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

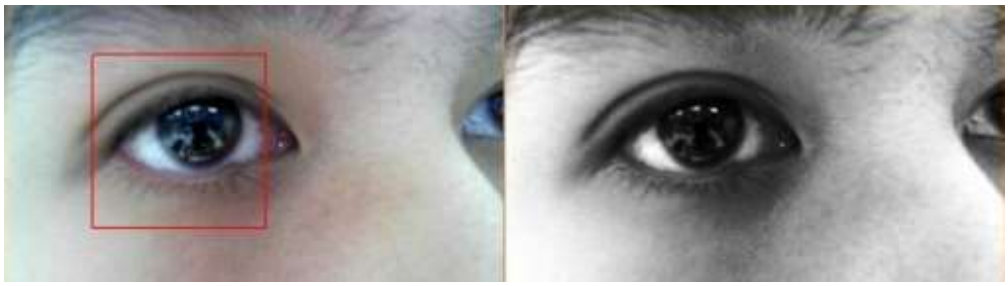
- Gamma Correct = 0.5d พบว่าได้ภาพที่มีระดับความเข้มปานกลางเห็นดวงตาชัดเจนทำให้หาตำแหน่งดวงตาได้ดี

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 ภาพ โดยมองตรงไปข้างหน้าที่ตำแหน่ง 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 ของรูม่านตาให้เหมาะสมซึ่งค่าจะอยู่ในช่วงประมาณ 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

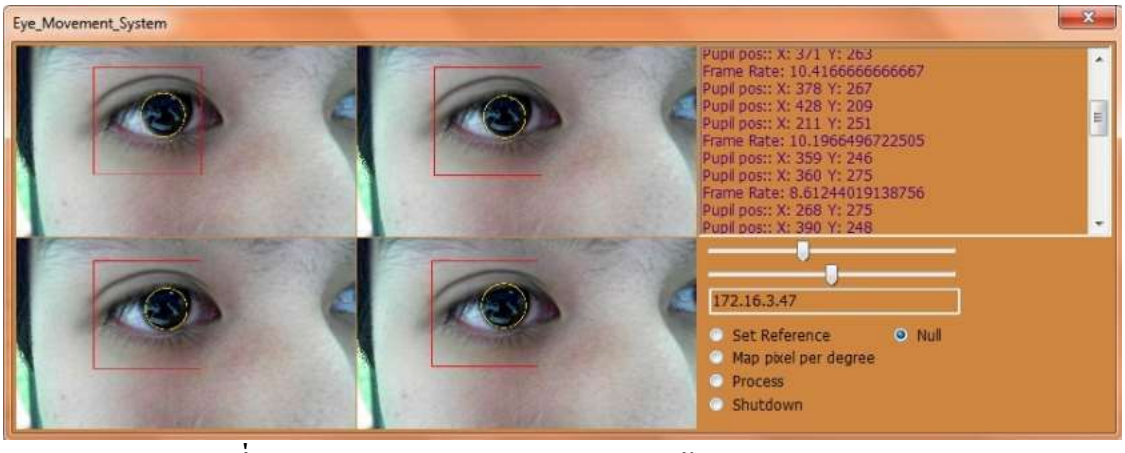
ภาพที่ 4.28 การทดลองการแปลง Pixel เป็นองศาโดยการมองในแกน Y ด้วยมุม 60 องศา

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 พร้อมๆกันโดยการแบ่งให้เกิดงานในการประมวลผล 3 งาน ซึ่งแต่ละงานประกอบด้วย

3 Threads การแบ่งงานในการประมวลผลภาพ 1 Frame ออกเป็นงานย่อย 3 งาน คือ

a) การดึงภาพจากกล้องและ Preprocessing image
b) การตรวจหาตำแหน่งของดวงตา
c) การตรวจหาตำแหน่งของรูม่านตา

ซึ่งการทดลองจะทดลองหาความเร็วในการตรวจจับรูม่านตาดังภาพที่ 4.30

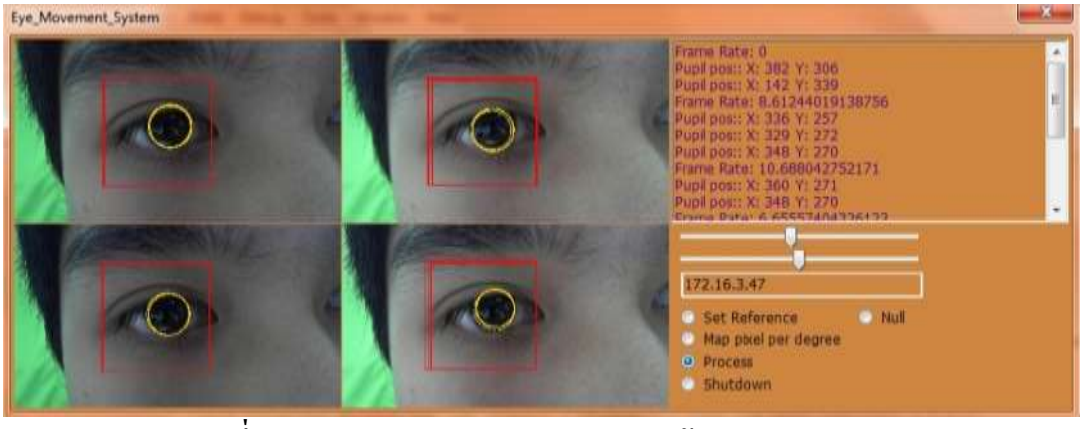
ภาพที่ 4.30 การทดลองการทำ Multithread ด้วยจำนวน 12 threads

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 พร้อมๆกันโดยการแบ่งให้มีเพียง 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