349 Y: 266

Pupil pos:: X: 352 Y: 264

Pupil pos:: X: 350 Y: 262

Pupil pos:: X: 370 Y: 279

Pupil pos:: X: 349 Y: 266

Pupil pos:: X: 354 Y: 265

Pupil pos:: X: 355 Y: 269

Pupil pos:: X: 360 Y: 262

Pupil pos:: X: 369 Y: 266

Pupil pos:: X: 355 Y: 262

Pupil pos:: X: 329 Y: 242

Pupil pos:: X: 369 Y: 258

Pupil pos:: X: 360 Y: 264

Pupil pos:: X: 352 Y: 269

Pupil pos:: X: 357 Y: 256

Pupil pos:: X: 359 Y: 259

Pupil pos:: X: 353 Y: 273

Pupil pos:: X: 349 Y: 263

Pupil pos:: X: 366 Y: 267

Pupil pos:: X: 361 Y: 269

Pupil pos:: X: 356 Y: 265

Pupil pos:: X: 347 Y: 266

Pupil pos:: X: 358 Y: 256

Pupil pos:: X: 365 Y: 263

Pupil pos:: X: 362 Y: 263

Pupil pos:: X: 370 Y: 271

Pupil pos:: X: 386 Y: 255

ตัวอย่าง 4.3 ตำแหน่งของรูม่านตาจากการตรวจจับเป็นจำนวน 35 ภาพ (ต่อ)

Pupil pos:: X: 356 Y: 264
Pupil pos:: X: 358 Y: 255
Pupil pos:: X: 347 Y: 258
Average of Pixel
X: 356.685714285714
Y: 263.6

จากการทดลองการตรวจหาตำแหน่งของรูม่านตาคำนวณ 35 ภาพเพื่อใช้เป็นค่าอ้างอิงจุดกึ่งกลางของรูม่านตาพบว่าได้ค่าเฉลี่ยของพิกัด X = 356.68 และ Y = 263.6 โดยค่าพิกัดที่ได้ของแกน X และ Y มีตำแหน่งที่ไม่แตกต่างกันมากซึ่งถือว่ามีความแม่นยำอยู่ในระดับสูง

4.1.8 การแปลง Pixel เป็นองศา

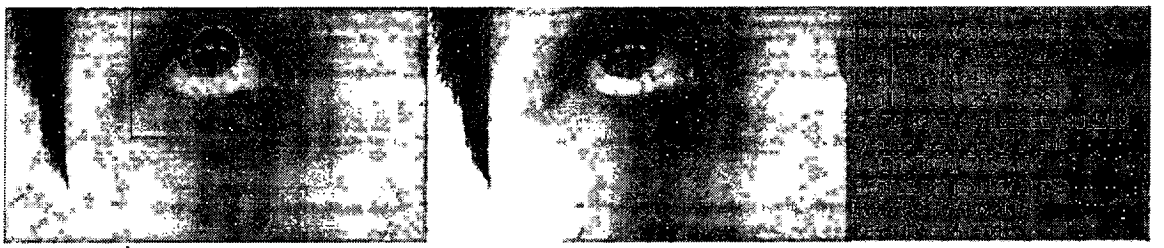
การแปลง Pixel เป็นองศาจะทำโดยการหาตำแหน่งของรูม่านตาในการมองในแนวแกน X และแนวแกน Y ด้วยมุม 60 องศาจากนั้นให้ทำการตรวจหาพิกัดของรูม่านตาจำนวน 35 ภาพติดต่อกันและทำการหาค่าเฉลี่ยจากนั้นนำมาหาความแตกต่างระหว่างพิกัดอ้างอิงดังนี้

- มองในแกน X ด้วยมุม 60 องศาทางด้านซ้าย ดังภาพที่ 4.27

ภาพที่ 4.27 การทดลองการแปลง Pixel เป็นองศาโดยการมองในแกน X ด้วยมุม 60 องศา

จากการทดสอบเพื่อหาตำแหน่งอ้างอิงของการมองไปในแนวแกน X พบว่าสามารถแปลงจาก Pixel เป็นองศาในแนวแกน X ได้ประมาณ 0.418 องศา/จุด Pixel ซึ่งค่าในการแปลง Pixel เป็นองศาของแต่ละคนจะมีความแตกต่างกันในการใช้งานจริงอาจจะต้องมีการปรับค่าใหม่ก่อนการใช้งาน

- มองในแกน Y ด้วยมุม 60 องศาทางด้านบน ดังภาพที่ 4.28



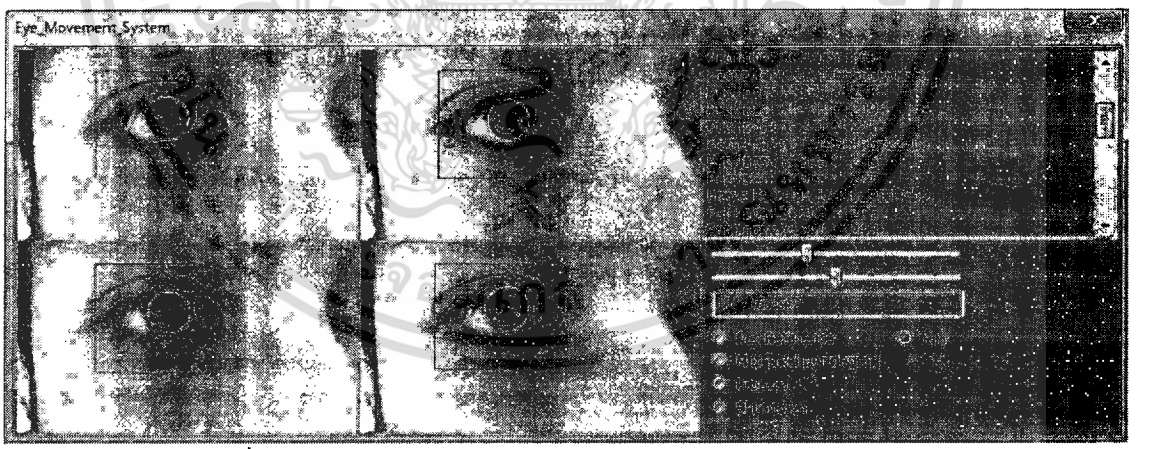
ภาพที่ 4.28 การทดลองการแปลง Pixel เป็นองศาโดยการมองในแกน Y ด้วยมุม 60 องศา

จากการทดสอบเพื่อหาตำแหน่งอ้างอิงของการมองไปในแนวแกน X พบว่าสามารถแปลงจาก Pixel เป็นองศาในแนวแกน X ได้ประมาณ 0.4628 องศา/จุด Pixel และการมองไปในแนวแกน Y พบว่าสามารถแปลงจาก Pixel เป็นองศาในแนวแกน Y ได้ประมาณ 0.6 องศา/จุดซึ่งค่าในการแปลง Pixel เป็นองศาของแต่ละคนจะมีความแตกต่างกันในการใช้งานจริงอาจจะต้องมีการปรับค่าใหม่ก่อนการใช้งาน

4.1.9 รูปแบบการทำ Multithreading

ออกแบบการทำ Multithread 3 แบบดังนี้

1) การออกแบบโดยการทำ 4 Thread Eye Processing ซึ่งแต่ละ Thread ทำงานเหมือนกันคือการดึงภาพจากกล้อง, การทำ Preprocess Image, การหาตำแหน่งของดวงตา, การทำหาตำแหน่งของรูม่านตา ซึ่งการทดลองจะทดลองหาความเร็วในการตรวจจับรูม่านตา ดังภาพที่ 4.29



ภาพที่ 4.29 การทดลองการทำ Multithread ด้วยจำนวน 4 threads

การทดลองโดยการทำ Multithreading ด้วย 4 Threads ทำให้การประมวลผลเร็วขึ้นจาก 5 Frame/s เป็น 8.6 – 14 Frame/s ซึ่งสามารถเพิ่มความเร็วได้มากขึ้นถึง 2.8 เท่า

2) การออกแบบโดยการทำ 12 Threads แต่ยังคงประมวลผลแบบ 4 Frame พร้อมๆ กันโดยการแบ่งให้เกิดงานในการประมวลผล 3 งาน ซึ่งแต่ละงานประกอบด้วย

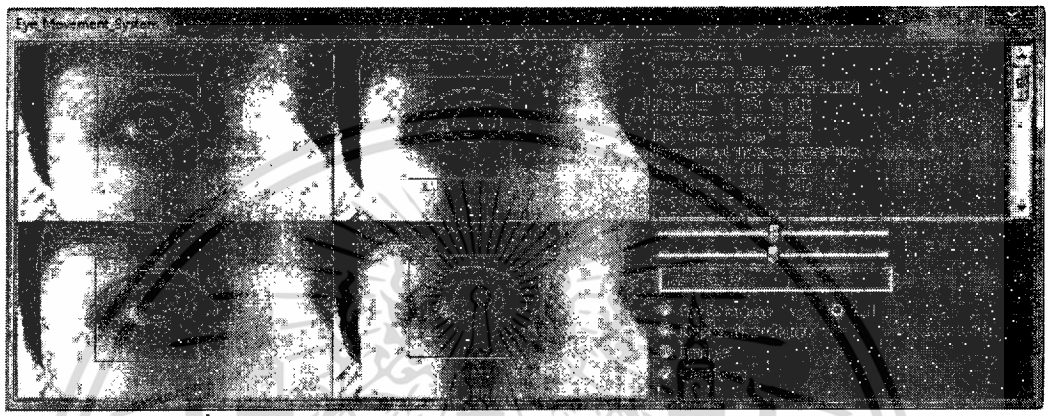
3 Threads การแบ่งงานในการประมวลผลภาพ 1 Frame ออกเป็นงานย่อย 3 งาน คือ

a) การดึงภาพจากกล้องและ Preprocessing image

b) การตรวจหาตำแหน่งของดวงตา

c) การตรวจหาตำแหน่งของรูม่านตา

ซึ่งการทดลองจะทดลองหาความเร็วในการตรวจจับรูม่านตาดังภาพที่ 4.30



ภาพที่ 4.30 การทดลองการทำ Multithread ด้วยจำนวน 12 threads

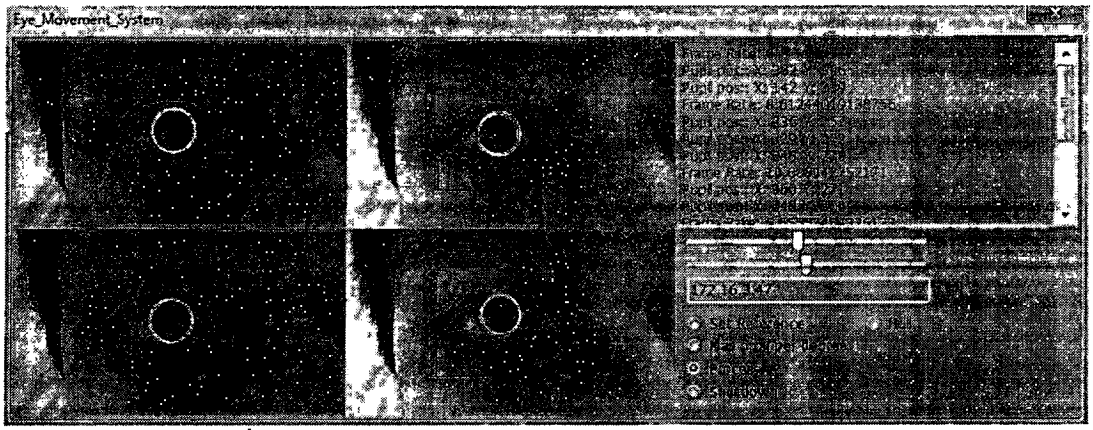
การทดลองโดยการทำ Multithreading ด้วย 12 Threads ทำให้การประมวลผลเร็วขึ้นจาก 5 Frame/s เป็น 6.83 – 12 Frame/s ซึ่งสามารถเพิ่มความเร็วได้มากขึ้นถึง 2.4 เท่า ซึ่งอาจเป็นเพราะการขึ้นรอข้อมูลของ Thread ก่อนหน้าและทรัพยากรของเครื่องคอมพิวเตอร์ที่ใช้มีน้อยกว่าจำนวน Thread ที่ใช้

3) การประมวลผลภาพแบบ 9 Threads การแบ่งการทำงานการประมวลผลภาพเป็น 9 Threads แต่ยังคงประมวลผลแบบ 4 Frame พร้อมๆกัน โดยการแบ่งให้มีเพียง 1 Thread เท่านั้นที่ทำการดึงภาพจากกล้องจับดวงตา และอีก 8 Threads ให้ทำงานโดยเหมือนมีการแบ่งงานเป็น 4 งานซึ่งแต่ละงานมี 2 Thread ช่วยกันประมวลผล โดยมีการแบ่งงานเป็น 2 Thread ดังนี้

a) การตรวจหาตำแหน่งของดวงตา

b) การตรวจหาตำแหน่งของรูม่านตา

ซึ่งการทดลองจะทดลองหาความเร็วในการตรวจจับรูม่านตาดังภาพที่ 4.31

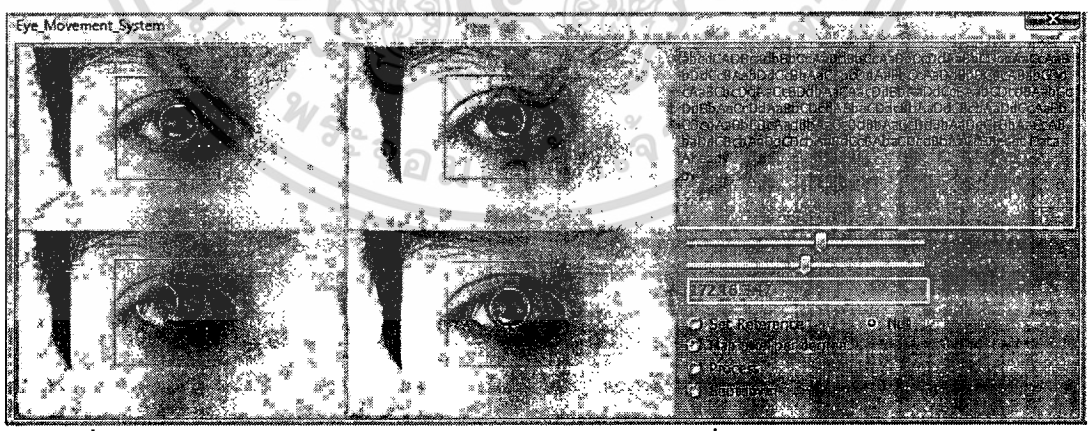


ภาพที่ 4.31 การทดลองการทำ Multithread ด้วยจำนวน 9 threads

การทดลองโดยการทำ Multithreading ด้วย 9 Threads ทำให้การประมวลผลเร็วขึ้นจาก 5 Frame/s เป็น 8.6 – 10.6 Frame/s ซึ่งสามารถเพิ่มความเร็วได้มากขึ้นถึง 2.12 เท่า ซึ่งอาจเป็นเพราะการเข้าถึงตัวแปรของภาพตัวเดียวกัน ซึ่งการแก้ไขทำได้โดยการเพิ่ม Buffer ของภาพให้มีมากขึ้นแต่การจัดการกับลำดับของภาพจะทำได้ยาก

4.1.10 Thread Order

การประมวลผลแบบ 4 Thread จะต้องมีลำดับการทำงานที่สอดคล้องกันซึ่งการทำงานของแต่ละ Thread จะถูกจัดการด้วยระบบปฏิบัติการว่าเมื่อใดให้ Thread A เข้าไปทำงานหรือให้ Thread B เข้าไปทำงาน ซึ่งการจัดลำดับการประมวลผลนี้จะส่งผลต่อการไม่สามารถทำนายว่า Thread ใดจะทำงานเสร็จก่อนได้ ซึ่งทดลองโดยการแสดงค่าหมายเลข Thread ก่อนเข้าทำงานและหลังจากทำงานเสร็จดังภาพที่ 4.32



ภาพที่ 4.32 การทดลองลำดับการทำงานของแต่ละ Thread เมื่อใช้การประมวลผล 4 threads

จากการทดลองเพื่อดูลำดับการทำงานของ Thread แต่ละ Thread ได้ผลของลำดับดังนี้

abcdCADBcadbBbCcAaDdBbCcAaBbCcDdAaBbCDcdAa

การเข้าทำงานของ Thread จะทำการ Print ค่า อักษรตัวเล็กแทน Thread ดังนี้ a = Thread 1 , b= Thread 2 , c= Thread 3 , c= Thread 4 และเมื่อ Thread ทำการประมวลผลเสร็จแล้วจะทำการ Print ค่า อักษรตัวใหญ่ ซึ่งจากการทดลองพบว่าในตอนแรก Thread ที่เข้าทำงานคือ a ,b ,c ,d ตามลำดับ แต่เมื่อทำงานแล้ว Thread ที่ประมวลผลภาพเสร็จคือ C, A, D, B ตามลำดับ ดังนั้นการเสร็จของ Thread มีโอกาสที่จะไม่เรียงลำดับกันเนื่องจากระบบปฏิบัติการจะจัดการ Schedule ของ Thread ทั้งหมด ดังนั้นลำดับของ Thread จะแสดงถึงภาพที่นำมาประมวลผลก่อนหลังซึ่งหากภาพแรกถูกประมวลผลเสร็จภายหลังจะทำให้การหมุนของ Motor มีความผิดพลาดขึ้น ทั้งนี้สามารถแก้ไขได้ด้วยการจัดลำดับของ Thread

4.2 การทดลองบนตัวหุ่นยนต์

4.2.1 ความเร็วในการหมุนของ Motor

Servo Motor แต่ละรุ่นจะมีความเร็วในการหมุนที่แตกต่างกัน ซึ่งทางกลุ่มผู้พัฒนาได้ใช้ Servo Motor 2 รุ่นคือ MKS DS1210 และ MG995 TowerPro

ทดลองโดยการหมุน Servo Motor MKS DS1210 ที่มุมต่างๆดังนี้

- ทดลองการหมุนด้วยมุม 60 องศาและโดยการกำหนดให้เวลาในการหมุน 200 ms พบว่าสามารถหมุนได้ 60 องศาตามที่กำหนด
- ทดลองการหมุนด้วยมุม 60 องศาและโดยการกำหนดให้เวลาในการหมุน 180 ms พบว่าสามารถหมุนได้ 60 องศาตามที่กำหนด
- ทดลองการหมุนด้วยมุม 60 องศาและโดยการกำหนดให้เวลาในการหมุน 150 ms พบว่าสามารถหมุนได้ 60 องศาตามที่กำหนด
- ทดลองการหมุนด้วยมุม 60 องศาและโดยการกำหนดให้เวลาในการหมุน 140 ms พบว่าสามารถหมุนได้น้อยกว่า 60 องศาตามที่กำหนด

ดังนั้นความเร็วในการหมุนของ MKS DS1210 อยู่ที่ประมาณ 0.15 วินาที /60 องศา หรืออยู่ที่ประมาณ 0.0025 วินาที/ องศา

ทดลองโดยการหมุน Servo Motor MG995 TowerPro ที่มุมต่างๆดังนี้

- ทดลองการหมุนด้วยมุม 60 องศาและโดยการกำหนดให้เวลาในการหมุน 240 Ms = พบว่าสามารถหมุนได้ 60 องศาตามที่กำหนด
- ทดลองการหมุนด้วยมุม 60 องศาและโดยการกำหนดให้เวลาในการหมุน 220 Ms = พบว่าสามารถหมุนได้ 60 องศาตามที่กำหนด

- ทดลองการหมุนด้วยมุม 60 องศาและโดยการกำหนดให้เวลาในการหมุน 200 Ms = พบว่าสามารถหมุนได้ 60 องศาตามที่กำหนด
- ทดลองการหมุนด้วยมุม 60 องศาและโดยการกำหนดให้เวลาในการหมุน 180 Ms = พบว่าสามารถหมุนได้น้อยกว่า 60 องศาตามที่กำหนด

ดังนั้นความเร็วในการหมุนของ MG995 TowerPro อยู่ที่ประมาณ 0.2 วินาที /60 องศา หรืออยู่ที่ประมาณ 0.0033 วินาที/ องศา

4.2.2 องศาการหมุนของ Motor

Servo Motor แต่ละรุ่นจะมีมุมในการหมุนที่แตกต่างกัน ซึ่งทางกลุ่มผู้พัฒนาได้ใช้ Servo Motor 2 รุ่นคือ MKS DS1210 และ MG995 TowerPro ซึ่งทดลองการหมุนของ Servo Motor โดยที่ไม่ได้ดัดแปลงจากโรงงานได้ผลดังนี้

1. MKS DS1210 ทดลองหมุนด้วยมุม 0 – 360 องศา พบว่าสามารถหมุนได้ 120 องศาตั้งแต่มุม 30 องศาจนถึง 150 องศา
2. MG995 TowerPro ทดลองหมุนด้วยมุม 0 – 360 องศา พบว่าสามารถหมุนได้ 210 องศาตั้งแต่มุม 0 องศาจนถึง 210 องศา

ดังนั้นในการเลือกใช้ Servo Motor เพื่อหมุนตามดวงตาทางกลุ่มผู้พัฒนาจะเลือกใช้ MKS DS1210 เนื่องจากสามารถหมุนด้วยความเร็วที่ต่อรอบที่มากกว่าและความต้องการในการใช้เพียง 120 เท่านั้น

4.2.3 การสื่อสารด้วย Serial Communication

การติดต่อด้วย Serial Communication ของ ATmega Microcontroller บน Arduino Board สามารถติดต่อแบบ Serial Communication ได้ด้วยความเร็วหลายระดับดังนี้

- Buadrate = 9600 bps, 19200, 38400 พบว่าสามารถส่งข้อมูลได้อย่างถูกต้องเกือบ 100 % ของการส่งข้อมูลแต่ละครั้ง
- Buadrate = 57600 bps พบว่าสามารถส่งข้อมูลได้อย่างถูกต้องเกือบ 100 % ของการส่งข้อมูลแต่ละครั้ง แต่ยังมีความผิดพลาดขึ้นบางครั้ง
- Baudrate = 115200 bps พบว่าสามารถส่งข้อมูลได้อย่างถูกต้องเกือบ 100 % ของการส่งข้อมูลแต่ละครั้ง แต่ยังมีความผิดพลาดขึ้นบางครั้งซึ่งทางกลุ่มผู้พัฒนาเลือกใช้ที่ความเร็วนี้เพื่อให้สามารถสื่อสารกับ WiFi Module ที่ความเร็วเท่ากันได้

4.2.4 Serial Port Buffer

การติดต่อระหว่าง Arduino และ WiFi Module จะติดต่อกันด้วย UART ซึ่งเป็น Serial Communication โดย WiFi Module จะส่งข้อมูลเข้าทางขา Rx ของ Arduino ในการทำ Socket Programming จะต้องทำการจัดการการรับข้อมูล Serial ให้เหมาะสมเนื่องจาก Arduino มี

Serial Buffer เพียง 64 Byte เท่านั้น และ Packet ที่ส่งข้อมูลองศาจาก Application มีขนาด 15 Byte ต่อ Packet ซึ่ง Arduino สามารถรองรับได้สูงสุดที่ 4 Packet และ Packet 5 จะ Overflow

ทดลองโดยการดูความเร็วในการรับ Packet และการส่ง Packet ของ Application ซึ่งได้ทดลองให้ Application ส่งข้อมูลองศาทุกๆ 300 ms ไปที่ตัวหุ่นยนต์แต่ตัวหุ่นยนต์จะทำการประมวลผลการแปล Packet และสั่งให้ Servo Motor หมุนด้วย Delay 300 ms ซึ่งอาจเกิดปัญหา Buffer Overflow ได้ดังนั้นสามารถแก้ไขได้โดยการจัดการให้ Microcontroller ทำการคำนวณว่าจะต้องใช้ Delay เท่าใดต่อการหมุน Servo Motor ไปยังตำแหน่งที่ต้องการแทนการกำหนดค่าแบบคงที่

4.2.5 ค่า Delay ที่เหมาะสม

การหมุนของ Servo Motor ไปยังตำแหน่งที่ต้องการจะต้องใช้ Delay ที่แตกต่างกันซึ่งขึ้นกับองศาที่จะหมุนไปจากตำแหน่งเดิม ดังนั้นจากการหาความเร็วในการหมุนของ Servo Motor MKS DS1210 ได้ซึ่งสามารถนำมาคำนวณ Delay ของการหมุน Servo Motor ไปยังตำแหน่งที่ต้องการได้จากการหาค่าความแตกต่างของมุมระหว่างมุมเดิมและมุมที่ต้องการจะหมุนไปคูณกับเวลา/ องศา ในการหมุนของ MKS DS1210 ดังนี้

- มุมเดิม คือ 90 องศาแต่ต้องการหมุนไปที่มุม 150 องศาจะต้องใช้เวลา = 0.0025 x (150-90) = 0.15 วินาที
- มุมเดิม คือ 73 องศาแต่ต้องการหมุนไปที่มุม 118 องศาจะต้องใช้เวลา = 0.0025 x (118-73) = 0.1125 วินาที

จากการทดลองด้วยการกำหนดค่า Delay ที่พอดีอาจทำให้มุมในการหมุนผิดพลาดเล็กน้อยดังนั้นให้กำหนด Delay ในการหมุนคือ Delay ที่คำนวณได้ + 0.01 วินาที ซึ่งให้ผลที่ดีกว่าการกำหนดค่าแบบพอดีดังนั้นการหมุนของมุมจะใช้ Delay ดังนี้

- มุมเดิม คือ 90 องศาแต่ต้องการหมุนไปที่มุม 150 องศาจะต้องใช้เวลา = (0.0025 x (150-90)) + 0.01 = 0.16 วินาที
- มุมเดิม คือ 73 องศาแต่ต้องการหมุนไปที่มุม 118 องศาจะต้องใช้เวลา = (0.0025 x (118-73)) + 0.01 = 0.1225 วินาที

4.3 การทดลองการสื่อสารระหว่างตัวหุ่นยนต์และ Application

ในการทดลองการสื่อสารระหว่างตัวหุ่นยนต์และ Application จะทำการทดสอบความเร็วในการติดต่อระหว่าง Application และ WiFi Module โดยทดสอบดังนี้

1) Ping Message จาก คอมพิวเตอร์ไปยัง WiFi Module และสังเกตค่า Ping Delay ซึ่งสามารถติดต่อกันได้ด้วย Delay ไม่เกิน 40 ms

2) การตรวจจับ Refreshing Rate ของการส่งข้อมูลจาก WiFi Module กลับมายัง Application พบว่าสามารถติดต่อกลับมาเพื่อ Refreshing Session ด้วยเวลาประมาณ 1 วินาที

4.4 การติดกล้องบนอุปกรณ์สวมใส่

การออกแบบการติดกล้องบนอุปกรณ์สวมใส่ ทางผู้พัฒนาได้เลือกใช้หมวกกันแสง UV โดยการติดกล้องจากส่วนที่กันแดดซึ่งยื่นออกมาด้านหน้าของหมวกและให้กล้องยื่นลงมาจากหมวก โดยการกำหนดให้มุมมองของกล้องมีขนาดที่พอดีกับดวงตาและเห็นเฉพาะดวงตาซ้าย

จากการทดลองพบว่าหมวกกันแสง UV ได้ผลดังนี้

1) หมวกทำจากวัสดุ Plastic มีความอ่อนตัวซึ่งทำให้การยึดของกล้องมีการสั่นได้เมื่อนำมาใช้งาน แต่สามารถใช้งานได้หากอยู่นิ่ง
2) หมวกกันแสง UV มีความยาวรอบหมวกที่กว้างซึ่งเมื่อนำไปใช้งานกับหลายๆคน พบว่าคนที่มีความยาวรอบศีรษะน้อยกว่าหมวกจะทำให้ใส่แล้วหลวมซึ่งส่งผลต่อกล้องที่ติดด้านหน้าเนื่องจากกล้องมีน้ำหนักทำให้ตำแหน่งของกล้องลดต่ำลงมาจากเดิม
3) ตำแหน่งดวงตาของแต่ละคนมีตำแหน่งที่แตกต่างกันไปในการสวมหมวกแต่ละครั้งจะต้องทำการปรับตำแหน่งของกล้องเล็กน้อยเพื่อให้เหมาะสมกับตำแหน่งดวงตาของคนๆนั้น
4) การเอียงของกล้องทำให้มีผลต่อการตรวจสอบรูม่านตาและอ้างอิงกับตำแหน่งของตำแหน่งที่ได้กำหนดไว้ ซึ่งหากมีการเอียงอาจทำให้การแปลงเป็นองศาผิดพลาดเล็กน้อย

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุป

ทางคณะผู้วิจัยสามารถพัฒนาได้ตรงตามขอบเขตที่วางไว้ทั้งหมด 3 ส่วน ดังนี้

5.1.1 อุปกรณ์ติดตั้งกล้องจับภาพดวงตา

1) สามารถถ่ายบริเวณดวงตาของผู้ใช้งานได้โดยเกิดการเอียงของภาพน้อยที่สุด
2) สามารถสวมบนศีรษะได้
3) สามารถเชื่อมต่อกับแอพพลิเคชั่นบนคอมพิวเตอร์ผ่านสาย USB 2.0 port ได้

5.1.2 แอพพลิเคชั่นประมวลผลภาพ

1) สามารถตรวจหาและระบุตำแหน่งดวงตาของผู้ใช้งานได้
2) สามารถตรวจหาและระบุตำแหน่งรูม่านตาของผู้ใช้งานได้
3) สามารถคำนวณทิศทางการมองของดวงตาได้
4) สามารถนำผลการคำนวณการเคลื่อนไหวของตาดำมาแปลงเป็นองศาที่จะสั่งให้มอเตอร์ทั้ง 2 ตัวหมุนได้
5) สามารถส่งข้อมูลองศาไปยัง Microcontroller บนตัวหุ่นยนต์เพื่อที่จะควบคุมให้มอเตอร์ทั้ง 2 ตัวหมุนไปยังองศาที่ต้องการได้ โดยสามารถส่งข้อมูลผ่าน Wi-Fi ด้วย Infrastructure Mode ได้
6) สามารถรับภาพจากกล้องไร้สายด้วยความถี่ 2.4 GHz มายังตัวรับบนคอมพิวเตอร์และแสดงผลบนแอพพลิเคชั่นได้
7) ตัวหุ่นยนต์และ Application สามารถทำงานแบบอัตโนมัติได้โดยการเชื่อมต่อ, ทำการส่งข้อมูล, ยกเลิกการเชื่อมต่อได้

5.1.3 ตัวหุ่นยนต์ติดกล้องถ่ายสภาพแวดล้อม

1) สามารถรับข้อมูลองศาจากแอพพลิเคชั่นผ่าน Wi-Fi ด้วย Infrastructure Mode เพื่อควบคุมให้มอเตอร์ทั้ง 2 ตัวหมุนตามองศาที่ต้องการได้
2) สามารถถ่ายสภาพแวดล้อมเสมือนการมองของดวงตาได้ด้วยความละเอียด 640 x 480
3) สามารถส่งภาพที่จับได้ไปยังแอพพลิเคชั่นผ่านทางการเชื่อมต่อไร้สายที่ความถี่ 2.4 GHz ได้

5.2 ปัญหาอุปสรรคและแนวทางการแก้ไข

1) มีความล่าช้าในการตั้งค่าเพื่อให้ WiFi Module บนตัวหุ่นยนต์สามารถเชื่อมต่อ WiFi ด้วย Infrastructure Mode ได้เนื่องจากต้องใช้เวลาในการค้นหา Access Point, การเชื่อมต่อกับ Access point ที่เข้ารหัสด้วยวิธีต่างๆกันและกำหนดค่า IP Address ของ Application ได้ ซึ่งแนวทางการแก้ไขทำได้โดยการใช้งานกับ Access Point ที่เป็น Open Network หรือทำการเปิด Hot-spot เองด้วยมือถือ
2) อุปกรณ์ติดกล้องจับภาพดวงตามีน้ำหนักค่อนข้างมากซึ่งเกิดจากตัวกล้องจับภาพจึงอาจเกิดความลำบากในการสวมใส่ นอกจากนี้การสวมใส่แต่ละครั้งอาจทำให้ตำแหน่งของการถ่ายดวงตาเปลี่ยนไปเล็กน้อย ซึ่งแนวทางการแก้ไขทำได้โดยการนำวัสดุที่มีความแข็งแรงและรองรับน้ำหนักที่มากขึ้นมาติดตั้งกล้องจับภาพดวงตา
3) แอพพลิเคชั่นไม่สามารถระบุตำแหน่งของตาดำได้เมื่อมองเห็นพื้นที่ของตาดำไม่ถึง ½ ของของพื้นที่ตาดำทั้งหมด ซึ่งจะเกิดปัญหาเมื่อมองไปยังสุดของแกน X, Y เช่นการมองไปยังซ้ายสุดหรือขวาสุด ทำให้รูม่านตามีเพียงครึ่งเดียวทำให้ไม่สามารถตรวจจับวงกลมภายในภาพได้ ซึ่งแนวทางการแก้ไขทำโดยการใช้อัลกอริทึ่มในการตรวจจับครึ่งวงกลมเมื่อตรวจจับวงกลมในภาพไม่พบ
4) แอพพลิเคชั่นคำนวณการแปลงจุด Pixel โดยแอพพลิเคชั่นจะมีการคำนวณค่าโดยการนำค่าคงที่ซึ่งได้จากการทดลองเป็นองศาและเกิดความผิดพลาดได้เมื่อใช้กับหลายคน ซึ่งเมื่อนำไปใช้กับคนที่มีดวงตาแตกต่างกันอาจเกิดความคลาดเคลื่อนเล็กน้อยในการแปลงเป็นองศาจริงได้ ซึ่งมีแนวทางการแก้ไขคือการทำ Dynamic Adaptive โดยการให้แอพพลิเคชั่นทำการคำนวณค่าที่ใช้ในการแปลงจาก Pixel เป็นองศาใหม่เมื่อทำการเริ่มการทำงานแอพพลิเคชั่น
5) การทำงานของ Multithreading เพื่อช่วยในการประมวลผลที่เร็วขึ้นแต่เกิดปัญหาเมื่อลำดับการทำงานของ Thread เสร็จไม่ตามลำดับทำให้ภาพก่อนประมวลผลเสร็จทีหลัง ส่งผลให้การส่งค่าข้อมูลองศาไปควบคุมการทำงานของตัวหุ่นยนต์ผิดไป ซึ่งมีแนวทางการแก้ไขคือการจัดลำดับการ Commit ข้อมูลของ Thread ใหม่ด้วย Tightly Commit คือการ Commit แบบตามลำดับของภาพที่ดึงจากกล้องเท่านั้น หรือการแก้ไขด้วย Loosely Commit โดยการ Commit แบบไม่เรียงลำดับได้แต่การ Commit ภาพสุดท้ายจะต้องเป็นภาพหลังสุดที่ดึงจากกล้องเท่านั้น
6) การ Debug แอพพลิเคชั่นเมื่อมีการทำงานผิดพลาด ซึ่งการทำงานแบบ Multithread ทำให้ Debug ได้ยากเนื่องจากการแสดงข้อมูลด้วย Thread จะต้องมีการขอเข้าใช้งาน User Interface และอาจมีการรอเพื่อเข้าถึงตัวแปร ทำให้ Thread ในการเข้าถึงตัวแปรไม่สามารถคาดเดาได้ดังนั้นการเข้าใช้งาน User Interface ก็มีโอกาสในการแสดงผลลำดับ

ของ Thread ผิดเช่นกัน ซึ่งแนวทางการแก้ไขทำโดยการใช้ตัวแปรแบบ Array ในการเก็บค่าลำดับของ Thread โดยมีตัวแปรตัวหนึ่งใช้เพื่อบอกลำดับล่าสุด และนำค่าของตัวแปรนั้นมาวิเคราะห์ต่อไป

7) แขนของตัวหุ่นยนต์ในการหมุนแนวแกน X ซึ่งมีน้ำหนักที่ไม่เท่ากันทำให้แกน Z เอียง ซึ่งส่งผลต่อกล้องที่ติดตั้งมอเตอร์ที่หมุนในแกน Y ซึ่งแนวทางการแก้ไขทำโดยการถ่วงน้ำหนักในข้างที่มีน้ำหนักเบาของแขนหุ่นยนต์
8) เสารับสัญญาณ Wi-Fi ของ Wi-Fi Module บนตัวหุ่นยนต์สามารถเสียหายได้ง่ายเนื่องจากสาย Interface ที่ยาวและหัวเชื่อมต่อของสาย Interface บนตัว Wi-Fi Module สามารถหลุดออกจากสายได้ง่าย ซึ่งแนวทางการแก้ไขทำโดยการนำ Wi-Fi Module ไปเชื่อมต่อกับ Shield เพื่อเปลี่ยน Interface แบบสายยาวเป็นการนำเสาสัญญาณมาติดบนตัว Wi-Fi Module แทน อีกวิธีหนึ่งในการแก้ปัญหาคือทำกล่องจากวัสดุที่แข็งแรง เพื่อใช้เป็นที่ห่อหุ้มสาย Interface ทำให้มีความแข็งแรงมากขึ้น
9) ตัวหุ่นยนต์มีน้ำหนักค่อนข้างมาก จึงอาจทำได้เกิดการเคลื่อนที่ส่วนเกินเมื่อแขนหุ่นยนต์ทำการหมุน ซึ่งแก้ไขได้โดยการนำวัสดุที่ทีความแข็งแรงและรองรับน้ำหนักที่มากขึ้นได้มาใช้งาน

5.3 แนวทางในการพัฒนาต่อ

1) พัฒนาส่วนของหน้าแอพพลิเคชั่นให้ใช้งานง่ายยิ่งขึ้น
2) นำการเคลื่อนไหวของดวงตาไปปรับใช้กับการควบคุมอย่างอื่นได้ เช่น การเคลื่อนที่ของหุ่นยนต์, การบังคับเป้าปืนด้วยสายตา เป็นต้น
3) พัฒนาให้หุ่นยนต์ให้แข็งแรงยิ่งขึ้นโดยการใช้โลหะ
4) พัฒนาความปลอดภัยในการส่งข้อมูลติดต่อกันระหว่างหุ่นยนต์และแอพพลิเคชั่น
5) พัฒนาการแสดงผลภาพให้ออกทางอุปกรณ์เสริมแทนการแสดงทางแอพพลิเคชั่น เช่น การแสดงผลภาพบนแว่นตา เป็นต้น

บรรณานุกรม/เอกสารอ้างอิง

Kun Peng, Liming Chen, Su Ruan, Georgy Kukharev. "A Robust Algorithm for Eye Detection on Gray Intensity Face without Spectacles." **JCS&T,** vol. 5, no. 3, October 2005. pp. 127-132.

รุสลี่ สุทธวีร์กูล, วิไลพร แซ่ลี้. "การตรวจจับใบหน้าด้วยวิธีการพื้นฐานของการจำลองรูปแบบ Haar-like" **วารสารวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ,** ปีที่ 6, ฉบับที่ 2, กรกฎาคม - ธันวาคม 2554.หน้า 34-43.

Tomasz Kocejko., Adam Bujnowski., Jerzy Wtorek. "Eye Mouse for Disabled." **HSI 2008,** Krakow, Poland, May 2008. pp. 199-201.

ฐิตวัฒน์ เตชจรัสชีวิน, ณพวีณา ฤกษ์ปรีดาพงศ์. **"ระบบติดตามลักษณะเด่นบนใบหน้าโดยใช้กล้องเพียงหนึ่งตัว"** วิทยานิพนธ์วิทยาศาสตรมหาบัณฑิต สาขาวิชาคณิตศาสตร์ ภาควิชาคณิตศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย. 2551.

นัฐพงษ์ หนูสิงห์. **"Multiple Peoples Detection and Tracking Designed for Video Surveillance System"** วิทยานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์. 2551.

นาถวัฒน์ สิริธรรมสกุล, นิลุบล น้อยหมื่น. **"การกำหนดอายุจากใบหน้า"** วิทยานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2551.

Zafersavas. **"TrackEye : Real-Time Tracking Of Human Eyes Using a Webcam."** [Online]. Available : http://www.codeproject.com/Articles/26897/TrackEye-Real-Time-Tracking-Of-Human-Eyes-Using-a.