รายงานโครงงาน

ภาคการศึกษาที่ 1 ปีการศึกษา 2566

รายวิชา 523480 โครงงานวิศวกรรมคอมพิวเตอร์

(Computer Engineering Project)

เรื่อง

IQ Puzzler Solve

ผู้จัดทำ

นายธีรวัฒน์ กูดกิ่ง รหัสนักศึกษา B6321031

อาจารย์ที่ปรึกษาโครงงาน

อาจารย์ ดร.วิชัย ศรีสุรักษ์

สาขาวิศวกรรมคอมพิวเตอร์

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

กิตติกรรมประกาศ

โครงงานเรื่อง “IQ Puzzle Solver” สำเร็จลุล่วงได้ด้วยดีเนื่องจากได้รับความกรุณาอย่างสูงจาก อาจารย์ ดร.วิชัย ศรีสุรักษ์ สาขาวิศกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ที่กรุณาแนะนำ และชี้แนะมาโดยตลอด รวมถึงสนับสนุนอุปกรณ์ทางด้านฮาร์ดแวร์ต่างๆที่เหมาะสม ตั้งแต่เริ่มต้นโครงงานจนสำเร็จด้วยดี ผู้จัดทำโครงงานขอขอบคุณด้วยความเคารพอย่างสูงไว้ ณ โอกาสนี้

ผู้จัดทำ

(นายธีรวัฒน์ กูดกิ่ง)

หัวข้อโครงงาน : IQ Puzzler Solve

ประเภท : **การประยุกต์ปัญญาประดิษฐ์และการประมวลผลภาพ**

ผู้เสนอโครงงาน : **นายธีรวัฒน์ กูดกิ่ง รหัสนักศึกษา B6321031**

อาจารย์ที่ปรึกษาโครงงาน : **อาจารย์ ดร.วิชัย ศรีสุรักษ์**

ปีการศึกษา : **1/2566**

บทคัดย่อ

โครงงานนี้มีวัตุประสงค์เพื่อออกแบบและพัฒนาโปรแกรมสำหรับแก้ปัญหาปริศนาเกม IQ Puzzler PRO โดยตรวจจับพื้นที่ว่างสำหรับวางบล็อกที่เหลืออยู่ด้วยวิธีการประมวลผลภาพ เนื่องจากการวางของชิ้นบล็อกแต่ละชิ้นนั้นสามารถปรับเปลี่ยนทิศทางและรูปแบบการวางได้มากสุดถึงชิ้นละ 8 รูปแบบ อัลกอริทึมในการแก้ปัญหาปริศนาสามารถแก้ไขในจุดนั้นโดยไม่จำเป็นต้องวางภาพต้นแบบให้เหมาะสมกับพื้นที่ที่เหลืออยู่ ทำให้มีความสะดวกและยืดหยุ่นในการใช้งานสูง อีกทั้งยังใช้การตรวจจับขอบวงกลมของชิ้นบล็อกในการประมวลผลภาพเพื่อให้ได้เอาต์พุตเป็น array โดยระบบที่ผู้จัดทำพัฒนาขึ้นสามารถเรียนรู้และประมวลผลลัพธ์ของปริศนาได้อย่างมีประสิทธิภาพ

ผลของโครงงานแสดงให้เห็นว่า วิธีในการแก้ปัญหาของจำนวนชิ้นบล็อกที่มากขึ้นจะทำให้เวลาของการแก้ปัญหาเพิ่มขึ้นไปด้วย ผู้จัดโครงงานทำจึงนำวิธีการเปรียบเทียบผลลัพธ์ที่เคยทำแล้วกับภาพต้นแบบ เพื่อให้ผลลัพธ์ที่เร็วขึ้น และทำการบันทึกผลลัพธ์ที่สมบูรณ์ลงไปเพื่อใช้ในการใช้งานโปรแกรมครั้งถัดไป

สารบัญ

บทที่ 1

บทนำ

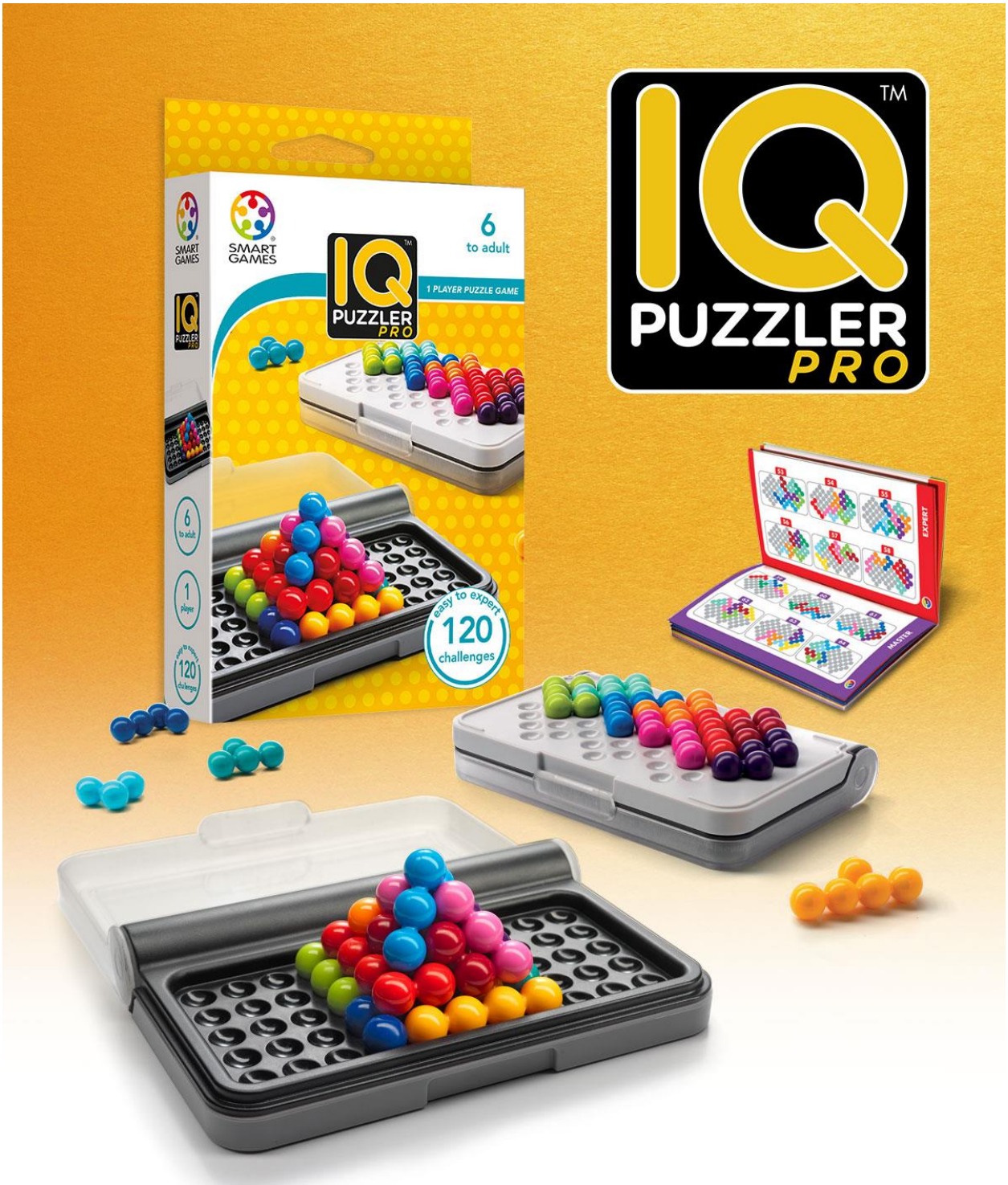
1.1 ความเป็นมาและความสำคัญ

ปัจจุบันมีของเล่นและเกมที่ใช้ฝึกฝนทักษะการคิด วิเคราะห์วางขายอยู่ทั่วไปทั้งในตลาดออนไลน์และท้องตลาดทั่วไป มีตั้งแต่ราคาถูกที่สามารถจับต้องได้ไปจนถึงหลักแสน แต่ก็มีของเล่นหรือเกมบางชนิดที่มีความยากในการเล่นที่แตกต่างกัน ทั้งในเรื่องของระยะเวลาในการเล่น ความยุ่งยากซับซ้อนของปริศนา และรูปแบบที่ไม่ซ้ำ เช่น LEGO , Tetris หรือแม้แต่เกมต่อบล็อกของเด็กประถม

ปัญหาเหล่านี้ อาจทำให้ผู้เล่นใช้เวลาอยู่กับของเล่นพวกนี้นานเกินไป จนทำให้เบื่อหรือเลิกเล่นไปในที่สุด แต่ปัจจุบันเทคโนโลยีปัญญาประดิษฐ์ (AI : Artificial Intelligence) กำลังเป็นที่นิยมในโลกออนไลน์ การเข้าถึงที่สะดวกและรวดเร็ว และสามารถใช้งานได้ทุกที่ ซึ่งมีผู้พัฒนาสิ่งนี้อยู่ทั่วโลก

ผู้จัดทำโครงงานจึงตระหนักได้ว่า จะดีกว่าไหมถ้าหากว่าเราสามารถใช้ปัญญาประดิษฐ์ในการแก้ไขปัญหาปริศนาเหล่านั้น และแสดงผลลัพธ์รวมไปถึงขั้นตอนของปริศนาต่าง ๆ ให้ผู้เล่นที่ไม่สามารถแก้ปริศนาได้ประหยัดเวลาในของเล่นหรือเกมที่พวกเขามี

จึงทำให้เกิดเป็นโครงงาน IQ Puzzler Solve ปัญญาประดิษฐ์ที่แก้ปัญหาการวางชิ้นส่วนบล็อกของ IQ Puzzler PRO ของเล่นพัฒนาทักษะการแก้ไขปัญหาการวางบล็อก ที่มีจำนวนจำกัดและพื้นที่ที่กำหนดมาให้ ผู้เล่นสามารถใช้กล้อง web cam เพื่อตรวจจับบล็อกแต่และบล็อกและพื้นที่วางบล็อกทั้งหมดของเกม IQ Puzzler PRO เพื่อให้ปัญญาประดิษฐ์ทำการบอกวิธีการวางบล็อกแบบสมบูรณ์ให้ผู้เล่น แล้วบันทึกผลลัพธ์ไว้เพื่อเปรียบเทียบกับการแก้ปัญหาครั้งหน้ากับรูปแบบที่ซ้ำหรือคล้ายคลึงกัน



รูปที่ 1.1 เกมทดสอบสมอง IQ Puzzler PRO

ที่มา (https://d32bxxnq6qs937.cloudfront.net/sites/default/files/smartgames-product-banner_IQ-Puzzler-Pro_0.jpg)

1.2 วัตถุประสงค์

- เพื่อตอบสนองความต้องการของผู้เล่นในการแก้ไขปัญหาปริศนาเกม IQ Puzzler PRO
- เพื่อให้ผลลัพธ์ที่ถูกต้องและแม่นยำสำหรับพื้นที่วางบล็อกแต่ละชิ้นของเกม IQ Puzzler PRO
- เพื่อลดความเครียดและเวลาในการวิเคราะห์ปริศนาเกม IQ Puzzler PRO ของผู้เล่น
- เพื่อใช้งาน Software ควบคู่กับ Hardware ได้อยากถูกต้องและเหมาะสม
- เพื่อให้ได้ Software ที่สามารถนำไปใช้งานได้จริง

1.3 ตัวชี้วัด และการบรรลุวัตถุประสงค์

- สามารถแก้ไขปัญหาการวางบล็อกเกม IQ Puzzler PRO ที่มีให้ในพื้นที่ที่กำหนดไว้ได้
- สามารถประยุกต์ใช้งานกับกล้อง web camera แบบเรียลไทม์ได้
- สามารถตรวจจับชิ้นบล็อกของเกม IQ Puzzler PRO และแยกประเภทได้
- สามารถตรวจจับพื้นที่ทั้งหมดของเกม IQ Puzzler PRO แล้วแปลงค่าเป็น array ได้
- สามารถเปรียบเทียบผลลัพธ์ที่คล้ายคลึงกับผลลัพธ์ที่เคยทำมาก่อน

1.4 ขอบเขตของโครงงาน

- พัฒนาปัญญาประดิษฐ์ให้สามารถใช้งานได้กับวัตถุของจริง
- วิเคราะห์รูปแบบการวางบล็อกของเกม IQ Puzzler PRO ที่มีในพื้นที่ที่กำหนดได้ถูกต้องและรวดเร็ว

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- ผู้จัดทำโครงงานได้นำความรู้มาประยุกต์ใช้ให้เข้ารูปแบบธุรกิจอุตสาหกรรม
- ผู้เล่นหรือผู้ใช้งานมีความสนุกสนานและคลายเครียดในการแก้ปัญหาปริศนาเกม IQ Puzzler PRO
- ลดปัญหาในการใช้เวลาเพื่อวิเคราะห์ผลลัพธ์ของปัญหาเกม IQ Puzzler PRO
- ตัวโครงงานหรือเอกสารที่เกี่ยวข้อง เป็นแนวทางในการพัฒนาปัญญาประดิษฐ์ และการตรวจจับวัตถุ แก่ผู้ที่สนใจศึกษา

1.6 แผนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงาน

หัวข้อการทำงาน	ระยะเวลา			
	ก.ค. 2566	ส.ค. 2566	ก.ย. 2566	ต.ค. 2566
เขียนโปรแกรมสำหรับการแก้ปัญหา	วันที่ 24 ถึง วันที่ 31	-	-	-
ทำ model object detection	-	วันที่ 1 ถึง วันที่ 31	วันที่ 1 ถึง วันที่ 30	-
ใช้กล้อง web camera ในการทดสอบ	-	-	-	วันที่ 1 ถึง วันที่ 10
เขียนโปรแกรมสำหรับ การเปรียบรูปภาพ	-	-	-	วันที่ 11 ถึง วันที่ 20
ปรับค่าพารามิเตอร์ทั้งหมดให้สมบูรณ์	-	-	-	วันที่ 21 - วันที่ 26

บทที่ 2

เอกสารและทฤษฎีที่เกี่ยวข้อง

โครงงานนี้ใช้ภาษาโปรแกรม Python ในการเขียนอัลกอริทึมในการแก้ปัญหาการวางบล็อกของเกม IQ Puzzler Pro ซึ่งทำงานบนแพลตฟอร์มของ Google Colab ในการวิเคราะห์คำตอบที่มีความไปได้จำนวนมาก และใช้ OpenCV บนแพลตฟอร์มของ Jupyter notebook ในการประมวลผลภาพ (Image Processing) โดยใช้วิธีการตรวจจับวัตถุ (Object Detection) เพื่อให้แยกชนิดของบล็อกเกม IQ Puzzler Pro แต่ละชนิด ซึ่งมรกระบวนการประมวลผลภาพต่าง ๆ ที่เกี่ยวข้องเช่น การหาขอบของภาพ (Edge Detection) และ Hough transforms เป็นต้น

2.1 Python

Python เป็นภาษาการเขียนโปรแกรมที่ใช้อย่างแพร่หลายในเว็บแอปพลิเคชัน การพัฒนาซอฟต์แวร์ วิทยาศาสตร์ข้อมูล และแมชชีนเลิร์นนิง (ML) นักพัฒนาใช้ Python เนื่องจากมีประสิทธิภาพ เรียนรู้ง่าย และสามารถทำงานบนแพลตฟอร์มต่างๆ ได้มากมาย ทั้งนี้ซอฟต์แวร์ Python สามารถดาวน์โหลดได้ฟรี ผสานการทำงานร่วมกับระบบทุกประเภท และเพิ่มความเร็วในการพัฒนา ซึ่งผู้จัดทำโครงงานใช้ภาษา Python ในการเขียนโปรแกรมเป็นหลัก

2.2 Object Detection

Object Detection (การตรวจจับวัตถุ) คือ เทคโนโลยีในทางคอมพิวเตอร์ หลักการที่เกี่ยวกับ Computer Vision และ Image Processing ที่ใช้ในงาน AI ตรวจจับวัตถุชนิดที่กำหนด เช่น มนุษย์ รถยนต์ อาคาร ที่อยู่ในรูปภาพ หรือวิดีโอ

2.3 Computer Vision

Computer Vision (คอมพิวเตอร์วิทัศน์) คือ แขนงหนึ่งของวิทยาการปัญญาประดิษฐ์ หรือ AI โดยการสร้างอวัยวะที่เสมือนดวงตาให้คอมพิวเตอร์หรือระบบ ทำให้สามารถจดจำ เข้าใจ และวิเคราะห์ข้อมูลภาพได้ เช่น รูปภาพนิ่ง วิดีโอ หรืออินพุตต่างๆ ที่เน้นด้านภาพเป็นหลัก ได้อย่างชาญฉลาด แม่นยำ และตอบสนองต่อข้อมูลภาพที่มองเห็นได้อย่างรวดเร็ว

2.4 Image Processing

Image Processing (การประมวลผลภาพ) หมายถึง กระบวนการจัดการและวิเคราะห์รูปภาพให้เป็นข้อมูลในแบบดิจิทัล โดยใช้คอมพิวเตอร์การ เพื่อให้ได้ข้อมูลที่เราต้องการทั้งในเชิงคุณภาพและปริมาณ (ขนาด รูปร่าง) หลังจากนั้นเราสามารถนำข้อมูลไปวิเคราะห์ และสร้างเป็นระบบ

2.5 OpenCV

OpenCV คือไลบรารีโอเพ่นซอร์สที่นิยมสำหรับการประมวลผลภาพขั้นพื้นฐาน เช่น การเบลอภาพ การผสมภาพ การเพิ่มคุณภาพของภาพ เพิ่มคุณภาพของวิดีโอ การรู้จำวัตถุต่าง ๆ ในภาพ หรือ การตรวจจับใบหน้าหรือวัตถุต่าง ๆ ในภาพและวิดีโอได้

2.6 Edge Detection

Edge Detection (การหาขอบของภาพ) เป็นการหาเส้นขอบของวัตถุที่อยู่ในภาพ หาได้จาก การใช้หลักการหาความชันของค่า intensity หากรูปนั้นไม่มีขอบภาพ ค่า intensity ของแต่ละ pixel ก็จะใกล้เคียงกัน ทำให้ไม่เกิดความชัน ซึ่งโดยปกติความชันสามารถหาได้จาก

$$f'(x) = \frac{df}{dx}(x)$$

แต่เนื่องจากค่า intensity ไม่ได้เป็นแบบ continuous เราจึงหาความชันในรูปแบบของ discrete แทน ซึ่งสามารถทำได้โดย

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{(u+1) + (u-1)} = \frac{f(u+1) - f(u-1)}{2}$$

2.7 Gaussian Blur

Gaussian blur เป็นการเบลอภาพด้วยการใช้ค่าเฉลี่ยแบบถ่วงน้ำหนัก ซึ่งภาพจะเบลอน้อยลงแต่เบลออย่างเป็นธรรมชาติ นอกจากนี้จากนี้จะสามารถรักษาเส้นขอบในภาพได้มากขึ้นเมื่อเทียบกับการเบลอภาพแบบอื่น ๆ Gaussian function คือ

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{\sigma^2}}$$

2.8 Hough Transform

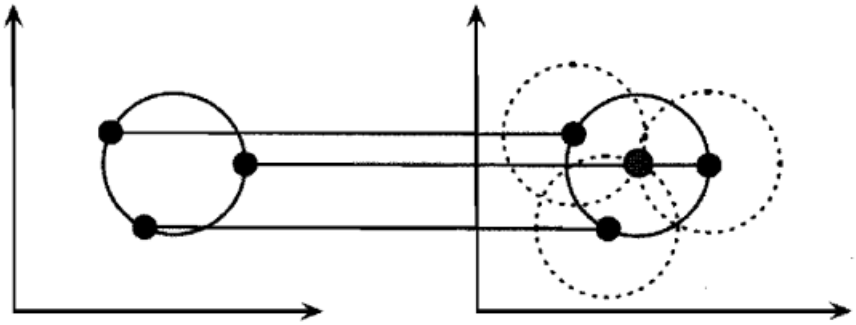
เป็นวิธีการที่นิยมในการวิเคราะห์หาวัตุภายในภาพที่มีรูปทรงเรขาคณิต (Parametric Shape) ที่สามารถสร้างรูปดังกล่าวจากสมการ เช่น วงกลม เส้นตรง โดยวิธีการประมวลผลเพื่อหาวัตถุดังกล่าวภายในภาพด้วย Hough Transform โดยในโครงงานนี้เราจะใช้วิธี Hough Circle Transform ในการวิเคราะห์หาวัตถุรูปทรงกลมภายในภาพ

2.9 Hough Circle Transform

การคำนวณหาตำแหน่งวงกลมในภาพด้วยวิธี Hough Circle ทำได้โดยสร้าง Accumulator (A) ของรัศมีที่ต้องการหา (r) แล้วทำการวาดวงกลมใน A ด้วยรัศมี r ของทุกจุดสีดำ (X,Y) ภายในภาพ ซึ่งตำแหน่งมี่มีค่า vote สูงของ A คือตำแหน่งที่พบวงกลมรัศมีดังกล่าว

จากสมการ $(x-a)^2 + (y-b)^2 = r^2$

และ $x = a + r \cdot cos\theta\ ,\ y = b + r \cdot sin\theta$



รูปที่ 2.1 การหาวงกลมด้วยวิธีการ Hough Circle Transform (ซ้าย) เป็นรูปภาพที่ต้องการหาวงกลมรัศมี R ซึ่งมีจุดอยู่ 3 จุด , (ขวา) วาดวงกลมรัศมี R ใน Parametric space (accumulator array) โดยจุดที่ตัดกันมากที่สุดคือจุดที่มีค่า vote สูงสุดซึ่งเป็นจุดศูนย์กลางของคำตอบ

บทที่ 3

ขั้นตอนและวิธีการดำเนินการ

3.1 กล่าวนำ

โครงงานนี้มุ่งเน้นการออกแบบและพัฒนาโปรแกรมให้สะดวกในการใช้งาน มีความยืดหยุ่นในการประมวลผลชิ้นส่วนบล็อกเกม IQ Puzzler PRO ได้หลายรูปแบบ และมีความสามารถในการแก้ปัญหาปริศนาได้อย่างรวดเร็วและถูกต้อง โดยระบบจะเปรียบเทียบภาพต้นแบบกับข้อมูลปริศนาที่เคยแก้ไว้ก่อนหน้านี้ทั้งหมดหากไม่พบคำตอบ ให้เรียนรู้ภาพต้นแบบของพื้นที่ที่กำหนดให้และบล็อกที่เหลืออยู่ของเกม IQ Puzzler PRO แล้วประมวลผลหาผลลัพธ์ที่ถูกต้องที่สุดให้กับผู้เล่น จากนั้นทำการบันทึกผลเพื่อใช้เปรียบเทียบกับผลรับในครั้งถัดไป

3.2 อุปกรณ์ที่ใช้

- ภาษาโปรแกรม Python รวมไปถึง OpenCV library
- แพลตฟอร์มการทำงาน (Google Colab , Jupyter notebook)
- กล้องเว็บแคม (Web camera) , กล้องถ่ายภาพแบบดิจิตอล (Digital Camera)
- ชุดของเล่น IQ Puzzler PRO (https://www.smartgames.eu/uk/one-player-games/iq-puzzler-pro)

3.3 ขั้นตอนการดำเนินการ

3.3.1 ค้นหว้าและจัดเตรียมข้อมูลที่จำเป็นต่อการทำโครงงาน

- จัดหาชิ้นงานตามร้านค้าออนไลน์หรือร้านค้าทั่วไป (https://www.smartgames.eu/uk/one-player-games/iq-puzzler-pro)
- แต้มจุดให้กับบล็อกที่สีเหมือนกัน เนื่องจากบล็อกทั้งหมดที่ได้มามี 12 ชิ้น แต่มีสีของบล็อกแค่ 6 สี จึงต้องนำสีมาแต้มจุดเพื่อสร้างความแตกต่างของบล็อกที่มีสีเดียวกัน

รูปที่ 3.3.1 ตัวอย่างการแต้มจุดของบล็อกที่มีสีเดียวกัน

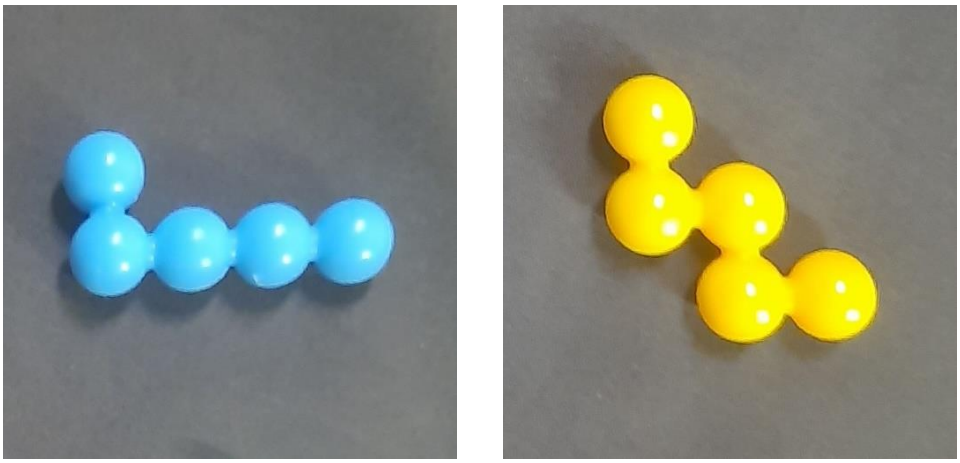
- สืบค้นรูปแบบโปรแกรมและเกมปริศนาที่คล้ายคลึงกัน

รูปที่ 3.3.2 เกมบล็อกปริศนา IQ Block Puzzle

รูปที่ 3.3.3 เกมวงจรปริศนา IQ Circuit

3.3.2 เขียนโปรแกรมอัลกอริทึมในการแก้ปัญหา

- กำหนดชุดข้อมูลทดสอบเพื่อใช้ในการทำงาน โดยการกำหนดข้อมูลของบล็อกเป็น array



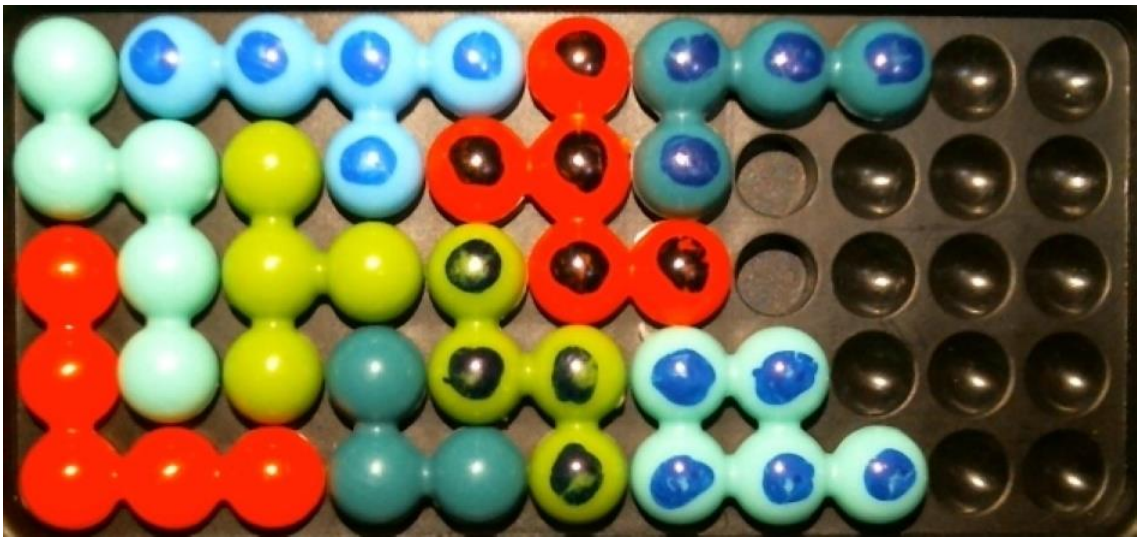
รูปที่ 3.3.4 ภาพตัวอย่างบล็อก สามารถแปลงเป็นค่า array ได้ดังนี้

Blue = [1,0,0,0],[1,1,1,1] (ภาพซ้าย) / Yellow = [1,0,0],[1,1,0],[0,1,1] (ภาพขวา)

- ผลลัพธ์ของการกำหนดบล็อกทั้งหมดที่ได้

```
import numpy as np
blue         = np.array([[1, 1, 1, 1], [1, 0, 0, 0]])
blue_dot     = np.array([[1, 1, 1, 1], [0, 1, 0, 0]])
green        = np.array([[1, 1], [1, 0]])
green_dot    = np.array([[0, 0, 1], [1, 1, 1]])
lb_dot       = np.array([[1, 1, 0], [1, 1, 1]])
lg_dot       = np.array([[1, 1, 0], [0, 1, 1]])
light_blue   = np.array([[0, 1, 1, 1], [1, 1, 0, 0]])
light_green = np.array([[0, 1, 0], [1, 1, 1]])
orange       = np.array([[0, 0, 1], [0, 0, 1], [1, 1, 1]])
orange_dot   = np.array([[0, 1, 1], [1, 1, 0], [0, 1, 0]])
yellow       = np.array([[0, 0, 1], [0, 1, 1], [1, 1, 0]])
yellow_dot   = np.array([[1, 0, 1], [1, 1, 1]])
```

- ในส่วนของพื้นที่เราจะให้ค่าพื้นที่ว่างเป็น ‘0’ ส่วนพื้นที่ที่มีบล็อกวางไว้อยู่แล้วจะเป็น ‘X’



รูปที่ 3.3.5 ภาพตัวอย่างพื้นที่บล็อก ที่มีขนาดกลว้าง 11 ช่อง สูง 5 ช่อง สามารถแปลงเป็นค่า array ได้ดังนี้

Area = [['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0'],
['X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0', '0'],
['X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0', '0'],
['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0'],
['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0']]

- เขียนฟังก์ชันตรวจสอบพื้นที่ที่มีว่าสามารถวางบล็อกได้หรือไม่

```
def can_place_block(block, row, col):
    block_rows, block_cols = block.shape
    return (
        row + block_rows <= area.shape[0] and
        col + block_cols <= area.shape[1] and
        not np.any(block * (area[row:row + block_rows, col:col + block_cols] != '0'))
```

***หมายเหตุ** : ฟังก์ชัน can_place_block(): เป็นฟังก์ชันที่ใช้เพื่อตรวจสอบพื้นที่ว่างใน area ว่าว่างหรือไม่ เพื่อที่จะทำการวางบล็อกที่ต้องการลงไป โดยมีเงื่อนไขคือ 1. จะต้องไม่ออกนอกขอบเขตของ area ที่มีขนาดแถวเป็น 11 ช่อง และขนาดคอลัมนเป็น 5 ช่อง และ 2. พิกัดใน area ที่กำลังจะวางต้องมีค่า ‘0’ หมายความว่า พื้นที่พิกัดนั้นสามารถรถวางได้เพราะมีที่ว่างอยู่*

- เขียนฟังก์ชันการวางบล็อกที่ต้องการ

```
def place_block(block, row, col, block_name):
    area[row:row + block.shape[0], col:col + block.shape[1]][block == 1] = block_name
```

***หมายเหตุ** : ฟังก์ชัน place_block(): เป็นฟังก์ชันที่ใช้เพื่อวางบล็อกที่พิกัดใน area โดยจะแทนค่าของตัวมันลงไปในพิกัดนั้น ๆ*

- เขียนฟังก์ชันการลบบล็อกที่ต้องการ

```
def remove_block(block, row, col):
    area[row:row + block.shape[0],col:col+block.shape[1]][block == 1] = '0'
```

***หมายเหตุ** : ฟังก์ชัน remove_block(): เป็นฟังก์ชันที่ใช้เพื่อลบบล็อกที่พิกัดใน area โดยจะแทนค่า '0' ลงไปในพิกัดนั้น ๆ เพื่อที่จะให้บล็อกอื่น ๆ มาแทนที่ในอนาคต*

- เขียนฟังก์ชันการแก้ไขปัญหา

```
def solve(block_names, remaining_blocks):
    if len(block_names) == 3:
        return True
    for row in range(area.shape[0]):
        for col in range(area.shape[1]):
            for block_name in remaining_blocks:
                block_matrix = blocks[block_name]
                for _ in range(8):
                    if can_place_block(block_matrix, row, col):
                        place_block(block_matrix, row, col, block_name)
                        block_names.append(block_name)
                        remaining_blocks.remove(block_name)
                        if solve(block_names,remaining_blocks):
                            return True
                        remove_block(block_matrix, row, col)
                        block_names.pop()
                        remaining_blocks.append(block_name)
                    block_matrix = np.rot90(block_matrix) if _ < 4 else np.flipud(block_matrix)
```

หมายเหตุ *: ฟังก์ชัน solve(): เป็นฟังก์ชันที่ใช้เพื่อแก้ปริศนาการวางบล็อกที่เป็นแบบฟังก์ชันเวียนบังเกิด (recursive function) โดยมีเงื่อนไขการจบฟังก์ชัน คือ เมื่อมีจำนวนบล็อกในลิสต์บล็อกที่วางแล้วจนครบ (ในตัวอย่างโค้ดนี้จะใช้เป็น 3 บล็อก) กรณีนี้ขอยกตัวอย่างชื่อบล็อกเป็น (A, B, C) โดยเริ่มจากบล็อก A ตามลำดับ ซึ่งการทำงานของฟังก์ชันเริ่มจากการตรวจสอบพื้นที่การวางบล็อกว่าสามารถวางได้ไหม หากวางได้ให้ทำการวางบล็อก A ลงไปที่พิกัดนั้น แล้วนำบล็อก A ไปใส่ในลิสต์ของบล็อกที่วางแล้ว จากนั้นเรียกใช้ฟังก์ชัน solve(): อีกครั้งเพื่อดึงบล็อกในลิสต์บล็อกที่เหลืออยู่ (B, C) ตามลำดับ หากสามารถวางบล็อก B ได้ให้ทำการนำบล็อก B ไปใส่ไว้ในลิสต์บล็อกที่วางแล้วกับบล็อก A แล้วเรียกใช้ฟังก์ชัน solve(): อีกครั้งที่บล็อก C ถ้าบล็อก C ไม่มีพื้นที่ที่สามารถวางได้ให้ทำการหมุนบล็อก C 90 องศา หากหมุดครบ 360 องศาแล้วยังไม่สามารถวางบล็อก C ได้ให้กลับด้านแนวตั้งแล้วทำการหมุน 90 อีก หากหมุนจนครบ 360 อีกแสดงว่าบล็อก C ไม่สามรถวางได้ ให้ทำการดึงบล็อก B ออกมาในลิสต์บล็อกที่เหลืออยู่ก่อนแล้วให้บล็อก C ใช้งานฟังก์ชัน solve() ถ้าบล็อก C วางได้ ให้เรียกใช้ฟังก์ชัน solve(): ที่บล็อก B ต่อ หากบล็อก B วางได้แสดงว่าจำนวนบล็อกที่ถูกวางแล้วเท่ากับจำนวนบล็อกที่มีตอนเริ่ม ดังนั้นเงื่อนไขการการจบฟังก์ชันเป็นจริง จึงจบการทำงาน*

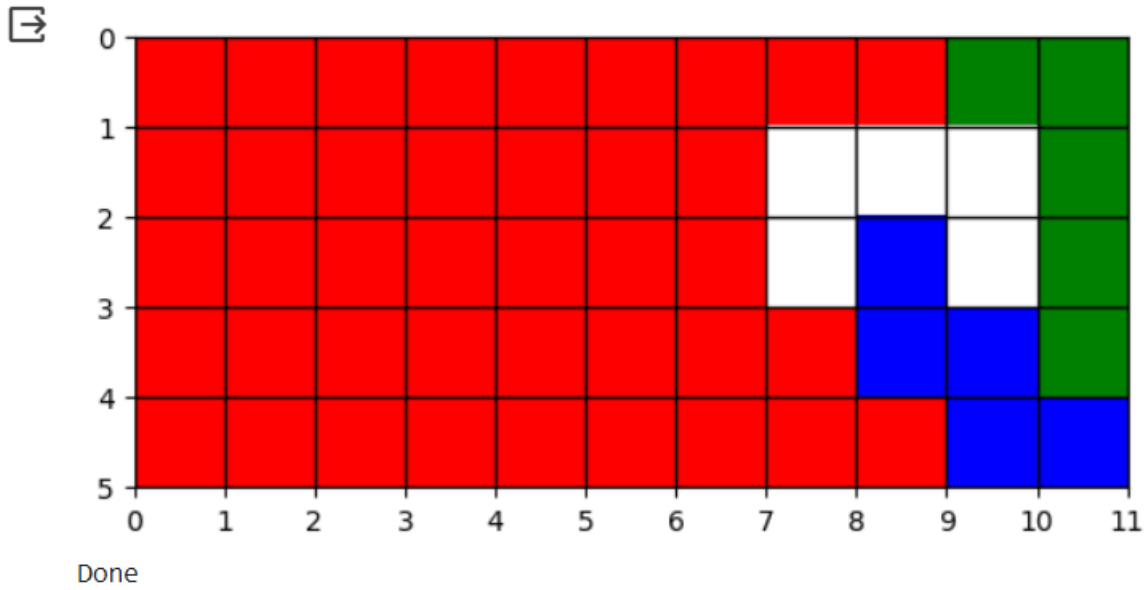
- อัลกอริทึมโปรแกรมแก้ไขปัญหาปริศนาทั้งหมด

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
blocks = {
    'A': np.array([[1, 0, 0, 0], [1, 1, 1, 1]]),
    'B': np.array([[1, 1, 1], [1, 0, 1]]),
    'C': np.array([[1, 0, 0], [1, 1, 0], [0, 1, 1]]),}
area = np.array([
    ['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0'],
    ['X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0', '0'],
    ['X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0', '0'],
    ['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0'],
    ['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0']],dtype='str')

def can_place_block(block, row, col):
    block_rows, block_cols = block.shape
    return (
        row + block_rows <= area.shape[0] and
        col + block_cols <= area.shape[1] and
        not np.any(block * (area[row:row + block_rows, col:col + block_cols] != '0'))
```

```
def place_block(block, row, col, block_name):
    area[row:row + block.shape[0], col:col + block.shape[1]][block == 1] = block_name

def remove_block(block, row, col):
    area[row:row + block.shape[0],col:col+block.shape[1]][block == 1] = '0'

def solve(block_names, remaining_blocks):
    if len(block_names) == 3:
        return True
    for row in range(area.shape[0]):
        for col in range(area.shape[1]):
            for block_name in remaining_blocks:
                block_matrix = blocks[block_name]
                for _ in range(8):
                    if can_place_block(block_matrix, row, col):
                        place_block(block_matrix, row, col, block_name)
                        block_names.append(block_name)
                        remaining_blocks.remove(block_name)
                        if solve(block_names,remaining_blocks):
                            return True
                        remove_block(block_matrix, row, col)
                        block_names.pop()
                        remaining_blocks.append(block_name)
                    block_matrix = np.rot90(block_matrix) if _ < 4 else np.flipud(block_matrix)
block_names = []
remaining_blocks = list(blocks.keys())
if solve(block_names, remaining_blocks):
  block_colors = {'A': 'red', 'B': 'blue', 'C': 'green'}
  cmap_colors = [block_colors.get(block_name, 'white') for block_name in block_names]
  cmap_colors.append('white')
  cmap = ListedColormap(cmap_colors)
  colored_area = np.zeros_like(area, dtype=int)
  for i, block_name in enumerate(block_names):
    colored_area[area == block_name] = i + 1
```

```
    colored_area[area == '0'] = 0


  plt.imshow(colored_area, cmap=cmap, interpolation='none', vmin=0, vmax=len(block_names)  ,extent=[0,
area.shape[1], area.shape[0], 0])
  plt.xticks(range(area.shape[1]))
  plt.yticks(range(area.shape[0]))
  plt.grid(True, color='black', linewidth=1)
  plt.show()
else:
  print("No solution found")
print("Done")
```

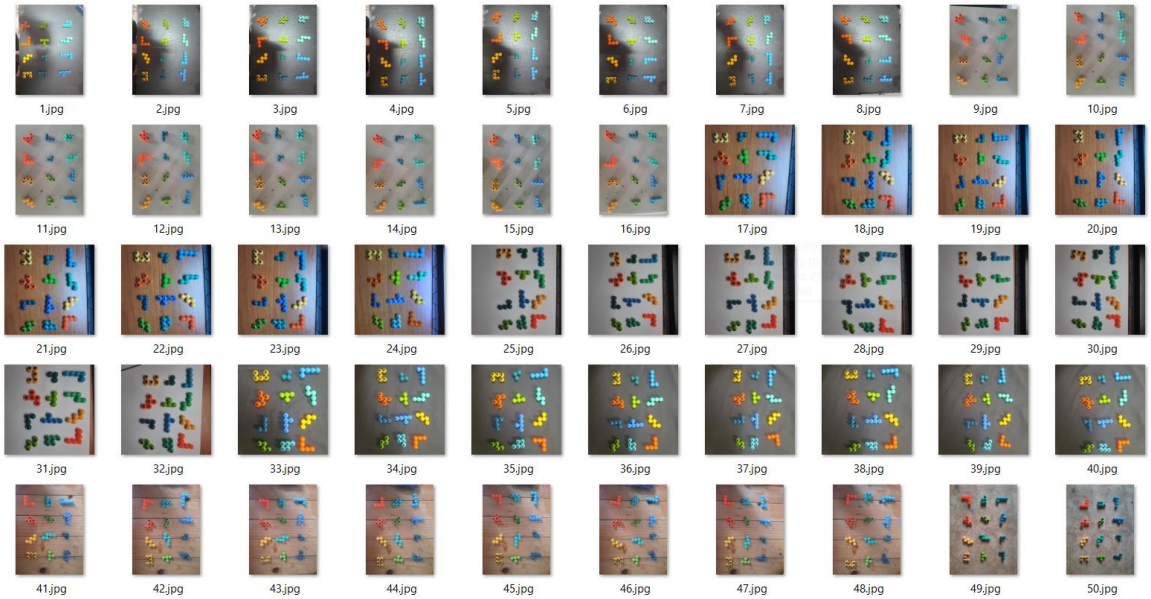
หมายเหตุ *: การเขียนโปรแกรมทดสอบนี้ใช้การกำหนดค่า พื้นที่ และ รูปแบบของบล็อก แบบ Hard code โดยใช้รูปที่ 3.3.5 เป็นต้นแบบ และผลลัพธ์ที่ได้เป็นดัง รูปที่ 3.3.6*



รูปที่ 3.3.6 ผลลัพธ์การทดสอบโปรแกรม

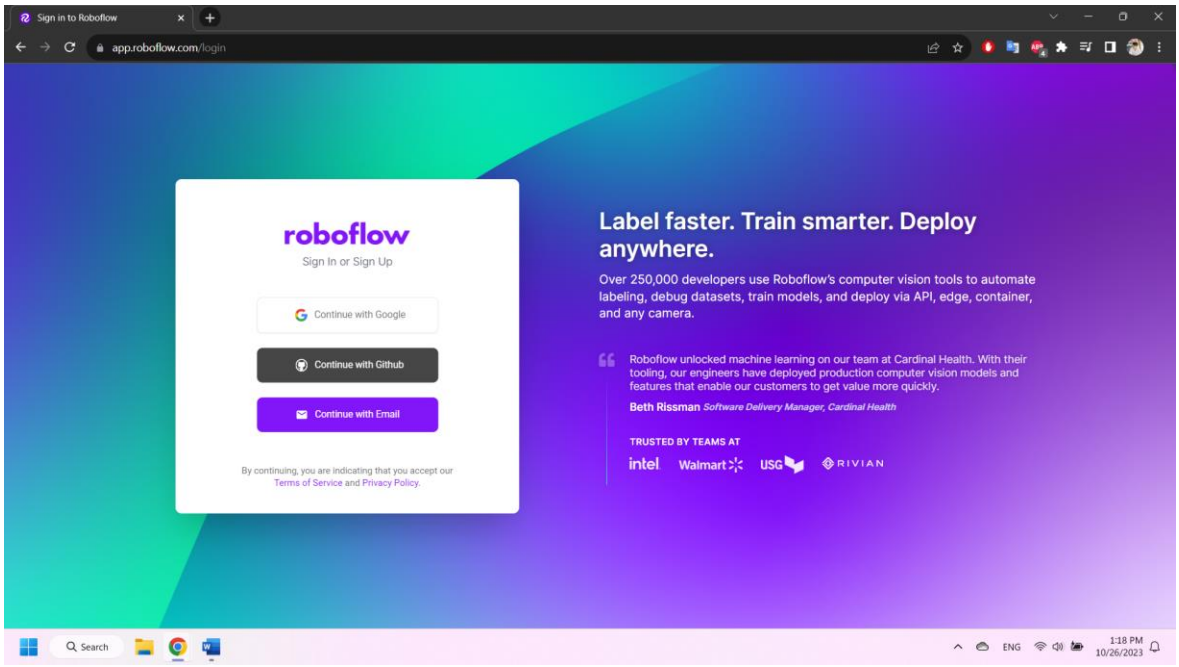
3.3.3 ทำ Model Object Detection ด้วย YOLOv8 บนแพลตฟอร์ม roboflow

- เตรียมข้อมูลที่ต้องการเทรน (รวบรวมชุดข้อมูลรูปภาพที่มีบล็อกทั้งหมด 12 ชิ้น)



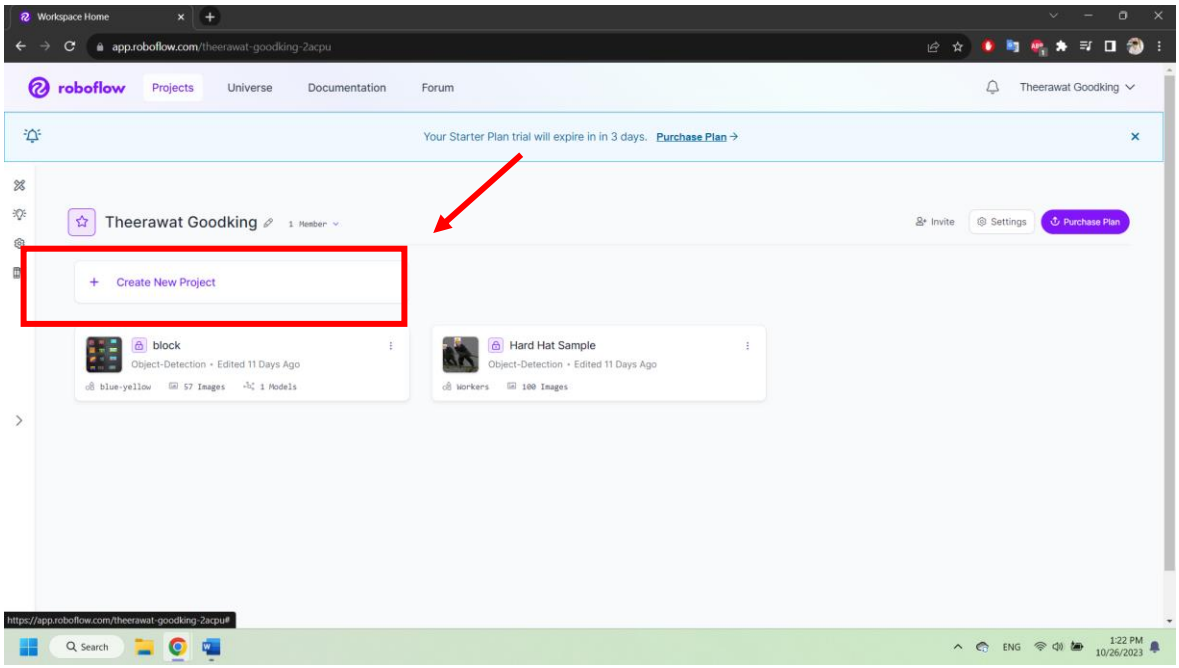
รูปที่ 3.3.7 ภาพตัวอย่างชุดข้อมูลของบล็อกทั้งหมด

- เข้าสู่ระบบเว็บไซต์ของ roboflow (*https://app.roboflow.com/login*)



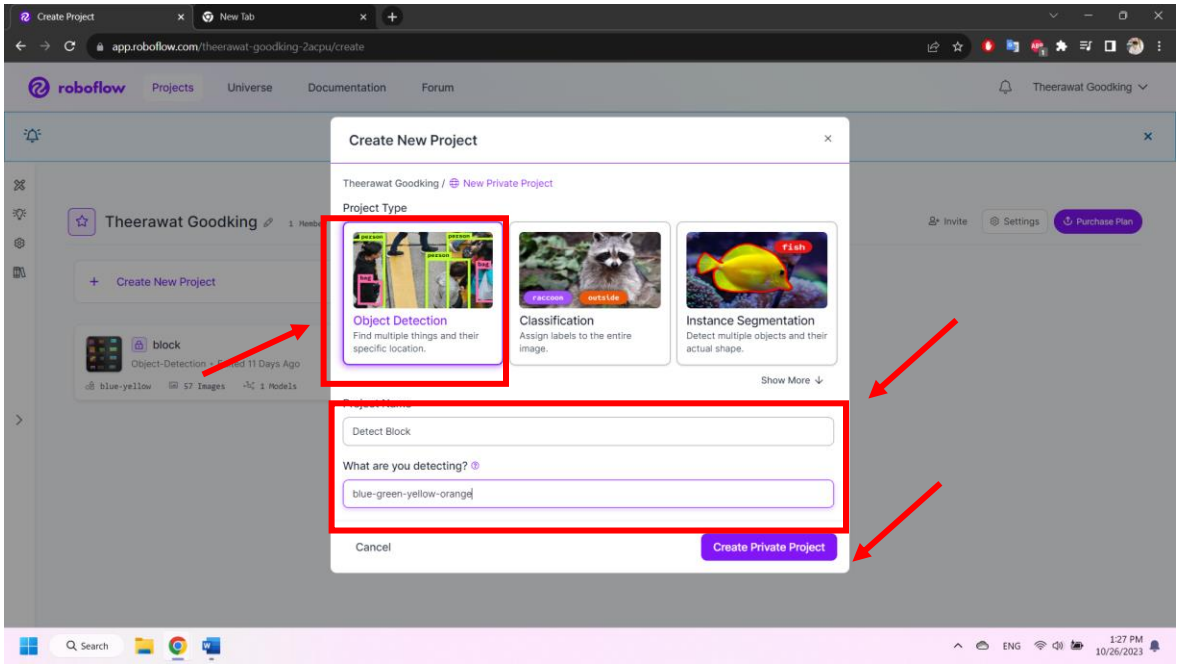
รูปที่ 3.3.8 หน้าเข้าสู่ระบบของ roboflow (https://app.roboflow.com/login)

- หลังจากเข้าสู่ระบบแล้วให้ทำการกดเลือก Create New Project



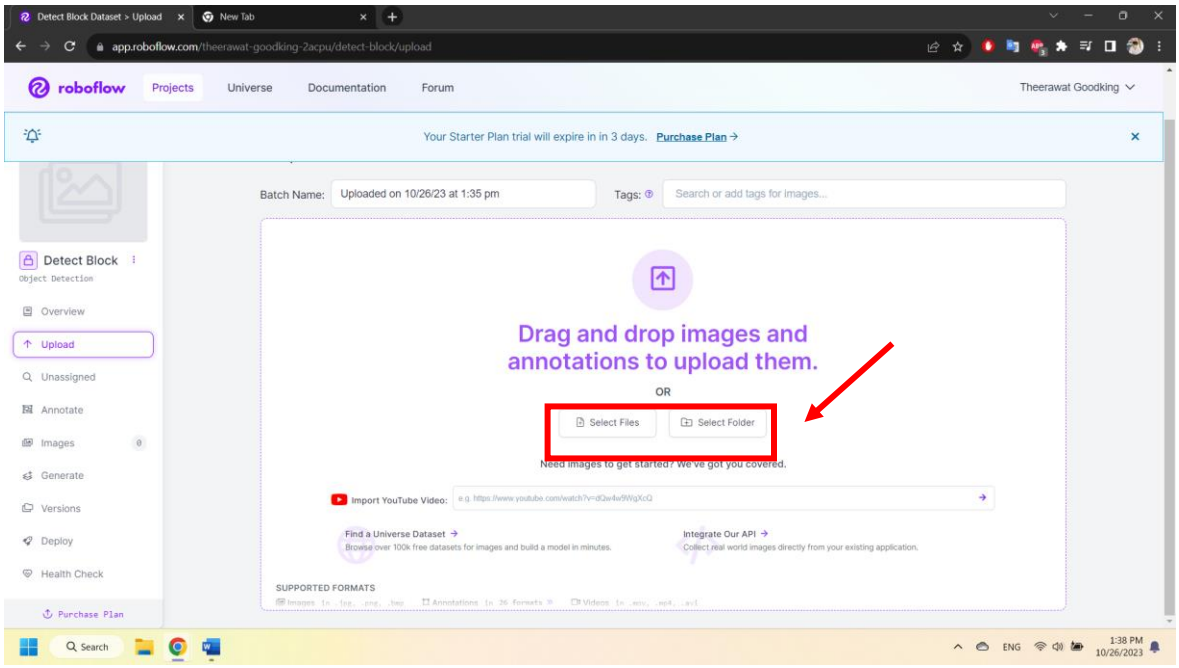
รูปที่ 3.3.9 หน้าแรกหลังจากเข้าสู่ระบบของ roboflow

- การสร้างโปรเจคบน roboflow



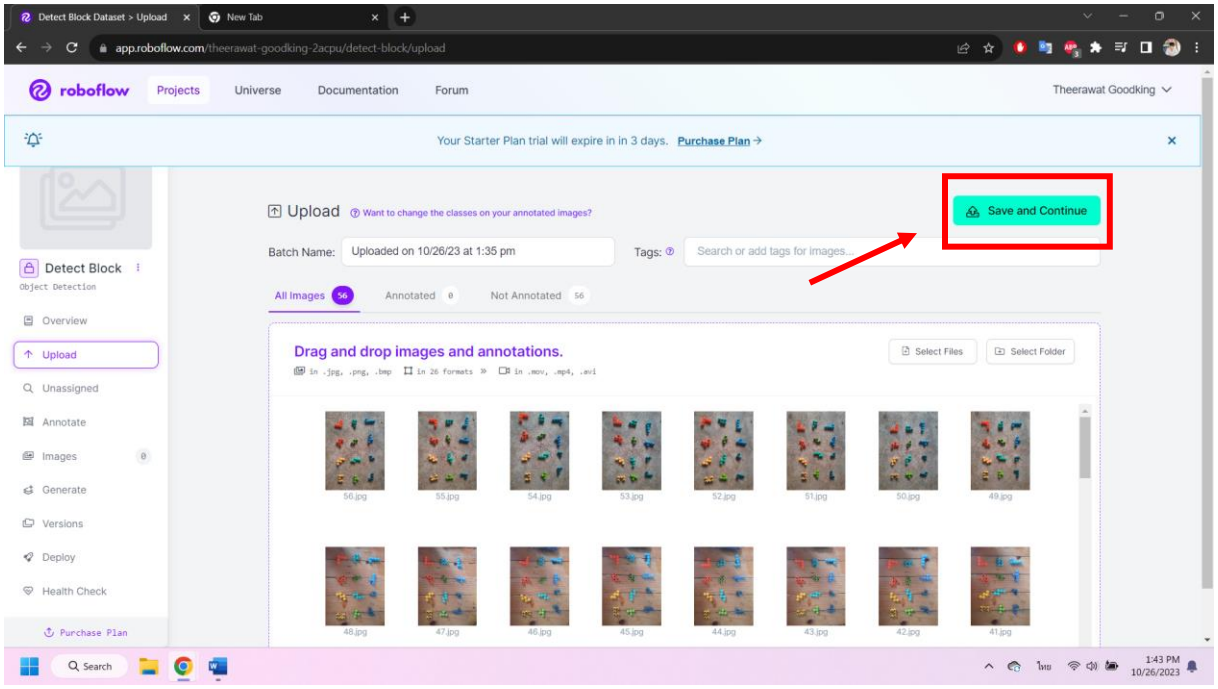
รูปที่ 3.3.10 เลือกประเภทของโปรเจค ในที่นี้ผู้จัดทำจะใช้การตรวจจับวัตถุ (Object Detection) จากนั้นให้ทำการใส่ชื่อโปรเจค และใส่ชนิดวัตถุที่ต้องการลงไป จากนั้นให้กด Create Private Project

- หลังจากกด แล้วให้ทำการ Upload ไฟล์ภาพชุดข้อมูลที่เราเตรียมไว้ลงไป



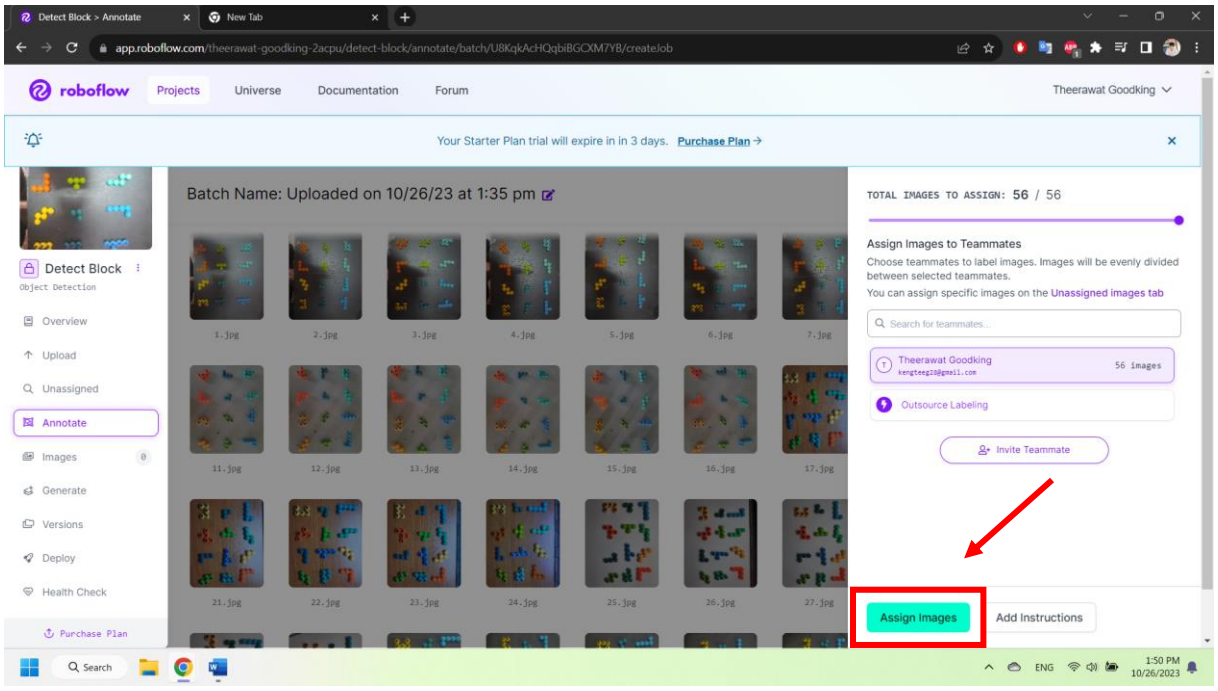
รูปที่ 3.3.11 หน้า Upload ไฟล์ภาพ โดยสามารถเลือกได้ว่าจะ Upload เป็นไฟล์หรือโฟลเดอร์ก็ได้

- หลังจาก Upload ไฟล์ทั้งหมดแล้วให้กด Save and Continue



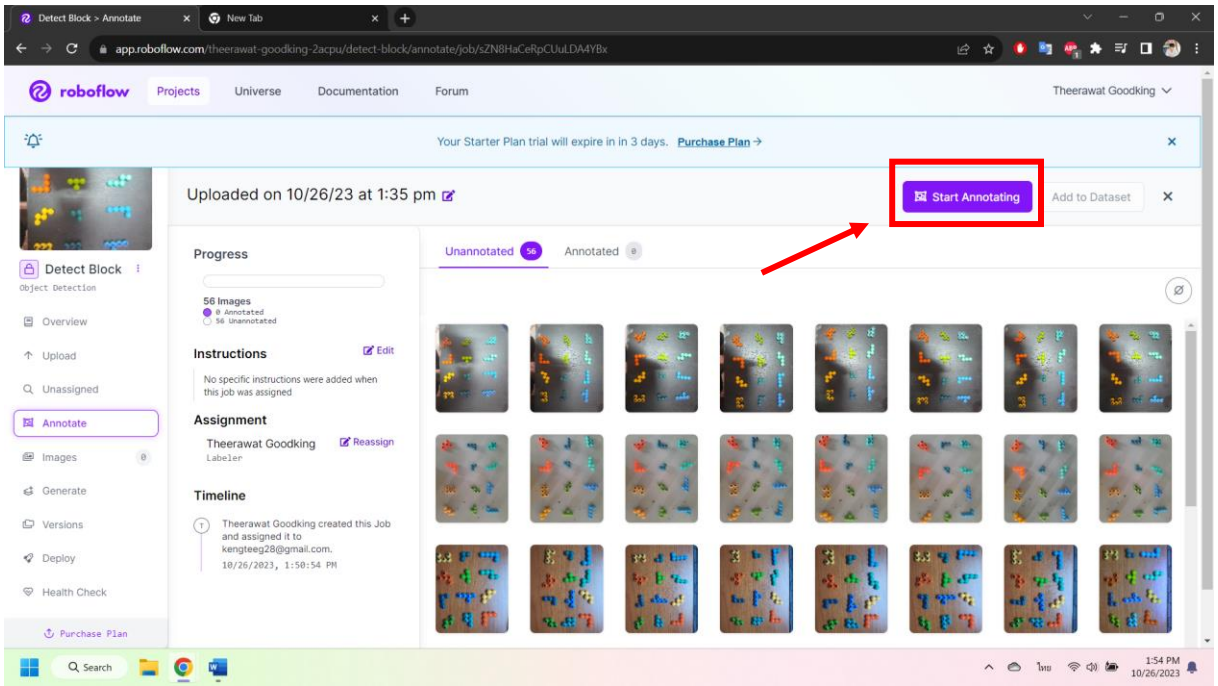
รูปที่ 3.3.12 ผลลัพธ์หลังจากการ Upload ไฟล์

- จากนั้นให้ทำการกด Assign Images เพื่อเสร็จสิ้นการ Upload ทั้งหมด



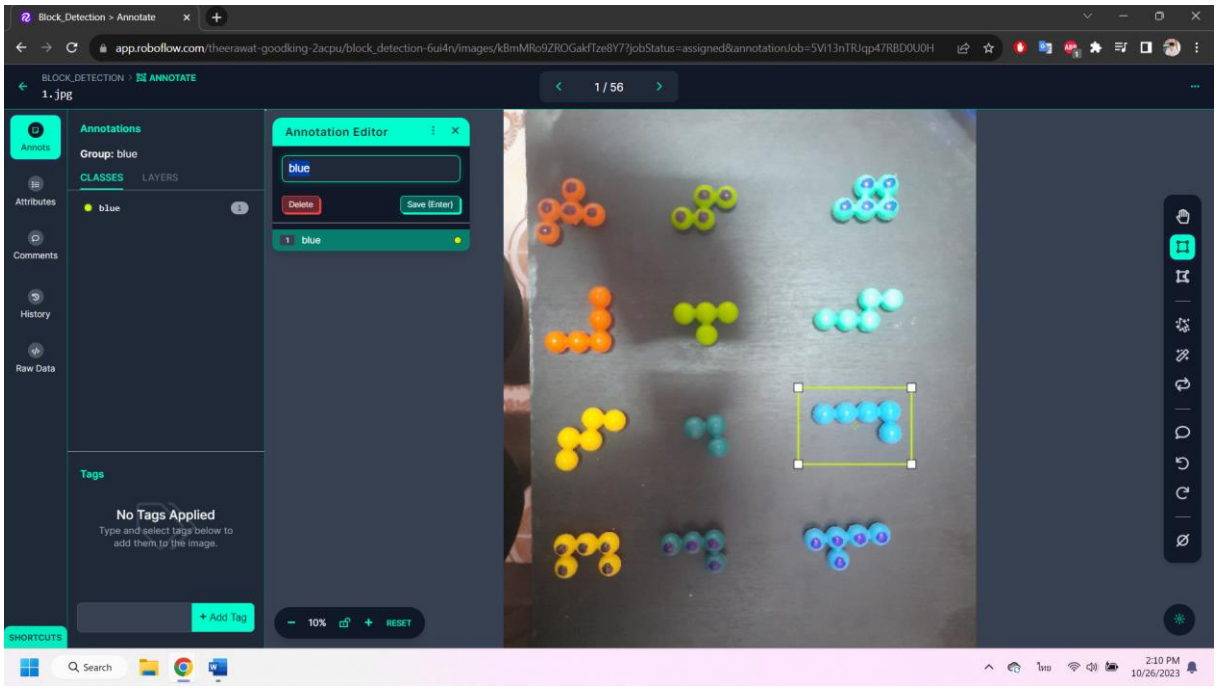
รูปที่ 3.3.13 ผลลัพธ์หลังจากการกด Save and Continue

- ทำการกด Start Annotating เพื่อเริ่มทำการ Label ของภาพ



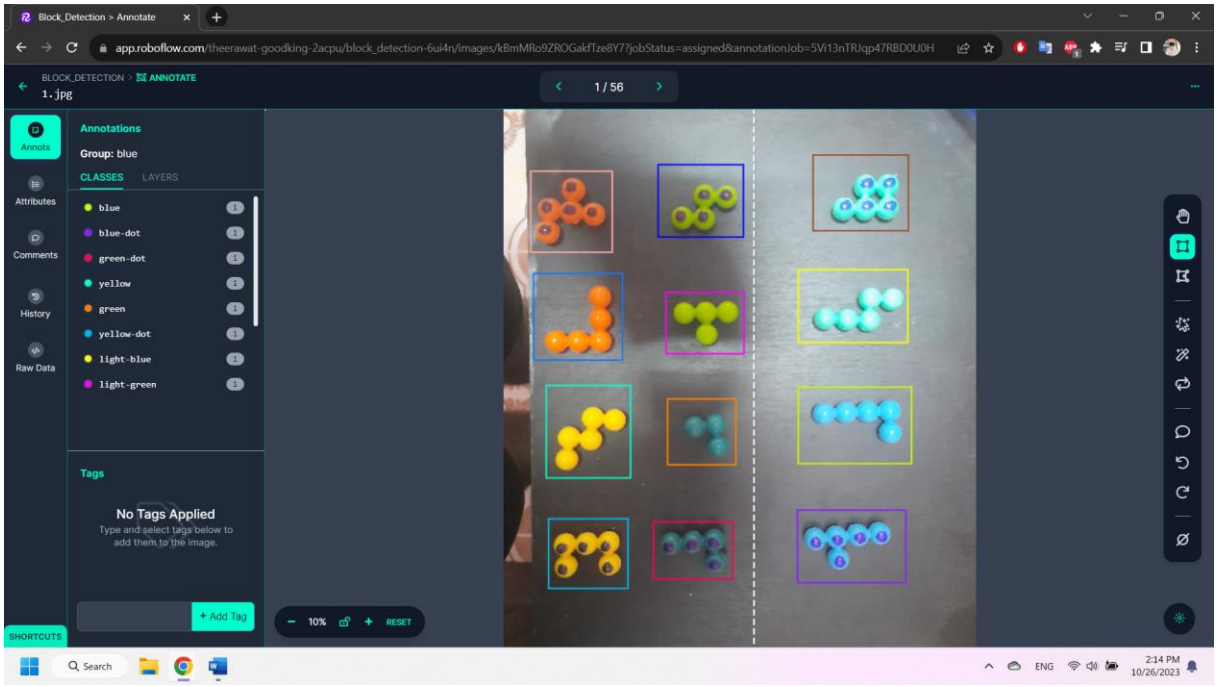
รูปที่ 3.3.14 หน้าหลักของการ Label รูปภาพ

- การ Label ภาพของวัตุ



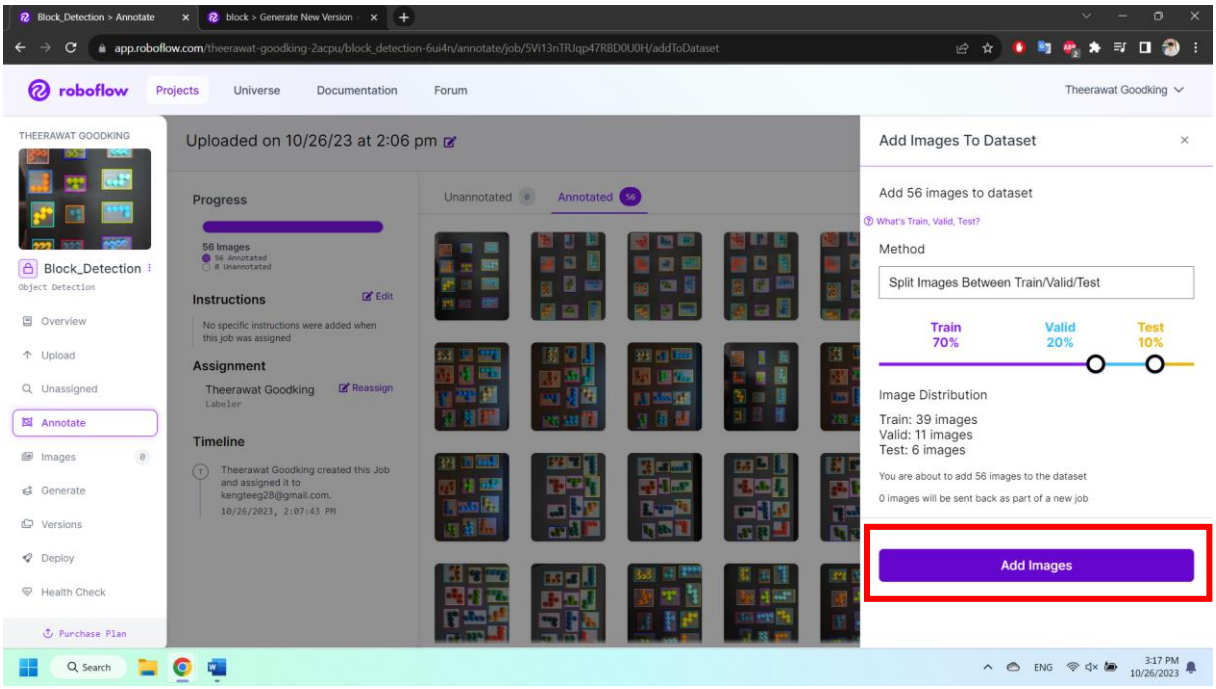
รูปที่ 3.3.15 หลังจากกด Start Annotating ให้เริ่มทำการ Label ภาพโดยการวาดกรอบขึ้นมาและใส่ Classes ที่ต้องการลงไป

- ทำการวาดกรอบของบล็อกทั้งหมดพร้อมใส่ Classes ให้กับบล็อกแต่ละชนิดจนครบทุกรูป



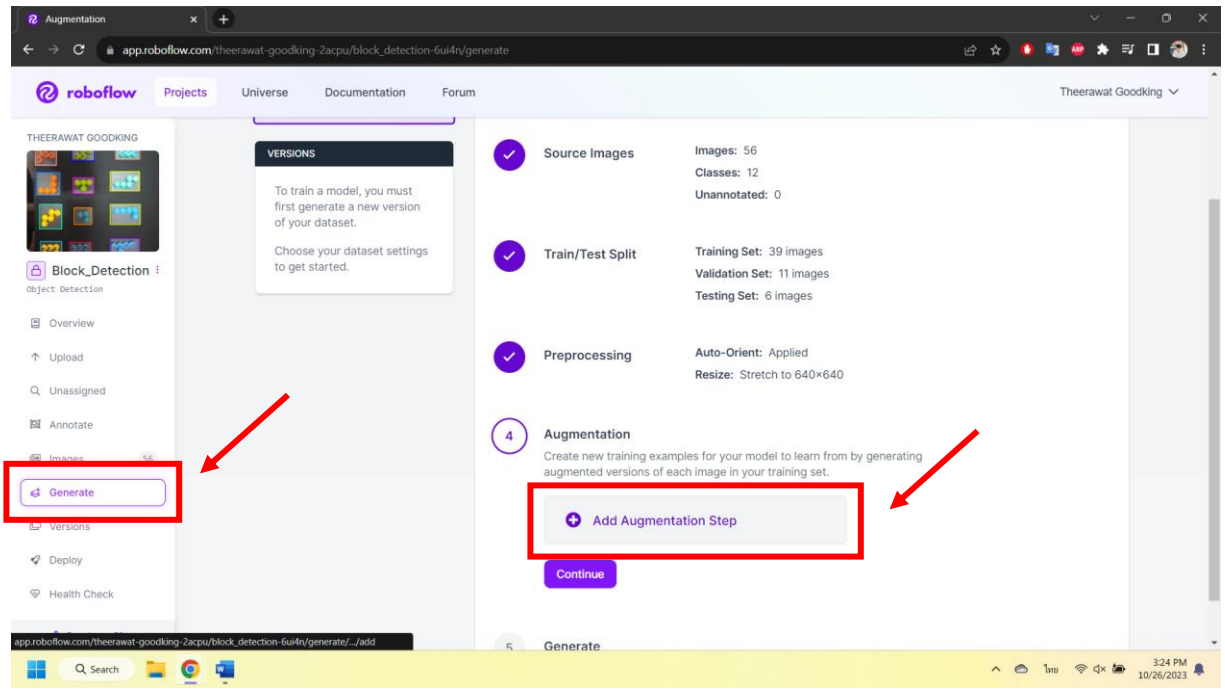
รูปที่ 3.3.16 หน้าจอของการ Label บล็อกทั้งหมด

- หลังจากทำการ Label ภาพทั้งหมดแล้วให้ทำกด Add Image เพื่อแบ่งสัดส่วนภาพ

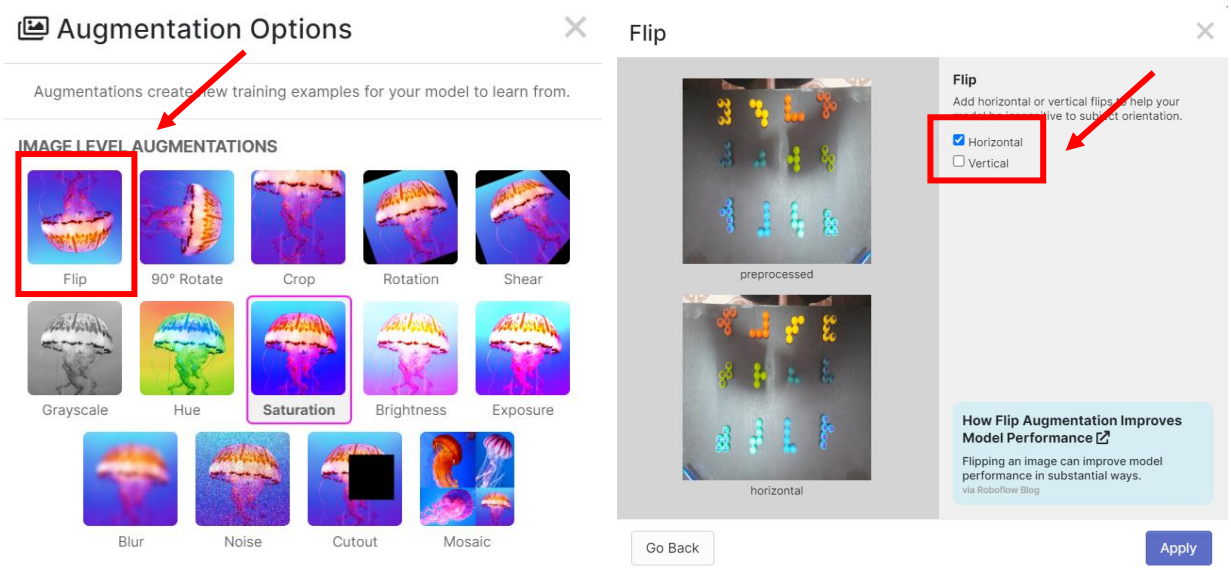


รูปที่ 3.3.17 หน้าจอหลักจากของการ Label ภาพทั้งหมด

- ทำการ Generate ชุดข้อมูล

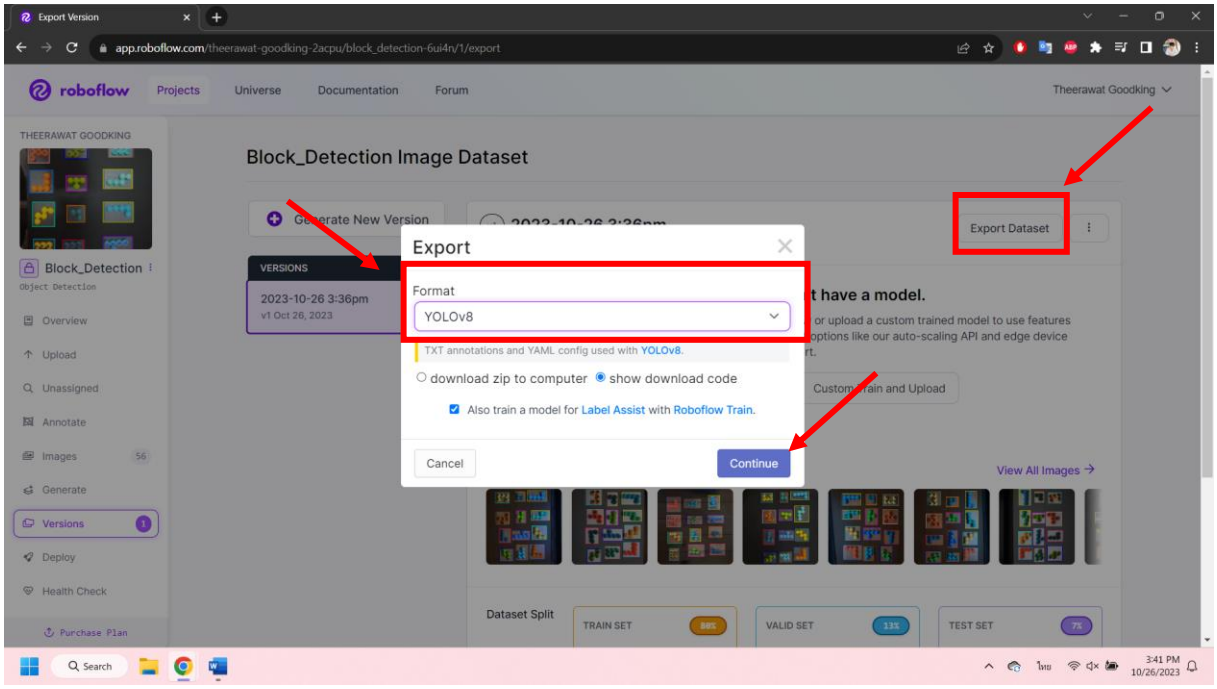


รูปที่ 3.3.18 มาที่หน้า Generate กด Continue ที่ขั้นตอน Preprocessing แล้วกด Add Augmentation Step



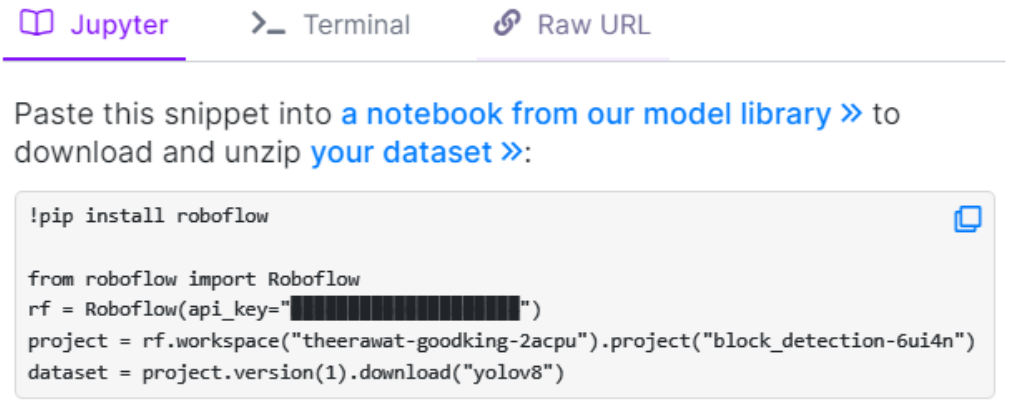
รูปที่ 3.3.19 ทำการเลือก Flip จากนั้นเลือก Horizontal แล้วกด Apply

- หลังจาก Generate ชุดข้อมูลแล้วให้ทำการเลือก version ของ dataset



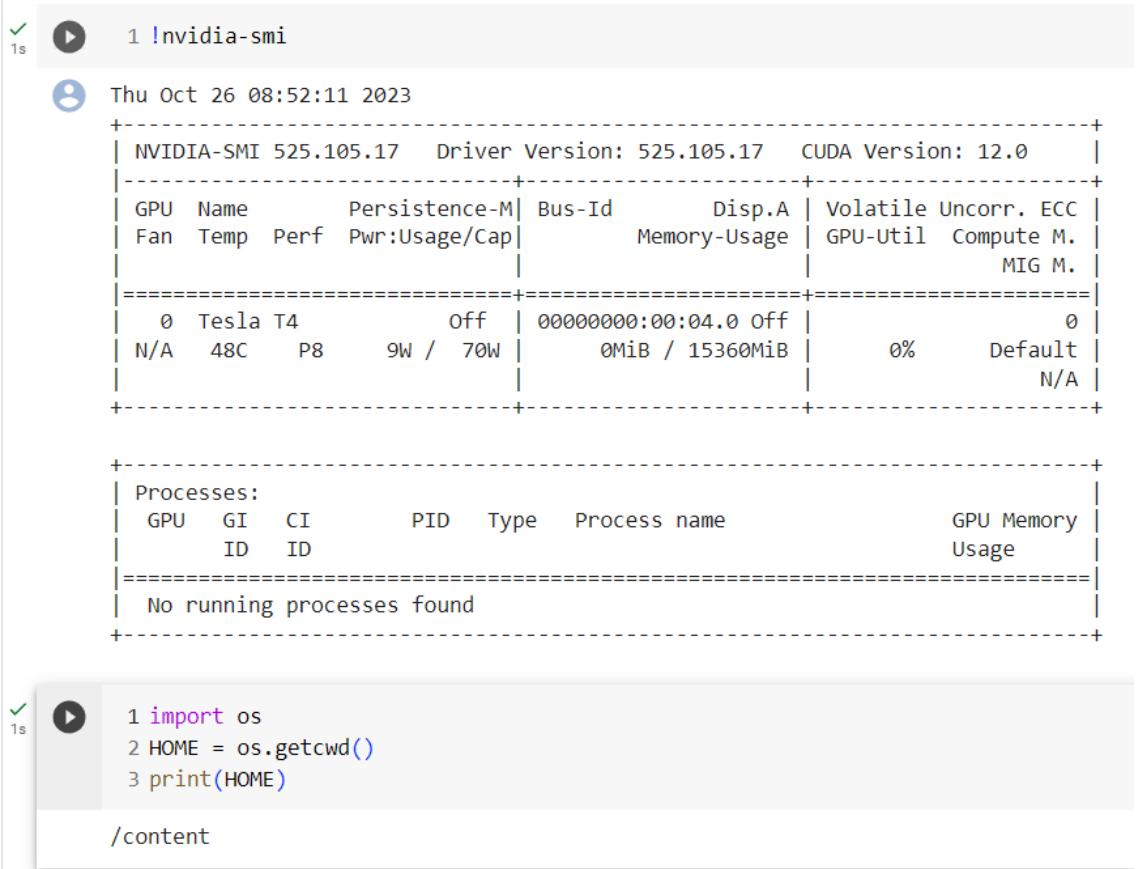
รูปที่ 3.3.20 กด Continue ที่ขั้นตอน Augmentation แล้วกด Generate จากนั้นกด Export Dataset แล้วเลือก Format เป็น YOLOv8 แล้วกด Continue

- เราจะได้ API KEY ของชุดข้อมูลที่เราสร้างมา



รูปที่ 3.3.21 ผลลัพธ์ API Key ของชุดข้อมูล

- มาที่ Google Colab แล้วตั้งค่า Runtime ให้เป็น GPU



รูปที่ 3.3.22 ตรวจสอบว่า GPU ในการทำงานหรือไม่ และกำหนดค่า HOME ให้เป็น part ของ Google colab

- ทำการติดตั้ง YOLOv8

```
!pip install ultralytics==8.0.20

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()
```

```
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 27.1/78.2 GB disk)
```

```
[4]
from ultralytics import YOLO

from IPython.display import display, Image
```

รูปที่ 3.3.23 ผลลัพธ์การติดตั้ง YOLOv8

- นำ roboflow API Key มาใช้งาน

```
!mkdir {HOME}/datasets
%cd {HOME}/datasets

!pip install roboflow --quiet

from roboflow import Roboflow
rf = Roboflow(api_key="mg8d8MEBGoGh7NAQG7j4")
project = rf.workspace("theerawat-goodking-2acpu").project("block_detection-6ui4n")
dataset = project.version(1).download("yolov8")
```

```
/content/datasets
──────────────────────────────────── 58.8/58.8 kB 2.0 MB/s eta 0:00:00
──────────────────────────────────── 155.3/155.3 kB 11.6 MB/s eta 0:00:00
──────────────────────────────────── 178.7/178.7 kB 25.8 MB/s eta 0:00:00
──────────────────────────────────── 58.8/58.8 kB 8.4 MB/s eta 0:00:00
──────────────────────────────────── 49.1/49.1 MB 18.7 MB/s eta 0:00:00
──────────────────────────────────── 67.8/67.8 kB 9.6 MB/s eta 0:00:00
──────────────────────────────────── 72.2/72.2 kB 11.3 MB/s eta 0:00:00
──────────────────────────────────── 54.5/54.5 kB 8.5 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
lida 0.0.10 requires fastapi, which is not installed.
lida 0.0.10 requires kaleido, which is not installed.
lida 0.0.10 requires python-multipart, which is not installed.
lida 0.0.10 requires uvicorn, which is not installed.
loading Roboflow workspace...
loading Roboflow project...
Dependency ultralytics==8.0.134 is required but found version=8.0.20, to fix: `pip install ultralytics==8.0.134`
Downloading Dataset Version Zip in Block_Detection-1 to yolov8:: 100%|██████████| 3576/3576 [00:01<00:00, 2924.15it/s]

Extracting Dataset Version Zip to Block_Detection-1 in yolov8:: 100%|██████████| 183/183 [00:00<00:00, 5564.43it/s]
```

รูปที่ 3.3.24 การใช้งาน roboflow API key ของผู้จัดทำโครงงาน

- ทำการเทรนโมเดล YOLOv8 โดยค่า epochs คือจำนวนรอบในการเทรน

```
%cd {HOME}
!yolo task=detect mode=train model=yolov8s.pt data={dataset.location}/data.yaml epochs=25 imgsz=800 plots=True
```

- ทำการประเมินโมเดลและนำไปใช้งาน

```
%cd {HOME}
!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt data={dataset.location}/data.yaml
```

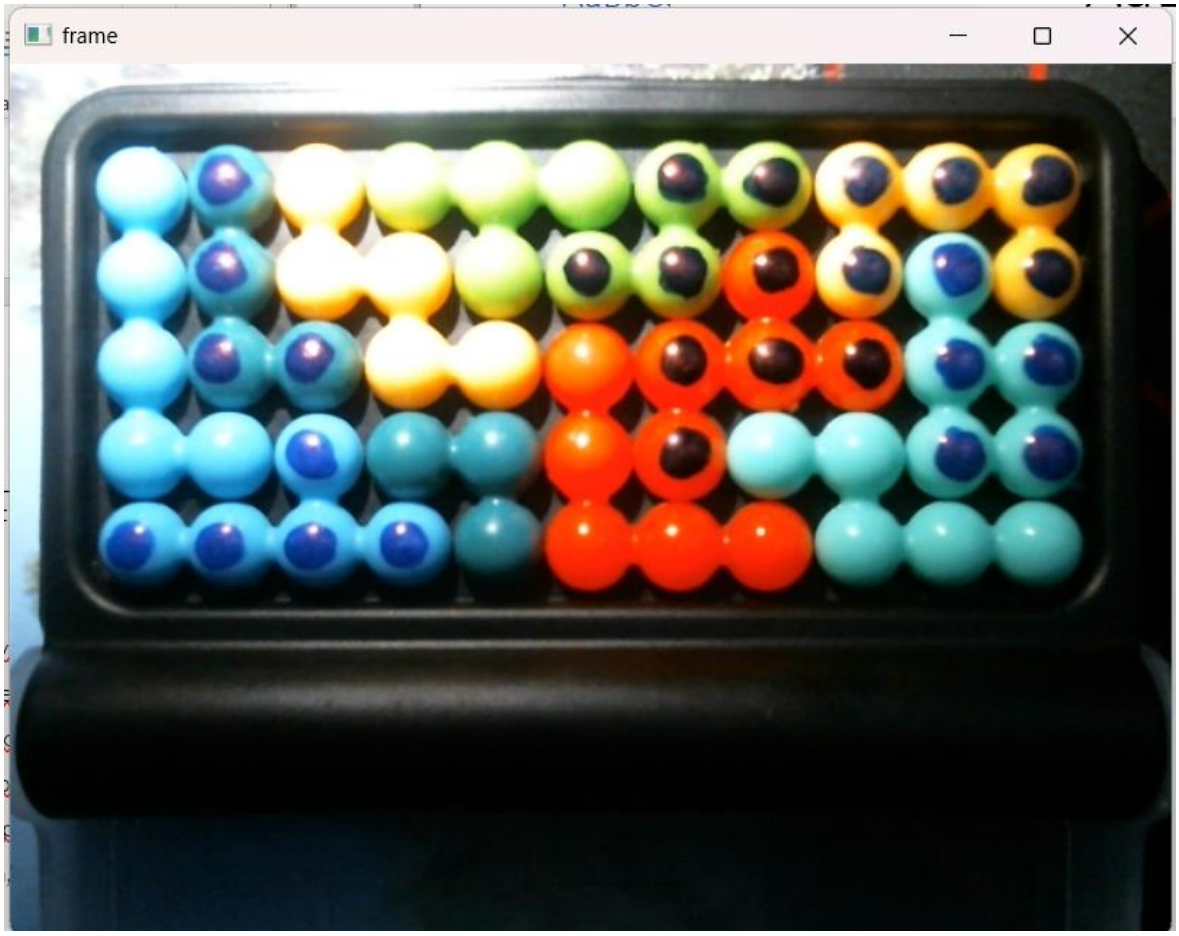
3.3.4 ปรับใช้งานกับกล้อง Web Camera

- ทำการติดตั้ง library OpenCV และ Jupyter notebook

pip install opencv-python	pip install notebook

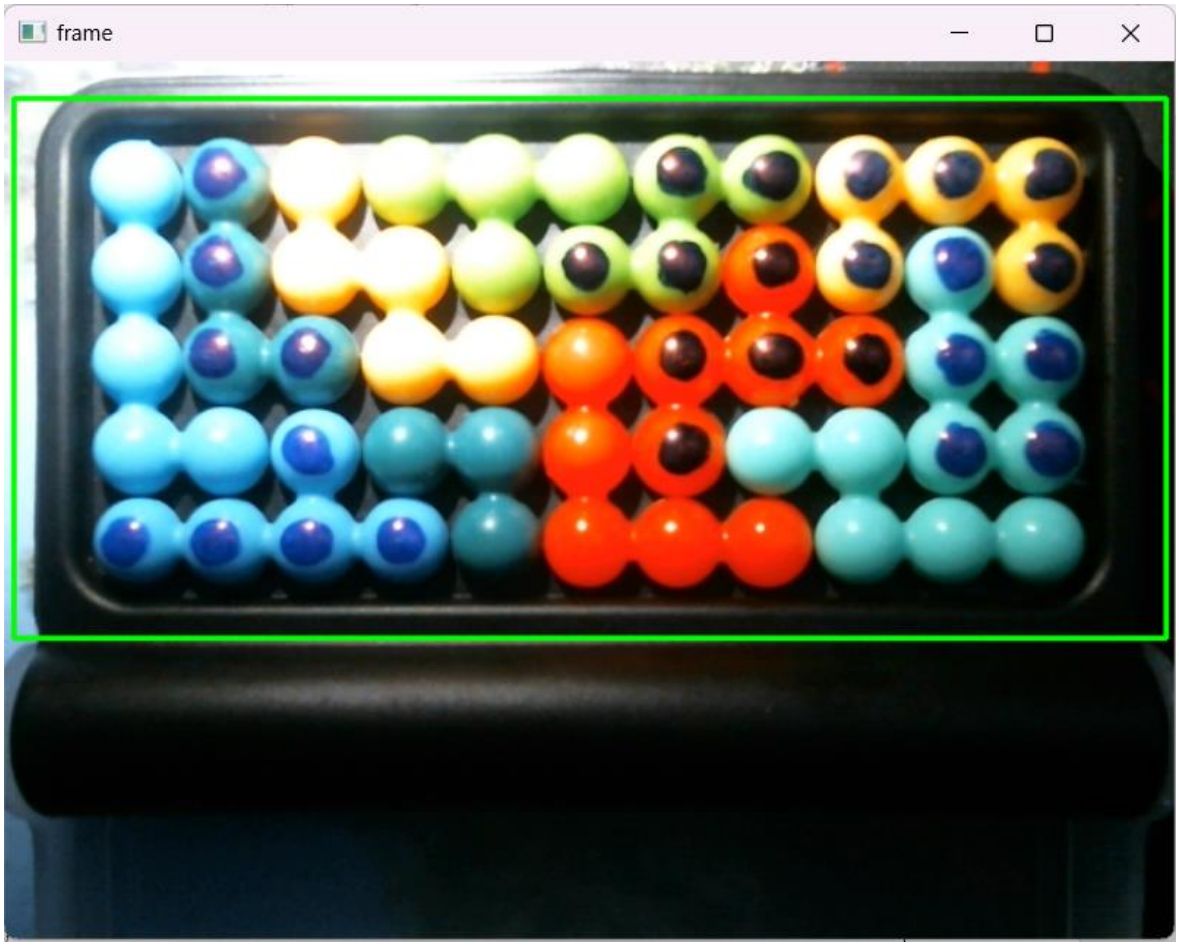
- ทดสอบการทำงานกับกล้องวีดีโอ

```
import cv2
cap = cv2.VideoCapture(0)
while cv2.waitKey(1) & 0xFF != ord('q'):
    ret, frame = cap.read()
    cv2.imshow('frame', frame)
cap.release()
cv2.destroyAllWindows()
```



รูปที่ 3.3.25 ผลการทำงานกับภาพกล้องวีดีโอ

- ใส่คำสั่ง cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2) เพื่อวาดกรอบสี่เหลี่ยมผืนผ้าลงบนภาพโดยค่า (x1, y1) คือจุดของมุมซ้ายบนของสีเหลี่ยม และค่า (x2, y2) คือจุดของมุมขวาล่า0E07

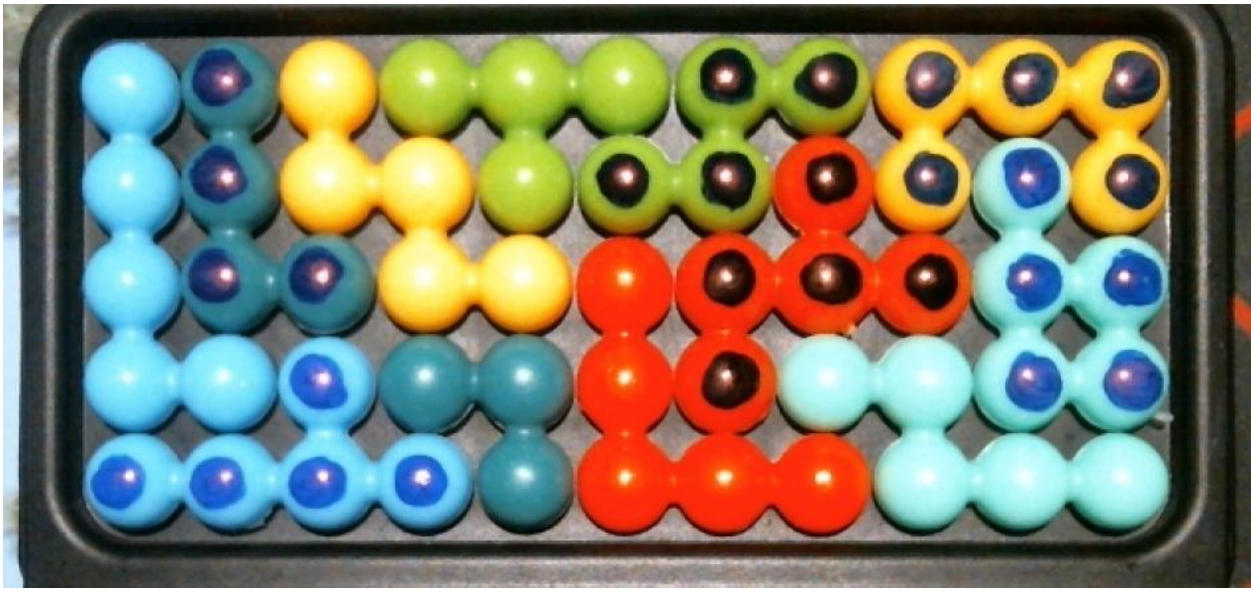


รูปที่ 3.3.26 ผลการวาดกรอบสี่เหลี่ยมลงบนภาพ

- ใส่คำสั่ง crop ภาพในพิกัดที่ต้องการและทำการขยายภาพเป็น 1.5 เท่าแล้วบันทึกภาพ

```
import cv2
cap = cv2.VideoCapture(1)
while cv2.waitKey(1) & 0xFF != ord('q'):
    ret, frame = cap.read()
    image = frame.copy()
    cv2.rectangle(image, (5, 20), (635, 315), (0, 255, 0), 2)
    cv2.imshow('frame', image)
```

```
roi_cropped = frame[20:315, 5:635]
                roi_width = int(roi_cropped.shape[1] * (150 / 100))
roi_height = int(roi_cropped.shape[0] * (150 / 100))
cropped = cv2.resize(roi_cropped, (roi_width,roi_height), interpolation = cv.INTER_AREA)
cv2.imwrite('./pic/test/save_result.jpg', cropped)
cap.release()
cv2.destroyAllWindows()
```



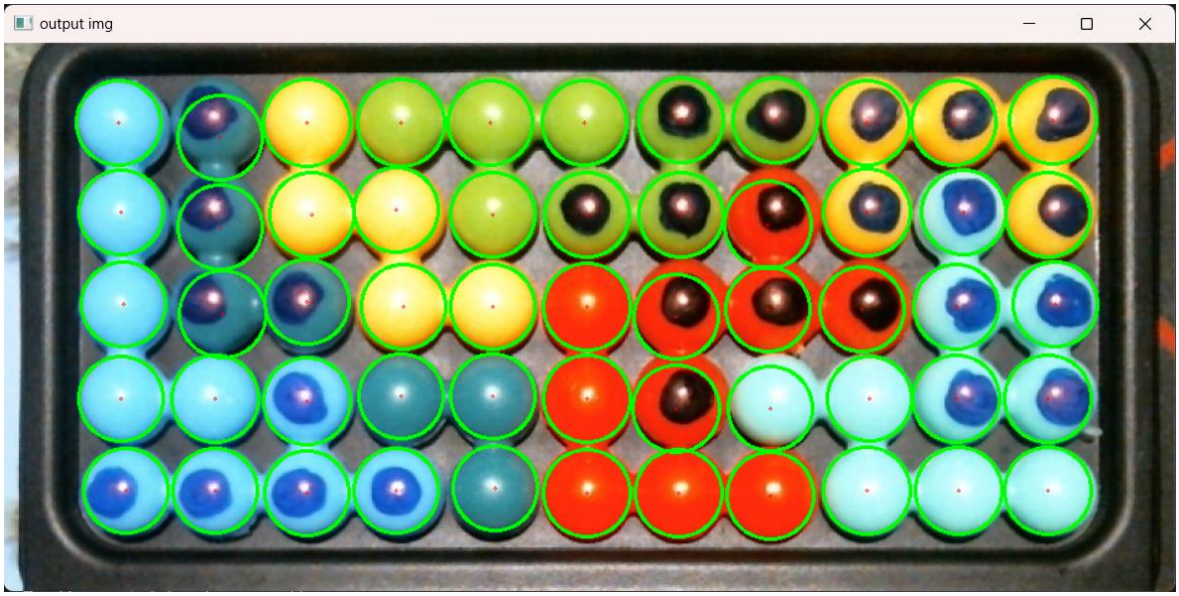
รูปที่ 3.3.27 ผลการ crop ภาพ

- เขียนฟังก์ชันเพื่อตรวจจับขอบของรูปภาพเพื่อวาดวงกลมลงไป

```
import cv2 as cv
import numpy as np
img = cv.imread('./pic/test/save_result.jpg')
img2 = img.copy()
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
gray = cv.GaussianBlur(gray, (7, 7), 1.5)
circles = cv.HoughCircles(gray, cv.HOUGH_GRADIENT, 1.5, 72, param1=50, param2=25, minRadius=25,
maxRadius=45)
circles = np.uint16(np.around(circles))
for i in circles[0, :]:
```

```
    center, radius = (i[0], i[1]), i[2]
    cv.circle(img2, center, radius, (0, 255, 0), 2)
    cv.circle(img2, center, 1, (0, 0, 255), 1)
cv.imshow("output img", img2)
cv.waitKey(0)
cv.destroyAllWindows()
```

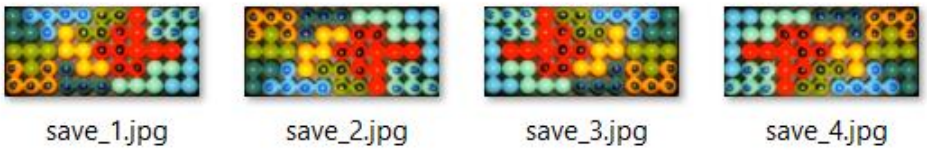
หมายเหตุ *: การทำงานของโค้ดนี้คือการอ่านภาพต้นแบบแล้วทำการเปลี่ยนเป็นภาพขาวดำ เพื่อทำการเบลอภาพด้วยเทคนิค Gaussian Filter เพื่อทำให้ตรวจจับขอบของวงกลมได้ง่ายขึ้น*



รูปที่ 3.3.28 ภาพผลการตรวจจับวงกลมภาพถ่าย

3.3.5 เขียนโปรแกรมเปรียบเทียบผลลัพธ์

- ทำการเก็บข้อมูลภาพผลลัพธ์ที่สมบูรณ์ของปริศนาเกม IQ Puzzler PRO เพื่อใช้ในการเปรียบเทียบกับภาพต้นแบบ โดยภาพผลลัพธ์ที่สมบูรณ์ 1 ภาพ สามารถบันทึกแยกได้เป็นรูปแบบภาพ 4 รูปแบบ ได้แก่ (ภาพต้นฉบับ ภาพพลิกแนวตั้ง ภาพพลิกแนวนอน ภาพพลิกทั้งแนวตั้งแนวนอน) เพราะว่าพื้นที่การวางบล็อกที่ทั้งหมดเป็นรูปสี่เหลี่ยมผืนผ้า ความเป็นไปได้ในการวางบล็อกสลับแนวตั้งแนวนอนจึงมีสูง



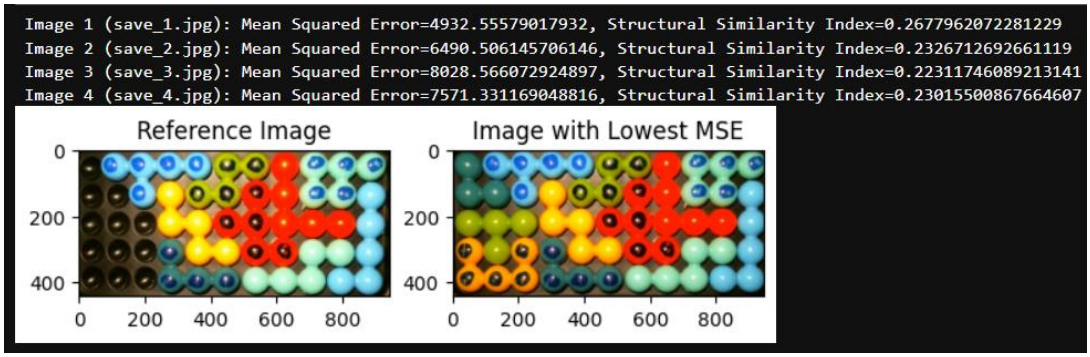
รูปที่ 3.3.29 ตัวอย่าง ภาพข้อมูลผลลัพธ์ที่เคยทำไว้

- เขียนโปรแกรมเพื่อเปรียบเทียบภาพต้นแบบกับภาพผลลัพธ์ที่เคยทำไว้

```
import os
import cv2
from skimage.metrics import structural_similarity as ssim
import matplotlib.pyplot as plt
import numpy as np

def mse(image1, image2):
    return np.sum((image1.astype("float") - image2.astype("float")) ** 2) / float(image1.size)

def compare_images(reference_image, image_folder):
    gray_reference_image = cv2.cvtColor(reference_image, cv2.COLOR_BGR2GRAY)
    results = []
    min_mse_image, min_mse_value = None, float('inf')
    for index, jpeg_file in enumerate(os.listdir(image_folder), start=1):
        image_path = os.path.join(image_folder, jpeg_file)
        current_image = cv2.imread(image_path)
        current_image = cv2.resize(current_image, (reference_image.shape[1], reference_image.shape[0]))
        gray_image = cv2.cvtColor(current_image, cv2.COLOR_BGR2GRAY)
        mse_value = mse(gray_reference_image, gray_image)
        ssim_value = ssim(gray_reference_image, gray_image)
        results.append((f"Image {index} ({jpeg_file})", mse_value, ssim_value))
        if mse_value < min_mse_value:
            min_mse_value, min_mse_image = mse_value, current_image
    return results, min_mse_image

reference_image_path = "./test/save_9.jpg"
```

```
reference_image = cv2.imread(reference_image_path)
image_folder = "./compare/"
results, min_mse_image = compare_images(reference_image, image_folder)
for name, mse_value, ssim_value in results:
    print(f"{name}: Mean Squared Error={mse_value}, Structural Similarity Index={ssim_value}")


plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(reference_image, cv2.COLOR_BGR2RGB))
plt.title("Reference Image")
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(min_mse_image, cv2.COLOR_BGR2RGB))
plt.title("Image with Lowest MSE")
plt.show()
```

หมายเหตุ *: การทำงานของโค้ดนี้เป็นการนำภาพสองภาพมาเปรียบเทียบโดยการหาค่า Mean Squared Error (MSE) ของทั้งสองภาพ โดยจะยึดภาพต้นแบบไว้แล้วทำการเปรียบเทียบกับทุกภาพผลลัพธ์ที่เคยบันทึกไว้ จากนั้นเหลือภาพที่มีค่า MSE มาแสดงผล*



รูปที่ 3.3.30 ภาพเปรียบเทียบของภาพต้นแบบกับภาพผลลัพธ์ที่เคยทำไว้ที่มีค่า Mean Squared Error น้อยที่สุด

3.3.6 การปรับพารามิเตอร์กับเขียนโปรแกรมเพิ่มเติม

เพื่อให้ส่วนประกอบทั้งหมดทำงานร่วมกันได้อย่างสมบูรณ์จึงต้องมีการปรับค่าพารามิเตอร์บางอย่างหรือเขียนโปรแกรมเพิ่มเติมเข้าไป ได้แก่

- การเพิ่มคำสั่งค้นหาคำในไฟล์ json เนื่องจากการเรียกใช้การตรวจจับวัตถุจาก API ของ roboflow การที่ตรวจจับวัตถุแล้วส่งผลลัพธ์เป็นข้อความเป็นไฟล์ json จึงทำให้ต้องเขียนคำสั่งเพื่อตรวจจับคำในไฟล์แล้วเพิ่มค่า array ของบล็อกที่ถูกตรวจพบลงไปในตัวแปร result_dict เพื่อใช้งานในการแก้ปัญหาปริศนา

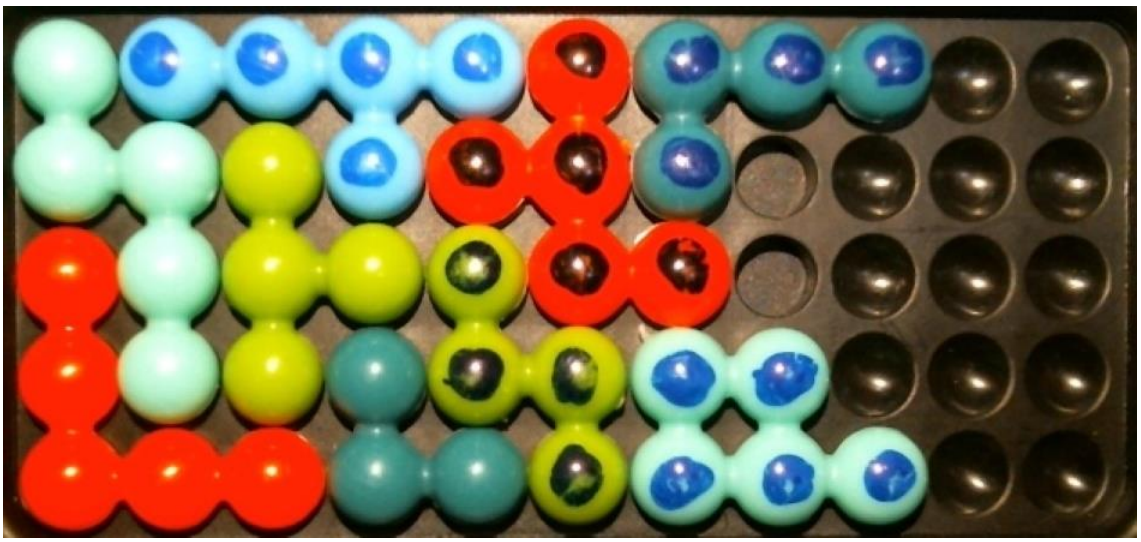
```
from roboflow import Roboflow
import numpy as np
import json
rf = Roboflow(api_key="mg8d8MEBGoGh7NAQG7j4")
project = rf.workspace().project("block-6bulo")
model = project.version(1).model
json_result = model.predict("./pic/test/test02.jpg", confidence=30, overlap=30).json()
str_result = json.dumps(json_result)
blue, blue_dot, green, green_dot, lb_dot, lg_dot, light_blue, light_green, orange, orange_dot, yellow_dot =
[
    np.array(matrix) for matrix in [
        [[1, 1, 1, 1], [1, 0, 0, 0]],
        [[1, 1, 1, 1], [0, 1, 0, 0]],[[1, 1], [1, 0]],
        [[0, 0, 1], [1, 1, 1]],
        [[1, 1, 0], [1, 1, 1]],
        [[1, 1, 0], [0, 1, 1]],
        [[0, 1, 1, 1], [1, 1, 0, 0]],
        [[0, 1, 0], [1, 1, 1]],
        [[0, 0, 1], [0, 0, 1], [1, 1, 1]],
        [[0, 1, 1], [1, 1, 0], [0, 1, 0]],
        [[1, 0, 1], [1, 1, 1]],]]
word_matrices = {
    "blue": blue, "blue-dot": blue_dot,
    "green": green, "green-dot": green_dot,
    "lb-dot": lb_dot, "lg-dot": lg_dot,
    "light-blue": light_blue, "light-green": light_green,
    "orange": orange, "orange-dot": orange_dot, "yellow-dot": yellow_dot}
result_word = [word for word in word_matrices if word in str_result]
result_dict = {word: word_matrices.get(word) for word in result_word}
print(result_dict)
```

- เพิ่มการตรวจจำนวนภาพในไฟล์ที่ต้องการบันทึกเพื่อที่จะทำการตั้งชื่อไฟล์อัตโรมัติ

```
import os
```

```
file_list = os.listdir("./pic/test")
num_images = sum(file.lower().endswith('.jpg') for file in file_list)
print(num_images)
```

- การแก้ไขพารามิเตอร์ของตรวจหาขอบวงกลม โดยแก้ไขและเพิ่มคำสั่งเพื่อตรวจจับวงกลมเพื่อให้ผลลัพธ์ที่ได้เป็น array ขนาดแถวเป็น 11 และคอลัมนเป็น 5 จากนั้นให้ทำการอ่านค่าและตรวจจับขนาดของพื้นที่ที่มีขนาดมากกว่า 4700 เพราะวงกลมของชิ้นบล็อกมีขนาดโดยเฉลี่ยอยู่ที่ 4700 – 4800 โดยหากวงกลมที่ตรวจพบมีพื้นที่มากกว่าที่ตั้งเงื่อนไขไวโปรแกรมจะทำการใส่ค่า ‘X’ ลงไปใน array และหากไม่ตรวจพบอะไรก็ให้ใส่ค่า ‘0’ ลงไปแทน



รูปที่ 3.3.31 ภาพพื้นที่ของเกม IQ Puzzler PRO จะเห็นได้ว่ามีขนาดวงกลมอยู่ 2 แบบคือ วงกลมของชิ้นบล็อกกับวงกลมของพื้นที่ โดยถ้าเราบล็อกเต็มพื้นจะมีวงกลมขนาดใหญ่ของชิ้นบล็อก และวงกลมขนาดเล็กของพื้นที่

```
import cv2 as cv
import numpy as np
img = cv.imread('./pic/test/save_8.jpg')
img2 = img.copy()
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
gray = cv.GaussianBlur(gray, (7, 7), 1.5)
circles=cv.HoughCircles(gray,cv.HOUGH_GRADIENT,1.5,72,param1=50,param2=25,minRadius=25,maxRadius=45)
circles = np.uint16(np.around(circles))
name, none = 'X', '0'
rows, cols = 5, 11
height, width = img.shape[:2]
cell_height, cell_width = height // rows + 1, width // cols + 1
grid_matrix = np.full((rows, cols), none, dtype=str)
for i in circles[0, :]:
    center, radius = (i[0], i[1]), i[2]
    area = np.pi * radius ** 2
    row_index, col_index = center[1]/ cell_height, center[0]//cell_width
    if area > 4700:
        cv.circle(img2, center, radius, (0, 255, 0), 2)
        cv.circle(img2, center, 1, (0, 0, 255), 1)
        grid_matrix[row_index, col_index] = name
result_list = grid_matrix.tolist()
print(result_list)
cv.imshow("output img", img2)
cv.waitKey(0)
cv.destroyAllWindows()
```

หมายเหตุ *: การทำงานของโค้ดนี้เป็นจะแสดงผลลัพธ์รูปที่ 3.3.13 และให้ผลลัพธ์ที่เป็น array ดังนี้*

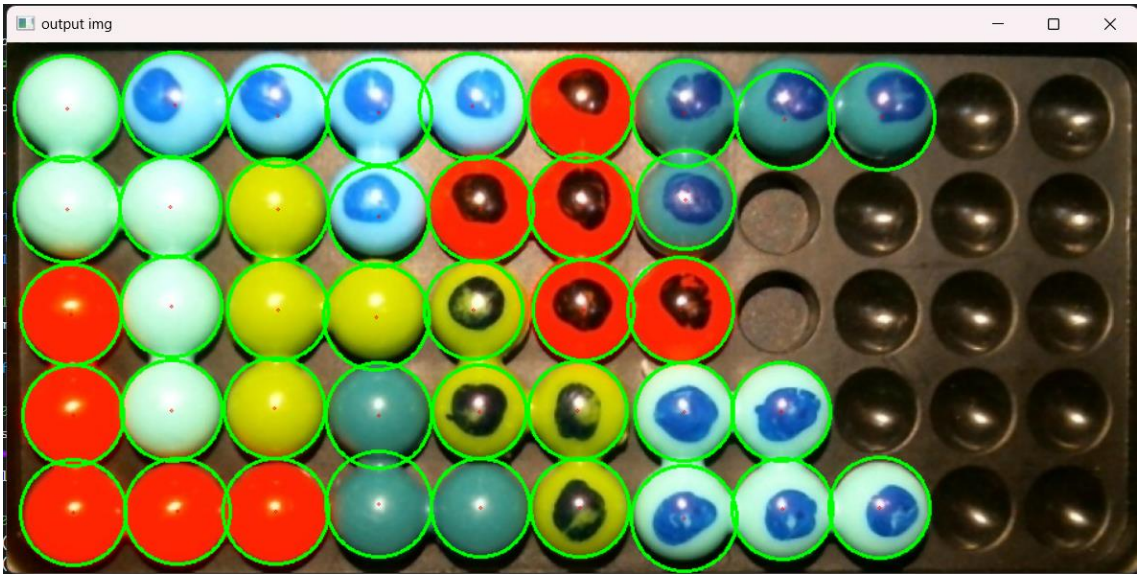
[['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0'],

['X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0', '0'],

['X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0', '0'],

['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0'],

['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', '0', '0']]



รูปที่ 3.3.32 ผลลัพธ์ของการแก้ไขพารามิเตอร์ของตรวจหาขอบวงกลมการ

3.3.7 โปรแกรมแบบสมบูรณ์

```
import os
import json
import cv2 as cv
import numpy as np
from roboflow import Roboflow
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

cap = cv.VideoCapture(0)
file_list = os.listdir("./pic/test")
num_images = sum(file.lower().endswith('.jpg') for file in file_list)

while cv.waitKey(1) & 0xFF != ord('q'):
    ret, frame = cap.read()
    image = frame.copy()
    cv.rectangle(image, (5, 20), (635, 315), (0, 255, 0), 2)
    cv.imshow('frame', image)

roi_cropped = frame[20:315, 5:635]
roi_width = int(roi_cropped.shape[1] * (150 / 100))
```

```
roi_height = int(roi_cropped.shape[0] * (150 / 100))
cropped=cv.resize(roi_cropped,(roi_width,roi_height),interpolation=cv.INTER_AREA)
cv.imwrite(f'./pic/test/save_{num_images}.jpg', cropped)
cap.release()
cv.destroyAllWindows()

img2 = cropped.copy()
gray = cv.cvtColor(cropped, cv.COLOR_BGR2GRAY)
gray = cv.GaussianBlur(gray, (7, 7), 1.5)
circles = cv.HoughCircles(gray, cv.HOUGH_GRADIENT, 1.5, 72, param1=50, param2=25, minRadius=25,
maxRadius=45)
circles = np.uint16(np.around(circles))
name, none = 'X', '0'
rows, cols = 5, 11
height, width = cropped.shape[:2]
cell_height, cell_width = height // rows + 1, width // cols + 1
grid_matrix = np.full((rows, cols), none, dtype=str)

for i in circles[0, :]:
    center, radius = (i[0], i[1]), i[2]
    area = np.pi * radius ** 2
    row_index, col_index = center[1]//cell_height, center[0]//cell_width
    if area > 4700:
        cv.circle(img2, center, radius, (0, 255, 0), 2)
        cv.circle(img2, center, 1, (0, 0, 255), 1)
        grid_matrix[row_index, col_index] = name

result_list = grid_matrix.tolist()
c.imshow("output img", img2)
cv.waitKey(0)
cv.destroyAllWindows()

rf = Roboflow(api_key="mg8d8MEBGoGh7NAQG7j4")
project = rf.workspace().project("block-6bulo")
```

```
model = project.version(1).model
json_result=model.predict("./pic/test/test.jpg",confidence=40,overlap=30).json()
str_result = json.dumps(json_result)
blue, blue_dot, green, green_dot, lb_dot, lg_dot, light_blue, light_green, orange, orange_dot, yellow,
yellow_dot = [
    np.array(matrix) for matrix in [
        [[1, 1, 1, 1], [1, 0, 0, 0]],
        [[1, 1, 1, 1], [0, 1, 0, 0]],
        [[1, 1], [1, 0]],
        [[0, 0, 1], [1, 1, 1]],
        [[1, 1, 0], [1, 1, 1]],
        [[1, 1, 0], [0, 1, 1]],
        [[0, 1, 1, 1], [1, 1, 0, 0]],
        [[0, 1, 0], [1, 1, 1]],
        [[0, 0, 1], [0, 0, 1], [1, 1, 1]],
        [[0, 1, 1], [1, 1, 0], [0, 1, 0]],
        [[0, 0, 1], [0, 1, 1], [1, 1, 0]],
        [[1, 0, 1], [1, 1, 1]],]]

word_matrices = {
    "blue": blue, "blue-dot": blue_dot,
    "green": green, "green-dot": green_dot,
    "lb-dot": lb_dot, "lg-dot": lg_dot,
    "light-blue": light_blue, "light-green": light_green,
    "orange": orange, "orange-dot": orange_dot,
    "yellow": yellow, "yellow-dot": yellow_dot}

result_word = []
blocks = {}

for word in word_list:
    if word in str_result:
        blocks [word] = word_matrices.get(word, None)
        result_word.append(word)
```

```
area = np.array((result_list),dtype='str')

def can_place_block(block, row, col):
    block_rows, block_cols = block.shape
    return (
        row + block_rows <= area.shape[0] and
        col + block_cols <= area.shape[1] and
        not np.any(block * (area[row:row + block_rows, col:col + block_cols] != '0')))

def place_block(block, row, col, block_name):
    area[row:row + block.shape[0], col:col + block.shape[1]][block == 1] = block_name

def remove_block(block, row, col):
    area[row:row + block.shape[0],col:col+block.shape[1]][block == 1] = '0'

def solve(block_names, remaining_blocks):
    if len(block_names) == 3:
        return True
    for row in range(area.shape[0]):
        for col in range(area.shape[1]):
            for block_name in remaining_blocks:
                block_matrix = blocks[block_name]
                for _ in range(8):
                    if can_place_block(block_matrix, row, col):
                        place_block(block_matrix, row, col, block_name)
                        block_names.append(block_name)
                        remaining_blocks.remove(block_name)
                        if solve(block_names, remaining_blocks):
                            return True
                        remove_block(block_matrix, row, col)
                        block_names.pop()
                        remaining_blocks.append(block_name)
                    block_matrix = np.rot90(block_matrix) if _ < 4 else np.flipud(block_matrix)
```

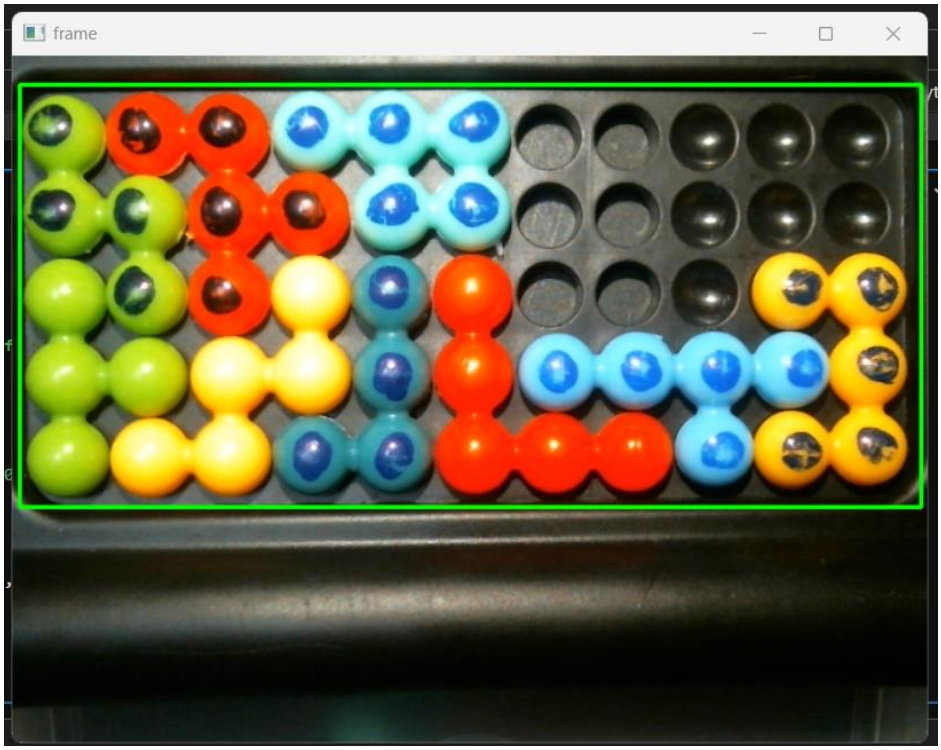
```
    return not remaining_blocks

block_names = []
remaining_blocks = list(blocks.keys())

if solve(block_names, remaining_blocks):
   if solve(block_names, remaining_blocks):
        block_colors = {block_name: plt.cm.tab20(i) for i, block_name in enumerate(block_names)}
    cmap = ListedColormap([block_colors[block_name] for block_name in block_names])
    for i in range(area.shape[0]):
        for j in range(area.shape[1]):
            plt.text(j+0.5,i+0.5,area[i,j],ha='center',va='center',fontsize=10)

    plt.imshow(np.zeros_like(area, dtype=int), cmap=cmap, interpolation='none', extent=[0, area.shape[1],
area.shape[0], 0])
    plt.xticks(range(area.shape[1]+1))
    plt.yticks(range(area.shape[0]+1))
    plt.grid(True, color='black', linewidth=1)
    plt.show()
else:
    print("No solution found")
print("Done")
```

บทที่ 4

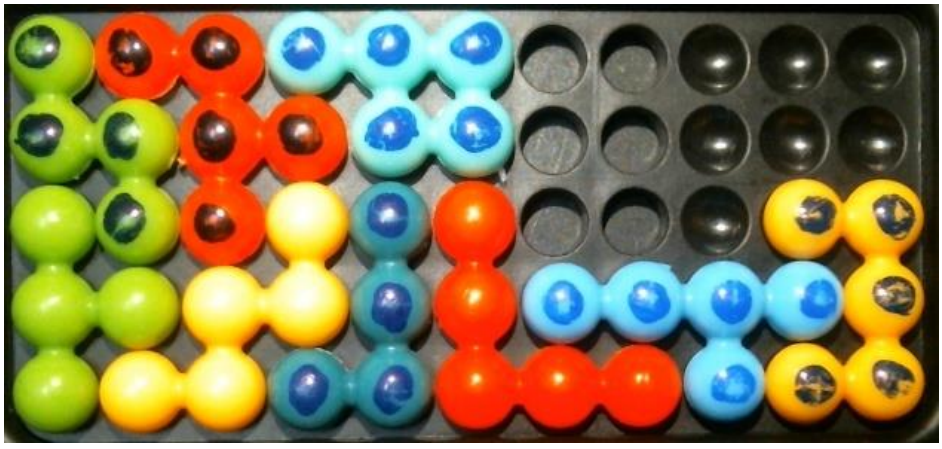
ผลการดำเนินงาน

4.1 ผลลัพธ์ของโครงงาน

- ทำการเปิดกล้องวีดีโอเผื่อบันทึกผลภาพพื้นที่ในการวางบล็อกเพื่อนำไปใช้งานกับการเปรียบเทียบภาพและการตรวจจับขอบวงกลม

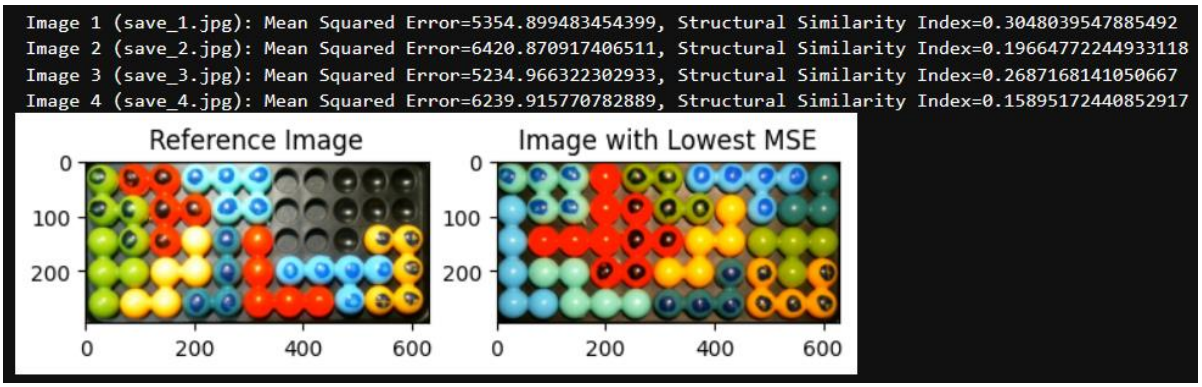


รูปที่ 4.1.1 ภาพขณะรันโปรแกรมการเปิดกล้องวีดีโอ



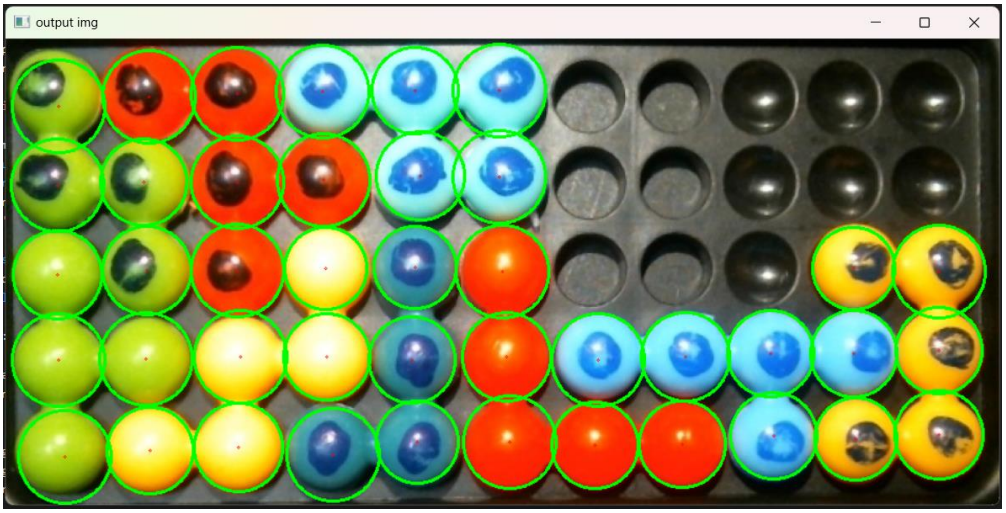
รูปที่ 4.1.2 ภาพผลการทดลองหลังการรันโปรแกรมการเปิดกล้องวีดีโอ

- ทำการเปรียบภาพที่ได้มากับภาพผลลัพธ์ที่มี



รูปที่ 4.1.3 ภาพผลการทดลองหลังการรันโปรแกรมการเปรียบเทียบภาพ จะเห็นได้ว่าภาพที่มีค่า MSE น้อยที่สุดไม่ตรงกับผลลัพธ์ที่ต้องการ แสดงว่าปริศนาปัญหาชุดนี้ยังไม่เคยได้รับการแก้ไขจากโปรแกรม

- ทำการตรวจจับขอบของวงกลมจากภาพผลลัพธ์ รูปที่ 4.1.2



รูปที่ 4.1.3 ภาพผลการทดลองหลังการรันโปรแกรมการตรวจจับขอบของวงกลม ผลลัพธ์ที่ได้ในรูปแบบของ array คือ

[['X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0', '0', '0'],

['X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0', '0', '0'],

['X', 'X', 'X', 'X', 'X', 'X', '0', '0', '0', 'X', 'X'],

['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X'],

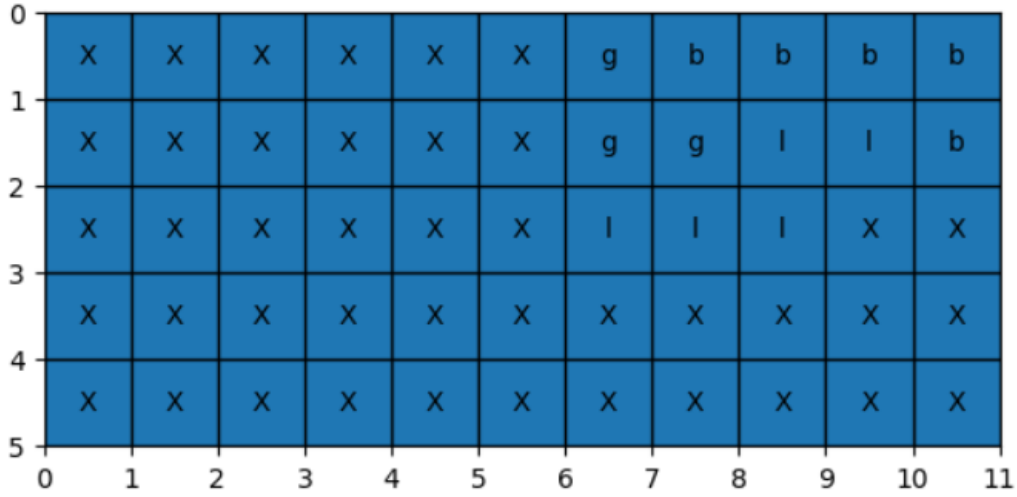
['X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X']]

- ทำการตรวจจับบล็อกที่เหลืออยู่ โดยใช้ API Key จาก roboflow โดยผลลัพธ์ที่ได้ดปีนดังนี้

```
{'green': array([[0, 1],[1, 1]]), 'light-blue': array([[1, 1, 1, 0],[0, 0, 1, 1]]), 'blue': array([[1, 0, 0, 0],[1, 1, 1, 1]])}
```

- ทำการใช้งานโปรแกรมแก้ปัญหาปริศนา

```
block found is {'green': array([[0, 1],
       [1, 1]]), 'light-blue': array([[1, 1, 1, 0],
       [0, 0, 1, 1]]), 'blue': array([[1, 0, 0, 0],
       [1, 1, 1, 1]])}
area is [['X' 'X' 'X' 'X' 'X' 'X' '0' '0' '0' '0' '0']
 ['X' 'X' 'X' 'X' 'X' 'X' '0' '0' '0' '0' '0']
 ['X' 'X' 'X' 'X' 'X' 'X' '0' '0' '0' 'X' 'X']
 ['X' 'X' 'X' 'X' 'X' 'X' 'X' 'X' 'X' 'X' 'X']
 ['X' 'X' 'X' 'X' 'X' 'X' 'X' 'X' 'X' 'X' 'X']]
```




```
Done
```

รูปที่ 4.1.4 ผลลัพธ์การแก้ปัญหาปริศนา โดยค่า 'X' คือพื้นที่ที่มีบล็อกอื่นวางไว้แล้ว , ค่า 'g' คือตำแหน่งที่บล็อก green ต้องวาง , ค่า 'l' คือตำแหน่งที่บล็อก light_blue ต้องวาง , ค่า 'b' คือตำแหน่งที่บล็อก blue ต้องวาง

- หลักจากที่ทดลองแก้ไขปัญหาปริศนาไปหลาย ๆ รูปแบบบน Google Colab ผลที่ได้พบว่า

ตารางที่ 1.2 ผลการทดสอบ

จำนวนบล็อกที่ยังไม่ได้วาง	เวลาที่โปรแกรมใช้แก้ปัญหา
1-3 ชิ้น	น้อยกว่า 3 วินาที
4 ชิ้น	ประมาณ 4 ถึง 10 วินาที
5 ชิ้น	ประมาณ 7 ถึง 10 นาที
6 ชิ้น	ประมาณ 57 นาที ถึง 72 นาที
7 ชิ้น	มากกว่า 7 ชั่วโมง

- วีดีโอผลลัพธ์ (https://www.youtube.com/playlist?list=PL8-PHAVyXiDoh7Tq6DP4Z5xRWlU7htIOa)

4.2 ผลที่คาดว่าจะได้รับ

- สามารถแสดงผลลัพธ์ในรูปแบบของสีแต่ละบล็อก
- สามารถตรวจจับชิ้นบล็อกบน Window แบบเรียลไทม์
- ใช้เวลาในการแก้ปัญหาเร็วกว่านี้ เช่น จำนวนบล็อกที่ยังไม่ได้วาง 5 ชิ้น จะใช้เวลาในการแก้ปัญหาต่ำกว่า 5 นาที เป็นต้น

4.3 การพัฒนาต่อยอด

- สามารถนำไปพัฒนาเป็น Web Application หรือ Mobile Application ได้
- สามารถนำอัลกอริทึมหรือโค้ดต้นแบบไปพัฒนาเพื่อแก้ปัญหาปริศนากับเกมอื่น ๆ ได้
- นำความรู้ในการตรวจจับวัตุไปใช้ในงานอุตสาหกรรมที่ต้องใช้การตรวจจับและแยกวัตุได้
- นำความรู้การหาขอบวัตุทรงกรมไปใช้งานกับงานรูปแบบที่คล้ายกันได้ เช่น การตรวจจับเหรียญ การตรวจการฝนกระดาษคำตอบของข้อสอบ เป็นต้น

บทที่ 5

ปัญหา และข้อเสนอแนะ

จากการดำเนินการศึกษาข้อมูลเกี่ยวกับการทำงานและการพัฒนาโครงงานได้พบปัญหาต่างๆและมีข้อเสนอแนะในการพัฒนาโปรแกรมสำหรับพัฒนาโปรแกรมในอนาคตเพื่อนำไปใช้งานจริงดังนี้

5.1 ปัญหาและข้อผิดพลาด

- การแก้ปัญหาปริศนาเพื่อหาผลลัพธ์สำหรับบล็อกที่ใช้วาง ยิ่งมีบล็อกเยอะยิ่งใช้เวลามากในการทำงานของโปรแกรมเพื่อหาความเป็นไปได้ของผลลัพธ์ที่ดีที่สุด
- การตรวจจับวัตุที่ยังมีความถูกต้องน้อย อีกทั้งคุณภาพกล้องวีดีโอที่ใช้ตรวจจับมีความคมชัดไปมากพอและรวมไปถึงส่งรบกวนจากธรรมชาติ เช่น แสงที่ตกกระทบกับชิ้นส่วนบล็อกจึงที่ให้สีหรือรูปแบบบล็อกคลาดเคลื่อนหรือลมที่พัดทพให้กล้องสั่นจนไม่สามารถโฟกัสกับภาพชิ้นส่วนบล็อกได้ จนทำให้การตรวจจับชิ้นส่วนบล็อกเกิดการผิดพลาด
- การหาค่าขอบของวงกลมที่เกิดจากแสงเงาจนทำให้เกิดวงกลมหลายวงซ้อนทับกันจนทำให้ Hough Circle Transform เกิดการหาขอบของวงกลมที่คลาดเคลื่อน
- โมเดลที่มีการตรวจจับวัตถุซ้ำซ้อนกันจนทำให้ไม่สามารถแก้ปัญหาปริศนาได้
- ผลลัพธ์การแก้ปัญหาปริศนาที่ได้ยังไม่สามารถแสดงผลค่าสีของแต่ละบล็อกได้ ทำได้เพียงแต่แสดงค่าตัวอักษรของบล็อกในผลลัพธ์

5.2 ข้อเสนอแนะและแนวทางการแก้ไขปัญหา

- ทำการเทรนโมเดลโดยใช้ข้อมูลที่มากขึ้นและดีขึ้น เพื่อให้การตรวจจับวัตุมีความถูกต้องมากขึ้น
- ทำการใช้งานกล้องวีดีโอที่มีคุณภาพดี และใช้ในพื้นที่ที่มีแสงคงที่และไม่มีลม
- ขัดผิวของชิ้นส่วนบล็อกให้มีลักษณะขรุขระเพื่อไม่ให้มีแสงสะท้อนบนตัวบล็อก
- ทาสีชิ้นส่วนบล็อกให้มีสีที่แตกต่างกันอยากชัดเจนแทนการแต้มจุดบนตัวบล็อก
- แก้ไขฟังก์ชันการแก้ปริศนาโดยศึกษาทฤษฎีต่าง ๆ เพื่อให้ได้ผลลัพธ์ที่รวดเร็ว

บรรณานุกรม

- การเขียนโปรเกรมภาษา Python พื้นฐาน [ออนไลน์] สืบค้นเมื่อวันที่ 20 กรกฎาคม 2566 แหล่งที่มา (https://www.youtube.com/watch?v=N1fnq4MF3AE)
- การแก้ปัญหาการวางบล็อกของเกม IQ Block Puzzle [ออนไลน์] สืบค้นเมื่อวันที่ 20 กรกฎาคม 2566 แหล่งที่มา (https://github.com/quantixed/iq-block-solver)
- การแก้ปัญหาการวางบล็อกของเกม IQ Circuit [ออนไลน์] สืบค้นเมื่อวันที่ 20 กรกฎาคม 2566 แหล่งที่มา (https://github.com/adrienduque/IQ_circuit_solver)
- การใช้ Hough Circle Transform [ออนไลน์] สืบค้นเมื่อวันที่ 20 ตุลาคม 2566 แหล่งที่มา (https://medium.com/@isinsuarici/hough-circle-transform-parameter-tuning-with-examples-6b63478377c9)
- การใช้ Jupyter notebook [ออนไลน์] สืบค้นเมื่อวันที่ 20 กรกฎาคม 2566 แหล่งที่มา (https://www.dataquest.io/blog/jupyter-notebook-tutorial/)
- การใช้ Google Colab [ออนไลน์] สืบค้นเมื่อวันที่ 20 กรกฎาคม 2566 แหล่งที่มา (https://web.eecs.umich.edu/~justincj/teaching/eecs442//WI2021/colab.html)
- การเทรน YOLOv5 model บน Google Colab [ออนไลน์] สืบค้นเมื่อวันที่ 20 กรกฎาคม 2566 แหล่งที่มา (https://github.com/roboflow/notebooks?ref=blog.roboflow.com)
- การสร้างชุดข้อมูลจากการ Label ภาพ [ออนไลน์] สืบค้นเมื่อวันที่ 20 กรกฎาคม 2566 แหล่งที่มา (https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/)
- การเปรียบเทียบรูปภาพด้วยภาษา Python [ออนไลน์] สืบค้นเมื่อวันที่ 20 กรกฎาคม 2566 แหล่งที่มา (https://pyimagesearch.com/2014/09/15/python-compare-two-images/)