

## การตรวจนับและการวัดขนาดของพัสดุ

จัดทำโดย

นายณัฐชนน วิทยาอารีย์กุล	62070502205
นายพิชญ์ รังษีจรัส	62070502215
นายสัจบรรณ ตันตินราศักดิ์	62070502224

### 1. บทนำ

ในปัจจุบันการขนส่งพัสดุเป็นที่นิยมเป็นอย่างมาก เห็นได้ชัดจากการที่มีบริษัทรับส่งพัสดุเพิ่มเข้ามาเป็นจำนวนมาก ยกตัวอย่างเช่น Kerry express, EMS, Lazada, Shopee Express ซึ่งพัสดุที่ถูกส่งนั้นจะถูกแบ่งออกเป็นหลายขนาดโดยแต่ละขนาดก็จะมีราคาและวิธีการจัดส่งที่แตกต่างกันออกไปตามขนาดเช่น พสดุขนาดเล็กอาจจะถูกนำมารวมกันเพื่อขนส่งภายในครั้งเดียว พสดุที่มีขนาดใหญ่ก็อาจจะจำเป็นต้องมีการขนส่ง ที่ระมัดระวังมากขึ้นเป็นต้น ดังนั้นการคัดแยกพัสดุด้วยขนาดจึงเข้ามามีบทบาทและมีความสำคัญโดยที่เพื่อให้เกิด ความสะดวกรวดเร็วและความปลอดภัยต่อพัสดุมากที่สุด กลุ่มของพวกเราจึงได้มีความสนใจและจัดทำโปรเจกต์ เกี่ยวกับการตรวจนับและการวัดขนาดของพัสดุ โดยนำความรู้ในรายวิชา ENE461 Digital Image Processing มาประยุกต์ใช้ในการจัดทำ

### 2. วัตถุประสงค์

- 2.1 เพื่อประยุกต์ใช้ Image processing กับ การตรวจวัดขนาดของพัสดุในหน่วยมาตรฐาน
- 2.2 เพื่อศึกษาการตรวจนับและการวัดขนาดของสิ่งของด้วย Image Processing
- 2.3 สามารถตรวจนับและวัดขนาดความกว้างและความยาวของพัสดุได้แบบ Real Time

### 3. ประโยชน์ที่คาดว่าจะได้รับ

- 3.1 นำความรู้ Image processing ที่ได้รับไปประยุกต์และพัฒนาต่อเพื่อให้เกิดประโยชน์ได้
- 3.2 ประยุกต์ใช้ความรู้ด้าน Image Processing เช่น การทำรูปภาพ Gray scale, การประยุกต์ใช้ภาพ Binary, การทำ Contour ภาพ, การใช้ Hough Transform

#### 4. วิธีการทำ(รายละเอียด)

ในการตรวจจับและวัดขนาดของพัสดุนั้นจะถูกแบ่งออกเป็น 2 ส่วนคือการตรวจจับพัสดุและการวัดขนาดของพัสดุ โดยส่วนที่ 1 คือ การตรวจจับพัสดุ และส่วนที่ 2 การวัดขนาดของพัสดุ

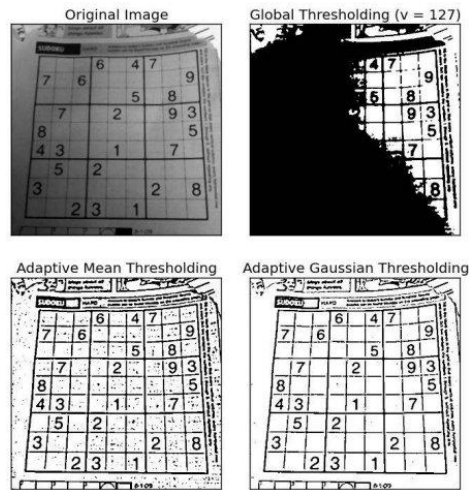
##### 4.1 การตรวจจับพัสดุ

โดยเริ่มจากการรับ Input ภาพเข้ามาในระบบ สามารถรับภาพ Input เข้ามาได้จากกล้อง Webcam แบบ Real time



รูปที่ 4.1 ตัวอย่างการรับInput จากกล้อง Webcam

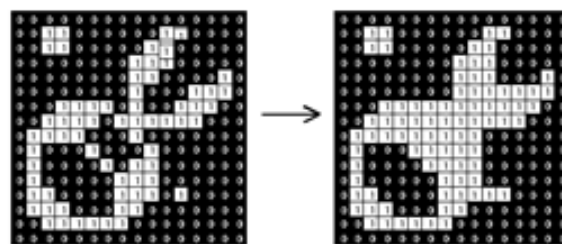
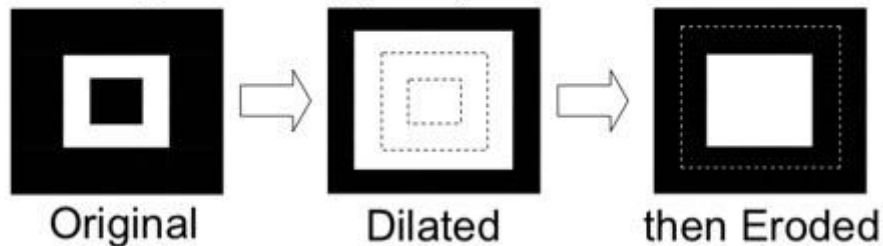
ซึ่งจะมีการกำหนดหรือตั้งข้อกำหนดคือ ฉากพื้นหลังของพื้นที่ทดสอบต้องเป็นพื้นที่สีขาวและไม่มีการกำหนดแสงที่แน่นอน ซึ่งหลักการที่ใช้ในการที่เราจะรู้ว่ามียวัตถุใหม่เข้ามาในกล้อง จะใช้หลักการของการหา Contour ของภาพเข้ามาตรวจจับ ว่าภาพที่รับเข้ามานั้นมี Contour เปลี่ยนไปหรือไม่ หลังจากนั้นจึงเริ่มกระบวนการ โดยเริ่มต้นจากการแปลงภาพที่ได้รับเป็นภาพ grayscale และนำไปทำเป็นภาพ Binary โดยวิธี Adaptive Thresholding ซึ่งทำด้วยวิธีการหาค่าเฉลี่ยของ Threshold จากค่าเฉลี่ยของทุกพิกเซลจากพื้นที่โดยรอบ โดยทำวนลูปไปเรื่อย ๆ กับบริเวณที่ไม่ซ้ำกันจนมีการกำหนดค่า Threshold ครบในทุก ๆ พิกเซล ถ้าค่า Intensity ของพิกเซลนั้นมีค่ามากกว่าค่า Threshold ของพิกเซลจะกำหนดให้เป็นสีขาว แต่ถ้าค่า Intensity ของพิกเซลนั้นน้อยกว่าค่า Threshold ของพิกเซลจะกำหนดให้เป็นสีดำ เมื่อทำครบทุกพิกเซลจะได้ผลลัพธ์เป็นภาพขาวดำ ที่เลือกใช้วิธี Adaptive Thresholding เพราะว่าสามารถแยกวัตถุกับพื้นหลังได้ถึงแม้ว่าจะมีความสว่างไม่สม่ำเสมอเช่นเดียวกับที่ทำการทดลอง



รูปที่ 4.2 ตัวอย่างการใช้ Adaptive Thresholding

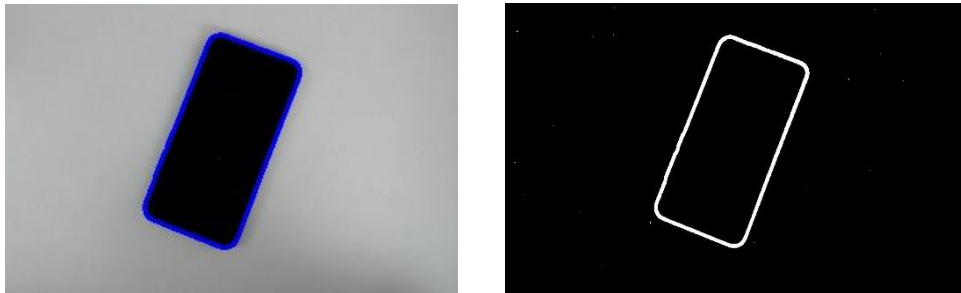
และทำการ Closed รูปภาพเพื่อลด noise โดยนำตัวดำเนินการพื้นฐานซึ่งก็คือการทำ การขยาย(Dilation) และตามด้วยการ การกร่อน(Erosion) ซึ่งเมื่อใช้กับภาพ Binary จะเป็นการเปลี่ยนค่าของตำแหน่งที่อยู่ใกล้ขอบเขตของวัตถุที่มีค่าเท่ากับ 0 ให้มีค่าเป็น 1 จากนั้นเป็นการเปลี่ยนค่าของตำแหน่งขอบเขตของวัตถุที่มีค่าเท่ากับ 1 ให้มีค่าเป็น 0 เพื่อใช้ในการปิดช่องว่างขนาดเล็กของภาพ Binary ให้มีความสมบูรณ์

Closed Image:  $A \bullet B = (A \oplus B) \ominus B$



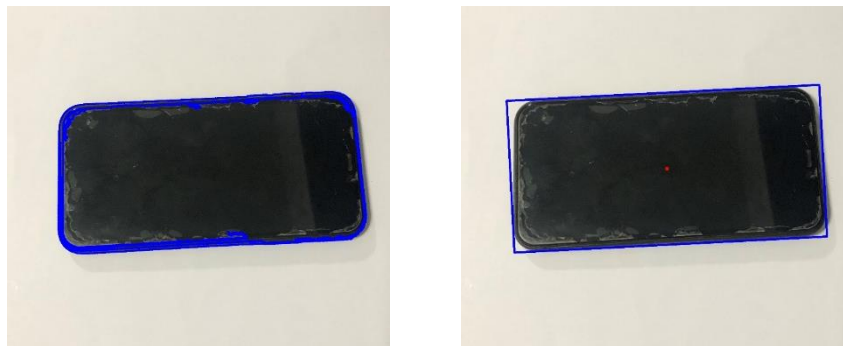
รูปที่ 4.3 ลักษณะการทำ Closed image

และตรวจสอบโดยหาขอบของวัตถุหรือการหา Contour นอกสุดของวัตถุ เราก็จะสามารถตรวจจับได้แล้วว่า มีวัตถุเพิ่มเข้ามาในกล้อง Webcam หรือไม่



ภาพที่ 4.4 ตัวอย่างการหาขอบโดยการหา Contour ของวัตถุ

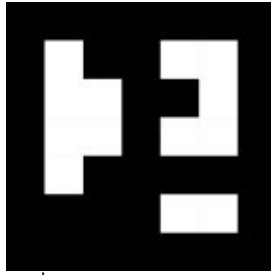
หลังจากที่ทำการหาขอบของวัตถุแล้ว เพื่อให้ง่ายต่อการกำหนดขอบเขตเพื่อนำไปใช้ในการคำนวณหาขนาดวัตถุจึงใช้รูปแบบขอบเขตแบบสี่เหลี่ยมในการระบุขอบเขตของวัตถุ



รูปที่ 4.5 ขอบเขตของ Contour ที่กำหนดให้เป็นสี่เหลี่ยม

## 4.2 การวัดขนาดของวัตถุ

การที่จะทราบขนาดของวัตถุใด ๆ ได้ จำเป็นที่จะต้อง มี วัตถุอ้างอิงหรือวัตถุที่เรารู้ขนาดในโลกของความเป็นจริง เพื่อการที่เราจะใช้ส่วนนั้นในการหาอัตราส่วนของ pixel ที่กล้องตรวจจับได้ในขณะนั้นเทียบกับขนาดจริง ที่เรียกว่า PixelsperMetric ซึ่งสำหรับในโปรเจกต์นี้ เราได้เลือกวัตถุที่จะนำมาเป็นวัตถุอ้างอิงก็คือ Aruco markers หรือว่า Augmented Reality University of Cordoba



รูปที่ 4.6 Aruco markers

ซึ่ง Aruco markers นี้เป็นวัตถุที่ถูกใช้เป็นวัตถุอ้างอิงอย่างแพร่หลายเพราะว่าสามารถตรวจจับเจอได้ง่าย เนื่องจากเป็นวัตถุที่มีลวดลายต่าง ๆ ที่ถูกสร้างขึ้นที่เป็นลักษณะของ Binary ที่มีพื้นหลังเป็นสีดำ ถูกใช้งานบ่อยใน Camera Pose Estimation และ Camera Calibration สำหรับในโปรเจกต์นี้เราจะเลือกใช้ Aruco 5x5 pixel ขนาด 5 cm x 5 cm ในการอ้างอิง โดย PixelsperMetric ของเราจะสามารถหาได้จากการนำความยาวรอบรูปทั้งหมดที่กล้องตรวจจับ Aruco markers ได้ มาหารกับความยาวรอบรูปจริงของ Aruco อ้างอิง ซึ่งก็คือ 20 cm หรือสามารถเขียนได้ดังสมการที่ 1

$$\text{PixelsperMetric} = \frac{\text{ความยาวรอบรูปของ Aruco markers ในกล้อง}}{20} \quad (1)$$

หลังจากที่ได้ PixelsperMetric ต่อไปคือการนำ Reference นี้ไปใช้กับวัตถุอื่น ๆ ที่สามารถตรวจจับได้ในพื้นที่ทดสอบ โดยนำขนาดของ Pixel ที่กล้องตรวจจับได้ของวัตถุนั้น ๆ หารด้วย PixelsperMetric ก็จะได้ขนาดจริงของวัตถุนั้นออกมา



รูปที่ 4.8 ภาพและขนาดที่กล้องสามารถจับได้ขณะนั้น



รูปที่ 4.9 ภาพและขนาดจริงที่ผ่านการหารด้วย PixelsperMetric

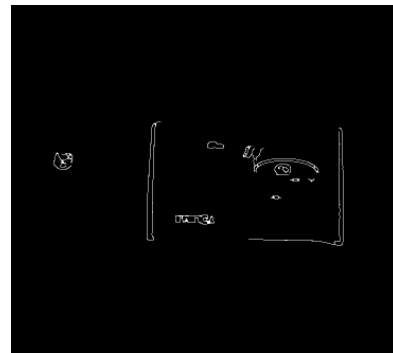
## 5. วิธีการทดลองและอุปสรรค

ก่อนที่จะได้มาซึ่งวิธีการทำโดยละเอียดในหัวข้อ 4 ที่ผ่านมา ได้มีการพบเจออุปสรรคต่างๆ จึงทำให้ได้มีการปรับเปลี่ยน Solution ให้มีความเหมาะสมที่สุดเพื่อให้สามารถตรวจจับและวัดขนาดของพัสดุดูออกมาได้อย่างแม่นยำและคลาดเคลื่อนน้อยที่สุด ซึ่งได้ปรับเปลี่ยน Solution ในส่วนต่างๆหลังจากได้ทำการทดลองดังต่อไปนี้

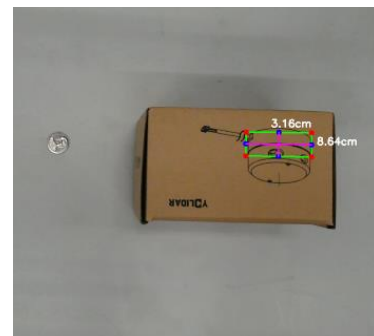
### 5.1 Canny edge detection vs Adaptive Thresholding

#### วิธีที่ 1


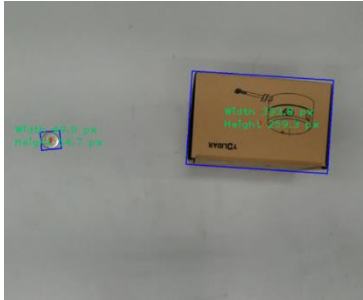
ทำการแปลงภาพที่เข้ามาจากกล้องให้เป็น Grayscale ด้วยฟังก์ชันของ Open CV จากนั้นทำการลด Noise ด้วยการ Blur โดยใช้ Gaussian Blur จากนั้นทำการหาขอบของวัตถุด้วยวิธีการ Canny edge detection



รูปที่ 5.1 ภาพจาก Canny edge




รูปที่ 5.2 ภาพการจับขนาดของวัตถุจากภาพที่ 5.1

<p>วิธีที่ 2</p> <p>ทำการแปลงภาพที่เข้ามาจากกล้องให้เป็น Grayscale ด้วยฟังก์ชันของ Open CV จากนั้นทำการเปลี่ยนภาพให้เป็น Binary โดยวิธี Adaptive Thresholding จากนั้นทำการ Closed ด้วยการทำให้ Dilation และตามด้วยการ Erosion รูปภาพเพื่อลด noise จากนั้นหาขอบของวัตถุหรือการหา contour นอกสุดของวัตถุด้วยฟังก์ชันของ Open CV</p>	 <p>รูปที่ 5.3 ภาพจากการทำ Binary โดยใช้ Adaptive Thresholding</p>  <p>รูปที่ 5.4 ภาพการจับ contour นอกสุดของวัตถุหลังจากผ่านการ Closed</p>
---	---

จากการทดลอง 5.1 พบว่าการใช้ Canny edge detection บาง Contour จะหายไปเนื่องจากค่าแสงที่เปลี่ยนไปทำให้มีผลต่อการทำ Contour แตกต่างกับการทำ Adaptive Thresholding ที่สามารถปรับค่า Threshold ของรูปภาพให้การทำ Contour ออกมาชัดเจนกว่า

## 5.2 Reference ด้วยวัตถุ vs Reference ด้วย Aruco (กำหนดระยะห่างที่แน่นอนของกล้อง)

<p>วิธีที่ 1</p> <p>หลังจากกระบวนการของหัวข้อ 5.1 ได้ทำการ Dilation และ Erosion เมื่อทำการหาขอบภาพจากวิธีการข้างต้นได้แล้วก็ทำการเขียน Loop เพื่อเข้าถึงพิกเซลที่เป็นขอบ จากนั้นก็ทำการสร้างกรอบรอบวัตถุและกำหนดให้วัตถุตัวหนึ่งเป็นวัตถุสำหรับอ้างอิงขนาดของวัตถุที่จะวัด ซึ่งในกรณีนี้ได้กำหนดให้ใช้วัตถุอ้างอิงเป็นเหรียญ</p>	 <p>รูปที่ 5.5 การสร้างกรอบของวัตถุ โดยมีเหรียญเป็นวัตถุอ้างอิง</p>
--	---

วิธีที่ 2.

หลังจากกระบวนการของหัวข้อ 5.1

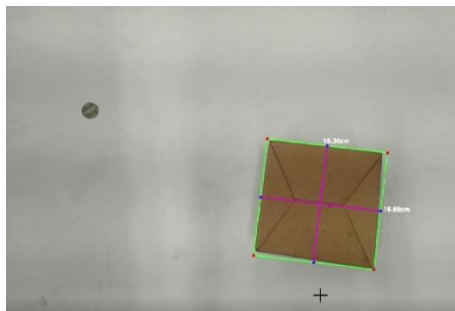
ได้ทำการสร้างกรอบสี่เหลี่ยมรอบขอบนอกสุดของวัตถุเพื่อง่ายต่อการคำนวณหาขนาดวัตถุในหน่วยพื้นฐาน ต่อมาทำการเขียน Loop เพื่อเข้าถึงทุกพิกเซลขอบของกรอบสี่เหลี่ยมเพื่อทำการหาขนาดโดยใช้ PixelsperMetric โดยใช้ Aruco อ้างอิง



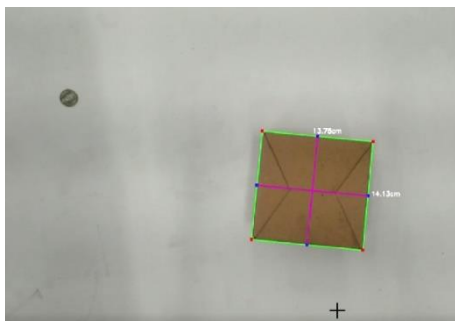
รูปที่ 5.6 การสร้างกรอบของวัตถุ โดยมี Aruco maker เป็นวัตถุอ้างอิง

จากการทดลอง 5.2 โดยใช้เหรียญในการอ้างอิงและ aruco marker ในการอ้างอิง พบว่า aruco marker จะสามารถถูกตรวจจับได้ง่ายกว่าและมีความคลาดเคลื่อนที่น้อยกว่า ทั้งนี้ทั้งนั้น แสงจะมีผลต่อ aruco marker น้อยกว่าเหรียญเนื่องจากเหรียญสามารถสะท้อนแสงได้

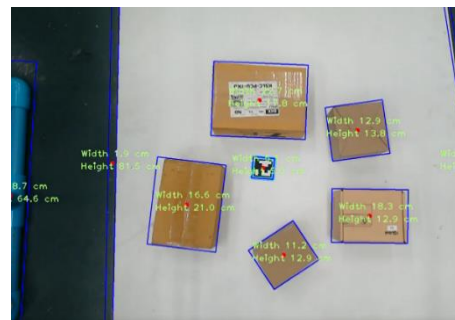
### 5.3 Reference ด้วยวัตถุ vs Reference ด้วย Aruco (ปรับระยะห่างกล้อง)



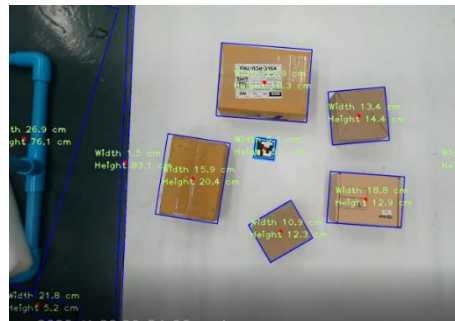
รูปที่ 5.7 การปรับระยะกล้อง โดยมีเหรียญเป็นวัตถุอ้างอิง



รูปที่ 5.8 การปรับระยะกล้องใกล้ขึ้น โดยมีเหรียญเป็นวัตถุอ้างอิง



รูปที่ 5.9 การปรับระยะกล้อง โดยมี Aruco maker เป็นวัตถุอ้างอิง

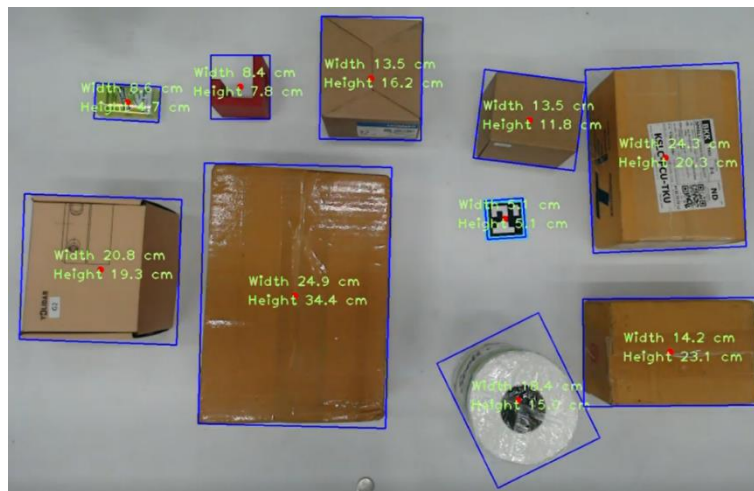


รูปที่ 5.10 8 การปรับระยะกล้องใกล้ขึ้น โดยมี Aruco maker เป็นวัตถุอ้างอิง

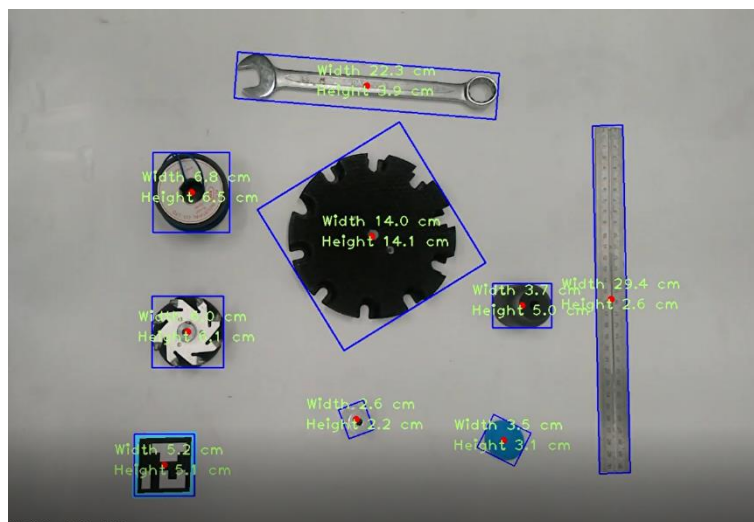


จากการทดลอง 5.3 โดยการนำวิธีที่ใช้เหรียญเป็นวัตถุอ้างอิงและการใช้ Aruco marker เป็นวัตถุอ้างอิง พบว่าทั้งสองวิธี จะมี Error หรือความคลาดเคลื่อนเกิดขึ้น หากตำแหน่งหรือระยะทางของกล่องเปลี่ยนไป แต่การใช้ Aruco marker เป็นวัตถุอ้างอิงจะมีความคลาดเคลื่อนที่น้อยกว่าเนื่องจาก Aruco marker มีขนาดที่ไม่เปลี่ยนแปลงไปแม้ระยะห่างของกล่องจะเปลี่ยนไป จึงทำให้ค่า PixelsperMetric เปลี่ยนแปลงน้อยกว่า การใช้เหรียญเป็นวัตถุอ้างอิง

## 6. ผลลัพธ์



รูปที่ 6.1 ผลลัพธ์ของการวัดขนาดกล่องโดยใช้ Aruco makerเป็นวัตถุอ้างอิง



รูปที่ 6.2 ผลลัพธ์ของการวัดขนาดวัตถุโดยใช้ Aruco makerเป็นวัตถุอ้างอิง

## 7. สรุปผลและอภิปรายผล

จากการทำการตรวจจับและการวัดขนาดของพัสดุ พบว่าการที่จะตรวจจับวัตถุด้วย Contour จะต้องเลือกวิธีการให้เหมาะสมกับงานนั้น ถ้าหากเลือกใช้วิธีที่ไม่เหมาะสมอาจจะได้รับผลลัพธ์ที่ไม่ตรงตามความต้องการยกตัวอย่างเช่นในโปรเจกต์นี้ ต้องการหาเพียงขอบนอกของวัตถุ แต่พอเลือกใช้วิธีการหา Contour โดย Canny edge detection พบว่าการตรวจจับขอบนอกสุดของวัตถุได้ผลลัพธ์ที่ไม่ถูกต้องและอาจจะเป็นที่จะต้องมีการกำหนดแสงด้วยเนื่องจากแสงก็เป็นอีกปัจจัยหนึ่งที่มีผลต่อการหา Contour ส่วนในการวัดขนาดของพัสดุสำหรับในโปรเจกต์นี้ เราได้ใช้กล้อง Webcam พบว่าหากเราไม่มีวัตถุที่จะมาเป็น Reference หรืออ้างอิงขนาดให้ กระบวนการนี้จะไม่สามารถทำได้หรือผลลัพธ์ที่ได้มาอาจจะไม่ตรงตามความเป็นจริงได้ ต่อมาถึงแม้ว่าจะมีวัตถุ Reference แล้ว ก็เกิดความคลาดเคลื่อนขึ้นได้จากที่ในบทที่ 5 ซึ่งได้ยกตัวอย่างใช้ Reference เป็นเหรียญ พบว่ามีความคลาดเคลื่อนพอสมควร และหากตำแหน่งของกล้องเปลี่ยนไปจะส่งผลให้ค่า Pixel ที่กล้องจับได้ ณ ขณะนั้นเปลี่ยนไปทำให้ค่า Pixel per Metric เปลี่ยนแปลงไป ท้ายที่สุดแล้วผลลัพธ์ที่ได้ออกมาก็จะคลาดเคลื่อนไปในที่สุด จึงทำให้วิธีที่ทดลองโดยใช้ Aruco marker เป็นวัตถุอ้างอิงจะเป็นวิธีที่ดีที่สุด มีความคลาดเคลื่อนน้อยกว่าการใช้วัตถุอ้างอิงเป็นเหรียญและสามารถปรับระยะห่างของกล้องได้พอสมควรทำให้สามารถ Adaptive ในเรื่องของระยะทางได้ แต่ก็ไม่มากนักโดยมีค่าความคลาดเคลื่อนที่น้อยที่สุดอยู่ที่ 2.0 เปอร์เซ็นต์และมีค่าความคลาดเคลื่อนที่เยอะที่สุดอยู่ที่ 35.9 เปอร์เซ็นต์ ทั้งนี้โปรเจกต์นี้อาจจะไม่จำเป็นต้องมี Reference ในการวัดขนาดวัตถุเลยถ้าหากว่าใช้กล้องประเภท Depth camera ที่สามารถรู้ถึงความลึกของภาพได้

## 8. หน้าที่และความรับผิดชอบใน Project

นายณัฐชนน วิทยาอารีย์กุล 62070502205

-ทำการทดสอบโปรแกรมการ Contour และการตรวจจับวัตถุ และทำรูปเล่มและทำการทดลองและบันทึกผล แก่ไฮสไลด์นำเสนอ

นายพิชญ์ รังษีจรัส 62070502215

-ทำการทดสอบโปรแกรมการวัดขนาดของวัตถุ ทำรูปเล่ม ทำการทดลองและบันทึกผล ทำการแก้ไขโปรแกรมโดยรวม แก่ไฮสไลด์นำเสนอ

นายสัจบรรณ ดันดินราศักติ 62070502224

-ทำการทดสอบโปรแกรมการ Input รูปภาพเข้า Webcam และตรวจสอบรูปเล่ม ทำการทดลองและบันทึกผล ทำสไลด์นำเสนอ

Link Video Presentation: <https://youtu.be/IBRTK7wSQN8>

## 9. อ้างอิง

8.1 Sergio Canu, **Measurement size of an Object** [Online], 2021, Available:

<https://pysource.com/2021/05/28/measure-size-of-an-object-with-opencv-aruco-marker-and-python/> [2022, November 11]

8.2 Adrian Rosebrock, **Measurement size of objects in an image with OpenCV**

[Online], 2016, Available: <https://learnopencv.com/augmented-reality-using-aruco-markers-in-opencv-c-python/> [2022, October 14]

8.3 Vizdx LLC, **Object Measurement** [Online], 2020, Available:

<https://www.computervision.zone/courses/object-size-measurement/> [2022, October 14]

8.4 Sunita Nayak, **Augmented Reality using Aruco Markers in OpenCV** [Online],

2020, Available: <https://learnopencv.com/augmented-reality-using-aruco-markers-in-opencv-c-python/> [2022, November 25]

8.5 Doxygen, **Detection of ArUco Markers** [Online], 2014, Available:

[https://docs.opencv.org/4.x/d5/dae/tutorial\\_aruco\\_detection.html](https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html) [2022, November 2]

## 10. ภาคผนวก

```
1 from scipy.spatial import distance as dist
2 from imutils import perspective
3 from imutils import contours
4 import numpy as np
5 import argparse
6 import imutils
7 import cv2
8 import utilis
9 import matplotlib.pyplot as plt
10
11 webcam = True
12 path = '1.jpg'
13 cap = cv2.VideoCapture(2)
14 cap.set(10,160)
15 cap.set(3,1920)
16 cap.set(4,1080)
17 scale = 3
18 wP = 210 *scale
19 hP= 297 *scale
20
21 def midpoint(ptA, ptB):
22     return ((ptA[0] + ptB[0]) * 0.5, (ptA[1] + ptB[1]) * 0.5)
23
24 if webcam:success,img = cap.read()
25 else: img = cv2.imread(path)
26
27 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
28 gray = cv2.GaussianBlur(gray, (7, 7), 0)
29
30 edged = cv2.Canny(gray, 50, 100)
31 edged = cv2.dilate(edged, None, iterations=1)
32 edged = cv2.erode(edged, None, iterations=1)
33 edged = cv2.dilate(edged, None, iterations=2)
34 edged = cv2.erode(edged, None, iterations=2)
35
36
37 cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
38                         cv2.CHAIN_APPROX_SIMPLE)
39 cnts = imutils.grab_contours(cnts)
40
41 # cnts = max(cnts, key = cv2.contourArea)
42 # cnts = cv2.minAreaRect(cnts)
43
44 # kernel = np.ones((21, 21), np.uint8)
45 # closing = cv2.morphologyEx(gray, cv2.MORPH_CLOSE, kernel)
46 # edges = cv2.Canny(closing, 50, 120)
47 # edges = cv2.resize(edges,(0,0),None,0.5,0.5)
48 # cv2.imshow(new,edges)
49
50
51 try :
52     (cnts, _) = contours.sort_contours(cnts)
53 except:
54     print("Error1")
55 finally:
56     pass
57     # pixelsPerMetric = None
58
59 i=1
60 for c in cnts:
61
62     if cv2.contourArea(c) >50000:
63         continue
64
65
66     c = max(cnts, key = cv2.contourArea)
67     orig = img.copy()
68     box = cv2.minAreaRect(c)
69     box = cv2.cv.BoxPoints(box) if imutils.is_cv2() else cv2.boxPoints(box)
70     box = np.array(box, dtype="int")
71     box = perspective.order_points(box)
72     cv2.drawContours(orig, [box.astype("int")], -1, (0, 255, 0), 2)
73
74     # loop over the original points and draw them
75     for (x, y) in box:
76         cv2.circle(orig, (int(x), int(y)), 5, (0, 0, 255), -1)
77
78     # unpack the ordered bounding box, then compute the midpoint
79     # between the top-left and top-right coordinates, followed by
80     # the midpoint between bottom-left and bottom-right coordinates
81     (t1, tr, br, bl) = box
82     (tltrX, tltrY) = midpoint(t1, tr)
83     (blbrX, blbrY) = midpoint(bl, br)
84
85     # compute the midpoint between the top-left and top-right points,
86     # followed by the midpoint between the top-right and bottom-right
87     (tlblX, tlblY) = midpoint(t1, bl)
88     (trbrX, trbrY) = midpoint(tr, br)
89
90     # draw the midpoints on the image
91     cv2.circle(orig, (int(tltrX), int(tltrY)), 5, (255, 0, 0), -1)
92     cv2.circle(orig, (int(blbrX), int(blbrY)), 5, (255, 0, 0), -1)
93     cv2.circle(orig, (int(tlblX), int(tlblY)), 5, (255, 0, 0), -1)
94     cv2.circle(orig, (int(trbrX), int(trbrY)), 5, (255, 0, 0), -1)
95
96     cv2.line(orig, (int(tltrX), int(tltrY)), (int(blbrX), int(blbrY)),
97             (255, 0, 255), 2)
98     cv2.line(orig, (int(tlblX), int(tlblY)), (int(trbrX), int(trbrY)),
99             (255, 0, 255), 2)
100
101     dA = dist.euclidean((tltrX, tltrY), (blbrX, blbrY))
102     dB = dist.euclidean((tlblX, tlblY), (trbrX, trbrY))
103
104     pixelsPerMetric = None
```

```

105     if pixelsPerMetric is None:
106         pixelsPerMetric = dB / 2.0
107
108     while True:
109         if webcam: success, img = cap.read()
110         else: img = cv2.imread(path)
111
112         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
113         gray = cv2.GaussianBlur(gray, (7, 7), 0)
114
115         edged = cv2.Canny(gray, 50, 100)
116         edged = cv2.dilate(edged, None, iterations=1)
117         edged = cv2.erode(edged, None, iterations=1)
118         edged = cv2.dilate(edged, None, iterations=2)
119         edged = cv2.erode(edged, None, iterations=2)
120
121         cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
122                                 cv2.CHAIN_APPROX_SIMPLE)
123         cnts = imutils.grab_contours(cnts)
124
125         cnts = max(cnts, key = cv2.contourArea)
126         cnts = cv2.minAreaRect(cnts)
127
128         kernel = np.ones((21, 21), np.uint8)
129         closing = cv2.morphologyEx(gray, cv2.MORPH_CLOSE, kernel)
130         edges = cv2.Canny(closing, 50, 120)
131         edges = cv2.resize(edges, (0,0), None, 0.5, 0.5)
132         cv2.imshow('Edge', edges)
133
134     try :
135         (cnts, _) = contours.sort_contours(cnts)
136     except:
137         print("Error1")
138     finally:

```

```

139         pass
140
141     i=1
142
143     for c in cnts:
144
145         if cv2.contourArea(c) > 50000:
146             continue
147
148         c = max(cnts, key = cv2.contourArea)
149         orig = img.copy()
150         box = cv2.minAreaRect(c)
151         box = cv2.cv.BoxPoints(box) if imutils.is_cv2() else cv2.boxPoints(box)
152         box = np.array(box, dtype="int")
153
154         box = perspective.order_points(box)
155         cv2.drawContours(orig, [box.astype("int")], -1, (0, 255, 0), 2)
156
157         # loop over the original points and draw them
158         for (x, y) in box:
159             cv2.circle(orig, (int(x), int(y)), 5, (0, 0, 255), -1)
160
161         # unpack the ordered bounding box, then compute the midpoint
162         # between the top-left and top-right coordinates, followed by
163         # the midpoint between bottom-left and bottom-right coordinates
164         (tl, tr, br, bl) = box
165         (tltrX, tltrY) = midpoint(tl, tr)
166         (blbrX, blbrY) = midpoint(bl, br)
167
168         # compute the midpoint between the top-left and top-right points,
169         # followed by the midpoint between the top-right and bottom-right
170         (tlblX, tlblY) = midpoint(tl, bl)
171         (trbrX, trbrY) = midpoint(tr, br)
172

```

```

173         # draw the midpoints on the image
174         cv2.circle(orig, (int(tltrX), int(tltrY)), 5, (255, 0, 0), -1)
175         cv2.circle(orig, (int(blbrX), int(blbrY)), 5, (255, 0, 0), -1)
176         cv2.circle(orig, (int(tlblX), int(tlblY)), 5, (255, 0, 0), -1)
177         cv2.circle(orig, (int(trbrX), int(trbrY)), 5, (255, 0, 0), -1)
178
179         cv2.line(orig, (int(tltrX), int(tltrY)), (int(blbrX), int(blbrY)),
180                 (255, 0, 255), 2)
181         cv2.line(orig, (int(tlblX), int(tlblY)), (int(trbrX), int(trbrY)),
182                 (255, 0, 255), 2)
183
184         dA = dist.euclidean((tltrX, tltrY), (blbrX, blbrY))
185         dB = dist.euclidean((tlblX, tlblY), (trbrX, trbrY))
186
187         # compute the size of the object
188         dimA = (dA / pixelsPerMetric)
189         dimB = (dB / pixelsPerMetric)
190
191         # draw the object sizes on the image
192         cv2.putText(orig, "{:.2f}cm".format(dimA),
193                     (int(tltrX - 15), int(tltrY - 10)), cv2.FONT_HERSHEY_SIMPLEX,
194                     0.65, (255, 255, 255), 2)
195         cv2.putText(orig, "{:.2f}cm".format(dimB),
196                     (int(trbrX + 10), int(trbrY)), cv2.FONT_HERSHEY_SIMPLEX,
197                     0.65, (255, 255, 255), 2)
198
199         i=i+1
200         #cv2.imwrite("result"+str(i)+".jpg",orig)
201     try :
202         orig = cv2.resize(orig,(0,0),None,0.5,0.5)
203         img = cv2.resize(img,(0,0),None,0.5,0.5)
204         edged = cv2.resize(edged,(0,0),None,0.5,0.5)
205     except:
206         print('Error2')

```

```

207     finally:
208         pass
209
210     cv2.imshow('Detected PIC',orig)
211     cv2.imshow('Webcam',img)
212     cv2.imshow('Edge',edged)
213
214     if cv2.waitKey(1) & 0xFF == ord('q'):
215         break

```

```

1  from scipy.spatial import distance as dist
2  from imutils import perspective
3  from imutils import contours
4  import numpy as np
5  import argparse
6  import imutils
7  import cv2
8  import matplotlib.pyplot as plt
9  from imutils import paths
10 from object_detector import *
11
12 webcam = True
13 path = '1.jpg'
14 cap = cv2.VideoCapture(0)
15 cap.set(10,160)
16 cap.set(3,1920)
17 cap.set(4,1080)
18
19 def midpoint(ptA, ptB):
20     return ((ptA[0] + ptB[0]) * 0.5, (ptA[1] + ptB[1]) * 0.5)
21
22 def detect_objects(frame):
23     # Convert Image to grayscale
24     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
25
26     # Create a Mask with adaptive threshold
27     mask = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 19, 5)
28
29     edged = cv2.dilate(mask, None, iterations=3)
30     edged = cv2.erode(edged, None, iterations=3)
31
32     # Find contours
33     contours, _ = cv2.findContours(edged, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
34
35     #cv2.imshow("mask", mask)
36
37     objects_contours = []
38
39     for cnt in contours:
40         area = cv2.contourArea(cnt)
41         if area > 2000:
42             #cnt = cv2.approxPolyDP(cnt, 0.03*cv2.arcLength(cnt, True), True)
43             objects_contours.append(cnt)
44
45     return objects_contours,mask,edged
46
47 # Load Aruco detector
48 parameters = cv2.aruco.DetectorParameters_create()
49 aruco_dict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_5X5_50)
50
51
52 while True:
53     _, img = cap.read()
54
55     corners, _ = cv2.aruco.detectMarkers(img, aruco_dict, parameters=parameters)
56     contours,mask,edged = detect_objects(img)
57     if corners:
58
59         # Draw polygon around the marker
60         int_corners = np.int0(corners)
61         cv2.polylines(img, int_corners, True, (232, 192, 73), 10)
62
63         # Aruco Perimeter
64         aruco_perimeter = cv2.arcLength(corners[0], True)
65
66         # Pixel to cm ratio
67         pixel_cm_ratio = aruco_perimeter / 20
68
69         for cnt in contours:
70             # Get rect
71             rect = cv2.minAreaRect(cnt)
72             (x, y), (w, h), angle = rect
73
74             # Get Width and Height of the Objects by applying the Ratio pixel to cm
75             object_width = w / pixel_cm_ratio
76             object_height = h / pixel_cm_ratio
77
78             # Display rectangle
79             box = cv2.boxPoints(rect)
80             box = np.int0(box)
81
82             cv2.circle(img, (int(x), int(y)), 7, (0, 0, 255), -1)
83             cv2.polylines(img, [box], True, (255, 0, 0), 2)
84             cv2.putText(img, "Width {} cm".format(round(object_width, 1)), (int(x - 100), int(y - 20)), cv2.FONT_HERSHEY_PLAIN, 2, (115, 252, 173), 2)
85             cv2.putText(img, "Height {} cm".format(round(object_height, 1)), (int(x - 100), int(y + 20)), cv2.FONT_HERSHEY_PLAIN, 2, (115, 252, 173), 2)
86
87
88     img = cv2.resize(img,(0,0),None,0.5,0.5)
89     mask = cv2.resize(mask,(0,0),None,0.5,0.5)
90     edged = cv2.resize(edged,(0,0),None,0.5,0.5)
91     cv2.imshow("edged",edged)
92     cv2.imshow("mask",mask)
93     cv2.imshow("Image",img)
94
95     if cv2.waitKey(1)& 0xFF == ord('q'):
96         break
97
98 cap.release()
99 cv2.destroyAllWindows()

```