การวัดค่าความขุ่นและการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าว ด้วยวิธีทางคอมพิวเตอร์วิทัศน์

MEASUREMENT OF TURBIDITY AND DETECTION OF ADULTERATED OBJECT IN COCONUT OIL USING COMPUTER VISION

อรรถพล พลานนท์

ATTAPON PALANANDA

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาดุษฎีบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2565

KMITL-2022-SC-D-002-064

MEASUREMENT OF TURBIDITY AND DETECTION OF ADULTERATED OBJECT IN COCONUT OIL USING COMPUTER VISION

ATTAPON PALANANDA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF DOCTOR IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE SCHOOL OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2022

KMITL-2022-SC-D-002-064

หัวข้อวิทยานิพนธ์	การวัดค่าความขุ่นและการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าวด้วยวิธีทางคอมพิวเตอร์วิทัศน์
ชื่อนักศึกษา	นายอรรถพล พลานนท์
รหัสประจำตัว	57605017
ปริญญา	ปรัชญาดุษฎีบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา	วิทยาการคอมพิวเตอร์
พ.ศ.	2565
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผู้ช่วยศาสตราจารย์ ดร.วรางคณา กิ้มปาน

บทคัดย่อ

การผลิตน้ำมันมะพร้าวด้วยกรรมวิธีแบบสกัดเย็นช่วยให้สามารถรักษาคุณภาพและประโยชน์ของน้ำมันมะพร้าวให้คงอยู่ ซึ่งในแต่ละขั้นตอนของการผลิตต้องอยู่ในความควบคุมดูแลอย่างใกล้ชิดเพื่อตรวจสอบคุณภาพของน้ำมันมะพร้าวให้ดีอยู่เสมอ ซึ่งปัจจัยที่ส่งผลกระทบต่อคุณภาพน้ำมันมะพร้าวคือ สีของน้ำมันมะพร้าวและวัตถุปลอมปนในน้ำมันมะพร้าวที่เป็นตัวแปรสำคัญต่อคุณภาพของน้ำมันมะพร้าวโดยตรง โดยเฉพาะสีของน้ำมันมะพร้าวนั้นมีผลต่อความรู้สึกของผู้บริโภคโดยตรงจากการมองเห็น และปัจจัยที่ส่งผลกระทบทำให้น้ำมันมะพร้าวมีสีที่แตกต่างกันไปคือสิ่งปลอมปนที่อยู่ในน้ำมันมะพร้าวซึ่งเป็นผลกระทบที่ต่อเนื่องกัน ดังนั้นในงานวิจัยนี้ได้ทำการวัดค่าความขุ่นร่วมกับการตรวจจับสิ่งปลอมปนที่อยู่ในน้ำมันมะพร้าว โดยเริ่มต้นจากการวัดค่าความขุ่นของน้ำมันมะพร้าวโดยวิธี MAMoH ซึ่งเป็นวิธีที่ดีที่สุดจาก 4 วิธีการที่ได้นำเสนอ คือวิธีการ MoGS วิธีการ MoH หรือ วิธีการ RPMoH โดยที่ค่าประสิทธิภาพของ MAMoH อยู่ที่ 99 เปอร์เซ็นต์ ในลำดับต่อมาได้ทำการตรวจจับสิ่งปลอมปนที่อยู่ในน้ำมันมะพร้าวโดยใช้วิธีการเรียนรู้เชิงลึก ซึ่งทำการเปรียบเทียบสถาปัตยกรรมการเรียนรู้เชิงลึกจำนวน 10 สถาปัตยกรรม โดยสถาปัตยกรรม MobileNet นั้น ให้ค่าประสิทธิภาพของความแม่นยำในการทำงานอยู่ที่ 80.20 เปอร์เซ็นต์ ซึ่งให้ผลการทำงานที่ดีที่สุดเมื่อเทียบกับสถาปัตยกรรมแบบอื่น

คำสำคัญ: การประมวลผลภาพ การหาค่าความขุ่น การเรียนรู้เชิงลึก คอมพิวเตอร์วิทัศน์

Thesis Title	Measurement of Turbidity and Detection of Adulterated Object in Coconut Oil using Computer Vision
Student Name	Mr. Attapon Palananda
Student ID	57605017
Degree	Doctor of Philosophy (Computer Science)
Department	Computer Science
Year	2022
Thesis Advisor	Asst.Prof.Dr.Warangkhana Kimpan

Abstract

Coconut oil that has been processed by cold extraction helps maintain the quality and benefits of coconut oil. Each stage of the production process must be closely monitored to ensure that the quality of the coconut oil is always maintained. Color, turbidity, and impurities are the important parameters of coconut oil quality. Especially the turbidity of coconut oil, which impacts consumers' feelings directly through their vision. Impurities in coconut oil are the factors that cause the oil to become hazy or change color. In this research, turbidity was measured together with the detection of impurities in coconut oil. The experiment started with the measurement of turbidity of coconut oil. The results showed that the MAMoH method showed the best results for coconut oil turbidity when compared to other proposed methods: MoGS, MoH, and RPMoH method. The efficiency of MAMoH method was 99 percent. Subsequently, contaminants in coconut oil were detected using deep learning methods. The experimental results revealed that the MobileNet architecture is superior to 10 architectures in terms of detecting and categorizing impurities. The accuracy of MobileNet is 80.20 percent, which is the best performance compared to other architectures.

Keywords: Image processing, Turbidity level, Deep learning, Computer vision

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงได้นั้น เพราะได้รับความกรุณาจากบุคคลหลายๆ ท่าน ดังนี้

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.วรางคณา กิ้มปาน อาจารย์ที่ปรึกษา ที่ได้สละเวลาดูแลเอาใจใส่ รวมทั้งให้ข้อคิด ให้คำแนะนำและองค์ความรู้ที่มีประโยชน์ รวมถึงแนวทางในการศึกษาที่สามารถนำมาใช้ประกอบการทำงานวิจัยจนสำเร็จลุล่วงได้

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.กฤษณะ ชินสาร รองศาสตราจารย์ ดร.จีรพร วีระพันธุ์ ผู้ช่วยศาสตราจารย์ ดร.นวลสวาท หิรัญสกลวงศ์ และผู้ช่วยศาสตราจารย์ ดร.อนันตพร หรรษคุณาฒัย คณะกรรมการสอบวิทยานิพนธ์ ที่กรุณาให้คำแนะนำตลอดจนข้อชี้แนะจนทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดี

ขอขอบพระคุณ คุณสุรเดช นิลเอก กรรมการผู้จัดการบริษัท ทรอปิคานา ออยล์ จำกัด ที่สนับสนุนงานวิจัยและสถานที่ในงานวิจัยจนสำเร็จลงได้ด้วยดี

ขอขอบคุณพระคุณบิดา มารดา ที่สนับสนุนทางด้านการศึกษาตามที่ได้ตั้งใจไว้ อีกทั้งยังได้สนับสนุนดูแลเรื่องค่าใช้จ่ายต่างๆ ในระหว่างการศึกษา

ท้ายสุดขอขอบคุณกัลยาณมิตร และพี่น้องทุกคนที่ให้คำปรึกษา และช่วยอำนวยความสะดวกในด้านต่างๆ

สำหรับคุณงามความดีและประโยชน์อันใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดา มารดา อาจารย์ทุกท่าน ซึ่งเป็นผู้ที่เคารพรักยิ่ง ตลอดจนญาติพี่น้องและผู้เกี่ยวข้องทุกท่านที่ให้การสนับสนุน และเป็นกำลังใจด้วยดีเสมอมา ขอน้อมคารวะแด่ผู้เขียนบทความวิชาการ ตำราวิชาการต่างๆที่ได้ศึกษาค้นคว้า และใช้อ้างอิงทุกท่าน

นายอรรถพล พลานนท์

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากน้ำมันมะพร้าวมีประโยชน์ต่อสุขภาพของมนุษย์ทั้งทางตรงและทางอ้อม โดยนำมาผลิตเป็นน้ำมันบำรุงเส้นผมหรือบำรุงผิวพรรณ รวมถึงเครื่องสำอางที่ใช้บำรุงใบหน้าและน้ำมันมะพร้าวรักษาช่องปาก ฯลฯ ซึ่งในกรรมวิธีการผลิตวิธีที่คงคุณค่าของน้ำมันมะพร้าวได้ดีที่สุดคือการใช้วิธีการสกัดแบบเย็น โดยวิธีการดังกล่าวทำให้ได้น้ำมันมะพร้าวที่บริสุทธิ์ออกมา โดยในอุตสาหกรรมการผลิตน้ำมันมะพร้าวบริสุทธิ์ด้วยวิธีการสกัดเย็นนั้นได้ถูกแปรรูปเป็นผลิตภัณฑ์ที่ใช้เพื่อดูแลสุขภาพและเป็นประโยชน์ต่อร่างกาย ซึ่งในขั้นตอนการผลิตน้ำมันมะพร้าว น้ำมันที่ได้หลังจากการบีบเนื้อมะพร้าวจนได้น้ำมันมะพร้าวออกมา น้ำมันมะพร้าวที่ได้จะถูกส่งต่อไปยังขั้นตอนการกรองสิ่งปลอมปนที่อยู่ในน้ำมันมะพร้าวโดยใช้เครื่องหมุนเหวี่ยง (Centrifuge) เพื่อกรองสิ่งปลอมปนในน้ำมันมะพร้าวออกทำให้น้ำมันมะพร้าวมีความขุ่นน้อยที่สุด ซึ่งถูกควบคุมการทำงานโดยพนักงาน สำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนของน้ำมันมะพร้าวนั้นถูกตรวจสอบด้วยสายตาของพนักงาน และเมื่อพนักงานตรวจสอบพบว่าน้ำมันยังมีความขุ่นหรือยังมีสิ่งปลอมปนอยู่จะทำการหมุนเหวี่ยงน้ำมันอีกรอบจนพนักงานที่ตรวจสอบไม่พบสิ่งปลอมปน หรือไม่พบความขุ่นของน้ำมันมะพร้าวแล้วจึงหยุดการทำงาน จากนั้นจึงนำส่งน้ำมันมะพร้าวที่ได้ ไปยังกระบวนการพักน้ำมันมะพร้าวในถังเก็บน้ำมันมะพร้าวเพื่อรอให้สิ่งปลอมปนที่เป็นอนุภาคแขวนลอยนั้นทิ้งตัวลงสู่ด้านล่างของถังพัก และสุดท้ายเป็นการนำน้ำมันมะพร้าวที่ได้ไปบรรจุลงภาชนะขวดบรรจุภัณฑ์เพื่อขายต่อไป สำหรับการผลิตในแต่ละรอบนั้นน้ำมันมะพร้าวอาจมีสีที่แตกต่างกันได้ ซึ่งปัจจัยที่เกิดขึ้นมาจากการแปรรูปวัตถุดิบในส่วนของเนื้อมะพร้าวที่มีส่วนของเปลือกกะลาปะปนอยู่ด้วย ทำให้น้ำมันมะพร้าวที่ได้ในแต่ละรอบมีสีของน้ำมันออกไปทางเหลืองอ่อนและแตกต่างกันไปในแต่ละรอบของการผลิต

สำหรับการตรวจสอบความขุ่นของน้ำมันมะพร้าวในปัจจุบัน ใช้วิธีการส่องน้ำมันมะพร้าวผ่านหลอดไฟเพื่อเปรียบเทียบความขุ่นของน้ำมันในแต่ละรอบ โดยนำน้ำมันมะพร้าวที่ได้มาเปรียบเทียบกับสีของน้ำมันมะพร้าวตัวอย่างที่อยู่ในเกณฑ์มาตรฐานของบริษัท จากนั้นจะวัดจากสายตาของพนักงาน ซึ่งถ้าผลลัพธ์ของน้ำมันมะพร้าวนั้นสีเหลืองมากเกินไปต้องนำน้ำมันมะพร้าวในรอบการผลิตนั้นมาผ่านการกรองใหม่อีกครั้ง ด้วยวิธีการหมุนเหวี่ยงและการบีบอัดน้ำมัน ซึ่งวิธีการนี้ทำให้ต้องเสียเวลาในการผลิตน้ำมันมะพร้าว และได้ค่าผลลัพธ์ความขุ่นของน้ำมันมะพร้าวที่ไม่แน่นอน

จากความสำคัญของปัญหาดังกล่าวจำเป็นต้องมีเครื่องมือที่ช่วยในการตรวจสอบน้ำมันมะพร้าว ดังนั้นงานวิจัยนี้จึงเสนอการวัดค่าความขุ่นและการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าวด้วยวิธีทางคอมพิวเตอร์วิทัศน์ (Computer vision) ซึ่งเป็นการหาความขุ่นของน้ำมันมะพร้าวด้วยการประมวลผลภาพ (Image processing) และการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าวด้วยวิธีการ

เรียนรู้เชิงลึก (Deep learning) ผลลัพธ์ที่ได้สามารถนำไปประยุกต์ใช้งานในการตรวจวัดค่าความขุ่นในน้ำมันมะพร้าวและการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าวของกระบวนการผลิตได้จริง ซึ่งช่วยสร้างมาตรฐานในการผลิตได้

1.2 วัตถุประสงค์ของงานวิจัย

1) พัฒนาอุปกรณ์ต้นแบบที่ประยุกต์ใช้งานร่วมกับเครื่องจักรผลิตน้ำมันมะพร้าวหรือเครื่องบรรจุน้ำมันมะพร้าวได้

2) พัฒนากระบวนการวัดความขุ่นในน้ำมันมะพร้าวด้วยวิธีทางคอมพิวเตอร์วิทัศน์

3) พัฒนารูปแบบการเรียนรู้เชิงลึกสำหรับการระบุสิ่งปลอมปนในน้ำมันมะพร้าว

1.3 ขอบเขตของงานวิจัย

1) การวัดค่าความขุ่นในน้ำมันมะพร้าวต้องใช้งานร่วมกับอุปกรณ์ต้นแบบที่พัฒนาขึ้นโดยเฉพาะ

2) ระบบแสงสว่างภายในอุปกรณ์ต้นแบบต้องได้รับการตรวจสอบความสว่างอย่างสม่ำเสมอ

3) ข้อมูลภาพที่ได้ต้องอยู่ภายใต้การควบคุมแสงในระบบปิดเพื่อควบคุมปริมาณแสงให้คงที่

4) ข้อมูลภาพที่ได้จากกล้องไมโครสโคปเป็นภาพที่ได้จากกำลังขยายของเลนส์ 500 เท่า

5) การวัดค่าความขุ่นและการตรวจจับวัตถุปลอมปนในน้ำมันมะพร้าว อยู่ในขั้นตอนการผลิตก่อนการบรรจุน้ำมันมะพร้าวลงภาชนะขวดบรรจุภัณฑ์

6) การตรวจจับวัตถุในน้ำมันมะพร้าวอยู่บนพื้นฐานวิธีการเรียนรู้เชิงลึก

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้อุปกรณ์ในการตรวจวัดความขุ่นของน้ำมันมะพร้าว

2) ได้กระบวนการทางการประมวลผลภาพในการวัดค่าความขุ่นของน้ำมันมะพร้าว

3) ได้กระบวนการจำแนกวัตถุปลอมปนในน้ำมันมะพร้าวด้วยวิธีการเรียนรู้เชิงลึก

4) สามารถนำค่าการตรวจวัดความขุ่นที่ได้มาทดแทนการตรวจสอบความขุ่นในปัจจุบัน

5) สามารถบอกค่าความขุ่นของน้ำมันมะพร้าวได้แบบเวลาจริงที่ใช้ในขั้นตอนการผลิต

6) สามารถตรวจจับวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าวได้

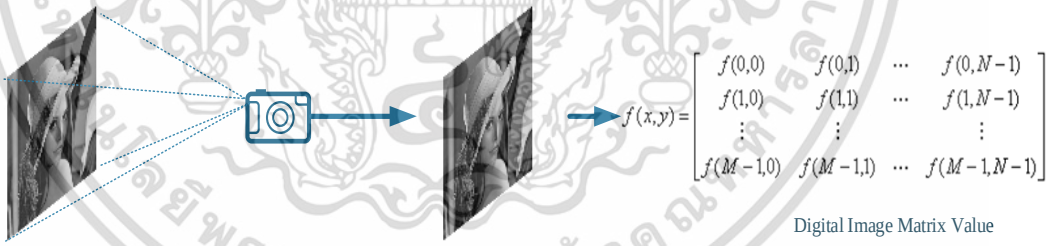
7) สามารถแยกแยะวัตถุปลอมปนที่ปะปนอยู่น้ำมันมะพร้าวได้

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในการศึกษาและค้นคว้าข้อมูลด้านคอมพิวเตอร์วิทัศน์นั้น สามารถนำมาประยุกต์ใช้ในการตรวจวัดค่าความขุ่นหรือสามารถประยุกต์ใช้ในการหาวัตถุปลอมปนแล้วจำแนกวัตถุปลอมปนนั้นได้ ผู้วิจัยได้ทำการศึกษาและทบทวนวรรณกรรมและงานวิจัยที่เกี่ยวข้อง โดยมีรายละเอียดดังนี้

2.1 ความรู้เกี่ยวกับการประมวลผลภาพ

การประมวลผลภาพ (Image processing) เป็นการประมวลผลข้อมูลที่ได้จากกล้องถ่ายภาพ โดยสัญญาณที่ได้จากกล้องถ่ายภาพต้องทำการแปลงให้อยู่ในรูปของภาพดิจิทัล (Digital image) [1] จึงจะสามารถนำไปประมวลผลต่อได้ โดยโครงสร้างของภาพดิจิทัลมีลักษณะการเก็บค่าข้อมูลแบบอาร์เรย์ (Array) ซึ่งในตำแหน่งของอาร์เรย์แต่ละช่องแสดงถึงคุณลักษณะของจุดภาพ (Pixel) ที่อยู่ในภาพ สามารถนิยามเป็นฟังก์ชันสองมิติ $f(x,y)$ โดยที่ x และ y เป็นพิกัดของภาพและแอมพลิจูดของ f ที่พิกัด (x,y) ภายในภาพคือค่าความเข้มแสงของภาพ (Intensity) ในตำแหน่งนั้น และถ้ากำหนดให้ภาพ $f(x,y)$ มีขนาด M แถวและ N คอลัมน์ และพิกัดของจุดกำเนิด (Origin) ของภาพ คือที่ตำแหน่ง (x,y) = (0,0) ซึ่งพื้นฐานของการได้มาของภาพดิจิทัล ดังรูปที่ 2.1

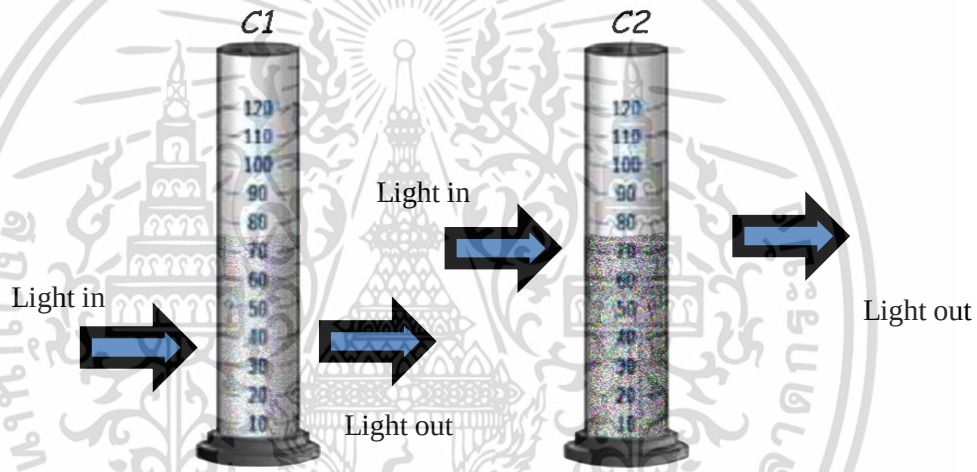


รูปที่ 2.1 การได้มาของภาพดิจิทัล

จากรูปที่ 2.1 ค่าแต่ละค่า ที่อยู่ใน $f(x,y)$ เรียกว่าจุดภาพ โดยตำแหน่ง (0,0) นั้นอยู่ทางด้านซ้ายมือสุดและด้านบนสุดของภาพ การจัดลำดับตำแหน่งของจุดภาพนั้นมีการเรียงจากซ้ายไปขวาและเรียงจากด้านบนลงสู่ด้านล่าง การเก็บค่าของความเข้มแสงของภาพมีการเก็บข้อมูลในลักษณะภาพบิตแมป (Bit-map image) หรือภาพแรสเตอร์ (Raster image) ซึ่งภาพดิจิทัลสามารถนำไปใช้งานร่วมกับฟังก์ชันทางคณิตศาสตร์หรือแบบจำลองเรขาคณิต เพื่อนำไปประมวลผลในงานต่าง ๆ ได้

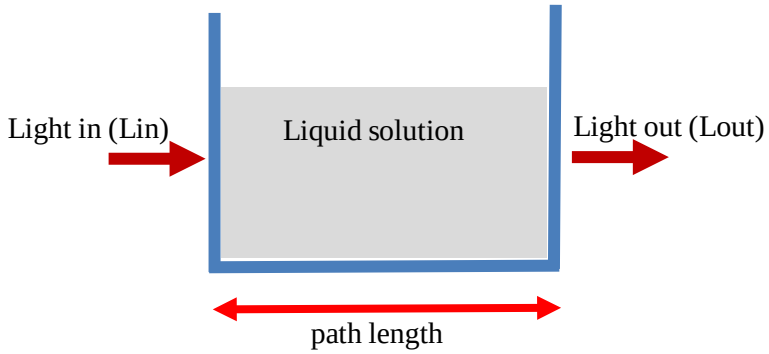
2.2 ความรู้เกี่ยวกับการหาค่าความขุ่นในของเหลว

ในการหาค่าความขุ่นในของเหลวมีด้วยกันหลายวิธี [2,3] ซึ่งวิธีการหนึ่งที่สามารถนำมาใช้ในการวัดความขุ่นในของเหลวได้ คือการวัดความสามารถของแสงที่ส่องลอดผ่านของเหลว สามารถนำมาใช้สำหรับการตรวจวัดได้ 2 วิธีการคือ วิธีการที่ 1 การวัดจากจานเซคิดิสก์ (Secchi disk) ซึ่งเป็นวิธีวัดความโปร่งแสงจากชั้นความลึกในของเหลว และวิธีการที่ 2 การใช้หลอดวัดความโปร่งแสง (Transparency tube) ซึ่งทั้ง 2 วิธีการสามารถนำมาใช้สำหรับการวัดความขุ่นในของเหลวที่ไหลผ่านได้โดยการอาศัยแสงที่ส่องลอดผ่านตัวกลางออกมา และสำหรับแสงที่ส่องลอดผ่านออกมาเมื่อถูกตัวกลางดูดกลืนแสงไว้ จะมีการแปรผันตรงกับปริมาณความขุ่นของตัวกลางที่ดูดกลืนแสงนั้นไว้ ดังรูปที่ 2.2



รูปที่ 2.2 การวัดค่าของแสงที่ส่องลอดผ่านตัวกลางที่แตกต่างกัน

จากรูปที่ 2.2 สามารถอธิบายได้ว่า ถ้าความเข้มข้น C2 > C1 ดังนั้นแสงที่ส่องลอดผ่านสารละลาย C2 ออกมาจะมีปริมาณความเข้มของแสงที่เหลือน้อยกว่าปริมาณความเข้มของแสงที่ส่องผ่านสารละลาย C1 เนื่องจากความเข้มข้นของสารละลายที่อยู่ใน C2 มีโมเลกุลที่สามารถดูดกลืนแสงหรือสามารถขวางทางเดินของแสงอยู่มากกว่าสารละลาย C1 เมื่อทำการวัดการดูดกลืนแสงของสารละลาย ปริมาณความเข้มของแสงที่ถูกดูดกลืนจะขึ้นอยู่กับค่าความเข้มข้นของสารละลายและค่าความหนาของสารละลายที่ลำแสงสามารถส่องลอดผ่านได้ โดยจากทฤษฎีของเบียร์-แลมเบิร์ต (Beer-Lambert law) [4] การวัดค่าการดูดกลืนแสงของสารตัวอย่าง สามารถทำได้โดยให้ลำแสงส่องลอดผ่านเข้าไป (Light in: Lin) ยังตัวอย่างของสารละลาย จากนั้นวัดปริมาณแสงที่ลอดผ่านออกมา (Light out: Lout) จากนั้นทำการเทียบกับแสงที่ผ่านออกมาเมื่อไม่มีสารตัวอย่าง ดังรูปที่ 2.3



รูปที่ 2.3 แสงที่ส่องลอดผ่านสารละลายของเหลว

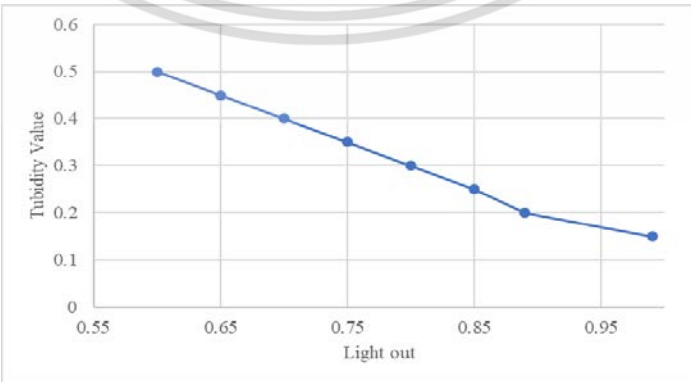
โดยที่ค่าการส่องผ่าน (Transmittance: T) เป็นอัตราส่วนปริมาณแสงที่ผ่านออกมา (Lout) ต่อปริมาณแสงที่ลอดผ่านเข้าไปในตัวอย่าง (Lin) โดยสามารถหาค่าได้จากสมการที่ (2.1)

$$T = \frac{Iout}{Iin} \tag{2.1}$$

สำหรับการหาค่าการดูดกลืนแสง (Absorbance: A) สามารถหาค่าได้จากสมการที่ (2.2)

$$A = \log\frac{Iout}{Iin} = -\log T \tag{2.2}$$

ซึ่งค่าที่ได้จากทั้ง 2 สมการ สามารถอธิบายความสอดคล้องกันของความเข้มแสงและการดูดกลืนของแสงที่ส่องผ่านตัวกลางได้ หรือสามารถอธิบายความสัมพันธ์ของแสงที่ขึ้นอยู่กับปริมาณของสารละลายหรือของเหลวได้ตามกฎของเบียร์และแลมเบิร์ต โดยเมื่อนำความสัมพันธ์ของค่าการดูดกลืนแสง มาสร้างกราฟความสัมพันธ์จะได้ความสัมพันธ์เชิงเส้นหรือกราฟความเข้มแสงมาตรฐาน ดังรูปที่ 2.4



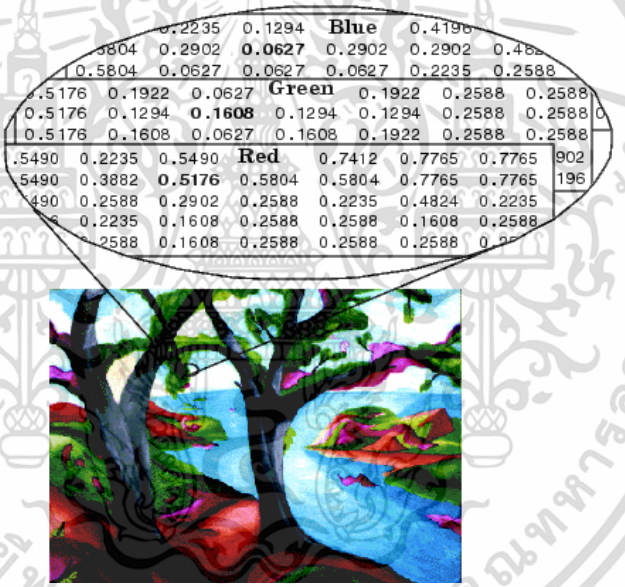
รูปที่ 2.4 ความสัมพันธ์เชิงเส้นของความเข้มแสงมาตรฐาน

2.3 ความรู้เกี่ยวกับโดเมนสี (Domain of color)

โดเมนสีเป็นรูปแบบในการกำหนดมาตรฐานสำหรับการแสดงผล โดยใช้ค่าตัวเลขแทนค่าของสีที่เกิดขึ้น โดเมนของสีที่ถูกนำมาใช้ในระบบวิดิทัศน์ ได้แก่ ระบบสีแบบ RGB ระบบสี YCbCr และระบบสี HSV เป็นต้น

2.3.1 ระบบสีแบบ RGB

ภาพสี RGB [5] – [7] หรือภาพสีจริง (True color Image) ซึ่งเป็นภาพสีที่มีความใกล้เคียงกับสีในธรรมชาติมากที่สุด ซึ่งข้อมูลที่อยู่ในภาพมีรูปแบบลักษณะคล้ายกับการเก็บข้อมูลในอาร์เรย์ (Array) 3 มิติ ขนาด m x n x 3 โดยที่ m แทนขนาดความยาว และ n แทนขนาดความกว้างในหน่วยของจุดภาพ (Pixel) ส่วน 3 เป็นค่าที่ใช้แทนมิติ โดยในแต่ละมิติจะเก็บค่าสีแยกจากกัน โดยภาพสี RGB นั้นมี ชั้นสีแดง (Red: R) ชั้นสีเขียว (Green: G) และชั้นสีน้ำเงิน (Blue: B) ดังรูปที่ 2.5



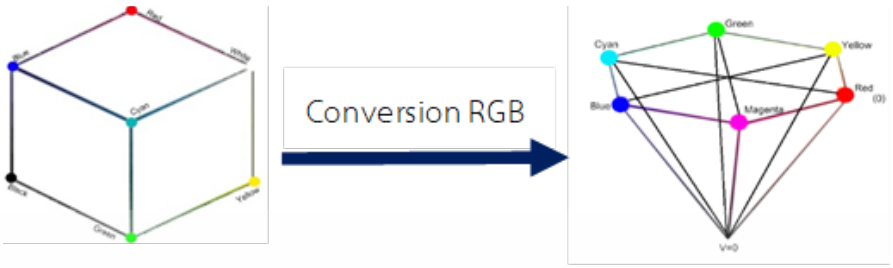
รูปที่ 2.5 ระบบสีแบบ RGB

ที่มา: http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/images/introa.gif

2.3.2 ระบบสีแบบ HSV

ระบบสีแบบ HSV (Hue: V, Saturation: S, Value: V) [8, 9] เป็นหนึ่งในระบบสัญญาณสีที่ใช้ในระบบภาพดิจิทัล โดยแตกต่างจากระบบสี RGB หรือ Y:Cb:Cr โดยภาพระบบสี HSV นั้น ประกอบด้วยค่า Hue คือ ค่าสีของสีหลัก หรือค่าสี RGB ซึ่งมีค่าอยู่ระหว่าง 0 และ 255 โดยถ้าค่า Hue มีค่าเท่ากับ 0 จะใช้แทนสีแดง และเมื่อค่า Hue มีค่าเพิ่มขึ้น สีภาพจะมีการเปลี่ยนแปลงไปตามสเปกตรัมของสีจนถึง 256 และกลับมาเป็นสีแดงอีกครั้ง ซึ่งสามารถเขียนในรูปขององศาได้

ดังนี้คือ สมมุติให้สีแดงเท่ากับ 0 องศา สีเขียวเท่ากับ 120 องศา และสีน้ำเงินเท่ากับ 240 องศา โดยมุมองศาของระบบ HSV แสดงได้ดังรูปที่ 2.6



รูปที่ 2.6 ระบบสีแบบ HSV

สำหรับในการแปลงระบบสี RGB ให้เป็นระบบสี HSV นั้น ทั้งสองระบบสีสามารถแปลงค่าซึ่งกันและกันได้ โดยระบบสี HSV เป็นการแปลงระบบสีที่ใกล้เคียงกับการประมวลผลสีของมนุษย์ ซึ่งมนุษย์สามารถเข้าใจได้ดีกว่าระบบสี RGB โดยสามารถแปลงค่าความสัมพันธ์ระหว่าง RGB และ HSV ซึ่งสามารถหาค่า H ได้จากสมการที่ (2.3) ค่า S สามารถหาได้จากสมการที่ (2.5) และค่า V สามารถหาได้จากสมการที่ (2.6)

$$H = \begin{cases} \delta, & B \leq G \\ 2\pi - \delta, & otherwise \end{cases} \tag{2.3}$$

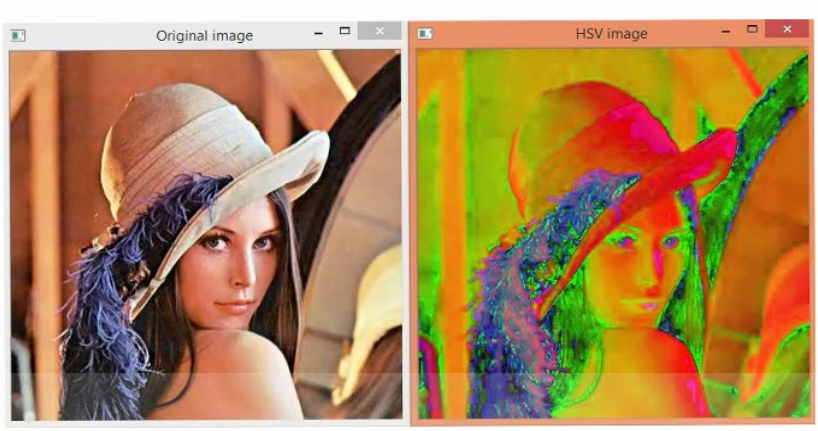
โดยที่ค่า δ หาได้จากสมการที่ (2.4)

$$\delta = \cos^{-1}\left(\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)\cdot(G-B)}}\right) \tag{2.4}$$

$$S = 1 - 3 \cdot \frac{Min(R,G,B)}{R+G+B} \tag{2.5}$$

$$V = \frac{R+G+B}{3} \tag{2.6}$$

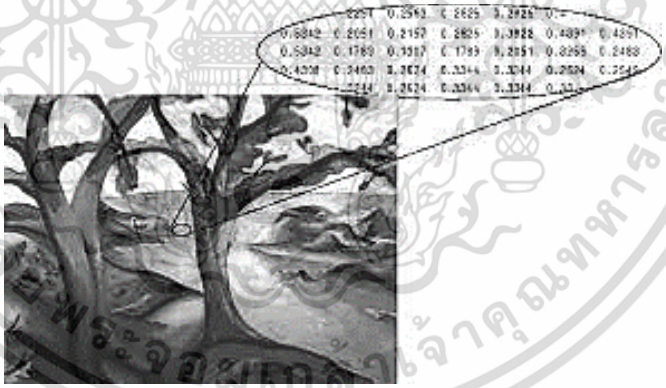
ผลลัพธ์ที่ได้จากการคำนวณการแปลงภาพระบบสี RGB ให้เป็นระบบสี HSV จากสมการที่ (2.3) - (2.6) สามารถแสดงได้ดังรูปที่ 2.7



รูปที่ 2.7 ผลการแปลงระบบสี RGB ให้เป็นระบบสี HSV

2.3.3 ระบบระดับสีเทา (Grayscale)

ภาพในระดับสีเทาเป็นภาพ [10]–[12] เป็นภาพที่แต่ละจุดภาพมีค่าความเข้มของสีอยู่ระหว่างขาวและดำ โดยมีค่าที่เป็นไปได้ของภาพระดับสีเทา (Grayscale image) จำนวน 8 บิต (Bits) อยู่ที่ 256 ระดับ ในการใช้งานสามารถกำหนดค่าให้อยู่ในช่วง 0-1 หรือ 0-255 ค่าได้ สามารถแสดงภาพระดับสีเทาได้ดังรูปที่ 2.8



รูปที่ 2.8 ภาพระดับสีเทา

การแปลงภาพสีให้เป็นภาพระดับเทาแบบค่าเฉลี่ย (Grayscale average) ทำได้โดยการนำจุดภาพที่อ่านค่าจากโดเมนสี RGB มาแยกออกเป็น ย่าน R ย่าน G และย่าน B จากนั้นนำมาหาค่าเฉลี่ยของจุดภาพ ด้วยสมการที่ (2.7)

$$Pixel\,(x,y) = (R + G + B)/3 \tag{2.7}$$

โดยที่

$Pixel\ (x,y)$ คือ ตำแหน่ง ณ จุดภาพใด ๆ ที่ (x,y)

R คือ ค่าระดับสีแดงของจุดภาพ

G คือ ค่าระดับสีเขียวของจุดภาพ

B คือ ค่าระดับสีน้ำเงินของจุดภาพ

ดังนั้น ค่าน้ำหนักที่ได้ของภาพระดับเทาสามารถหาได้จาก ค่า R 33% ค่า G 33% และค่า B 33% ซึ่งวิธีการนี้เป็นวิธีการเบื้องต้นในการแปลงภาพระดับเทา ซึ่งอาจแยกแยะรายละเอียดบางส่วนของภาพไม่ได้ ภาพที่ได้อาจมีสีดำกลืนกันไป โดยสาเหตุที่เกิดขึ้นเนื่องมาจากความยาวคลื่นของแสงแต่ละแสงมีไม่เท่ากัน โดยความยาวคลื่นของแสงสีแดงนั้นยาวที่สุด และแสงสีเขียวและแสงสีน้ำเงินตามลำดับ ผลที่ได้จากการแปลงภาพระดับเทาดังรูปที่ 2.9

(ก) ภาพต้นฉบับ (ข) ภาพระดับเทาแบบค่าเฉลี่ย

รูปที่ 2.9 เปรียบเทียบผลลัพธ์ของภาพระดับเทาแบบค่าเฉลี่ย

2.4 การแบ่งบางส่วนของภาพ (Image Segmentation)

ในการแบ่งบางส่วนของภาพ [13,14] มีด้วยกันหลายวิธีการ ซึ่งวิธีการหนึ่งที่ได้เลือกนำมาใช้คือการนำเอาเฉพาะส่วนที่ต้องการของภาพ (Region of interest) มาประมวลผล โดยวิธีการนี้ต้องทราบถึงตำแหน่งที่ไม่ต้องการและตำแหน่งที่ต้องการอย่างชัดเจน โดยมีขั้นตอนการทำงานเมื่อได้ตำแหน่งที่ต้องการในภาพแล้ว จะทำการตีกรอบสี่เหลี่ยมล้อมรอบพื้นที่ที่ต้องการเอาไว้ ซึ่งเป็นไปตามการระบุตำแหน่งในภาพเฉพาะส่วนที่ต้องการดังกล่าวมาประมวลผล โดยใน 1 ภาพ สามารถกำหนดได้หลายส่วนของภาพ ผลที่ได้แสดงดังรูปที่ 2.10

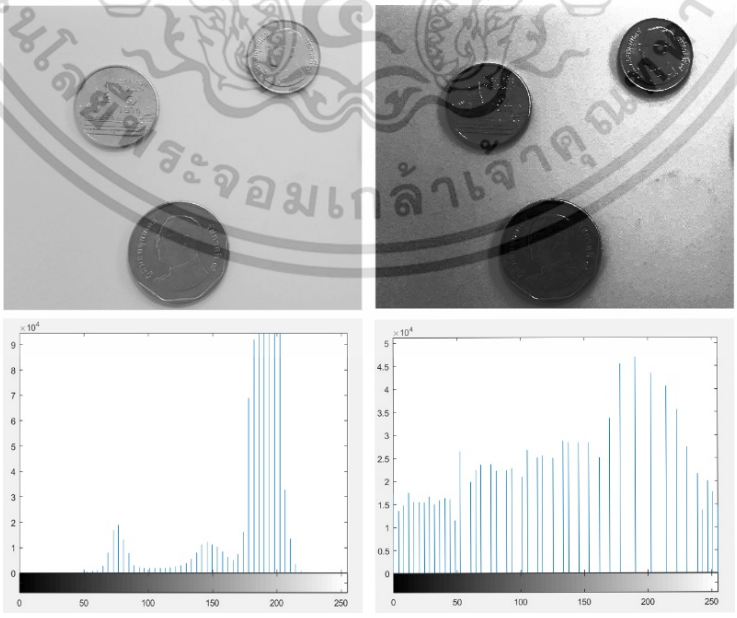
รูปที่ 2.10 การแบ่งภาพเฉพาะส่วนที่ต้องการ

2.5 การปรับปรุงภาพ (Image Enhancement)

สำหรับการปรับปรุงภาพ [15,16] เป็นการทำให้คุณภาพของภาพนั้นดีขึ้น ทำให้ส่วนของภาพนั้นมีความเด่นชัดขึ้นมา ซึ่งมีวิธีการทางการปรับปรุงภาพด้วยกันหลายวิธี เช่นการลดสัญญาณรบกวนที่มีอยู่ในภาพ หรือการเกลี่ยเฉดสีของภาพให้มีความเรียบเพื่อใช้ในการประมวลผลภาพต่อไป

2.5.1 การปรับปรุงภาพด้วยวิธีการปรับเท่าฮิสโตแกรม (Histogram Equalization)

เป็นการปรับรูปภาพเพื่อเสริมสร้างคุณลักษณะบางอย่างของภาพให้มีความเด่นชัดขึ้น [5,10,17] หรือเพื่อให้เห็นถึงรายละเอียดที่อยู่ในภาพ โดยอาศัยการกระจายตัวของฮิสโตแกรม ซึ่งเป็นการทำให้ภาพนั้นมีความเข้มของแสงที่เท่าเทียมกันหรือการไล่แสงที่สม่ำเสมอกัน และด้วยวิธีการปรับปรุงการกระจายความถี่ของกราฟสะสมความเข้มแสงนั้น ผลที่ได้จากการปรับปรุงภาพด้วยวิธีปรับเท่าฮิสโตแกรม นั้นแสดงดังรูปที่ 2.11



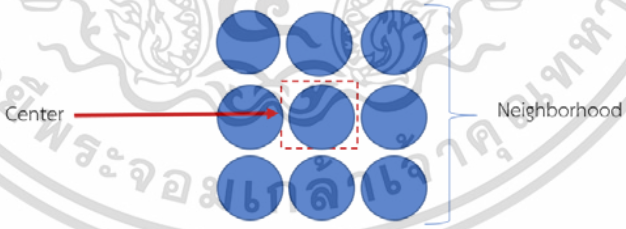
รูปที่ 2.11 การปรับปรุงภาพด้วยวิธีการปรับเท่าฮิสโตแกรม

จากรูปที่ 2.11 ลักษณะของการกระจายตัวของฮิสโตแกรมช่วยแสดงลักษณะเด่นหรือความคมชัดของภาพ โดยฮิสโตแกรมที่มีการกระจายเป็นแถบกว้าง แสดงถึงภาพมีความคมชัดที่สูง และฮิสโตแกรมที่มีการกระจายเป็นแถบแคบจะแทนภาพที่มีความคมชัดที่ต่ำ

2.5.2 การลดสัญญาณรบกวนในภาพด้วยมอร์โฟโลยี

วิธีการลดสัญญาณรบกวนในภาพมีด้วยกันหลายวิธี ซึ่งวิธีการมอร์โฟโลยี [5,10] (Morphology) ได้ถูกนำมาใช้ในการลดสิ่งรบกวนในภาพ ด้วยการเปลี่ยนแปลงลักษณะรูปร่างหรือโครงร่างของภาพ จากการแยกส่วนประกอบของภาพเพื่อประมวลผลแสดงรูปร่างใหม่ด้วยการใช้องค์ประกอบโครงสร้าง (Structuring element) โดยดำเนินการอยู่บนภาพขาวดำที่มีค่าระดับแค่ 0 หรือ 1 เท่านั้น การทำงานขององค์ประกอบโครงสร้างถูกใช้เป็นตัวกำหนดรูปร่าง และมีการดำเนินการพื้นฐานของมอร์โฟโลยีอยู่ 2 ส่วน ได้แก่ การกัดเซาะ (Erosion) และการขยาย (Dilation) ซึ่งสามารถนำตัวดำเนินการมาพัฒนาพื้นฐานดำเนินการได้อีก 2 ส่วน คือการเปิด (Opening) และการปิด (Closing)

การกำหนดองค์ประกอบโครงสร้างเป็นการกำหนดเมทริกซ์ย่อยเพื่อใช้ดำเนินการกับรูปภาพ ซึ่งสามารถเรียกได้อีกชื่อหนึ่ง คือ เคอร์เนล (Kernels) ขนาดขององค์ประกอบโครงสร้างนั้นส่วนใหญ่แล้วจะเป็นจำนวนคี่ เช่น 3x3 หรือ 5x5 เป็นต้น ดังรูปที่ 2.12 เพื่อให้ค่าน้ำหนักของคำตอบอยู่ที่ตำแหน่งกึ่งกลาง (Center) ขององค์ประกอบโครงสร้าง โดยนำมาประมวลผลร่วมกับจุดรอบข้าง (Neighborhood) ขององค์ประกอบโครงสร้าง จากนั้นนำคำตอบที่ได้แทนในตำแหน่งเดียวกันของภาพใหม่

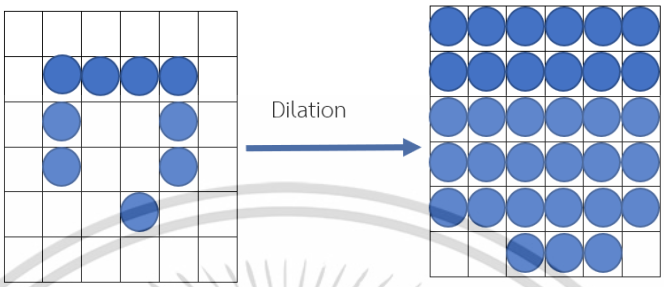


รูปที่ 2.12 ตัวอย่างองค์ประกอบโครงสร้าง

1) การขยาย (Dilation)

การขยายเป็นการพิจารณาข้อมูลภาพขาวดำ ซึ่งค่าของจุดภาพมีค่าเท่ากับ 0 หรือ 1 เท่านั้น โดยที่ค่า 0 แทนด้วยสีดำ และค่า 1 แทนสีขาว การขยายภาพให้ใหญ่ขึ้นเพื่อเพิ่มพื้นที่ของจุดภาพสีดำ ซึ่งการขยายพื้นที่ของวัตถุจะถูกดำเนินการผ่านองค์ประกอบโครงสร้าง โดยการเคลื่อนที่ผ่านครั้งละตำแหน่งบนข้อมูลภาพและทำการเดินตลอดทั้งภาพ โดยถ้าข้อมูลในภาพ Hit กับ

องค์ประกอบโครงสร้าง หมายถึง บริเวณที่องค์ประกอบโครงสร้างตัดกับส่วนใดส่วนหนึ่งของจุดภาพบนรูปภาพ ซึ่งถูกกำหนดให้คำตอบมีค่าเป็น 1 และในกรณี Miss กับองค์ประกอบโครงสร้าง หมายถึง บริเวณที่องค์ประกอบโครงสร้างไม่ตัดกับส่วนใดส่วนหนึ่งของจุดภาพที่บนรูปภาพ ซึ่งจะถูกกำหนดให้คำตอบมีค่าเป็น 0 ผลลัพธ์ที่ได้จากวิธีการขยายแสดงดังรูปที่ 2.13

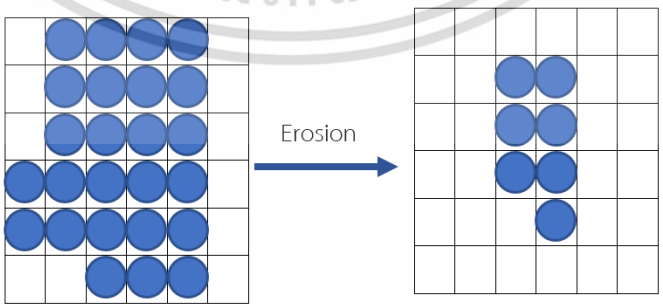


รูปที่ 2.13 ผลลัพธ์ที่ได้จากวิธีการขยาย

ผลที่ได้จากการทำการขยาย เป็นการเพิ่มจำนวนจุดภาพดำเพื่อให้ขยายขนาดของกลุ่มจุดภาพดำ ซึ่งช่วยลดช่องว่างให้น้อยลงส่งผลให้เกิดการเชื่อมต่อกันระหว่างกลุ่มของจุดภาพดำที่อยู่ใกล้กันได้

2) การกร่อน (Erosion)

เป็นการพิจารณาข้อมูลภาพขาวดำเช่นเดียวกันกับวิธีการขยาย แต่วิธีการกร่อนเป็นการกร่อนขนาดบริเวณขอบของวัตถุ ซึ่งการกร่อนพื้นที่ของวัตถุจะถูกดำเนินการผ่านองค์ประกอบโครงสร้าง จากนั้นเคลื่อนที่ผ่านทีละตำแหน่งบนข้อมูลภาพโดยทำเดินการตลอดทั้งภาพ โดยถ้าข้อมูลในภาพ hit หรือ fit กับองค์ประกอบโครงสร้าง จะถูกกำหนดให้คำตอบมีค่าเป็น 1 และในกรณี miss จะถูกกำหนดให้คำตอบมีค่าเป็น 0 ดังรูปที่ 2.14

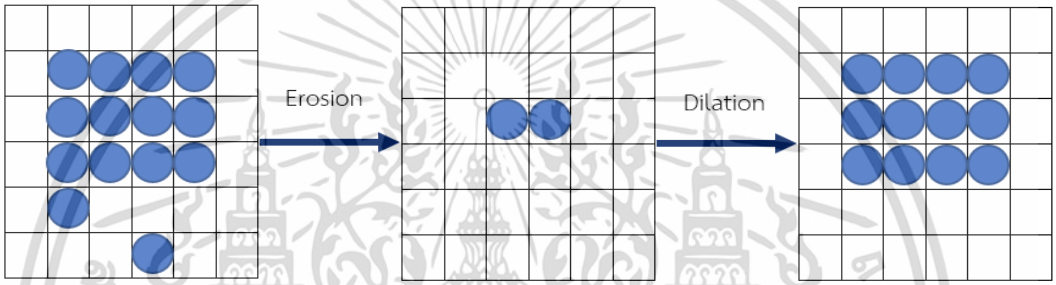


รูปที่ 2.14 ผลลัพธ์ที่ได้จากวิธีการกร่อน

วิธีการกร่อนภาพนั้นเป็นวิธีการลดจำนวนจุดภาพดำที่อยู่ในภาพ ซึ่งช่วยเพิ่มระยะห่างให้กว้างขึ้น ระหว่างกลุ่มของจุดภาพดำที่อยู่ในตำแหน่งใกล้กัน หรือเป็นการกำจัดกลุ่มของจุดดำที่มีขนาดเล็กออกไปได้

3) วิธีการเปิด (Opening)

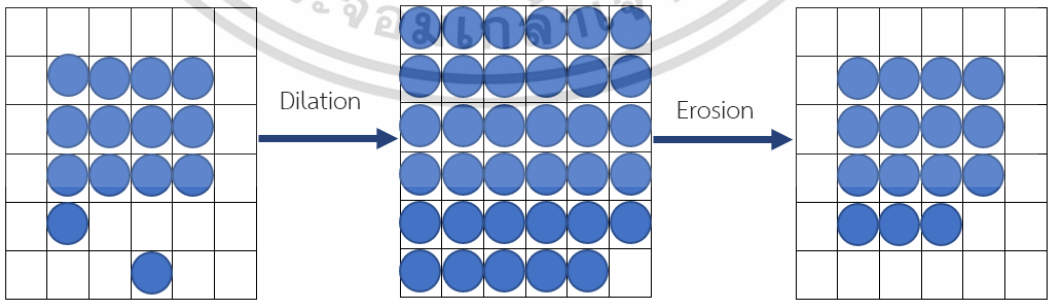
เป็นการทำงานร่วมกันระหว่างการขยายและการกร่อน ซึ่งแบ่งการทำงานเป็น 2 รอบ โดยรอบแรกเป็นการดำเนินการด้วยการกร่อน จากนั้นในรอบที่สองเป็นการดำเนินการด้วยการขยาย ซึ่งช่วยในการลดสิ่งรบกวน (Noise) ที่มีอยู่ภายในภาพและคืนขอบของภาพด้วยการขยายได้ ซึ่งทำให้ขอบของวัตถุในภาพไม่เสียหาย ผลลัพธ์ที่ได้จากวิธีการเปิดแสดงดังรูปที่ 2.15



รูปที่ 2.15 ผลลัพธ์ที่ได้จากวิธีการเปิด

4) วิธีการปิด (Closing)

เป็นการทำงานร่วมกันระหว่างการขยายและการกร่อนเช่นเดียวกันกับวิธีการเปิด แต่แตกต่างกันตรงที่รอบแรกถูกดำเนินการด้วยการขยาย จากนั้นในรอบที่สองเป็นการดำเนินการด้วยการกร่อน ผลลัพธ์ที่ได้จากวิธีการปิดแสดงดังรูปที่ 2.16



รูปที่ 2.16 ผลลัพธ์ที่ได้จากวิธีการปิด

2.5.3 การกรองภาพทางตำแหน่ง

การกรองภาพ (Filtering image) [18] เป็นวิธีการที่ช่วยเน้นส่วนของรายละเอียดของภาพช่วยทำให้ภาพมีความเด่นชัดขึ้น ด้วยการลดการเบลอของภาพ หรือการเน้นขอบของภาพ โดยภาพนั้นอาจมีความความผิดปรกติที่เกิดจากการถ่ายภาพ หรือจากสัญญาณรบกวนต่าง ๆ ได้ ซึ่งสามารถทำการกรองภาพด้วยการกรองข้อมูลด้วยวิธีการดังต่อไปนี้

1) การกรองแบบค่าเฉลี่ยเคลื่อนที่ (Mean moving average filter)

เป็นการกรองโดยอาศัย Spatial convolution [18] - [20] ซึ่งมีค่าน้ำหนักที่เท่ากัน คือ $\frac{1}{s^2}$ เมื่อ S คือขนาดของ Kernel โดย เช่น น้ำหนักในแต่ละช่องจะเป็น 1/9 ถ้า Kernel นั้น มีขนาด 3x3 ดังรูปที่ 2.17 โดยการกรองแบบนี้เป็นการกรองข้อมูลแบบเชิงเส้นและเป็นแบบยินยอมให้ความถี่ต่ำผ่านได้ (Low-pass filter)

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

รูปที่ 2.17 หน้าต่างเคลื่อนที่ขนาด 3x3

สำหรับการกรองด้วยวิธีการนี้ สามารถทำได้โดยใช้การคอนโวลูชัน (Convolution) ด้วยการเลื่อนหน้าต่างเคลื่อนที่กวาดไปบนภาพที่ละจุดภาพจากซ้ายไปขวา และจากบนลงล่าง ตามสมการกรองภาพที่มีสัญญาณรบกวนแสดงดังสมการที่ (2.8)

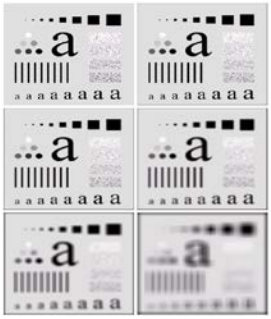
$$g(x,y) = \frac{1}{N_t}\sum_{i_t}\sum_{j\in H} i(x-i, y-i) \tag{2.8}$$

โดยที่ $i(x,y)$ เป็นตำแหน่งของภาพที่มีสัญญาณรบกวนเกิดขึ้น

$g(x,y)$ เป็นภาพที่ได้หลังจากทำการกรองภาพแล้ว

N_t เป็นจำนวนช่องของ Kernel H

การกรองแบบค่าเฉลี่ยเคลื่อนที่สามารถกำจัดสัญญาณรบกวนแบบเกาส์เซียน (Gaussian noise) ได้ แต่ในขณะเดียวกันจะส่งผลทำให้ภาพเบลอขึ้นได้ ทำให้เส้นหรือขอบของภาพไม่ชัดเจน ตัวอย่างดังรูปที่ 2.18



รูปที่ 2.18 ผลของการใช้การกรองแบบค่าเฉลี่ยเคลื่อนที่ที่มีขนาดต่างกัน

นอกจากกรองแบบค่าเฉลี่ยเคลื่อนที่กับภาพในลักษณะ 2 มิติ แล้วค่าเฉลี่ยเคลื่อนที่นั้นยังสามารถช่วยปรับปรุงข้อมูลที่มีความไม่เรียบและไม่คงตัว หรือข้อมูลที่มีสิ่งรบกวนสูงแบบ 1 มิติได้ด้วย โดยเส้นค่าเฉลี่ยที่เกิดขึ้นใหม่นั้นจะมีการนำค่าในอดีตมาคิดและคำนวณประมวลผลร่วมอยู่ด้วย เพื่อทำให้ข้อมูลมีความเรียบมากขึ้นโดยค่าเฉลี่ยเคลื่อนที่สามารถหาได้จากสมการที่ (2.9)

$$\bar{p}_{sm} = \frac{p_m + p_{m-1} + p_{m-2} + \cdots + p_{m(n-1)}}{n} \qquad (2.9)$$

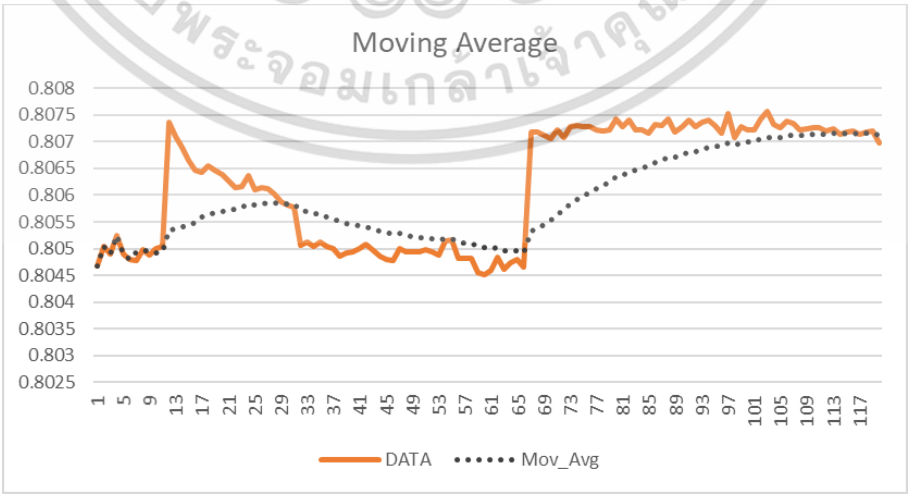
$$= \frac{1}{n} \sum_{i=0}^{n-1} p_{m-i}$$

โดย $\bar{p}_{sm}$ คือ ค่าที่ได้จากการคำนวณค่าเฉลี่ยเคลื่อนที่

p_m คือ ค่าที่ได้ในปัจจุบัน

n คือ ข้อมูลจำนวนก่อนหน้า

ซึ่งเมื่อนำค่าข้อมูล (Data) เส้นสีส้มมาผ่านการกรองข้อมูลในสมการที่ (2.9) สามารถกรองสัญญาณโดยใช้วิธีการหาค่าเฉลี่ยเคลื่อนที่ได้ ผลดังเส้นไข่ปลาดังรูปที่ 2.19



รูปที่ 2.19 ผลของการกรองข้อมูลด้วยเส้นค่าเฉลี่ยเคลื่อนที่แบบ 1 มิติ

2) การกรองแบบเกาส์เซียน (Gaussian)

เป็นการกรองโดยอาศัย Spatial convolution [21]-[23] ในลักษณะรูปสมมาตรเชิงวงกลม โดยค่าน้ำหนักจะมีค่าแปรผันตามการกระจายตัวแบบเกาส์เซียน (Gaussian distribution) ดังสมการที่ (2.10)

$$f(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad (2.10)$$

โดยที่ x แทนขนาดของสัญญาณรบกวน

σ แทนส่วนเบี่ยงเบนมาตรฐาน (Standard Deviation: S.D.) และ

μ แทนค่าเฉลี่ยกลาง (Mean)

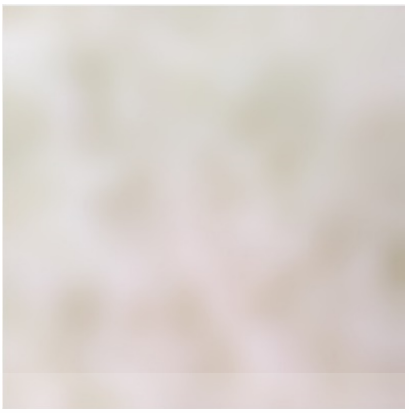
ซึ่งค่า σ เป็นตัวกำหนดความกว้างของฟังก์ชันความหนาแน่นความน่าจะเป็น (Probability density function: pdf) ที่อยู่ในรูปของเกาส์เซียน และค่า σ ยังเป็นค่าส่วนเบี่ยงเบนมาตรฐานของความกว้างจากการขยายแตกออก รวมถึงผลลัพธ์ของภาพที่เรียบขึ้นและถ้าค่า σ มีค่ามาก ส่งผลให้ช่วงกว้างของตัวกรองเกาส์เซียนจะยิ่งกว้างขึ้นตามไปด้วย ส่วนค่า μ เป็นตัวกำหนดตำแหน่งของฟังก์ชันความหนาแน่นความน่าจะเป็น โดยลักษณะของหน้าต่างเคลื่อนที่ขนาด 5x5 ดังรูปที่ 2.20

3/25	10/25	14/25	14/25	14/25
6/25	14/25	13/25	2/25	15/25
8/25	7/25	8/25	14/25	5/25
5/25	5/25	9/25	6/25	5/25
5/25	7/25	8/25	5/25	5/25

รูปที่ 2.20 หน้าต่างเคลื่อนที่ขนาด 5x5

การกรองแบบเกาส์เซียน สามารถกำจัดสัญญาณรบกวนแบบเกาส์เซียนได้ดี ซึ่งจุดภาพที่อยู่ตรงจุดกึ่งกลางจะมีค่าน้ำหนักจากตัวกรองมากที่สุด และสามารถส่งผลทำให้เกิดภาพเบลอได้เช่นเดียวกันกับการกรองแบบค่าเฉลี่ยเคลื่อนที่ ผลของการกรองภาพด้วยวิธีการกรองแบบเกาส์เซียน ดังรูปที่ 2.21

(ก) ภาพต้นฉบับ



(ข) ภาพที่ผ่านกระบวนการแบบเกาส์เซียน

รูปที่ 2.21 ผลที่ได้จากวิธีการเกาส์เซียน

2.6 การหาวัตถุที่อยู่ในภาพ

การหาวัตถุในภาพนั้นเป็นการหาสิ่งที่เราสนใจในภาพ มีขั้นตอนดังต่อไปนี้

2.6.1 วิธีการลบภาพ

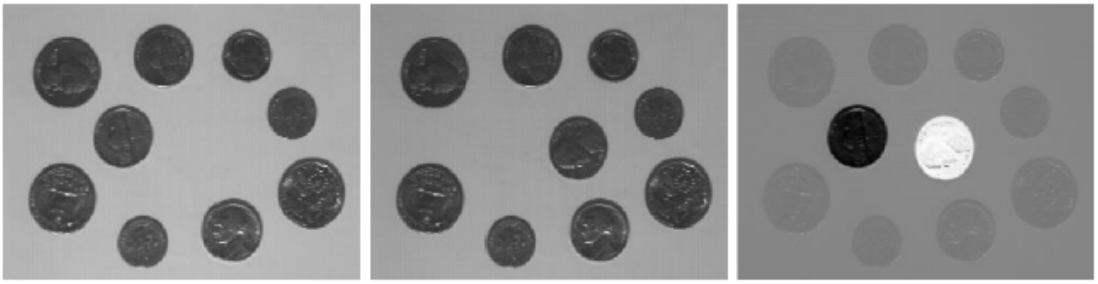
วิธีการลบภาพ (Image subtraction) เป็นวิธีการขจัดพื้นหลังของภาพออก ซึ่งดำเนินการบนภาพระดับเทา (Gray image) จำนวน 2 ภาพที่มีขนาดของภาพ (Resolution) และชั้นของสี (Class) ที่เท่ากัน จึงสามารถนำมาลบ (Subtraction) กันได้ โดยผลลัพธ์ที่ได้ถูกเก็บไว้ในอาเรย์ (Array) ที่มีขนาดเท่ากับขนาดของภาพ ซึ่งเมื่อกำหนดให้ $f_1(M,N)$ และ $f_2(M,N)$ แทนภาพระดับเทาที่มีขนาดเท่ากัน จากนั้นสามารถดำเนินการลบภาพโดยใช้สมการการลบภาพที่ (2.11) ซึ่งผลลัพธ์ที่ได้ดังรูปที่ 2.22

$$f_r(M,N) = f_1(M,N) - f_2(M,N) \tag{2.11}$$

เมื่อ $f_r(M,N)$ คือ ภาพของผลลัพธ์

$f_1(M,N)$ คือ ภาพต้นฉบับ

$f_2(M,N)$ คือ ภาพพื้นหลัง



(ก) ภาพต้นฉบับ A (ข) ภาพต้นฉบับ B (ค) ผลลัพธ์ของวิธีการลบภาพ

รูปที่ 2.22 วิธีการลบภาพ

2.6.2 การหาขอบของภาพ

การหาขอบของภาพ (Edge detection) [23] เป็นการหาเส้นรอบวัตถุ ซึ่งเมื่อหาเส้นรอบวัตถุนั้นได้ทำให้สามารถคำนวณพื้นที่ของวัตถุนั้นตามมาด้วยเช่นกัน การหาขอบของวัตถุที่สมบูรณ์นั้นเป็นเรื่องยากและมีปัจจัยที่สำคัญอยู่ที่ความละเอียดของภาพ ยิ่งความละเอียดภาพมีคุณภาพต่ำจะส่งผลทำให้มีความแตกต่างระหว่างความเข้มแสงของจุดภาพระหว่างพื้นหน้าและพื้นหลังน้อย หรือความสว่างที่ไม่สม่ำเสมอกันของภาพ ทำให้เกิดขอบภาพที่ไม่พึงประสงค์ เนื่องจากเกิดความแตกต่างของความเข้มแสงจากจุดหนึ่งไปยังอีกจุดหนึ่ง ซึ่งถ้าความเข้มแสงต่างกันมากขอบภาพนั้นจะมีความเด่นชัดมาก และถ้าหากความเข้มแสงต่างกันน้อยขอบภาพนั้นจะไม่ชัดเจน

1) การหาขอบภาพด้วยวิธีโซเบล (Sobel Edge Detection)

การหาขอบภาพโดยวิธีโซเบล [24,25] เป็นการหาขอบภาพโดยใช้หน้าต่างเคลื่อนที่ขนาด 3x3 จำนวนสองหน้าต่างเคลื่อนที่ โดยให้หน้าต่างเคลื่อนที่อันแรกในการค้นหาความแตกต่างของความเข้มแสงในแนวนอน (E_x) และให้หน้าต่างเคลื่อนที่อันที่สองในการหาความแตกต่างของความเข้มแสงในแนวตั้ง (E_y) โดยหน้าต่างเคลื่อนที่ทั้งสองแบบดังรูปที่ 2.23

$$E_x = \begin{matrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{matrix} \quad , \quad E_y = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

รูปที่ 2.23 หน้าต่างเคลื่อนที่ในการหาค่าเฉลี่ย

โดยเมื่อนำไปใช้กับภาพต้นฉบับในการค้นหาความแตกต่างของความเข้มของแสงในแต่ละแนว (E_x และ E_y) ด้วยการไล่ระดับในทุกจุดภาพและทิศทาง โดยค่าความสัมบูรณ์ในความแตกต่างของความเข้มของแสงสามารถหาได้จากผลรวมของ E_x และ E_y เข้าด้วยกัน ดังสมการที่ (2.12) และผลที่ได้จากการดำเนินการแสดงดังรูปที่ 2.24

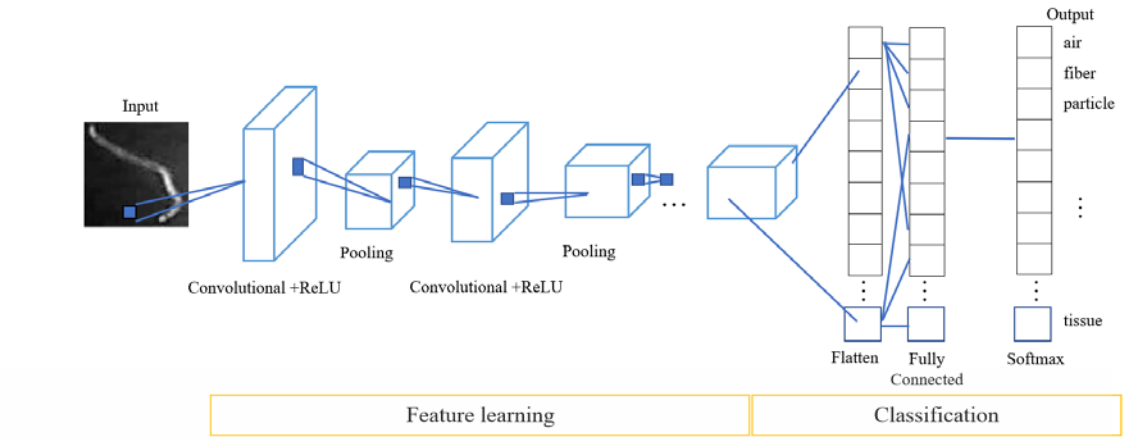
$$|E| = \sqrt{{E_x}^2 + {E_y}^2} \tag{2.12}$$

(ก) ภาพต้นฉบับ (ข) ขอบภาพด้วยวิธีโซเบล

รูปที่ 2.24 การหาขอบภาพด้วยวิธีโซเบล

2.7 โครงข่ายประสาทเทียม

การเรียนรู้เชิงลึก (Deep learning) คือการเรียนรู้ของเครื่อง (Machine learning) ซึ่งเป็นการฝึกฝนให้คอมพิวเตอร์นั้นสามารถทำงานได้เหมือนมนุษย์ โดยวิธีการเรียนรู้เชิงลึกจะกำหนดค่าพารามิเตอร์พื้นฐานเกี่ยวกับข้อมูลและฝึกให้คอมพิวเตอร์เรียนรู้ด้วยตัวเองโดยการจดจำรูปแบบจากการใช้การประมวลผลหลายชั้น ซึ่งมีกระบวนการในการคำนวณคล้ายกับระบบโครงข่ายประสาท (Neurons) ของมนุษย์ หรือเรียกว่า โครงข่ายประสาทเทียม (Artificial neural networks: ANN) [26,27] จากนั้นต่อมาได้มีการเพิ่มประสิทธิภาพการทำงานด้วยการใช้โครงข่ายประสาทแบบคอนโวลูชัน (Convolutional neural network: CNN) [28] ซึ่งเป็นโครงข่ายประสาทเทียมระดับลึก โดยมักถูกนำมาใช้ในการวิเคราะห์ภาพ จากหลักการทำงานพื้นฐานของ CNN นั้นมีการทำงานอยู่ 3 ส่วนคือ ส่วนแรกคือส่วนรับเข้า (Input) เป็นการรับข้อมูลหรือภาพวัตถุเข้ามาเปรียบเทียบได้กับการมองเห็นของมนุษย์ ส่วนที่สองคือส่วนของ Hidden layer [29]-[31] เป็นส่วนสำหรับการประมวลผลข้อมูลที่ถูกป้อนเข้ามา หรือเปรียบเทียบได้กับการทำงานของโครงข่ายประสาทสมองของมนุษย์ที่สามารถเรียนรู้ในการคัดแยกสิ่งต่าง ๆ ได้ และส่วนสุดท้ายส่วนที่สามคือส่วน Output เป็นส่วนแสดงผลลัพธ์ที่ได้จากการประมวลผลในส่วนที่สอง โครงสร้างทั่วไปของ CNN แสดงดังรูปที่ 2.25



รูปที่ 2.25 สถาปัตยกรรมของโครงข่ายประสาทแบบคอนโวลูชัน

โดยจากรูปที่ 2.25 อธิบายโครงสร้างโดยทั่วไปของ CNN ซึ่งประกอบด้วยส่วนต่าง ๆ ดังนี้

1. Input Layer: ทำหน้าที่อ่านข้อมูลรูปภาพที่เป็น Input เข้ามาในโครงข่ายประสาทเทียม

2. Convolutional Layer: ทำหน้าที่กรองคุณลักษณะของภาพให้ได้ Features จากการดำเนินวิเคราะห์ภาพแต่ละจุดภาพ (Pixel) ของภาพที่ได้อ่านเข้ามา ซึ่งผลลัพธ์ที่ได้คือ Convolution Feature Map

3. Rectified Linear Unit (ReLU): ทำหน้าที่ส่งต่อ Linear function ออกไป โดยเลือกปิด Neuron บางตัว ถ้า Neuron นั้นมีค่าน้อย

4. Pooling Layer: ทำหน้าที่ สกัดเอาส่วนที่สำคัญที่สุดของข้อมูลและเพิ่มประสิทธิภาพการประมวลผลให้รวดเร็วขึ้น

5. SoftMax Layer: ทำหน้าที่ในกำหนดค่า Output ให้แสดงผลลัพธ์ออกมาในรูปแบบ Multiclass Logistic Classifier

6. Output Layer: ทำหน้าที่แสดงผลลัพธ์ที่ได้ของการคัดแยก (Classification)

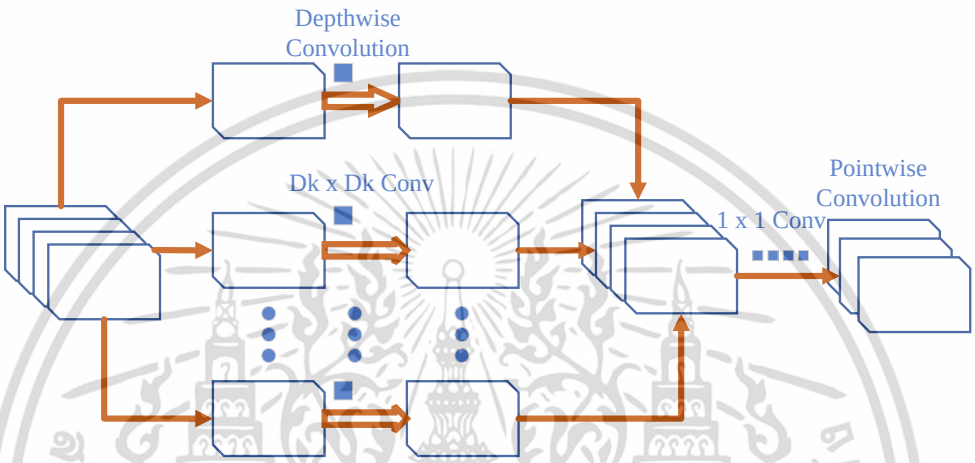
ซึ่งเมื่อนำมาสร้างเป็นลำดับชั้นสำหรับการฝึกฝนแบบจำลอง ได้ผลดังรูปที่ 2.26

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 112, 112, 32)	0
conv2d_2 (Conv2D)	(None, 112, 112, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 56, 56, 64)	0
conv2d_3 (Conv2D)	(None, 56, 56, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 28, 28, 128)	0
flatten_1 (Flatten)	(None, 100352)	0
dropout_1 (Dropout)	(None, 100352)	0
dense_1 (Dense)	(None, 256)	25690368
dense_2 (Dense)	(None, 10)	2570

รูปที่ 2.26 ลำดับชั้นมาตรฐานของ CNN

2.7.1 MobileNet

สถาปัตยกรรม MobileNet [32,33] เป็นโครงข่ายประสาทเทียมแบบ CNN ที่มีวัตถุประสงค์เพื่อลดขนาด (Size) และความซับซ้อนของแบบจำลอง CNN ลง ซึ่งมีประโยชน์สำหรับการใช้งานผ่านอุปกรณ์สมาร์ตโฟน (Smartphone) โดยประสิทธิภาพของโมเดลไม่ได้ลดลง ซึ่งค่าประสิทธิภาพการทำงานได้ใกล้เคียงกับโครงข่ายประสาทเทียมแบบการเรียนรู้เชิงลึก การลดขนาดของแบบจำลองสามารถทำได้ดังรูปที่ 2.27



รูปที่ 2.27 การบิดแบบแยกส่วนเชิงลึก

ในการทำให้ MobileNet มีขนาดของโมเดลที่เล็กลง สามารถทำได้ด้วยการทำการบิดแบบแยกส่วนเชิงลึก (Depthwise separable convolution) ซึ่งมีขั้นตอนดังนี้

1) Depthwise convolution คือ channel-wise $D_k \times D_k$ spatial convolution โดยกำหนดช่องขึ้นมา 5 ช่อง ทำให้มี 5 $D_k \times D_k$ spatial convolution

2) Pointwise convolution คือการเปลี่ยนมิติข้อมูลเป็นขนาด 1×1 convolution

จากการดำเนินการเบื้องต้นในการคำนวณค่า Convolution เชิงลึก (Deep) แบบแยกส่วนได้ดังสมการที่ (2.13)

$$D_k \times D_k \times m \times D_f \times D_f + m \times n \times D_f \times D_f \qquad (2.13)$$

โดยที่

m แทนจำนวนช่องสัญญาณเข้า

n แทนจำนวนช่องสัญญาณออก

D_k แทนขนาดเคอร์เนล

D_f แทนขนาดแผนที่คุณลักษณะ (Feature)

และสำหรับการหาค่า Standard Convolution สามารถหาค่าได้ดังสมการที่ (2.14)

$$D_k \times D_k \times m \times D_f \times D_f \tag{2.14}$$

ดังนั้นสามารถคำนวณและลดรูปของ model ได้จากการคำนวณตามสมการที่ (2.15)

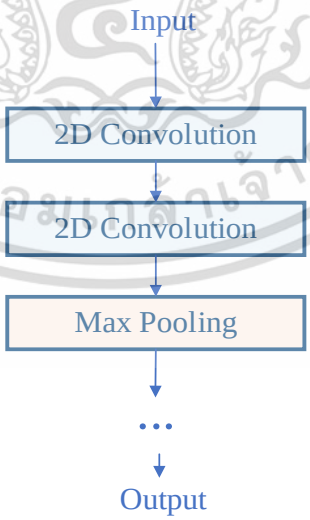
$$\frac{D_k \times D_k \times m \times D_f \times D_f + m \times n \times D_f \times D_f}{D_k \times D_k \times m \times D_f \times D_f} \tag{2.15}$$

$$= \frac{1}{n} + \frac{1}{D_k^2}$$

เมื่อ $D_k \times D_k$ มีขนาดเท่ากับ 3×3 ทำให้สามารถคำนวณได้น้อยลง 8 ถึง 9 เท่า และทำให้ค่าความแม่นยำลดลงเพียงเล็กน้อย

2.7.2 VGGNet

VGGNet คิดค้นโดย VGG (Visual Geometry Group) จาก University of Oxford โดยในปี ค.ศ. 2014 Simonyan และ Zisserman [29,30,34] เสนอสถาปัตยกรรมเครือข่าย VGG ขึ้น โดยแบบจำลอง VGG นั้น มีการใช้ชั้น (Layer) แบบ 2D Convolutional ขนาด 3×3 ซึ่งเป็นการเพิ่มขั้นตอนด้านบนของแบบจำลอง CNN มาตรฐาน (Max pooling, Fully-connected และ SoftMax layer) โดยโครงสร้างของชั้นที่เพิ่มเข้าไปดังรูปที่ 2.28

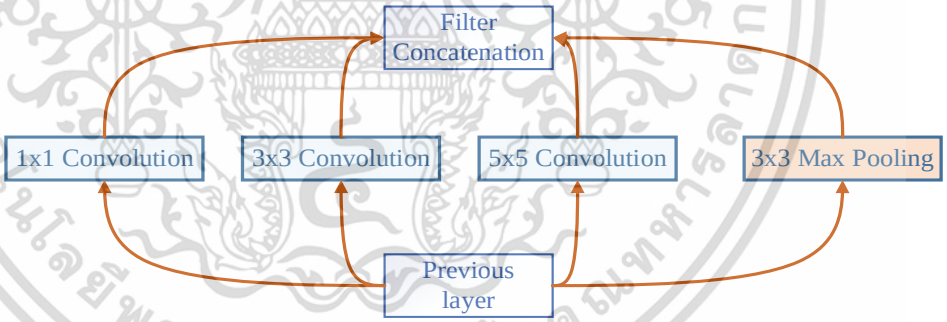


รูปที่ 2.28 สถาปัตยกรรม VGGNet

สำหรับ Convolutional neural network (CNN) เมื่อจำนวนของชั้น layer ใน Network เพิ่มขึ้น ความสามารถของแบบจำลองที่ได้จะดีขึ้นและมีความซับซ้อนขึ้น โดยจากการศึกษาผลการทำงานของ VGG11, VGG11 (Local Response Normalization: LRN), VGG13, VGG16 (Conv 1), VGG16 และ VGG19 พบว่า VGG16 และ VGG19 มีค่าความผิดพลาดต่ำที่สุด โดยความหมายของเลข 16 และเลข 19 นั้นหมายถึง จำนวนของชั้น layer โดยค่าความผิดพลาดของ VGG19 ที่มี 19 layer มีค่าความผิดพลาดที่มากกว่า VGG16 ที่มี 16 layer ซึ่งหมายถึง จำนวนชั้นของ layer ไม่สามารถลดค่าความผิดพลาดลงได้ อีกทั้ง VGG19 ยังมีขนาดของแบบจำลองที่มากกว่า VGG16

2.7.3 GoogLeNet

GoogLeNet เป็นโครงข่ายประสาทเทียมแบบ deep convolution ความลึกขนาด 22 ชั้น ซึ่งเป็นรูปแบบหนึ่งของ Inception network ซึ่งเป็น Deep convolutional neural network ที่พัฒนาโดยนักวิจัยของ Google โดยเริ่มต้นขึ้นในปี ค.ศ. 2014 ที่ Inception V1 ได้ถูกพัฒนาขึ้นโดย Christian Szegedy [35] และทีม เพื่อเพิ่มความสามารถในการตรวจสอบและจำแนกวัตถุแต่ละชนิดให้ดีขึ้น ประกอบด้วย ขนาดของการคอนโวลูชันที่แตกต่างกัน (ขนาด 1×1, ขนาด 3×3 และ ขนาด 5×5) และ ขนาด 3×3 Max-pooling node ดังรูปที่ 2.29



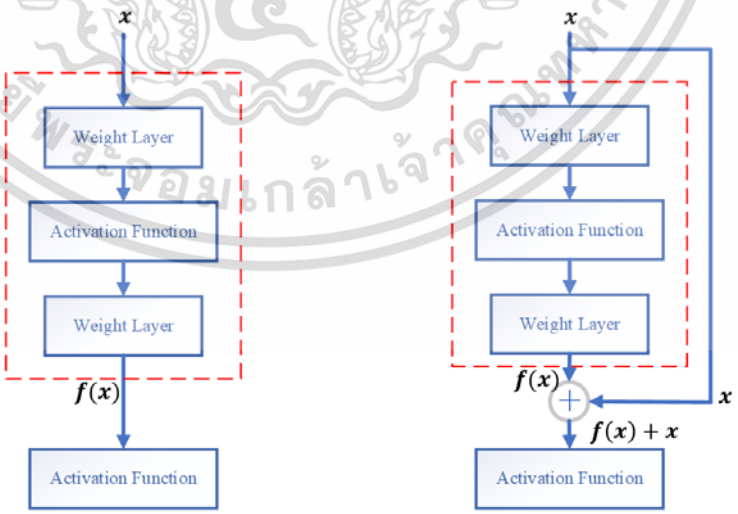
รูปที่ 2.29 โครงสร้างเริ่มต้นของโมเดล Inception

ซึ่งต่อมาได้มีการปรับปรุงทำให้มีประสิทธิภาพให้ดีขึ้น โดยจากการทดสอบ Large Scale Visual Recognition Challenge 2014 (ILSVRC14) พบว่ามีค่า Error จากการทดสอบน้อยที่สุด โดยวิธีการหนึ่งที่ GoogLeNet ได้รับประสิทธิภาพคือการลดขนาดของภาพนำเข้า ในขณะเดียวกันยังรักษาข้อมูลเชิงพื้นที่ ที่สำคัญไปพร้อม ๆ กันได้ และต่อมาหลังจากประสบความสำเร็จในงานแข่งขันเมื่อปี ค.ศ. 2014 GoogLeNet ได้ทำการปรับปรุงกระบวนการหาค่าประสิทธิภาพของการฝึกอบรมหรือฝึกฝน (Training) ด้วยการทำ Batch normalization technique หรือเรียกว่า Inception V2 หรือ BN-Inception ReLU ที่มีการปรับในเรื่องการอิ่มตัวของค่าสีและการไล่เฉดสีที่

หายไป ซึ่งส่งผลทำให้การกระจายของค่า X ที่เริ่มคงที่หรือที่มีการเปลี่ยนแปลงเล็กน้อยจะถูกขยายออกเมื่อลงไปในชั้นที่ลึกขึ้นของเครือข่าย ซึ่งทำให้อัตราการเรียนรู้สูงขึ้น และใน Inception V2 ได้นำ 3x3 convolution มาแทนที่ 5x5 convolution เพื่อเป็นการลดจำนวนพารามิเตอร์ และต่อมาปรับปรุงเพิ่มขึ้นเป็น Inception V3 ที่มีการแยกตัวประกอบของ node convolutional ในโมดูล Inception และทำให้มีขนาดเล็กลง และปัจจุบันคือ Inception V4 ที่ปรับปรุงจาก Inception V3 โดยทำการเพิ่มโมดูลการเริ่มต้นที่มากกว่า Inception V3 ซึ่งจากการทดสอบประสิทธิภาพแสดงให้เห็นว่า Inception V4 มีอัตราความผิดพลาดน้อยที่สุด และใช้เวลาสำหรับการ Training model มากที่สุด เมื่อเทียบกับรุ่นก่อนหน้าทั้ง 3 รุ่น

2.7.4 ResNet

สถาปัตยกรรม ResNet หรือ Residual neural network ถูกนำเสนอครั้งแรกโดย Kaiming He และคณะ [36] โดยมีโครงสร้างที่ประกอบด้วย Residual block ที่นำมาต่อกัน ซึ่งสามารถให้ประสิทธิภาพที่ดี แต่จะมีความลึกของชั้นที่เพิ่มขึ้นตามมา โดยโครงสร้างที่ของ ResNet นั้นมีชั้นความลึกมาตรฐานอยู่ที่ 18 ชั้น 50 ชั้น 101 ชั้น และ 152 ชั้นตามลำดับ สำหรับ Residual block นั้นเป็นส่วนประกอบพื้นฐาน โดยภายในของ Residual block นั้นมีชั้นที่มีพารามิเตอร์สำหรับการฝึกทั้งหมดเอาไว้ จากนั้นใช้วิธีการ Skip connection ซึ่งเป็นการเชื่อมเส้นข้อมูลข้ามชั้นคอนโวลูชันในบางชั้น เนื่องจากการทำคอนโวลูชันในบางชั้นไม่มีค่าถ่วงน้ำหนักอยู่ ซึ่งสามารถกำหนดค่าเป็น 0 ได้ โดยที่เมื่อข้อมูลเข้ามาจะถูกส่งผ่านไปยังชั้นถัดไปโดยไม่มีการสูญหาย สามารถเปรียบเทียบโครงสร้างแบบปกติที่ไม่มี Skip connection ดังรูปที่ 2.30 (ก) และโครงสร้างของ Residual block ดังรูปที่ 2.30 (ข)

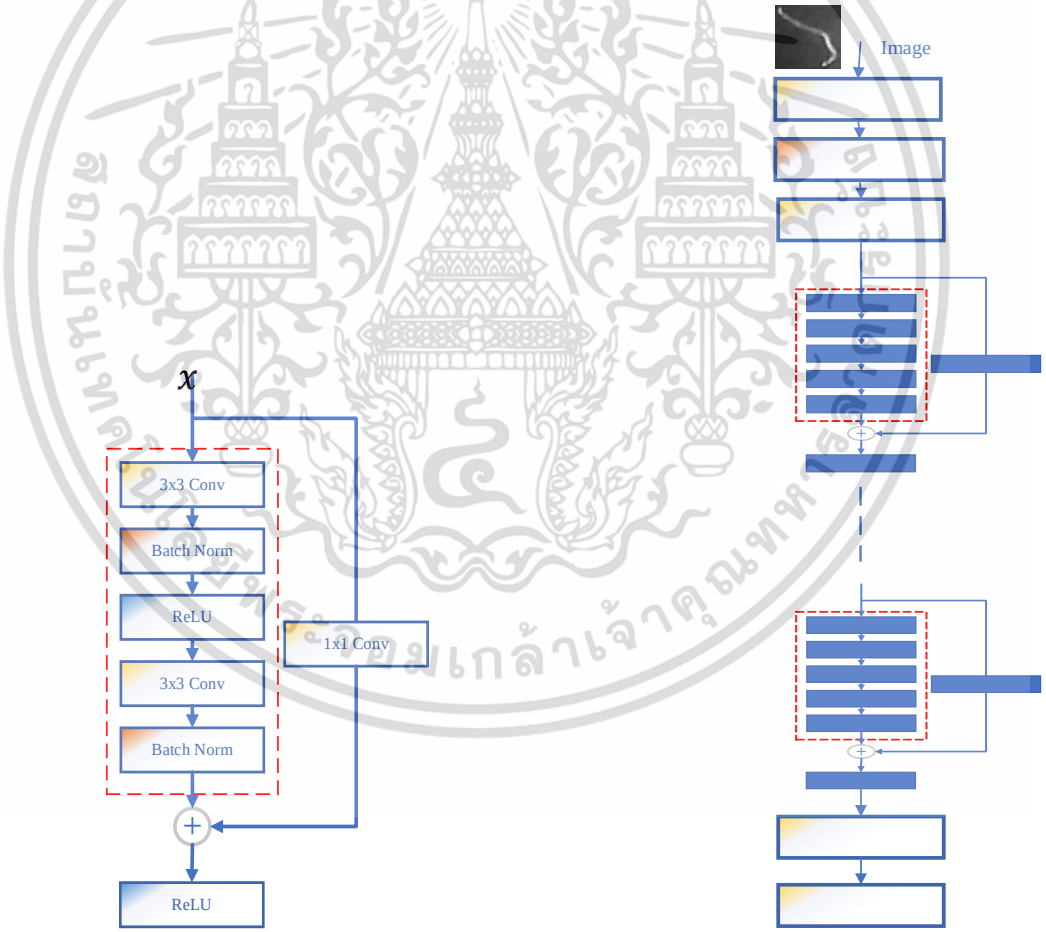


(ก) ชั้นโครงสร้างมาตรฐานของ ResNet (ข) โครงสร้างของ Residual block

รูปที่ 2.30 โครงสร้างสถาปัตยกรรมของโมเดล Inception

เมื่อ x คือข้อมูลนำเข้า โดยข้อมูลนำเข้ามาจะผ่านการถ่วงน้ำหนักด้วย $f(x)$ จากนั้นส่งค่าต่อไปยัง Activation function แต่สำหรับ Residual block นั้น เมื่อมีข้อมูลนำเข้ามาจะผ่านการถ่วงน้ำหนัก จากนั้นนำมารวมกับข้อมูลนำเข้าอีกรอบด้วย $f(x)+x$ ก่อนส่งไปยัง Activation function อีกครั้ง

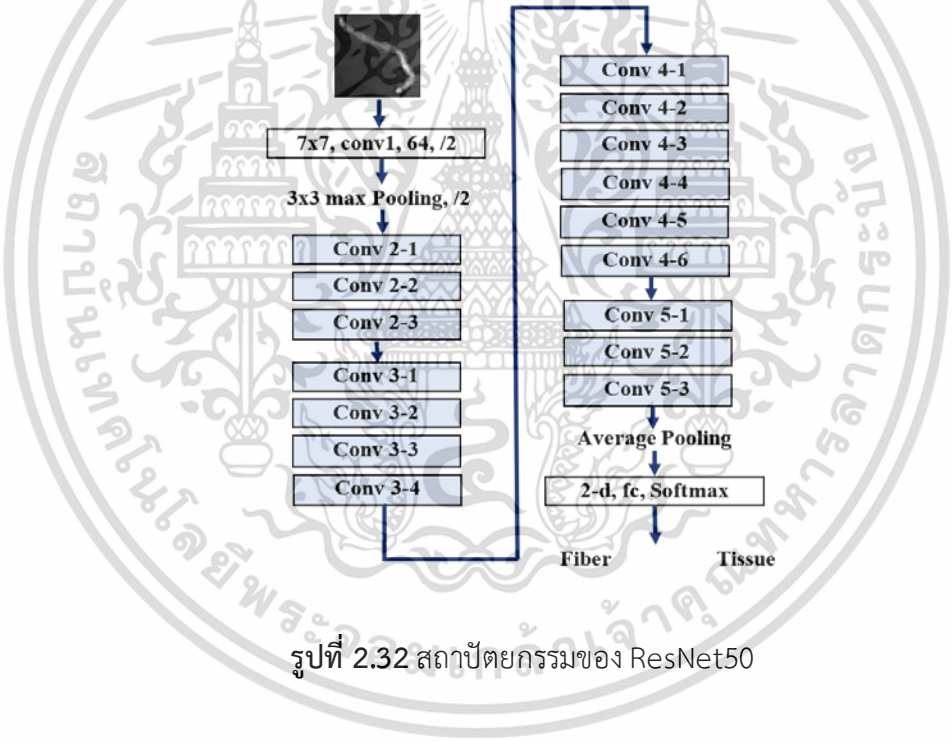
สำหรับ ResNet จะมีโครงสร้างที่ประกอบขึ้นจาก Residual block ที่นำมาต่อกัน โดยภายในโครงสร้าง Residual block ดังรูปที่ 2.31(ก) ซึ่งประกอบด้วย Convolution layer ขนาด 3x3, Batch normalization และ ReLU ซึ่งเมื่อนำมาต่อรวมกันตามสถาปัตยกรรม ResNet โดยชั้นแรกของโครงข่ายใช้ filter ขนาด 7x7 ที่รับข้อมูลภาพเข้า และให้ Output ขนาด 64 และกำหนดให้ stride เป็น 2 ตามด้วยชั้น Batch normalization และการทำ Max pooling ในชั้นถัดไป ประกอบไปด้วยชั้นของ Residual blocks และส่วนสุดท้ายของโครงข่ายใช้ Average pooling ก่อนเข้าสู่การจำแนก (Classification) ต่อไป โดยโครงสร้างสถาปัตยกรรม Resnet ดังรูปที่ 2.31(ข)



(ก) ส่วนประกอบภายในของ Residual block (ข) สถาปัตยกรรมของ ResNet

รูปที่ 2.31 โครงสร้างสถาปัตยกรรมของ ResNet

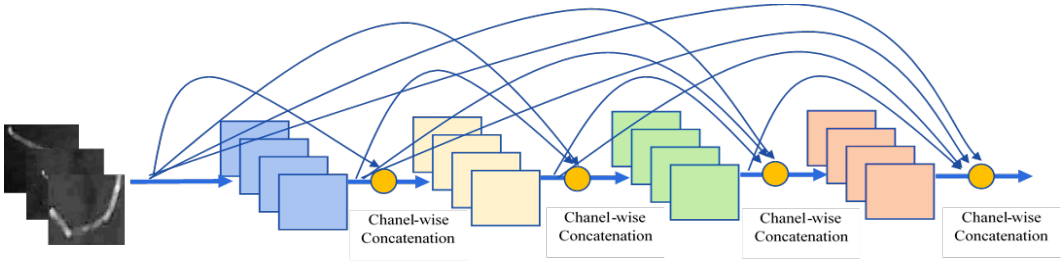
ResNet ได้ถูกนำเสนอเพื่อแก้ปัญหาการสูญหายของเกรเดียน (Vanishing gradient problem: VGP) ที่ค่าของผลลัพธ์เกรเดียนน้อยลงไปเรื่อย ๆ จนมีค่ากลายเป็น 0 ซึ่งเป็นผลจากการกำหนดชั้นเลเยอร์ (Network layer) ให้มีจำนวนที่ลึกมาก ด้วยวิธีการทำงานของ RestNet ที่มีการใช้งาน Residual block เป็นส่วนประกอบพื้นฐาน โดยภายในของ Residual block จะมีชั้นที่มีพารามิเตอร์สำหรับการฝึกทั้งหมดเอาไว้ ซึ่งจำนวนของ Residual block กลายเป็นจำนวนชั้นของ Network แทน และเนื่องจากโครงสร้างของ RestNet นั้นมีชั้นของ Network เป็นจำนวนมาก จึงมีเส้นเชื่อมโยงระหว่างชั้นเพิ่มขึ้น โดยในทุก ๆ 2 ชั้นของการคอนโวลูชันจะมีเส้นข้ามชั้นเชื่อมต่ออยู่เพื่อช่วยลดความผิดพลาดของการฝึกฝน สำหรับการฝึกฝนแบบจำลองของ ResNet จำนวนของพารามิเตอร์คือชั้นที่ใช้สำหรับการเรียกชื่อ เช่น ResNet50 หรือ ResNet101 ซึ่งโครงสร้างของ ResNet50 จะมีขนาด [3,4,6,3] ซึ่งหมายถึง (3+4+6+3) x 3 = 48 layer + 2 layer หรือหมายถึง 50 layer แสดงดังรูปที่ 2.32



รูปที่ 2.32 สถาปัตยกรรมของ ResNet50

2.7.5 DenseNet

สถาปัตยกรรม ResNet [37] เป็นโครงสร้างที่ในแต่ละชั้นจะได้รับข้อมูล Input เพิ่มเติมจากชั้นก่อนหน้าทั้งหมด และส่งค่าคุณลักษณะ (Feature) ที่ได้จากชั้นของตัวเอง ไปยังชั้นอื่นที่ตามมาทั้งหมด ซึ่งการเชื่อมต่อแบบนี้ทำให้ในแต่ละชั้นจะได้รับข้อมูลร่วมกันหรือเป็นความรู้ร่วมจากทุกชั้นก่อนหน้า โดยที่ DenseNet ถูกพัฒนาขึ้นมาเพื่อช่วยลดปัญหา Vanishing gradient problem ช่วยเพิ่มคุณลักษณะ Strengthen feature propagation และช่วยปรับปรุงประสิทธิภาพของพารามิเตอร์ให้ดีขึ้น ดังรูปที่ 2.33



รูปที่ 2.33 การเชื่อมต่อแต่ละชั้นของ DenseNet

DenseNet หรือ Dense Convolutional Network ได้ถูกนำเสนอในปี 2017 โดย Gao Huang และคณะ [38] จากความร่วมมือในการคิดค้นโดย Cornwell University และ Tsinghua University อ้างอิงด้วยการเชื่อมต่อกันระหว่างชั้นด้วย Feed-forward fashion

2.8 งานวิจัยที่เกี่ยวข้อง

ในปี ค.ศ. 2017 MobileNet ได้ถูกนำเสนอโดย Andrew G. Howard [32] มีวัตถุประสงค์เพื่อลดขนาดของแบบจำลอง CNN มาตรฐานให้เหมาะสมกับการใช้งานในอุปกรณ์พกพา โดยยังสามารถรักษาประสิทธิภาพการทำงานได้ใกล้เคียงกับโครงข่ายประสาทเทียมแบบการเรียนรู้เชิงลึก

ในปี ค.ศ. 2018 Yiting Tao และคณะ [39] ได้นำวิธีการ DenseNet ไปใช้ในการแก้ไขปัญหาการจัดกลุ่มคำตอบที่มีความเลื่อมล้ำของข้อมูลที่อยู่บนภาพถ่ายดาวเทียม ด้วยวิธีการ Depth-width-reinforced deep neural network (DNN) ด้วยการเพิ่มชั้นความลึกของเครือข่ายและขนาดความกว้าง (Width) ของการ Training model เพื่อให้ได้การจัดกลุ่มคำตอบที่ดีขึ้น

ปี ค.ศ. 2019 Edna Chebet Tooa และคณะ [40] ได้นำวิธีการ DenseNet มาเปรียบเทียบกับสถาปัตยกรรมอื่น ในการเปรียบเทียบประสิทธิภาพของแบบจำลองการจัดกลุ่มคำตอบโรคพืชจากภาพถ่ายของใบไม้ จากชุดข้อมูลภาพ PlantVillage ประกอบด้วย 54,306 ภาพ มี 26 โรคสำหรับพืช 14 ชนิด โดยผลการวิจัยพบว่า วิธีการ DenseNet นั้นให้ความแม่นยำกับชุดข้อมูล PlantVillage ในการทดสอบที่ 99.75% ซึ่งเมื่อเปรียบเทียบค่าความแม่นยำนั้นมีค่าสูงกว่า สถาปัตยกรรมอื่นที่ได้ทดลองด้วยกัน

และในปีเดียวกัน Tammina [41] ได้เสนอวิธีการแก้ปัญหาในการจำแนกรูปภาพโดยจำกัดจำนวนตัวอย่างของการฝึกแบบจำลอง VGG16 ด้วยจำนวนตัวอย่างที่น้อยที่สุด จากการทดลองแสดงให้เห็นถึงความแม่นยำ (Accuracy) และค่าความสูญเสีย (Loss) หลังจากมีการปรับแต่งค่าพารามิเตอร์ของ CNN โดยค่าความแม่นยำต่ำสุดเท่ากับ 79.20% จากนั้นมีการเพิ่มจำนวนรูปภาพให้กับการฝึกฝนแบบจำลองทำให้ค่าความแม่นยำเท่ากับ 95.40%

ต่อมาในปี ค.ศ. 2020 [33] ได้เสนอวิธีการ MobileNet รวมกับ Transfer learning algorithm สำหรับการวิเคราะห์ภาพและการจดจำเป้าหมาย ซึ่งนำมาแก้ไขปัญหาในเรื่องความแม่นยำ

ในการจำแนกรูปภาพใหม่ โดยใช้ชื่อโครงสร้างว่า TL-MobileNetโดยการเพิ่ม DropBlock และ Global average เข้าไปในชั้นของ MobileNet เดิม ผลจากการทดลองพบว่าโมเดลที่นำเสนอมีความแม่นยำในการทำนายแบบจำลองอยู่ที่ 97.69% ของชุดข้อมูลมาตรฐาน Modified National Institute of Standards and Technology (MNIST) dataset

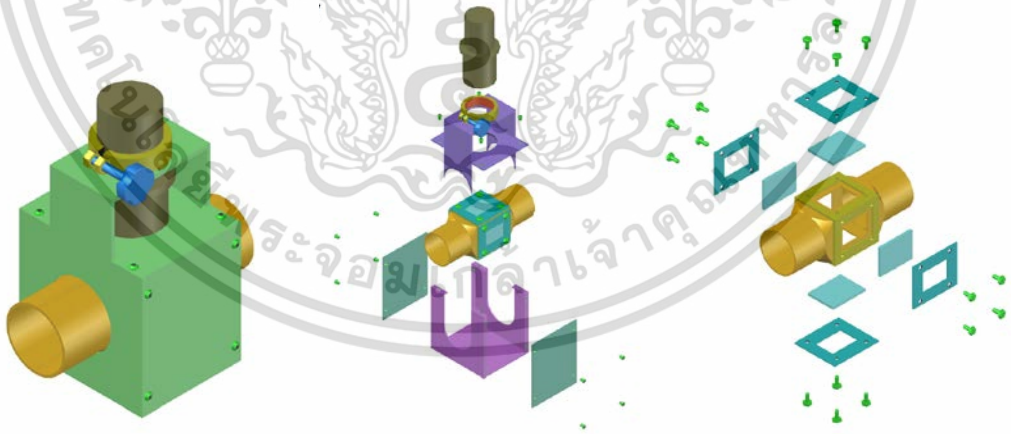
บทที่ 3

วิธีการดำเนินงานวิจัย

ในส่วนขั้นตอนการดำเนินงานวิจัยได้พัฒนาอุปกรณ์สำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนต้นแบบในน้ำมันมะพร้าว เพื่อให้สามารถใช้งานร่วมกับระบบการผลิตน้ำมันมะพร้าวของโรงงานได้ โดยออกแบบให้มีผลกระทบต่อการทำงานของโรงงานน้อยที่สุด การทำงานได้แบ่งการดำเนินงานเป็น 2 ส่วนหลัก ได้แก่ 1) ส่วนการพัฒนาอุปกรณ์ท่อสำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าว และ 2) ส่วนการพัฒนาขั้นตอนวิธีการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าว มีรายละเอียดดังนี้

3.1 ส่วนการพัฒนาอุปกรณ์ท่อสำหรับตรวจจับน้ำมัน

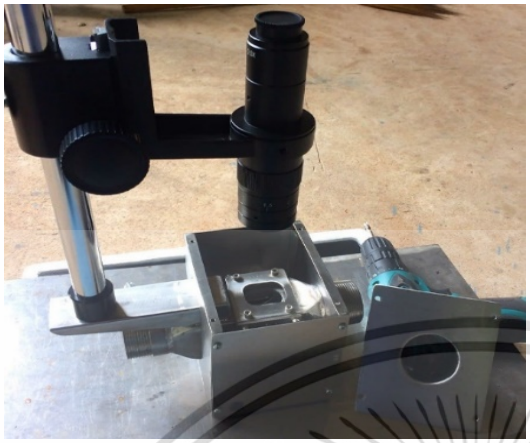
สำหรับอุปกรณ์ท่อสำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าวนั้นได้ออกแบบให้สามารถใช้งานร่วมกับท่อลำเลียงน้ำมันมะพร้าว หลังจากกระบวนการกรองน้ำมันมะพร้าวด้วยเครื่องหมุนเหวี่ยง (Centrifugation) ซึ่งจะนำน้ำมันมะพร้าวมาเก็บไว้ในถังพักเพื่อรอการบรรจุ โดยอุปกรณ์ท่อสำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าวนั้นมีการออกแบบดังรูปที่ 3.1



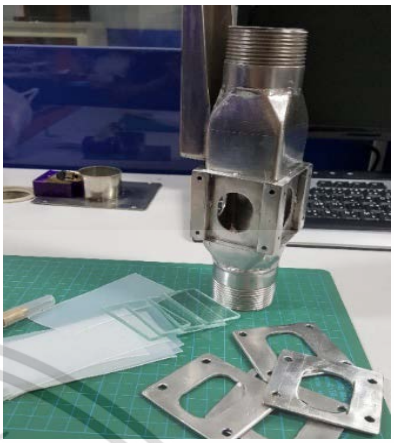
รูปที่ 3.1 ต้นแบบอุปกรณ์ท่อสำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าว

จากนั้นทำการเลือกวัสดุตามมาตรฐานการผลิต โดยเลือกใช้สแตนเลส เบอร์ 304 ที่ใช้สำหรับอุตสาหกรรมอาหาร (Food grade) และไม่เกิดสนิม เพื่อให้สอดรับกับระบบการทำงานภายใน

โรงงานผลิตน้ำมันมะพร้าวที่มีการส่งผ่านน้ำมันมะพร้าวผ่านท่อสแตนเลส ผลของการดำเนินการแสดงดังรูปที่ 3.2



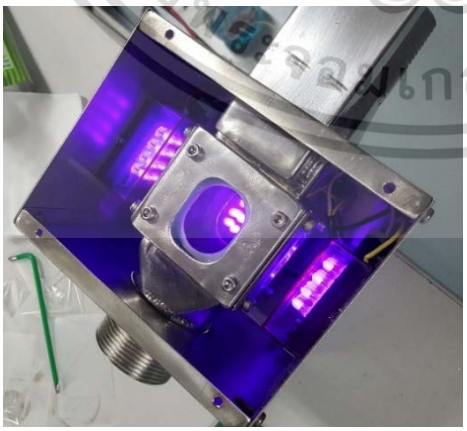
(ก) โครงสร้างและแท่นยึดกล้อง



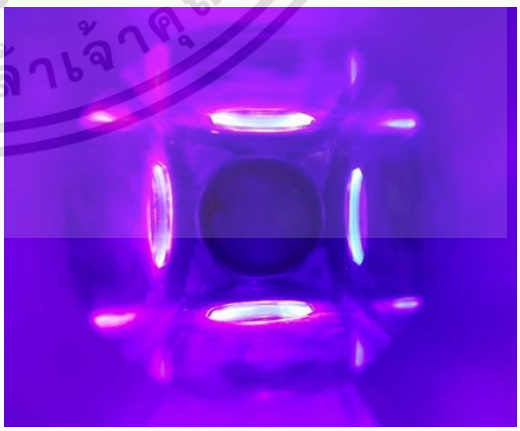
(ข) กระจกและซิลิโคนสำหรับอุตสาหกรรมอาหาร

รูปที่ 3.2 อุปกรณ์ท่อสำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าว

จากรูปที่ 3.2 เมื่อนำอุปกรณ์ท่อสำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าวมาทำการติดกระจกและแผ่นซิลิโคนสำหรับอุตสาหกรรมอาหาร (Food grade) โดยไม่ทำให้มีการปนเปื้อนของสารเคมีอื่นๆ เป็นการป้องกันการปนเปื้อนสารต่าง ๆ ที่เกิดจากซิลิโคนแบบปกติ จากนั้นทำการสร้างระบบส่องสว่างภายในเพื่อให้แสงสามารถลอดผ่านน้ำมันมะพร้าวมายังเลนส์ของกล้องไมโครสโคป เพื่อสำหรับการใช้งานกล้องให้สามารถตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าวได้ จากการทดลองพบว่าแสงอัลตราไวโอเลต (Ultraviolet) นั้นมีความเหมาะสม สามารถแสดงได้ดังรูปที่ 3.3



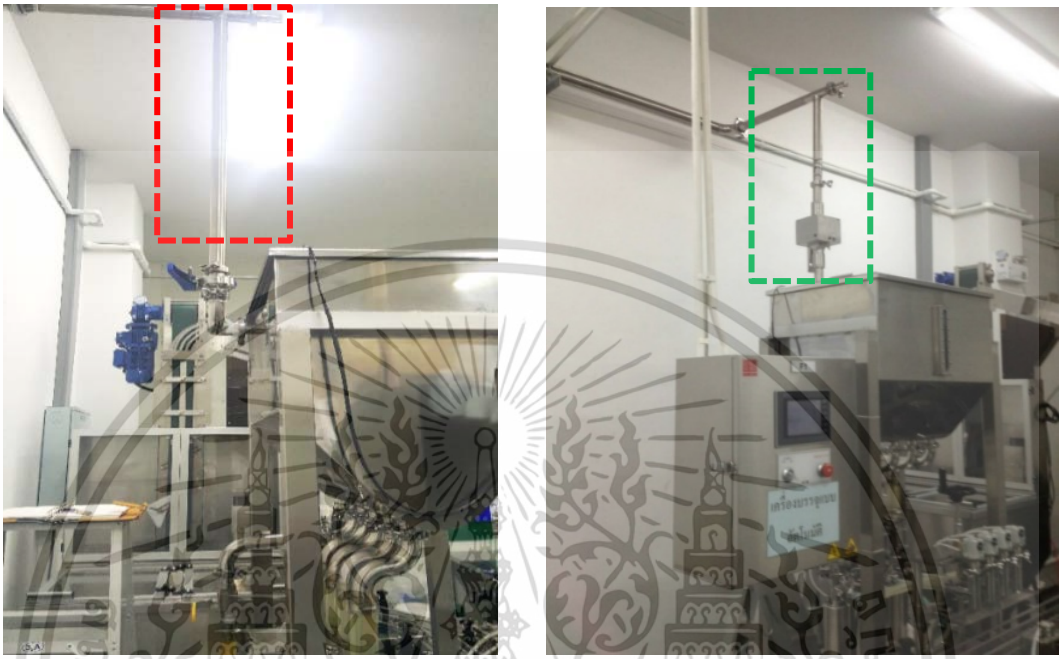
(ก) ภายนอกท่อ



(ข) ภายในท่อ

รูปที่ 3.3 ระบบการให้แสงของท่อสำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าว

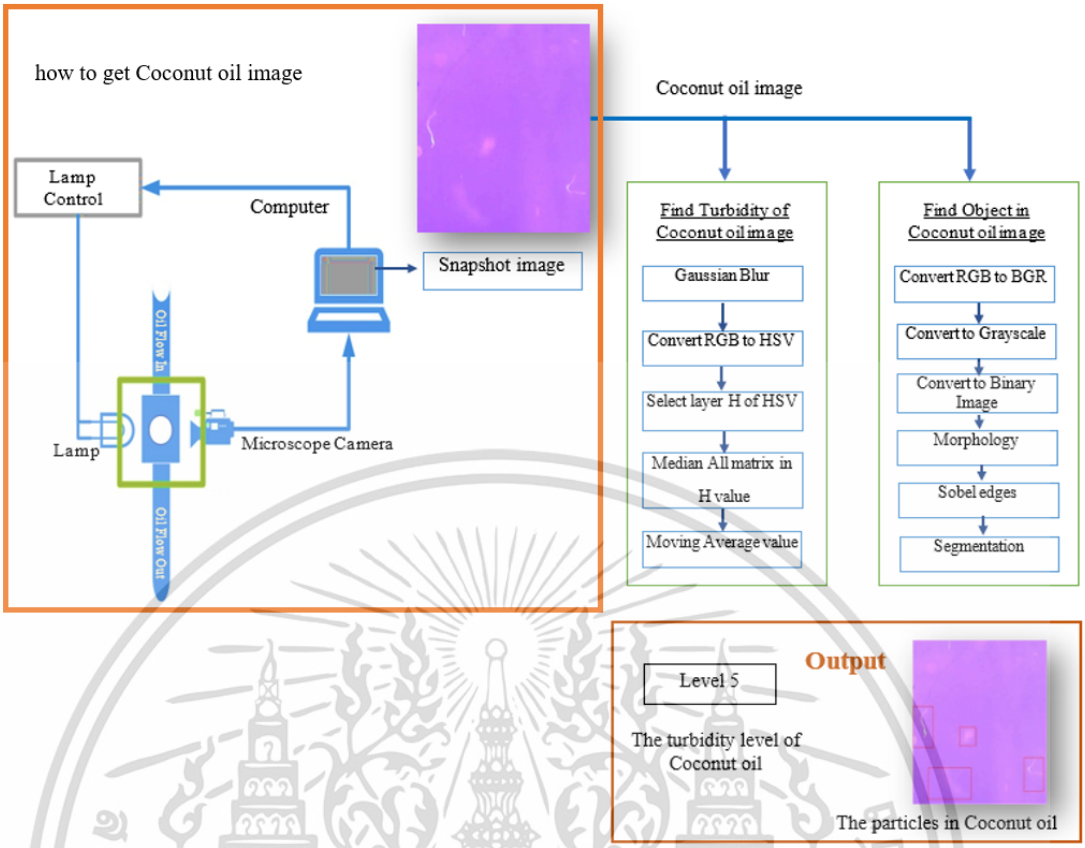
จากนั้นจึงนำอุปกรณ์ท่อสำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าวไปติดตั้งและทดสอบการรั่วไหล และแก้ไขจนสามารถนำไปติดตั้งร่วมกับระบบการผลิตน้ำมันมะพร้าวของโรงงานได้ ดังรูปที่ 3.4



(ก) ภาพก่อนการติดตั้ง (ข) ภาพหลังการติดตั้ง

รูปที่ 3.4 การติดตั้งอุปกรณ์ท่อสำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าว

ในส่วนต่อมาผู้วิจัยได้ดำเนินการพัฒนาโครงสร้างและขั้นตอนวิธีการสำหรับการตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าวด้วยวิธีทางคอมพิวเตอร์วิทัศน์ประกอบด้วย การนำภาพเข้าสู่การประมวลผลภาพ การแบ่งส่วนของภาพ (Image segmentation) ที่ต้องการในภาพ จากนั้นทำการปรับปรุงภาพ (Image enhancement) เพื่อทำให้ภาพมีความเด่นชัดขึ้นและลดสัญญาณรบกวนที่เกิดขึ้นภายในภาพ การประมวลผลหาค่าความขุ่นของน้ำมันมะพร้าวที่ได้จากภาพ และการตรวจหาแล้วจำแนกวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าว จากนั้นแสดงค่าความขุ่นและสิ่งปลอมปน โดยภาพรวมของระบบการวัดค่าความขุ่นและการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าวแสดงดังรูปที่ 3.5



รูปที่ 3.5 ภาพรวมของระบบการวัดค่าความขุ่นและการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าว

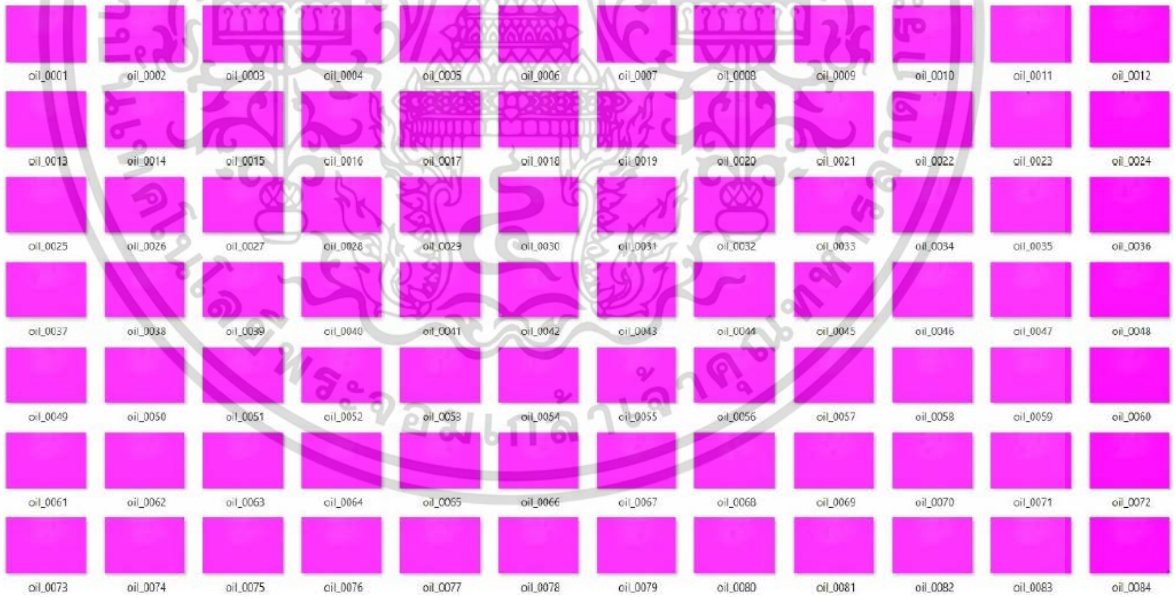
3.2 วิธีการรับภาพน้ำมันมะพร้าว

ในขั้นตอนการทำงานที่ได้นำเสนอ [41] เริ่มต้นจากการเปิดให้น้ำมันมะพร้าวที่ได้จากกระบวนการหมุนเหวี่ยงน้ำมันมะพร้าวเพื่อกรองสิ่งปลอมปนที่อยู่ในน้ำมันมะพร้าวออกมา โดยน้ำมันมะพร้าวที่ได้จะถูกลำเลียงผ่านท่อนำส่งน้ำมันมายังตำแหน่งการตรวจจับความขุ่นและการตรวจจับสิ่งปลอมปนของน้ำมันมะพร้าว โดยในตำแหน่งการตรวจจับความขุ่นและการตรวจจับสิ่งปลอมปนประกอบไปด้วย 4 ส่วนสำคัญ ได้แก่ กล้องไมโครสโคป ท่อสแตนเลสที่สามารถให้แสงลอดผ่านน้ำมันมะพร้าวได้ โดยใช้สแตนเลส เบอร์ 304 ที่ใช้สำหรับอุตสาหกรรมอาหาร กล่องป้องกันแสงรบกวนจากภายนอก และอุปกรณ์ให้แสงสว่างในการส่องแสงให้ลอดผ่านน้ำมันมะพร้าว ซึ่งระบบการทำงานถูกควบคุมการถ่ายภาพน้ำมันมะพร้าวด้วยคอมพิวเตอร์ ซึ่งภาพถ่ายน้ำมันมะพร้าวเป็นภาพจากกล้องไมโครสโคปที่มีกำลังขยายอยู่ระหว่าง 50 - 100 เท่า โดยมีความละเอียดของภาพขนาด 1,280 x 1,024 จุดภาพ ภาพที่ได้เป็นภาพที่เกิดจากแสงที่ลอดผ่านน้ำมันมะพร้าวมาตกกระทบที่เลนซ์ของกล้องไมโครสโคป ซึ่งได้เลือกใช้แสงแบบ Blacklight ที่ความยาวคลื่น 410 nm (นาโนเมตร) แสงแบบ Blacklight นั้นมีผลทำให้เกิดการสะท้อนกลับของแสงกับวัตถุน้อยที่สุด เมื่อเปรียบเทียบกับแสงจาก

หลอดไฟทั่วไป โดยในการทดลองได้ทำการเก็บบันทึกข้อมูลภาพชนิด bmp (Bitmap) เพื่อใช้ในการประมวลผลภาพต่อไป

เริ่มต้นของการนำข้อมูลมาทำการวิเคราะห์ ได้ทำการเก็บข้อมูลภาพถ่ายน้ำมันมะพร้าวที่เกิดจากกระบวนการผลิตในโรงงานต้นแบบ ที่มีการผลิตในแต่ละช่วงเวลาแตกต่างกัน เป็นจำนวน 3 ครั้ง รวมเป็นภาพตัวอย่างของน้ำมันมะพร้าวที่มีความขุ่นของน้ำมันมะพร้าวที่แตกต่างกัน โดยเลือกนำมาใช้สำหรับอ้างอิงต้นแบบทั้งหมดจำนวน 1,409 ภาพ เรียกชุดข้อมูลนี้ว่า Coconut Oil Determine Turbidity (CCODT) [41] ซึ่งข้อมูลภาพน้ำมันมะพร้าวที่ได้ แสดงดังรูปที่ 3.6

(ก) ระดับความขุ่นสูงสุด (ข) ระดับความขุ่นน้อยที่สุด



(ค) ตัวอย่างภาพของชุดข้อมูล CCODT

รูปที่ 3.6 ตัวอย่างภาพน้ำมันมะพร้าวที่ใช้ในการทดลอง

โดยรูปที่ 3.6(ก) เป็นตัวอย่างภาพน้ำมันมะพร้าวที่มีความขุ่นมาก และรูปที่ 3.6(ข) เป็นตัวอย่างภาพน้ำมันมะพร้าวที่มีความขุ่นน้อย ซึ่งปัจจัยที่มีผลทำให้น้ำมันมะพร้าวมีความขุ่นคือสิ่งปลอมปนต่างๆ ที่อยู่ในน้ำมันมะพร้าวซึ่งอาจเกิดขึ้นจากขั้นตอนของการผลิตได้

3.3 การหาค่าความขุ่นของน้ำมันมะพร้าว

ในการหาค่าความขุ่นของน้ำมันมะพร้าวได้นำภาพจากชุดข้อมูล CCODT มาเข้าสู่การประมวลผล ซึ่งทำการทดสอบและเปรียบเทียบกันทั้งหมด 4 วิธีการ ได้แก่ 1) วิธีการหาค่าเฉลี่ยของภาพจากระดับสีเทา 2) วิธีการหาค่าเฉลี่ยจากความแข็งแรงของสี 3) วิธีการสุ่มพื้นที่จากความแข็งแรงของสี และ 4) วิธีการหาค่าเฉลี่ยเคลื่อนที่จากความแข็งแรงของสี [41] สามารถอธิบายรายละเอียดได้ดังนี้

3.3.1 วิธีการหาค่าเฉลี่ยของภาพจากระดับสีเทา (Median of Gray Scale: MoGS)

วิธีการนี้เป็นวิธีการแปลงภาพสี RGB ให้เป็นภาพระดับเทา (Grayscale) ก่อน จากนั้นจึงนำค่าของจุดภาพแต่ละจุดภาพ ของภาพระดับเทามาหาค่าเฉลี่ยกึ่งกลาง (Median) ของจุดภาพทั้งหมด ทำให้ได้ค่ากึ่งกลางของภาพระดับเทานั้นโดยมีขั้นตอนแสดงดังรูปที่ 3.7

รูปที่ 3.7 ขั้นตอนการหาค่าความขุ่นของน้ำมันมะพร้าวด้วยการหาค่าเฉลี่ยจากภาพระดับเทา

จากขั้นตอนในรูปที่ 3.7 เมื่อนำขั้นตอนมาพัฒนาโปรแกรมตามวิธี MoGS แสดงใน Algorithm 1

Algorithm 1: MoGS Method

Input: Im = coconut oil image // image with values from R = 0 to 255 G = 0 to 255 and B = 0 to 255 for each pixel

1 **FOR** i = 1 to 1,409 images

2 image = Im (i) // convert RGB image to grayscale image number of shades are between 0-255

3 **FOR** all the RGB image pixels

ConvertFactor = 255 / ((Number of shades > 1) - 1) // Number of shades must be greater than 1

Avg_Value = (Red+Green+Blue) / 3

Im_Gray = Integer ((Avg_Value/ConvertFactor) + 0.5) x ConvertFactor

END FOR

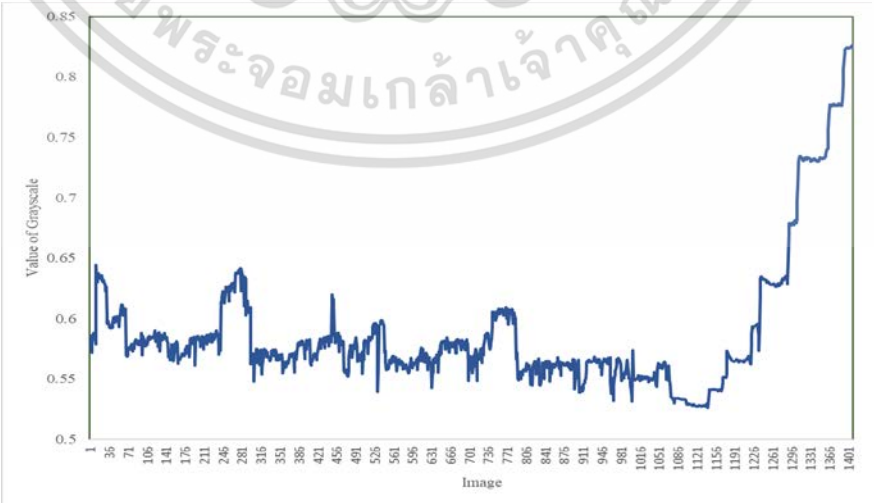
// generate (Im_Gray) Matrix of grayscale image from the value obtained

4 M_Gray <_ Median (Im_Gray) // _nd the median for value in Im_Gray

5 store [i] D M_Gray // save to array

6 **END FOR**

จากนั้นนำข้อมูลภาพจากชุดข้อมูล CCODT ทุกภาพ มาผ่านขั้นตอน Algorithm 1 โดยทำการบันทึกผลค่าเฉลี่ยของภาพระดับเทาที่ได้ทั้งหมด ซึ่งค่าเฉลี่ยกึ่งกลางของภาพระดับเทาจากวิธีการ MoGS ทั้งหมดนั้น มีค่าสูงสุดอยู่ที่ 0.826049 และมีต่ำสุดอยู่ที่ 0.526254 ค่าของผลลัพธ์ที่ได้จากวิธีการวิธีการหาค่าเฉลี่ยของภาพจากระดับสีเทา MoGS แสดงดังรูปที่ 3.8 โดยกำหนดให้แกน x แทนลำดับภาพ และแกน y แทนค่าเฉลี่ยของภาพระดับเทา



รูปที่ 3.8 ค่าของผลลัพธ์ที่ได้จากวิธีการ MoGS

โดยที่ค่าสูงสุดสามารถนำมาแทนค่าความขุ่นของน้ำมันมะพร้าวที่น้อยที่สุด และค่าต่ำสุดสามารถนำมาแทนค่าความขุ่นของน้ำมันมะพร้าวที่มากที่สุดได้ ขอบเขตของข้อมูลตามขั้นตอนวิธีการ MoGS นั้นแสดงค่าดังตารางที่ 3.1

ตารางที่ 3.1 การเปรียบเทียบค่าขอบเขตข้อมูลของแต่ละวิธีการที่นำเสนอ

Value	Method			
	MoGS	MoH	RPMoH	MAMoH
Minimum	0.526254	0.804513	0.80307	0.80468
Maximum	0.826049	0.833719	0.83515	0.83355

3.3.2 วิธีการหาค่าเฉลี่ยกึ่งกลางจากความแข็งแรงของสี (Median of Hue: MoH)

วิธีการนี้เป็นวิธีการที่อาศัยค่าความแข็งแรงของสี (Hue) ที่อยู่ในโดเมนสี HSV (Hue, Saturation, Value) เข้ามาช่วย ซึ่งจากการศึกษาค้นคว้าพบว่าค่าความแข็งแรงของสีนั้นมีความคงทนต่อสิ่งรบกวน (Noise) ที่มีอยู่ในภาพได้ดีกว่าโดเมนสี RGB ขั้นตอนแสดงได้ดังรูปที่ 3.9

รูปที่ 3.9 ขั้นตอนการหาค่าความขุ่นของน้ำมันมะพร้าวด้วยค่าเฉลี่ยกึ่งกลางของความแข็งแรงของสี

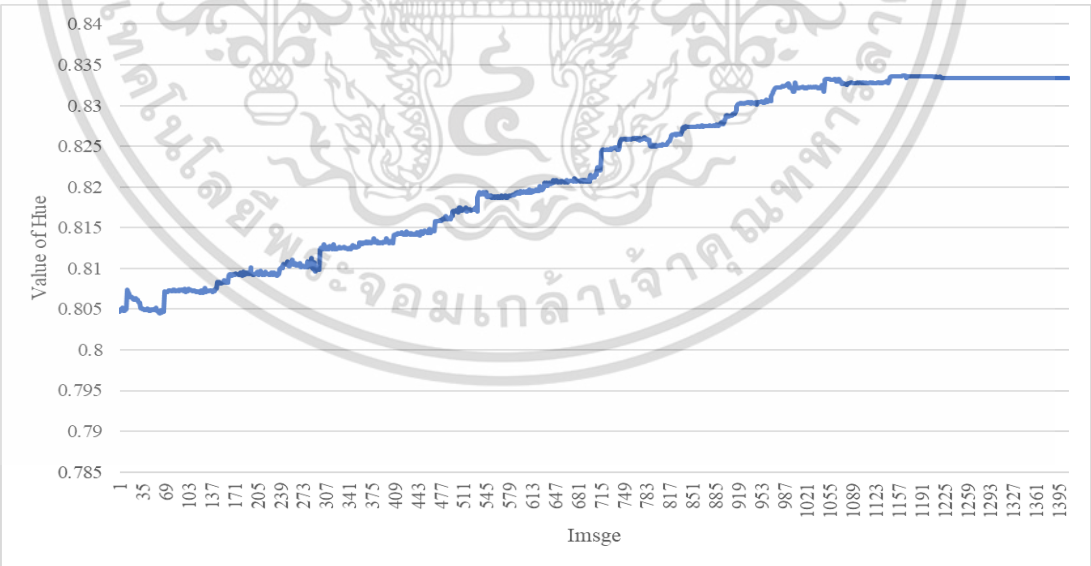
จากขั้นตอนในรูปที่ 3.9 เมื่อนำขั้นตอนมาพัฒนาโปรแกรมตามวิธี MoH โดยเริ่มต้นจากการนำภาพจากชุดข้อมูล CCODT เข้ามาผ่านกระบวนการเบลอภาพด้วยวิธีการ Gaussian ที่ค่า 70 เปอร์เซ็นต์เพื่อช่วยลดสิ่งรบกวนในภาพ โดยการเบลอวัตถุในภาพให้กลืนไปกับพื้นหลังของภาพ จากนั้นทำการเปลี่ยนโดเมนสี จาก RGB เป็น HSV เนื่องจากโดเมนสี HSV นั้นมีความคงทนต่อสัญญาณรบกวนที่ดีกว่าและมีค่าความแข็งแรงของสีที่สามารถนำมาใช้ประโยชน์ได้ จากนั้นนำชั้นข้อมูลของค่าสี H มาหาค่ากึ่งกลางของข้อมูลจากทุกจุดภาพทั้งหมด และทำการบันทึกค่าเฉลี่ยกึ่งกลางที่ได้ และแสดงผลลัพธ์ซึ่งผลลัพธ์ที่ได้แทนค่าความขุ่นในน้ำมันมะพร้าว โดยมีขั้นตอนการพัฒนาโปรแกรมตามวิธี MoH แสดงใน Algorithm 2

Algorithm 2: MoH Method

Input: Im = coconut oil image // image with values from R = 0 to 255 G = 0 to 255 and B = 0 to 255 for each pixel

```
1 FOR i = 1 to 1,409 images
2       image = Im (i)    // convert RGB image to grayscale image number of shades
                            are between 0-255
3       Blur_Im ← Gaussian (image,70)        // blur image for reduce noise at 70%
4       Im_HSV = HSV ← RGB(Blur_Im(i))       // convert domain color RGB to HSV
5       H_Im = Select H layer from Im_HSV    // split H color layer from Im_HSV
6       M_H_Im ← Median(H_Im)                // find the median for value in H_Im
7       store [i] = M_H_Im                   // save median value to array
8 END FOR
```

เมื่อนำชุดข้อมูล CCODT มาผ่านขั้นตอน Algorithm 2 โดยทำการบันทึกผลลัพธ์ที่ได้ในแต่ละภาพ ซึ่งเมื่อทำการจัดเรียงข้อมูลผลลัพธ์ที่ได้พบว่ามีค่าสูงสุดอยู่ที่ 0.833719 และมีต่ำสุดอยู่ที่ 0.804513 ผลลัพธ์ทั้งหมดของวิธีการหาค่าเฉลี่ยกึ่งกลางจากความแข็งแรงของสีแสดงดังรูปที่ 3.10 โดยที่กำหนดให้ แกน x แทนลำดับภาพ และ แกน y แทนค่าเฉลี่ยกึ่งกลางจากค่าความแข็งแรงของสี

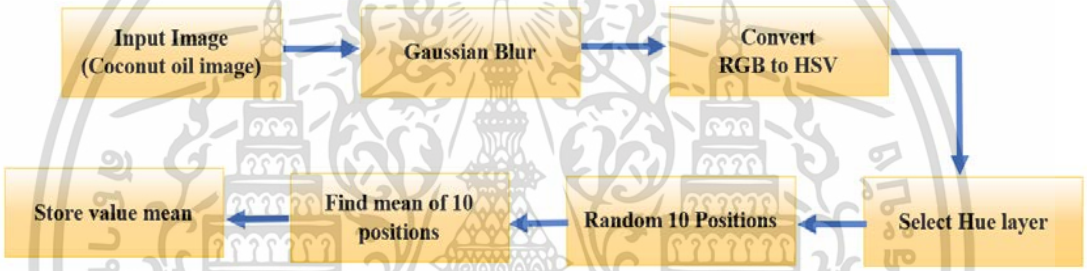


รูปที่ 3.10 ค่าของผลลัพธ์ที่ได้จากวิธีการ MoH

สำหรับที่ค่าสูงสุดสามารถนำมาแทนค่าความขุ่นของน้ำมันมะพร้าวที่น้อยที่สุด และค่าต่ำสุดสามารถนำมาแทนค่าความขุ่นของน้ำมันมะพร้าวที่มากที่สุดได้ โดยขอบเขตของข้อมูลตามขั้นตอนวิธีการ MoH นั้นแสดงค่าดังตารางที่ 3.1 ที่ได้กล่าวไปแล้ว

3.3.3 วิธีการสุ่มพื้นที่จากความแข็งแรงของสี (Random Position Median of Hue: RPMoH)

วิธีการนี้เป็นวิธีการสุ่มตำแหน่งที่อยู่ในภาพน้ำมันมะพร้าว เพื่อนำค่าของจุดภาพในตำแหน่งนั้นมาประมวลผลแทนการประมวลผลทั้งภาพ โดยคาดหวังในเรื่องเวลาในการประมวลผลที่น้อยลงจากวิธีการ MoH ซึ่งวิธีการนี้มีการเริ่มต้นเช่นเดียวกันกับวิธีการ MoH แต่แตกต่างในส่วนของการหาค่ากึ่งกลางจากการหาค่าจากข้อมูลทั้งภาพเป็นการหาค่ากึ่งกลางจากตำแหน่ง 10 ตำแหน่งแทน ซึ่งมีขั้นตอนดังรูปที่ 3.11



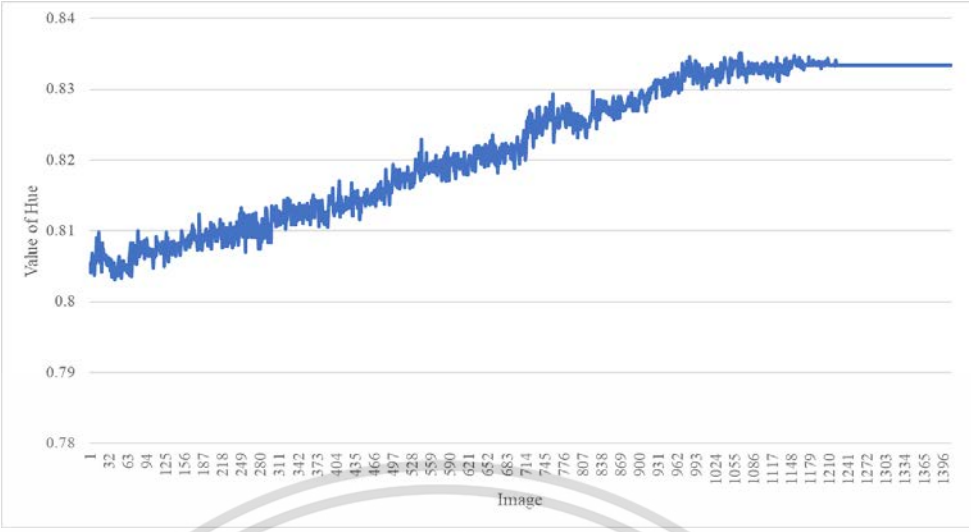
รูปที่ 3.11 ขั้นตอนการหาค่าความขุ่นของน้ำมันมะพร้าวด้วยการสุ่มพื้นที่จากความแข็งแรงของสี

จากขั้นตอนในรูปที่ 3.11 เมื่อนำขั้นตอนมาพัฒนาโปรแกรมตามวิธี RPMoH โดยเริ่มต้นขั้นตอนวิธีการเช่นเดียวกันกับวิธีการ MoH ซึ่งหลังจากได้ชั้น H จาก โดเมนสี HSV มา จะทำการสุ่มตำแหน่งเพื่อใช้อ้างอิงการอ่านข้อมูลของชั้น H โดยสุ่มตำแหน่งทั้งหมด 10 ตำแหน่ง จากนั้นจึงอ่านค่าจากตำแหน่งอ้างอิงที่ได้ และนำมาหาค่ากึ่งกลางของค่า 10 ตำแหน่งนั้น ซึ่งผลลัพธ์จากค่ากึ่งกลางที่ได้แทนค่าความขุ่นในน้ำมันมะพร้าว โดยมีขั้นตอนพัฒนาโปรแกรมแสดงใน Algorithm 3

Algorithm 3: RPMoH Method

```
Input: Im = coconut oil image   // image with values from R = 0 to 255 G = 0 to 255
                                   and B = 0 to 255 for each pixel
1 FOR i = 1 to 1,409 images
2       image = Im (i)          // convert RGB image to grayscale image number of
                                   shades are between 0-255
3       Blur_Im ← Gaussian (image,70)      // blur image for reduce noise at 70%
4       Im_HSV = HSV ← RGB(Blur_Im(i))     // convert domain color RGB to HSV
5       H_Im = Select H layer from Im_HSV  // split H color layer from Im_HSV
                                   // random position (x, y) for select pixel in image
6       [XIm,YIm] = size of H_Im           // find size Image
7             FOR 10 Positions
                  X[i] D Random [0-XIm]
                  Y[i] D Random [0-YIm]
                  STORE RH_Im[i] ← Read Data H layer in Position
                  X[i], Y[i]
              END FOR
8       M_RH_Im ← Median (RH_Im)           // find the median for value in RH_Im
9       store [i] ← M_RH_Im                // save value to array
10 END FOR
```

เมื่อนำชุดข้อมูล CCODT มาผ่านขั้นตอน Algorithm 3 โดยทำการบันทึกผลลัพธ์ที่ได้ในแต่ละภาพ ซึ่งเมื่อทำการจัดเรียบข้อมูลผลลัพธ์ที่ได้พบว่ามีค่าสูงสุดอยู่ที่ 0.83515 และมีต่ำสุดอยู่ที่ 0.80307 ผลลัพธ์ที่ได้ทั้งหมดของวิธีการหาค่าเฉลี่ยกึ่งกลางจากความแข็งแรงของสีแสดงดังรูปที่ 3.12 โดยที่ แกน x แทนลำดับภาพ และ แกน y แทนค่าเฉลี่ยกึ่งกลางจากค่าความแข็งแรงของสี

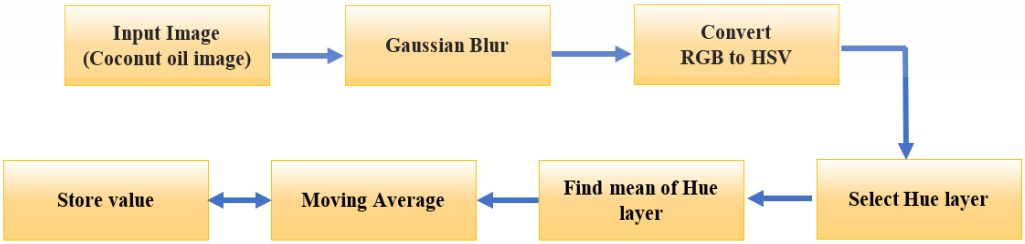


รูปที่ 3.12 ค่าของผลลัพธ์ที่ได้จากวิธีการ RPMoH

โดยที่ค่าสูงสุดสามารถนำมาแทนค่าความขุ่นของน้ำมันมะพร้าวที่น้อยที่สุด และค่าต่ำสุดสามารถนำมาแทนค่าความขุ่นของน้ำมันมะพร้าวที่มากที่สุดได้ โดยขอบเขตของข้อมูลตามขั้นตอนวิธีการ RPMoH นั้นมีค่าดังตารางที่ 3.1 ที่ได้กล่าวไปแล้ว

3.3.4 วิธีการหาค่าเฉลี่ยเคลื่อนที่จากความแข็งแรงของสี (Moving Average Median of Hue: MAMoH)

เป็นวิธีการปรับปรุงจากวิธีการ MoH โดยการนำค่าผลลัพธ์ที่ได้จากการหาค่ากึ่งกลางของชั้น H ปัจจุบัน มาหาค่าเฉลี่ยร่วมกับค่าผลลัพธ์ที่ได้จากการหาค่ากึ่งกลางของชั้น H ที่ได้ก่อนหน้า ซึ่งจากวิธีการหาค่าเฉลี่ยแบบเคลื่อนที่ได้ (Moving average) ต้องมีการกำหนดขนาดหน้าต่างของข้อมูลเพื่อใช้ในการพิจารณา ซึ่งได้ทำการทดลองเพื่อเปรียบเทียบหน้าต่างขนาดที่แตกต่างกันที่เหมาะสมต่อการนำไปใช้งาน ซึ่งวิธีการนี้มีการเริ่มต้นเช่นเดียวกันกับวิธีการ MoH แต่แตกต่างในส่วนของผลลัพธ์ที่ได้จะถูกนำมาหาค่าเฉลี่ยร่วมกับค่าก่อนหน้า ซึ่งมีขั้นตอนแสดงดังภาพที่ 3.13



รูปที่ 3.13 ขั้นตอนการหาค่าความขุ่นของน้ำมันมะพร้าวด้วยการหาค่าเฉลี่ยเคลื่อนที่จากความแข็งแรงของสี

จากขั้นตอนในรูปที่ 3.13 เมื่อนำขั้นตอนมาพัฒนาโปรแกรมตามวิธี MAMoH โดยเริ่มต้นขั้นตอนวิธีการเช่นเดียวกันกับวิธีการ MoH ซึ่งหลังจากได้ค่ากึ่งกลางของภาพชั้น H จาก โดเมนสี HSV มา จะนำไปหาค่าเฉลี่ยแบบเคลื่อนที่ได้ โดยมีขั้นตอนพัฒนาโปรแกรมตามวิธี MAMoH แสดงใน Algorithm 4

Algorithm 4: MAMoH Method

Input: Im = coconut oil image // image with values from R = 0 to 255 G = 0 to 255 and B = 0 to 255 for each pixel

1 **FOR** i = 1 to 1,409 images

2 image = Im (i) // convert RGB image to grayscale image number of shades are between 0-255

3 Blur_Im ← Gaussian (image,70) // blur image for reduce noise at 70%

4 Im_HSV = HSV ← RGB(Blur_Im(i)) // convert domain color RGB to HSV

5 H_Im = Select H layer from Im_HSV // split H color layer from Im_HSV

6 M_H_Im ← Median(H_Im) // find the median for value in H_Im

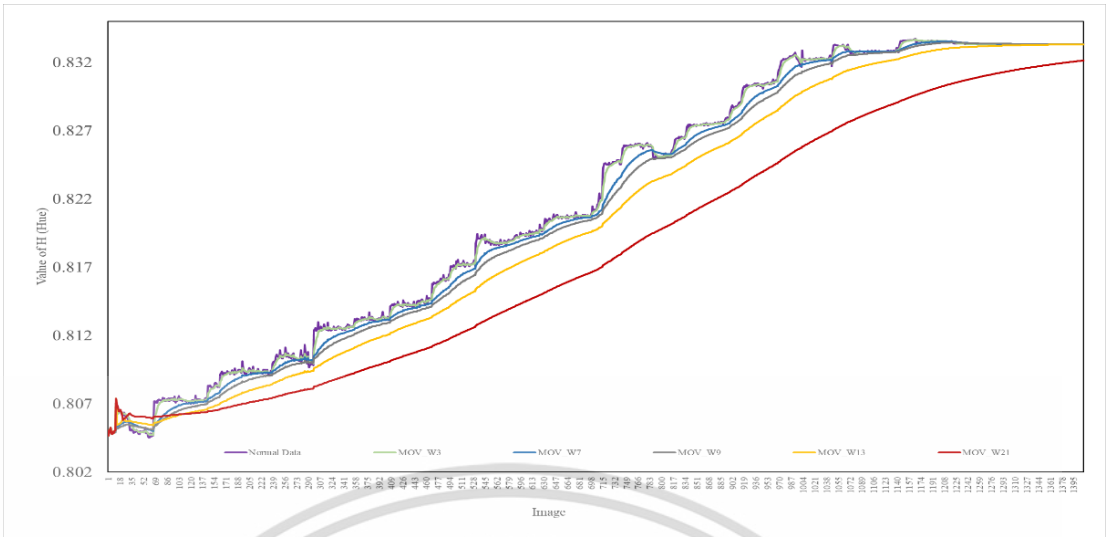
// moving average value

7 store [i] = (M_H_Im(i-1) + M_H_Im(i-2) + M_H_Im(i-3) +...+ M_H_Im(i-n))/n

//save moving average value to array

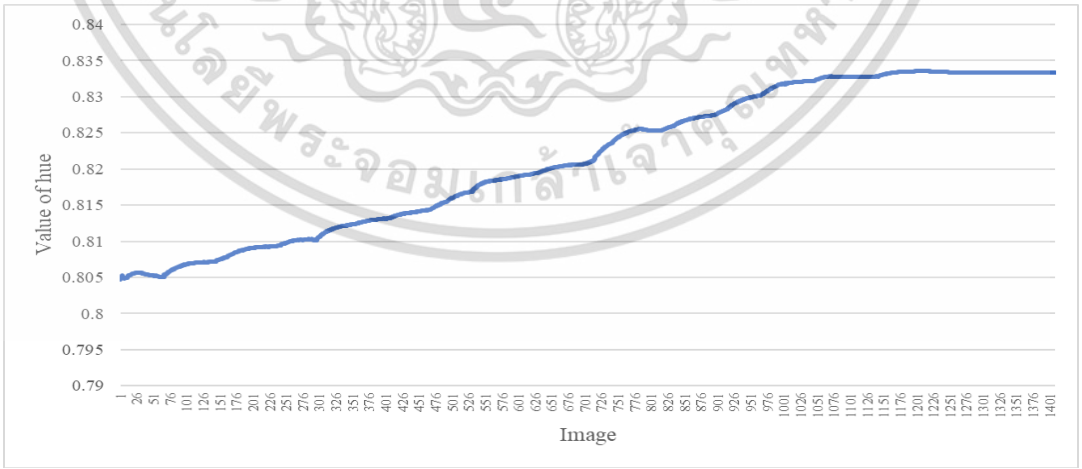
8 **END FOR**

โดยที่วิธีการหาค่าเฉลี่ยเคลื่อนที่จากความแข็งแรงของสีได้ทำการทดลองจำนวน 5 รอบ เพื่อหาค่าของหน้าต่างเคลื่อนที่ (Moving Windows) ที่เหมาะสมต่อการใช้งานมากที่สุด โดยกำหนดให้หน้าต่างมีขนาดเท่ากับ 3, 7, 9, 13 และ 21 ตามลำดับ เมื่อนำผลลัพธ์ที่ได้จากชุดข้อมูล CCODT มาเปรียบเทียบผลการใช้งานของหน้าต่างเคลื่อนที่พบว่าเมื่อใช้ขนาดหน้าต่างที่เกินกว่า 9 ข้อมูลขึ้นไปนั้น ส่งผลให้ค่าของข้อมูลที่ได้อยู่ต่ำกว่าค่าความเป็นจริง ของข้อมูลมากเกินไป ทำให้ได้ค่าคำตอบที่ได้มีความคลาดเคลื่อนต่ำกว่าค่าที่ควรเป็น การเปรียบเทียบผลลัพธ์ที่ได้จากการใช้ขนาดหน้าต่างที่แตกต่างกัน โดยที่กำหนดให้ แกน x แทนลำดับภาพ และ แกน y แทนค่าเฉลี่ยกึ่งกลางจากค่าความแข็งแรงของสี ดังรูปที่ 3.14



รูปที่ 3.14 เปรียบเทียบผลที่ได้จากการใช้ขนาดหน้าต่างที่แตกต่างกัน

จากรูปที่ 3.14 ผลที่ได้มาเปรียบเทียบผลการใช้งานหน้าต่างเคลื่อนที่ขนาดแตกต่างกัน พบว่าเมื่อใช้ขนาดหน้าต่างที่เกินกว่า 9 ข้อมูล ขึ้นไปนั้นส่งผลให้ค่าของข้อมูลที่ได้ มีค่าต่ำกว่าค่าความเป็นจริงของข้อมูลมากเกินไป ทำให้ค่าคำตอบที่ได้ มีความคลาดเคลื่อนต่ำกว่าความเป็นจริง ซึ่งจากการทดลองพบว่าขนาดที่เหมาะสมในการทำงานมีขนาดหน้าต่างเคลื่อนที่อยู่ในช่วง 7 ข้อมูล โดยหลังจากการนำข้อมูลผ่านวิธีการค่าเฉลี่ยแบบเคลื่อนที่ได้นั้น ค่าที่ได้ยังอยู่ในช่วงของข้อมูลเดิมและได้เส้นข้อมูลใหม่ที่มีความเรียบเพิ่มขึ้น โดยที่ แกน x แทนลำดับภาพ และ แกน y แทนค่าเฉลี่ยกึ่งกลางจากค่าความแข็งแรงของสี ดังรูปที่ 3.15



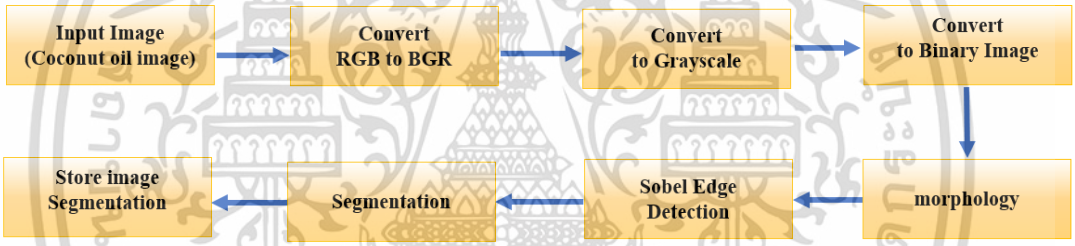
รูปที่ 3.15 ค่าของผลลัพธ์ที่ได้จากวิธีการ MAMoH

3.4 การหาวัตถุที่อยู่ในน้ำมันมะพร้าว

สำหรับการทดลองหาวัตถุที่อยู่ในน้ำมันมะพร้าว ได้นำภาพที่ได้เก็บบันทึกไว้จากกระบวนการผลิตน้ำมันมะพร้าวในช่วงเวลาที่แตกต่างกัน เพื่อหาวัตถุปลอมปนที่มีขนาดเล็กที่อาจเล็ดลอดจากขั้นตอนการกรองน้ำมันมะพร้าวได้ ซึ่งวัตถุปลอมปนนั้นส่งผลกระทบโดยตรงต่อคุณภาพของน้ำมันมะพร้าว สำหรับการหาและการตรวจสอบวัตถุปลอมปนในน้ำมันมะพร้าวนั้นสามารถแบ่งได้เป็น 2 ส่วน คือ 1) ส่วนของการหาวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าวและ 2) ส่วนการตรวจสอบจำแนกวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าว

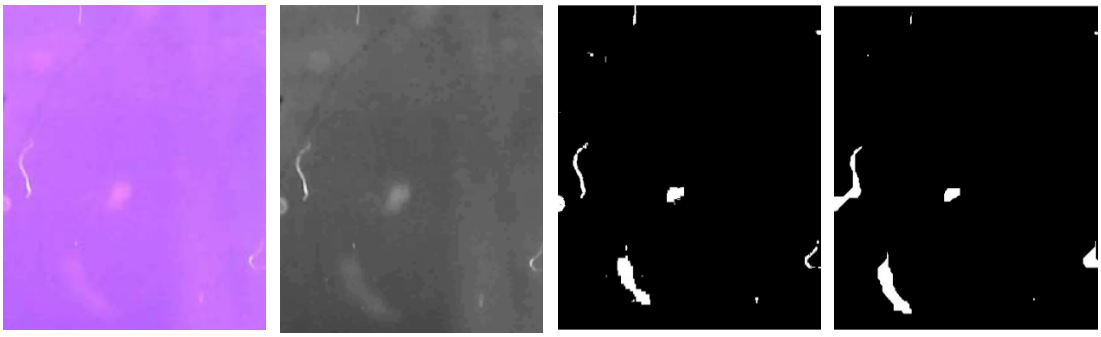
3.4.1 การหาวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าว

สำหรับการหาวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าวนั้น สามารถทำได้โดยใช้หลักการประมวลผลภาพ ตามขั้นตอนการหาวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าว ขั้นตอนการหาวัตถุในภาพแสดงได้ดังรูปที่ 3.16



รูปที่ 3.16 ขั้นตอนการหาวัตถุในภาพ

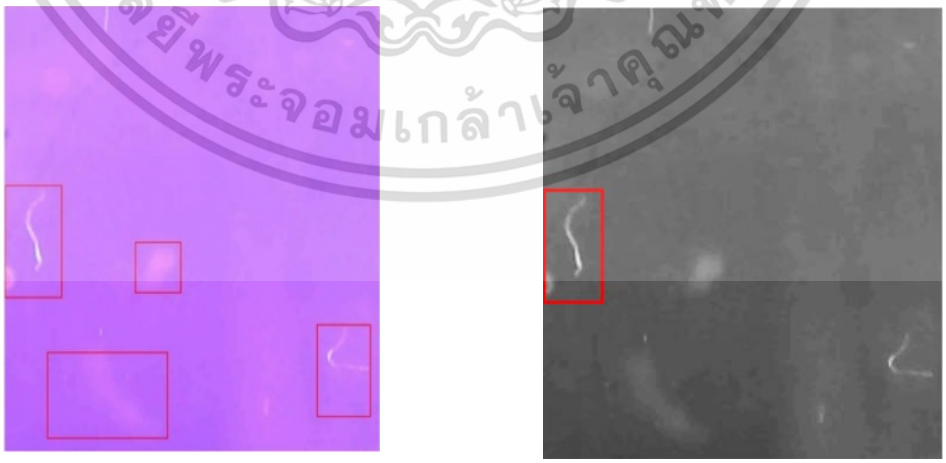
จากขั้นตอนการทำงานในรูปที่ 3.16 เมื่อนำภาพน้ำมันมะพร้าวเข้าสู่การประมวลผลเริ่มต้นจากการแปลงโดเมนของภาพน้ำมันมะพร้าวที่อยู่ในโดเมนสี RGB (Red: R, Green: G, Blue: B) ให้อยู่ในโดเมนสี BGR (Blue: B, Green: G, Red: R) ซึ่งมีนัยสำคัญแบบเดียวกัน แตกต่างกันในส่วนของการสลับของชั้น ซึ่งเครื่องมือที่ใช้ในการพัฒนามีการอ้างอิงโดเมนสี BGR จึงต้องทำการเปลี่ยนโดเมนสีตามเครื่องมือที่ใช้งาน จากนั้นจึงทำการแปลงภาพให้อยู่ในรูปแบบภาพระดับเทา ซึ่งเป็นการทำภาพสี BGR ดังรูปที่ 3.17(ก) ให้เป็นภาพระดับเทาการลดขนาดภาพ จากภาพ 3 ชั้นข้อมูลให้เป็น 1 ชั้นข้อมูล ซึ่งทำให้จุดภาพมีค่าระดับจุดภาพตั้งแต่ 0-255 ระดับ ดังรูปที่ 3.17(ข) จากนั้นทำการแปลงภาพระดับเทาให้อยู่ในรูปแบบภาพไบนารี (Binary image) หรือภาพขาว-ดำ ที่มีค่าระดับจุดภาพ 2 ระดับที่มีค่า 0 หรือ 1 เท่านั้น โดยค่า 0 แทนสีดำและค่า 1 แทนสีขาว ดังรูปที่ 3.17(ค) ซึ่งผลที่ได้ ดังรูปที่ 3.17



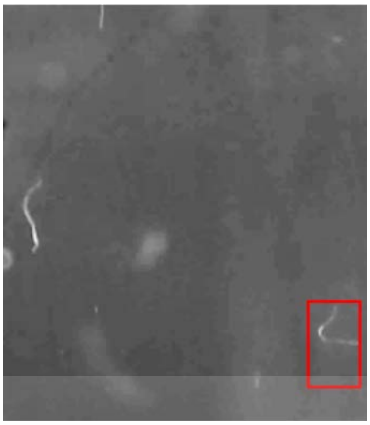
(ก) ภาพ BGR (ข) ภาพระดับเทา (ค) ภาพขาวดำ (ง) ภาพจากวิธีการปิด

รูปที่ 3.17 ผลลัพธ์การแปลงภาพ BGR เป็นภาพขาวดำ

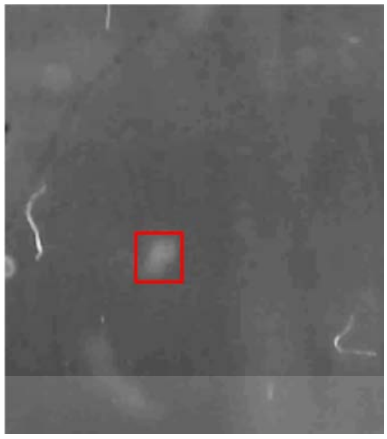
จากนั้นในขั้นตอนต่อไปทำการปรับปรุงภาพด้วยวิธีสัณฐานวิทยา (Morphology) เพื่อลดสิ่งรบกวนในภาพด้วยวิธีการปิด (Closing) ซึ่งเป็นการทำงานร่วมกันระหว่างการขยาย (Dilation) และการกร่อน (Erosion) โดยเริ่มต้นการทำงานในรอบแรกด้วยวิธีการขยาย จากนั้นในรอบที่สองดำเนินการด้วยวิธีการกร่อน ผลที่ได้ดังรูปที่ 3.17(ง) จากนั้นได้ใช้วิธีการหาวัตถุที่อยู่ในภาพจากการใช้การหาขอบของวัตถุโดยวิธีโซเบล (Sobel edge detection) ซึ่งอาศัยการค้นหาความแตกต่างของความเข้มแสง จากนั้นทำการเก็บตำแหน่งจุดที่มีความแตกต่างของความเข้มแสงที่หาได้ เมื่อการหาวัตถุในภาพเสร็จสิ้นลง นำตำแหน่งที่หาได้จากวิธีการหาขอบของวัตถุโดยวิธีโซเบล มาทำการตัดภาพวัตถุที่อยู่ในภาพน้ำมันมะพร้าวตามตำแหน่งที่ได้เก็บไว้ โดยพิจารณาภาพวัตถุที่มีขนาดจุดภาพรวมกันมากกว่า 8 จุดภาพ เพื่อเป็นการหลีกเลี่ยงสิ่งรบกวนที่อยู่ในภาพเพิ่มเติมโดยผลลัพธ์ของการทดลองดังรูปที่ 3.18



(ก) วัตถุที่อยู่ในภาพ (ข) ภาพวัตถุที่หนึ่ง



(ค) ภาพวัตถุที่สอง



(ง) ภาพวัตถุที่สาม

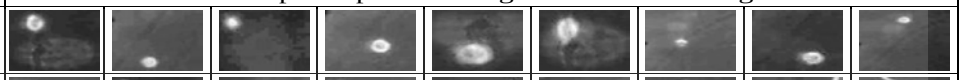
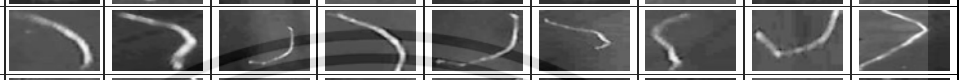
(จ) ภาพวัตถุที่สี่

รูปที่ 3.18 ผลลัพธ์จากการหาวัตถุที่อยู่ในภาพถ่ายน้ำมันมะพร้าว

จากนั้นทำการเก็บรวบรวมข้อมูลจากภาพถ่ายน้ำมันมะพร้าวเพื่อหาสิ่งปลอมปนในน้ำมันมะพร้าวเพิ่มเติม โดยจากการเก็บบันทึกภาพถ่ายน้ำมันมะพร้าวจากกระบวนการผลิตน้ำมันมะพร้าวเพิ่มเติม ทำให้ได้ภาพถ่ายน้ำมันมะพร้าวทั้งหมดจำนวน 4,725 ภาพ ซึ่งเมื่อนำภาพถ่ายน้ำมันมะพร้าวมาทั้งหมดมาผ่านกระบวนการหาวัตถุที่อยู่ในภาพด้วยกระบวนการ ในรูปที่ 3.16 ทำให้ได้ภาพวัตถุที่อยู่ในภาพถ่ายน้ำมันมะพร้าวทั้งหมด จำนวน 14,784 ภาพ ซึ่งมีภาพวัตถุในน้ำมันมะพร้าวที่ไม่สมบูรณ์ปะปนอยู่ด้วย จากนั้นจึงนำภาพทั้งหมดมาทำการคัดกรองภาพวัตถุที่อยู่ในน้ำมันมะพร้าวโดยพิจารณาจำแนกจากรูปทรงของวัตถุที่เป็นรูปร่าง และภาพวัตถุที่ไม่สามารถระบุรูปร่างหรือภาพวัตถุที่ไม่สมบูรณ์ ไม่สามารถจำแนกวัตถุได้ ซึ่งจากการคัดแยกตามรูปทรงของวัตถุทำให้เหลือภาพวัตถุที่อยู่ในน้ำมันมะพร้าวจำนวน 7,861 ภาพ จากนั้นภาพวัตถุทั้งหมดได้ถูกจัดกลุ่มโดยผู้เชี่ยวชาญ ซึ่งผู้เชี่ยวชาญได้มีการแบ่งกลุ่มของภาพวัตถุปลอมปนออกเป็นกลุ่มได้ทั้งหมด 10 กลุ่ม โดยแบ่งเป็นกลุ่ม

ของฟองอากาศ 1 กลุ่ม (Air) กลุ่มของกากใย fiber จำนวน 4 กลุ่ม (FiberT1, FiberT2, FiberT3 และ FiberT4) กลุ่มของเศษฝุ่นผงจำนวน 4 กลุ่ม (ParticleT1, ParticleT2, ParticleT3 และ ParticleT4) และกลุ่มของเนื้อเยื่อเนื้อมะพร้าว จำนวน 1 กลุ่ม (Tissue) โดยข้อมูลภาพวัตถุปลอมปนในแต่ละกลุ่ม ดังตารางที่ 3.2

ตารางที่ 3.2 ภาพวัตถุปลอมปนในน้ำมันมะพร้าวจำนวน 10 กลุ่ม

Class	Example of particle image for model training
Air	
FiberT1	
FiberT2	
FiberT3	
FiberT4	
ParticleT1	
ParticleT2	
ParticleT3	
ParticleT4	
Tissue	

ตารางที่ 3.2 เป็นข้อมูลรายละเอียดของภาพวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าวที่ต้องทำการตรวจสอบ ซึ่งจากข้อมูลภาพวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าวจำนวน 7,861 ภาพ นำมาจัดเป็นชุดข้อมูลเพื่อใช้สำหรับการตรวจสอบวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าว โดยแบ่งออกเป็น 2 ชุดข้อมูล ซึ่งชุดข้อมูลที่ 1 ถูกนำไปใช้สำหรับการฝึกฝนแบบจำลอง (Training model) ของการเรียนรู้เชิงลึก และชุดข้อมูลที่ 2 ถูกนำไปใช้สำหรับการทดสอบประสิทธิภาพของแบบจำลอง (Model) โดยที่

ชุดข้อมูลที่ 1 เรียกว่า “Particle in Coconut Oil_V2 (PiCO_V2)” เป็นชุดข้อมูลที่ใช้สำหรับการฝึกฝนแบบจำลอง กำหนดจำนวนข้อมูลภาพวัตถุปลอมปน particle ทั้งหมด 6,861 ภาพ ประกอบด้วย ภาพ Air จำนวน 1,529 ภาพ ภาพ FiberT1 จำนวน 1,107 ภาพ ภาพ FiberT2 จำนวน 311 ภาพ ภาพ FiberT3 จำนวน 398 ภาพ ภาพ FiberT4 จำนวน 391 ภาพ ภาพ ParticleT1 จำนวน 516 ภาพ ภาพ ParticleT2 จำนวน 661 ภาพ ภาพ ParticleT3 จำนวน 483 ภาพ ภาพ ParticleT4 จำนวน 898 ภาพ และภาพ Tissue จำนวน 567 ภาพ

ชุดข้อมูลที่ 2 เรียกว่า “Particle in Coconut Oil_V1 (PiCO_V1)” เป็นชุดข้อมูลที่ใช้สำหรับการทดสอบวัดค่าประสิทธิภาพแบบจำลอง โดยกำหนดจำนวนข้อมูลภาพวัตถุปลอมปน particle ทั้งหมด 1,000 ภาพ ประกอบด้วย ภาพ Air จำนวน 135 ภาพ ภาพ FiberT1 จำนวน 186 ภาพ ภาพ FiberT2 จำนวน 72 ภาพ ภาพ FiberT3 จำนวน 111 ภาพ ภาพ FiberT4 จำนวน 76 ภาพ ภาพ ParticleT1 จำนวน 63 ภาพ ภาพ ParticleT2 จำนวน 91 ภาพ ภาพ ParticleT3 จำนวน 64 ภาพ ภาพ ParticleT4 จำนวน 86 ภาพ และ ภาพ Tissue จำนวน 116 ภาพ

โดยชุดข้อมูลทั้ง 2 ชุด PiCO_V1 และ PiCO_V2 ถูกจัดเก็บเป็นภาพระดับเทาที่มีการปรับขนาดภาพตามมาตรฐานที่ขนาดภาพเท่ากับ 224 x 224 และขนาดภาพเท่ากับ 299 x 299 เพื่อใช้งานในลำดับถัดไป

3.4.2 การตรวจสอบจำแนกวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าว

สำหรับการตรวจสอบจำแนกวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าว ได้อาศัยการสร้างรูปแบบจำลองต่าง ๆ ตามสถาปัตยกรรมแบบ CNN เพื่อช่วยในการจำแนกวัตถุปลอมปนที่อยู่ในน้ำมันมะพร้าว โดยทำการทดลองร่วมกับข้อมูล PiCO_V2 ซึ่งสถาปัตยกรรม CNN ที่ได้นำมาทดลองประกอบด้วย สถาปัตยกรรม Standard CNN, MobileNetV2, ResNet50, ResNet101, DenseNet121, VGG16 และ VGG19 ซึ่งอาศัยภาพมีขนาดภาพเท่ากับ 224 x 224 จุดภาพ (Pixel) ในการฝึกฝน และสถาปัตยกรรม Xception, InceptionV3 และ InceptionResNetV2 architectures จะอาศัยภาพมีขนาดภาพเท่ากับ 299 x 299 จุดภาพ ในการฝึกฝน โดยการพัฒนาการฝึกฝนแบบจำลอง ได้ใช้โปรแกรม Jupyter Notebook และ Python 3.7 kernel พร้อมกับไลบรารี TensorFlow, Keras และ OpenCV โดยหลังจากดำเนินการพัฒนาฝึกฝนแบบจำลองที่ต้องการได้ จะทำการนำแบบจำลองมาทำการทดสอบด้วยชุดข้อมูล PiCO_V1 ต่อไป

ในการทดลองส่วนแรกของการฝึกฝนแบบจำลองตามสถาปัตยกรรม CNN นั้นถูกดำเนินการบนโครงสร้างสถาปัตยกรรม MobileNet เท่านั้น ดังรูปที่ 3.19

Model: "mobilenet_1.00_224"

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 224, 224, 3)]	0
conv1_pad (ZeroPadding2D)	(None, 225, 225, 3)	0
conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1_bn (BatchNormalization	(None, 112, 112, 32)	128
conv1_relu (ReLU)	(None, 112, 112, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv_dw_1_bn (BatchNormaliza	(None, 112, 112, 32)	128
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2048
conv_pw_1_bn (BatchNormaliza	(None, 112, 112, 64)	256
conv_pw_1_relu (ReLU)	(None, 112, 112, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, 113, 113, 64)	0
conv_dw_2 (DepthwiseConv2D)	(None, 56, 56, 64)	576
conv_dw_2_bn (BatchNormaliza	(None, 56, 56, 64)	256
conv_dw_2_relu (ReLU)	(None, 56, 56, 64)	0
conv_pw_2 (Conv2D)	(None, 56, 56, 128)	8192

(ก) ลำดับชั้นที่ 1-16

conv_pw_2_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_pw_2_relu (ReLU)	(None, 56, 56, 128)	0
conv_dw_3 (DepthwiseConv2D)	(None, 56, 56, 128)	1152
conv_dw_3_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_dw_3_relu (ReLU)	(None, 56, 56, 128)	0
conv_pw_3 (Conv2D)	(None, 56, 56, 128)	16384
conv_pw_3_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_pw_3_relu (ReLU)	(None, 56, 56, 128)	0
conv_pad_4 (ZeroPadding2D)	(None, 57, 57, 128)	0
conv_dw_4 (DepthwiseConv2D)	(None, 28, 28, 128)	1152
conv_dw_4_bn (BatchNormaliza	(None, 28, 28, 128)	512
conv_dw_4_relu (ReLU)	(None, 28, 28, 128)	0
conv_pw_4 (Conv2D)	(None, 28, 28, 256)	32768
conv_pw_4_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_pw_4_relu (ReLU)	(None, 28, 28, 256)	0
conv_dw_5 (DepthwiseConv2D)	(None, 28, 28, 256)	2304
conv_dw_5_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_dw_5_relu (ReLU)	(None, 28, 28, 256)	0

(ข) ลำดับชั้นที่ 17-34

conv_pw_5 (Conv2D)	(None, 28, 28, 256)	65536
conv_pw_5_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_pw_5_relu (ReLU)	(None, 28, 28, 256)	0
conv_pad_6 (ZeroPadding2D)	(None, 29, 29, 256)	0
conv_dw_6 (DepthwiseConv2D)	(None, 14, 14, 256)	2304
conv_dw_6_bn (BatchNormaliza	(None, 14, 14, 256)	1024
conv_dw_6_relu (ReLU)	(None, 14, 14, 256)	0
conv_pw_6 (Conv2D)	(None, 14, 14, 512)	131072
conv_pw_6_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_6_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_7 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_7_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_7_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_7 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_7_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_7_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_8 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_8_bn (BatchNormaliza	(None, 14, 14, 512)	2048

(ค) ลำดับชั้นที่ 35-52

conv_dw_8_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_8 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_8_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_8_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_9 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_9_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_9_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_9 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_9_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_9_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_10 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_10_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_dw_10_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_10 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_10_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_pw_10_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_11 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_11_bn (BatchNormaliz	(None, 14, 14, 512)	2048

(ง) ลำดับชั้นที่ 53-70

conv_dw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_11 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_11_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_pw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, 15, 15, 512)	0
conv_dw_12 (DepthwiseConv2D)	(None, 7, 7, 512)	4608
conv_dw_12_bn (BatchNormaliz	(None, 7, 7, 512)	2048
conv_dw_12_relu (ReLU)	(None, 7, 7, 512)	0
conv_pw_12 (Conv2D)	(None, 7, 7, 1024)	524288
conv_pw_12_bn (BatchNormaliz	(None, 7, 7, 1024)	4096
conv_pw_12_relu (ReLU)	(None, 7, 7, 1024)	0
conv_dw_13 (DepthwiseConv2D)	(None, 7, 7, 1024)	9216
conv_dw_13_bn (BatchNormaliz	(None, 7, 7, 1024)	4096
conv_dw_13_relu (ReLU)	(None, 7, 7, 1024)	0
conv_pw_13 (Conv2D)	(None, 7, 7, 1024)	1048576
conv_pw_13_bn (BatchNormaliz	(None, 7, 7, 1024)	4096
conv_pw_13_relu (ReLU)	(None, 7, 7, 1024)	0
global_average_pooling2d_2 (	(None, 1024)	0

(จ) ลำดับชั้นที่ 71-88

reshape_1 (Reshape)	(None, 1, 1, 1024)	0
dropout (Dropout)	(None, 1, 1, 1024)	0
conv_preds (Conv2D)	(None, 1, 1, 10)	10250
reshape_2 (Reshape)	(None, 10)	0
act_softmax (Activation)	(None, 10)	0

Total params: 3,239,114
Trainable params: 3,217,226
Non-trainable params: 21,888

(ฉ) ลำดับชั้นที่ 89-93

รูปที่ 3.19 ลำดับชั้นตามโครงสร้างสถาปัตยกรรม MobileNet

โดยผลจากการทดลองการฝึกฝนตามสถาปัตยกรรม MobileNetV2 ทำให้ได้โมเดลเพื่อใช้ในกาคัดแยกสิ่งปลอมปนที่อยู่ในน้ำมันมะพร้าว ซึ่งได้ถูกนำมาเปรียบเทียบเพื่อหาค่าประสิทธิภาพที่ดีที่สุดต่อไป

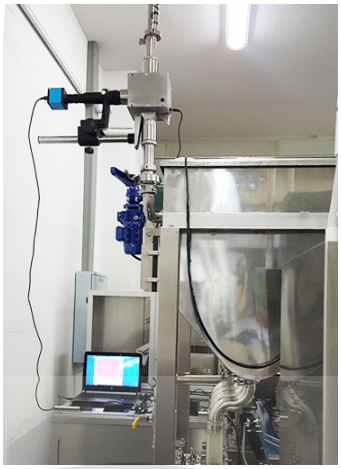
บทที่ 4
ผลการวิจัยและการอภิปรายผล

ในบทนี้ได้กล่าวถึงผลที่ได้จากการทดลองตรวจสอบความขุ่นและสิ่งปลอมปนในน้ำมันมะพร้าว โดยการทำงานได้แบ่งการดำเนินงานเพื่อหาค่าประสิทธิภาพออกเป็น 2 ส่วนหลัก ประกอบด้วยการหาประสิทธิภาพของวิธีการหาค่าความขุ่นในน้ำมันมะพร้าว และการหาค่าประสิทธิภาพในการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าว

4.1 ส่วนสภาพแวดล้อมในการทดลองและชุดข้อมูล

สำหรับการดำเนินการหาประสิทธิภาพของวิธีการหาค่าความขุ่นในน้ำมันมะพร้าวได้ใช้โปรแกรม MatLab 2018b ในการทดลองหาค่าประสิทธิภาพได้ใช้ชุดข้อมูลที่เรียกว่า Coconut Oil Determine Turbidity (CCODT) เพื่อทำการวัดค่าประสิทธิภาพของแต่ละขั้นตอนวิธีที่นำเสนอ จากนั้นจึงทำการพัฒนาเป็นโปรแกรมสำเร็จรูปสำหรับใช้งานตามขั้นตอนวิธีการหาค่าเฉลี่ยเคลื่อนที่จากความแข็งแรงของสี (Moving Average Median of Hue: MAMoH) โดยใช้ Microsoft Visual C# 2019 ร่วมกับ library EmguCV และ Aforge.Net ในการทดลองเพื่อหาค่าประสิทธิภาพในการใช้งาน สำหรับการหาค่าประสิทธิภาพในการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าว ได้ใช้โปรแกรม Python ร่วมกับ library CV, Keras และ Tensorflow ในการทดลองหาค่าประสิทธิภาพได้ใช้ชุดข้อมูลที่เรียกว่า PiCO_V1 สำหรับการตรวจจับสิ่งปลอมปนจากการเรียนรู้เชิงลึก โดยการทดลองทั้งหมดอยู่บนพื้นฐานของหน่วยประมวลผลกลาง Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz หน่วยความจำขนาด 16.0 GB บนระบบปฏิบัติการ Windows 10 แบบ 64 บิต และสำหรับการทดลองได้ใช้พื้นที่ของ บริษัท ทรอปิคานา ออยล์ จำกัด ในการทำงาน โดยติดตั้งอุปกรณ์ตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าวร่วมกับระบบสายการผลิตน้ำมันมะพร้าว ซึ่งตำแหน่งในการติดตั้งอุปกรณ์สำหรับตรวจสอบสิ่งปลอมปนในน้ำมันมะพร้าวนั้นแสดงดังรูปที่ 4.1

ในการทดลองได้ทำการบันทึกภาพของน้ำมันมะพร้าวที่ไหลผ่านอุปกรณ์ตรวจดูน้ำมันมะพร้าวในช่วงเวลาของการผลิตน้ำมันมะพร้าวที่แตกต่างกัน ซึ่งภาพที่บันทึกได้เหล่านี้จะถูกเรียกว่าชุดข้อมูล CCODT ที่มีภาพถ่ายความขุ่นของน้ำมันมะพร้าวจำนวน 1,409 ภาพ



รูปที่ 4.1 ตำแหน่งในการติดตั้งอุปกรณ์

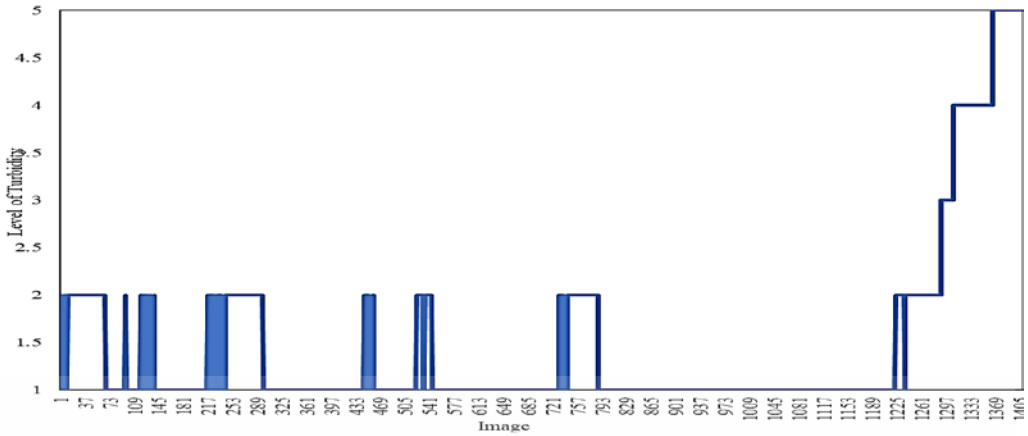
4.2 วิธีการทดลอง

สำหรับการทดลองได้นำชุดข้อมูล CCODT มาผ่านวิธีการที่นำเสนอ ซึ่งมีด้วยกัน 4 วิธีการคือ วิธีการหาค่าเฉลี่ยของภาพจากระดับสีเทา (MoGS) วิธีการหาค่าเฉลี่ยกึ่งกลางจากความแข็งแรงของสี (MoH) วิธีการสุ่มพื้นที่จากความแข็งแรงของสี (RPMoH) และวิธีการหาค่าเฉลี่ยเคลื่อนที่จากความแข็งแรงของสี (MAMoH) ซึ่งจากตารางที่ 3.1 เมื่อนำค่าสูงสุดและต่ำสุดมาทำการแบ่งชั้นของข้อมูลออกมาเป็น 5 ระดับ ได้ผลดังตารางที่ 4.1

ตารางที่ 4.1 เปรียบเทียบค่าความขุ่นของแต่ละระดับชั้นที่ได้จากวิธีการที่นำเสนอ

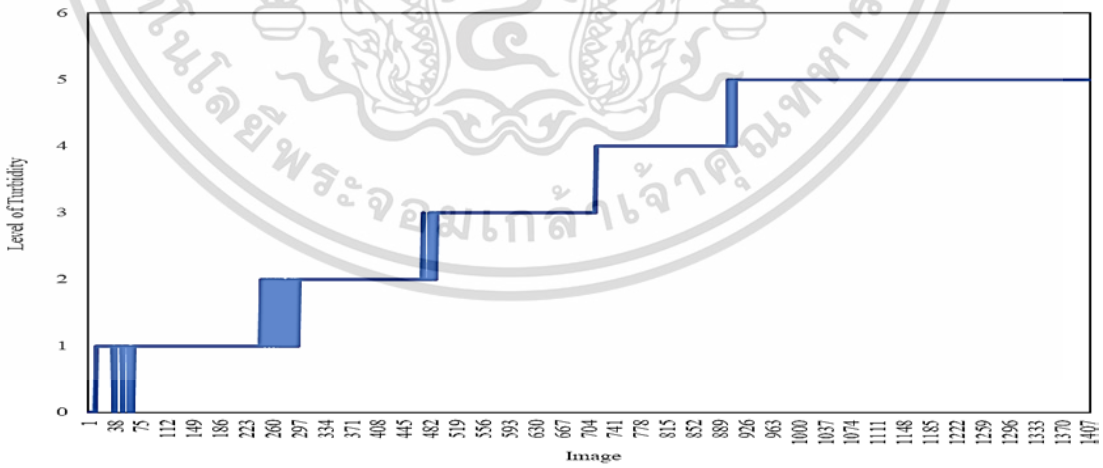
Level	Methods								Value
	MoGS		MoH		RPMoH		MAMoH		
	Min	Max	Min	Max	Min	Max	Min	Max	
5	0.7661	0.8260	0.8279	0.8337	0.8287	0.8352	0.8278	0.8336	Excellent
4	0.7061	0.7661	0.8220	0.8279	0.8223	0.8287	0.8220	0.8277	Good
3	0.6462	0.7061	0.8162	0.8220	0.8159	0.8223	0.8162	0.8220	Fair
2	0.5862	0.6462	0.8104	0.8162	0.8095	0.8159	0.8105	0.8162	Communicable
1	0.5263	0.5862	0.8045	0.8104	0.8031	0.8095	0.8048	0.8105	Bad

จากนั้นในครั้งแรกได้ทำการเปรียบเทียบค่าความขุ่นของน้ำมันมะพร้าวด้วยวิธีการ MoGS ซึ่งเป็นวิธีการหาค่าความขุ่นของน้ำมันมะพร้าวจากการแปลงภาพระดับเทา ในการทดลองใช้ชุดข้อมูล CCODT จำนวน 1,409 ภาพ โดยนำค่าความขุ่นของน้ำมันมะพร้าวที่ได้ในแต่ละภาพมาเปรียบเทียบกับค่าระดับความขุ่นของ MoGS ในตารางที่ 4.1 ผลลัพธ์ที่ได้จากชุดข้อมูล CCODT แสดงดังรูปที่ 4.2 โดยแกน x แทนจำนวนรูปภาพและแกน y แทนค่าเฉลี่ยของภาพระดับเทา



รูปที่ 4.2 ผลการเปรียบเทียบค่าความขุ่นของน้ำมันมะพร้าวกับค่าระดับความขุ่นของ MoGS

จากรูปที่ 4.2 เป็นการนำค่าความขุ่นที่ได้จากการประมวลผลภาพจากชุดข้อมูลภาพ CCODT มาเปรียบเทียบกับค่าระดับความขุ่นของน้ำมันมะพร้าว โดยจากผลการทดลองแสดงให้เห็นว่าการหาค่าความขุ่นจากวิธีการ MoGS ไม่สามารถจำแนกระดับความขุ่นของน้ำมันมะพร้าวได้อย่างชัดเจน ในครั้งที่สองได้ทำการเปรียบเทียบค่าความขุ่นของน้ำมันมะพร้าวด้วยวิธีการ MoH ซึ่งเป็นวิธีการที่อาศัยความแข็งแรงของสี ในการทดลองใช้ชุดข้อมูล CCODT จำนวน 1,409 ภาพ โดยนำค่าความขุ่นของน้ำมันมะพร้าวที่ได้ในแต่ละภาพนำมาเปรียบเทียบกับค่าระดับความขุ่นของ MoH ในตารางที่ 4.1 ผลลัพธ์ที่ได้จากชุดข้อมูล CCODT แสดงดังรูปที่ 4.3 โดยแกน x แทนจำนวนรูปภาพและแกน y แทนค่าความแข็งแรงของสี

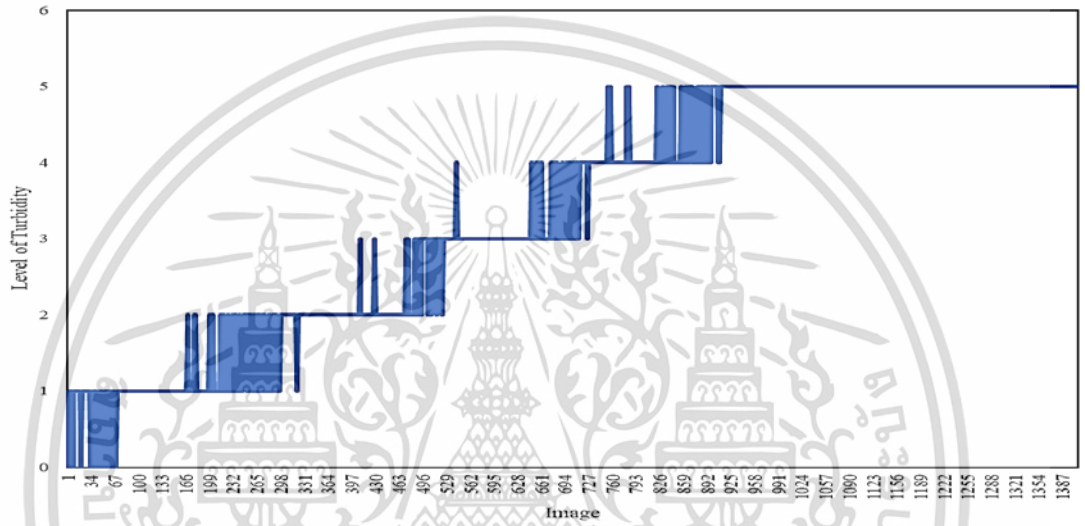


รูปที่ 4.3 ผลการเปรียบเทียบค่าความขุ่นของน้ำมันมะพร้าวกับค่าระดับความขุ่นของ MoH

จากรูปที่ 4.3 เป็นการนำค่าความขุ่นที่ได้จากการประมวลผลภาพจากชุดข้อมูลภาพ CCODT มาเปรียบเทียบกับค่าระดับความขุ่นของน้ำมันมะพร้าว โดยจากผลการทดลองแสดงให้เห็นว่าการหา

ค่าความขุ่นจากวิธีการ MoH ยังมีข้อผิดพลาดในการจำแนกระดับความขุ่นของน้ำมันมะพร้าวระหว่างค่าระดับ 1 และค่าระดับ 2

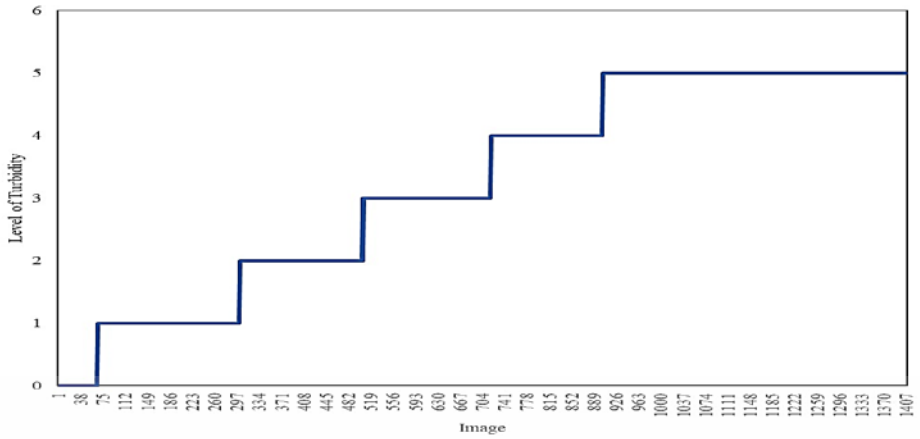
ต่อมาในครั้งที่สาม ได้ทำการเปรียบเทียบค่าความขุ่นของน้ำมันมะพร้าวด้วยวิธีการ RPMoH ซึ่งเป็นวิธีการที่อาศัยการสุ่มตำแหน่งที่อยู่ในภาพ จากนั้นอ่านค่าเฉลี่ยของความแข็งแรงของสี ในการทดลองใช้ชุดข้อมูล CCODT จำนวน 1,409 ภาพ โดยนำค่าความขุ่นของน้ำมันมะพร้าวที่ได้ในแต่ละภาพมาเปรียบเทียบกับค่าระดับความขุ่นของ RPMoH ในตารางที่ 4.1 ผลลัพธ์ที่ได้จากชุดข้อมูล CCODT แสดงดังรูปที่ 4.4 โดยแกน x แทนจำนวนรูปภาพและแกน y แทนค่าความแข็งแรงของสี



รูปที่ 4.4 ผลการเปรียบเทียบค่าความขุ่นของน้ำมันมะพร้าวกับค่าระดับความขุ่นของ RPMoH

จากรูปที่ 4.4 เป็นการนำค่าความขุ่นที่ได้จากการประมวลผลภาพจากชุดข้อมูลภาพ CCODT มาเปรียบเทียบกับค่าระดับความขุ่นของน้ำมันมะพร้าว โดยจากผลการทดลองแสดงให้เห็นว่าการหาค่าความขุ่นจากวิธีการ RPMoH พบข้อผิดพลาดในการจำแนกระดับความขุ่นของน้ำมันมะพร้าวมากกว่าวิธีการ MoH

สำหรับในการทดลองครั้งสุดท้าย ได้ทำการเปรียบเทียบค่าความขุ่นของน้ำมันมะพร้าวด้วยวิธีการ MAMoH ซึ่งเป็นวิธีการที่อาศัยการสุ่มตำแหน่งที่อยู่ในภาพจากนั้นอ่านค่าเฉลี่ยของความแข็งแรงของสี ในการทดลองใช้ชุดข้อมูล CCODT จำนวน 1,409 ภาพ โดยนำค่าความขุ่นของน้ำมันมะพร้าวที่ได้ในแต่ละภาพมาเปรียบเทียบกับค่าระดับความขุ่นของ MAMoH ในตารางที่ 4.1 ผลลัพธ์ที่ได้จากชุดข้อมูล CCODT แสดงดังรูปที่ 4.5 โดยแกน x แทนจำนวนรูปภาพและแกน y แทนค่าความแข็งแรงของสี

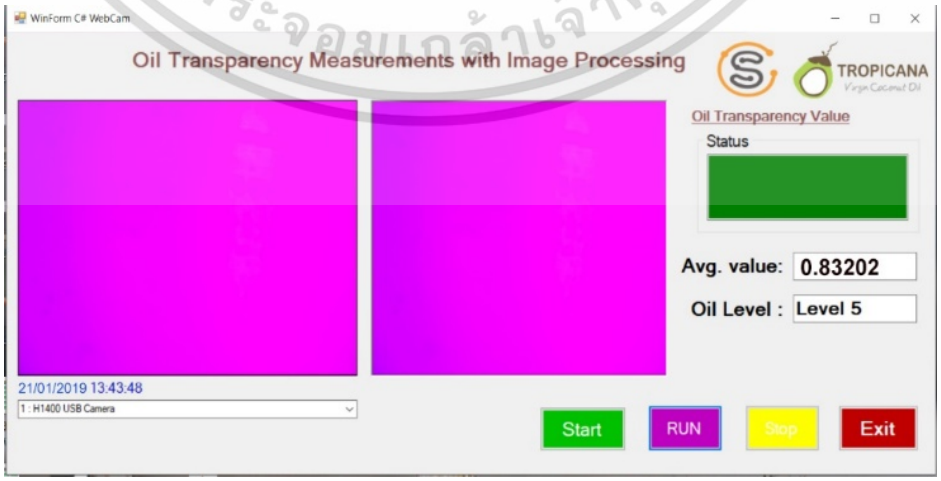


รูปที่ 4.5 ผลการเปรียบเทียบค่าความขุ่นของน้ำมันมะพร้าวกับค่าระดับความขุ่นของ MAMoH

จากรูปที่ 4.5 เป็นการนำค่าความขุ่นที่ได้จากการประมวลผลภาพจากชุดข้อมูลภาพ CCODT มาเปรียบเทียบกับค่าระดับความขุ่นของน้ำมันมะพร้าว โดยจากผลการทดลองแสดงให้เห็นว่าการหาค่าความขุ่นจากวิธีการ MAMoH มีค่าความแม่นยำกว่าเมื่อเปรียบเทียบกับอีก 3 วิธีการที่นำเสนอ ดังนั้นจึงเลือกวิธี MAMoH สำหรับการทดสอบครั้งแรกเพื่อใช้ในการทดสอบครั้งที่สองต่อไป

4.3 การทดลองใช้วิธี MAMoH ในกระบวนการผลิตน้ำมันมะพร้าว

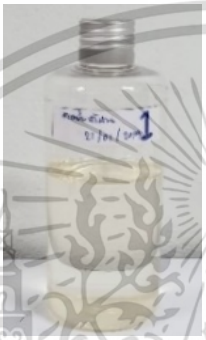
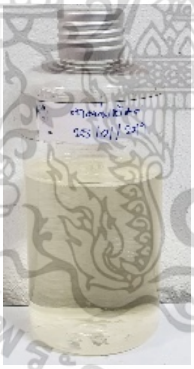
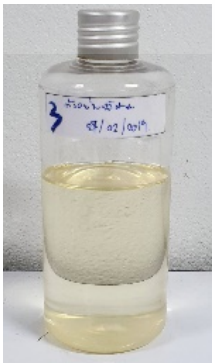
สำหรับการทดลองได้นำวิธีการ MAMoH มาทดลองใช้งานจริง เนื่องจากผลลัพธ์ที่ได้ของการทดลองจากชุดข้อมูล CCODT นั้นให้ผลลัพธ์ที่ดีกว่าวิธีการอื่น ในการแบ่งค่าระดับความขุ่นของน้ำมันมะพร้าว ซึ่งในการทดลองผู้วิจัยได้รับความอนุเคราะห์น้ำมันที่ผ่านกระบวนการเหวี่ยงน้ำมันมะพร้าวทั้งหมด 3 รอบการผลิตโดยในแต่ละรอบการผลิตต่างวันและเวลา จากนั้นผลที่ได้นำมาหาค่าประสิทธิภาพสำหรับการใช้งานจริง ผ่านโปรแกรมที่ถูกพัฒนาขึ้นตามวิธีการ MAMoH หน้าต่างโปรแกรมหาค่าระดับความขุ่นของน้ำมันมะพร้าว แสดงดังรูปที่ 4.6



รูปที่ 4.6 หน้าต่างการทำงานของโปรแกรมหาค่าระดับความขุ่นของน้ำมันมะพร้าว

ในการทดลองได้วิเคราะห์ภาพน้ำมันมะพร้าวที่ผ่านการแปรรูปและผ่านการกรอง โดยวิธีการหมุนเหวี่ยงเพื่อสกัดน้ำมันมะพร้าว โดยเริ่มต้นทำการทดลองใช้งานรอบแรกเมื่อวันที่ 21 มกราคม พ.ศ. 2562 รอบที่สองวันที่ 25 มกราคม พ.ศ. 2562 และรอบที่สามวันที่ 27 กุมภาพันธ์ พ.ศ. 2562 โดยตัวอย่างน้ำมันมะพร้าวในแต่ละรอบของการผลิตแสดงในตารางที่ 4.2

ตารางที่ 4.2 ข้อมูลการผลิตน้ำมันมะพร้าวในแต่ละรอบ

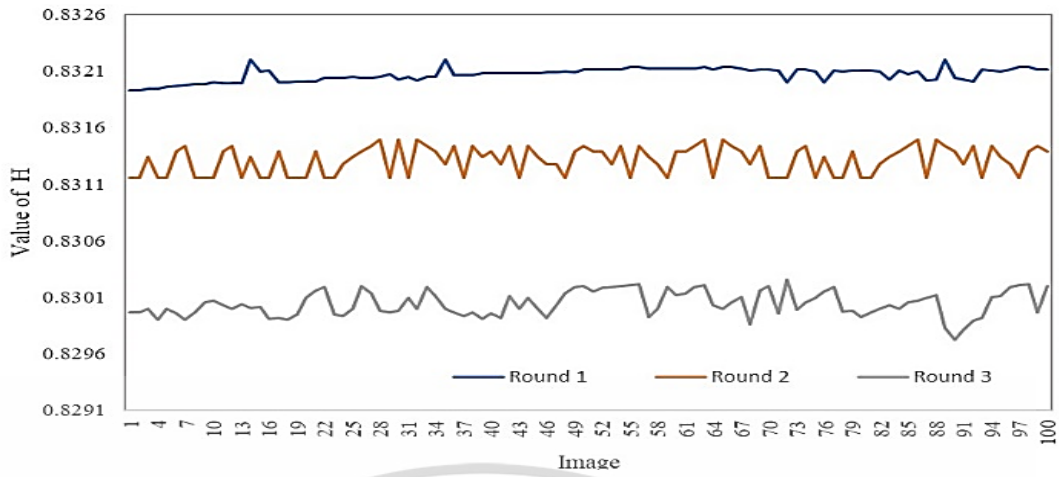
รอบการผลิต	ภาพตัวอย่างน้ำมันมะพร้าว	คำอธิบาย
รอบที่ 1 21 มกราคม พ.ศ. 2562		น้ำมันมะพร้าวที่ได้จากรอบการผลิตนี้นั้น มีสีใสเหมือนน้ำปกติ เนื่องจากในกระบวนการผลิตน้ำมันมะพร้าว มีส่วนของเปลือกกะลามะพร้าวผสมอยู่น้อยเท่านั้น ดังนั้นเมื่อใช้น้ำมันมะพร้าวผ่านการกรองน้ำมันมะพร้าวแบบละเอียดน้ำมันมะพร้าวจะใสเหมือนน้ำ
รอบที่ 2 25 มกราคม พ.ศ. 2562		น้ำมันมะพร้าวที่ได้จากรอบการผลิตนี้ มีสีเหลืองเพิ่มขึ้น เนื่องจากในกระบวนการผลิตน้ำมันมะพร้าวมีการเปลี่ยนแปลงไป โดยใช้เครื่องจักรขนาดใหญ่ช่วยแตกกะลามะพร้าวออก จึงส่งผลทำให้มีเศษกะลามะพร้าวเพิ่มขึ้นในกระบวนการผลิต และในทางกลับกันส่งผลทำให้มีปริมาณการผลิตน้ำมันมะพร้าวที่เพิ่มขึ้นเช่นกัน
รอบที่ 3 27 กุมภาพันธ์ พ.ศ. 2562		น้ำมันมะพร้าวที่ได้จากรอบการผลิตนี้ มีสีเหลืองมากกว่ารอบที่สอง เนื่องจากกระบวนการผลิตที่มีการให้ความร้อนกับเนื้อมะพร้าว เพื่อไล่ความชื้นของมะพร้าวที่มากเกินไป ซึ่งใช้เวลาที่นานกว่ารอบที่ 2 จึงทำให้สีของน้ำมันมะพร้าวเป็นสีเหลืองมากกว่ารอบที่ 2

การทดลองเริ่มต้นจากโปรแกรมทำการถ่ายภาพโดยใช้กล้องถ่ายภาพ ทุก ๆ 5 วินาที จากนั้นนำภาพถ่ายที่ได้ผ่านการประมวลผลตามวิธี MAMoH และเก็บบันทึกค่าความขุ่นของน้ำมันมะพร้าวไว้ โดยเวลาที่ใช้ในการทดลองแต่ละรอบอยู่ที่ 14 นาที ข้อมูลการประมวลผลภาพตามวิธี MAMoH จำนวน 3 รอบการผลิตนั้นแสดงค่าดังตารางที่ 4.3

ตารางที่ 4.3 ค่าความขุ่นของน้ำมันมะพร้าวในแต่ละรอบการผลิต

รอบการผลิตที่	ค่าต่ำสุด	ค่าสูงสุด
1	0.83198	0.83211
2	0.83114	0.83145
3	0.82996	0.83021

จากค่าในตารางที่ 4.3 แสดงให้เห็นว่าทั้ง 3 รอบของการทดลองและการบันทึกข้อมูลค่าระดับความขุ่นของน้ำมันมะพร้าวอยู่ที่ระดับ 5 ในตารางที่ 4.1 โดยในรอบแรกของการบันทึกผลการทดลองมีค่าอยู่ระหว่าง 0.83198 ถึง 0.83211 มีความขุ่นของน้ำมันมะพร้าวน้อยที่สุด ซึ่งเกิดจากกระบวนการผลิตที่มีสิ่งปลอมปน เช่น เปลือกด้านในกะลามะพร้าว หรือกากใยจากเปลือกมะพร้าวต่าง ๆ มากับเนื้อมะพร้าวน้อย ทำให้ได้น้ำมันนะพร้าวที่มีความขุ่นน้อยสำหรับในรอบที่สองจากการบันทึกผลการทดลองมีค่าอยู่ระหว่าง 0.83114 ถึง 0.83145 ซึ่งค่าที่ได้แสดงให้เห็นว่าในรอบการผลิตรอบนี้ มีสิ่งปลอมปนมากกว่ารอบแรก เมื่อพิจารณาด้วยตาเปล่าสังเกตเห็นได้ว่าน้ำมันเริ่มมีสีเหลืองอ่อนๆ ซึ่งสีของน้ำมันมะพร้าวที่เปลี่ยนแปลงไปมาจากกระบวนการผลิตที่มีสิ่งปลอมปน จำพวกเปลือกที่ติดกับเนื้อมะพร้าว หรือกากใยจากเปลือกมะพร้าวต่าง ๆ เข้ามาในกระบวนการผลิต และในรอบที่ 3 มีค่าอยู่ระหว่าง 0.82996 ถึง 0.83021 ในรอบการผลิตนี้ มีช่วงของผลที่กว้างกว่ารอบที่ 1 และรอบที่ 2 เนื่องจากเป็นรอบที่เปลี่ยนรูปแบบจากกระบวนการผลิตจากการใช้พนักงานขูดเนื้อมะพร้าวเป็นการใช้เครื่องกะเทาะเปลือกมะพร้าวแทนทำให้กระบวนการผลิตในการบีบคั้นน้ำมันจึงมีเปลือกมะพร้าวปลอมปนเข้าไปรวมกับน้ำมันมะพร้าวที่ออกมา ส่งผลให้สีของน้ำมันมะพร้าวที่ได้มีสีเหลืองอ่อนมากกว่ารอบที่ 2 และเมื่อทำการบันทึกค่าพบว่าวิธีการ MAMoH นั้นยังสามารถทำงานได้อยู่ในระดับเดียวกับรอบที่หนึ่งและรอบที่สอง โดยผลการเปรียบเทียบความขุ่นของน้ำมันมะพร้าวแสดงดังรูปที่ 4.7 โดยแกน x แทนจำนวนรูปภาพและแกน y แทนค่าความแข็งแรงของสี



รูปที่ 4.7 ผลการเปรียบเทียบผลการวัดค่าน้ำมันมะพร้าวในกระบวนการผลิตจำนวน 3 รอบ

สำหรับการตรวจวัดความขุ่นของน้ำมันมะพร้าวนั้นอาจมีได้หลากหลายวิธีการ ซึ่งวิธีการที่ผู้วิจัยได้นำเสนอนั้นเป็นเพียงวิธีการหนึ่งที่สามารถช่วยสร้างความเชื่อมั่นให้กับกระบวนการผลิตภายในโรงงานน้ำมันมะพร้าวได้ วิธีการ MAMoH นั้นสามารถแบ่งระดับความขุ่นของน้ำมันมะพร้าวได้ชัดเจนที่สุด เมื่อเปรียบเทียบกับวิธีการอื่น ๆ ที่ได้นำเสนอ และจากการทดลองร่วมกับระบบการผลิตน้ำมันมะพร้าวทั้ง 3 รอบ พบว่าวิธีการ MAMoH มีความคงทนต่อการเปลี่ยนแปลงของข้อมูลตั้งแต่เริ่มต้นจนมีการเปลี่ยนแปลงในขั้นตอนการผลิตน้ำมันมะพร้าวใหม่ ซึ่งวิธีการ MAMoH ยังสามารถแสดงผลลัพธ์ของคำตอบได้อย่างถูกต้อง

4.4 การวัดประสิทธิภาพของโมเดลในการจำแนกวัตถุในน้ำมันมะพร้าว

ในการทดสอบวัดค่าประสิทธิภาพการจำแนกวัตถุที่ปะปนอยู่ในน้ำมันมะพร้าว ได้ใช้ชุดข้อมูล PiCO_V1 สำหรับการวัดค่าประสิทธิภาพของโมเดลตามสถาปัตยกรรมต่าง ๆ ที่นำเสนอ โดยทำการทดลองหาค่าไฮเปอร์พารามิเตอร์ที่เหมาะสม จากนั้นทำการวัดค่าประสิทธิภาพของโมเดลจากการใช้ตาราง Confusion matrix

4.4.1 การทดลองค่าไฮเปอร์พารามิเตอร์

ในการทดลองเพื่อหาค่าไฮเปอร์พารามิเตอร์ที่เหมาะสมกับสถาปัตยกรรม MobileNet นั้น มีจำนวน 2 ไฮเปอร์พารามิเตอร์ที่เกี่ยวข้องกัน คือ ตัวคูณความกว้าง (Width Multipliers) และความละเอียดของภาพ (Resolution) ซึ่งผลลัพธ์ที่ได้จากการทดลองสามารถแสดงให้เห็นถึงความแตกต่างในการใช้ค่าไฮเปอร์พารามิเตอร์และแสดงให้เห็นถึงค่าประสิทธิภาพของสถาปัตยกรรม MobileNet ที่ได้จากชุดข้อมูล PiCO_V2 โดยผลจากการทดลองกำหนดค่าของความกว้างที่แตกต่างกันพบว่า ค่าของความกว้าง Width Multipliers นั้นส่งผลกระทบต่อค่าประสิทธิภาพความแม่นยำและขนาดของโมเดล ซึ่งแสดงให้เห็นได้อย่างชัดเจนในตารางที่ 4.4

ตารางที่ 4.4 ประสิทธิภาพ MobileNet บนตัวคูณความกว้างที่ต่างกัน (ความละเอียดคงที่ = 224)

ตัวคูณความกว้างที่ต่างกัน	ความแม่นยำ (%)	ขนาดพื้นที่จัดเก็บ (Mb)
1.00	94.05	12.52
0.75	88.72	8.38
0.50	71.54	5.60
0.25	74.25	3.88

โดยผลลัพธ์ในตารางที่ 4.4 แสดงให้เห็นถึงความสัมพันธ์กันระหว่างตัวคูณความกว้างและขนาดของโมเดล ซึ่งเมื่อทำการทดลองปรับค่าของตัวคูณความกว้างให้มีค่าลดลงส่งผลให้ขนาดของโมเดลนั้นมีค่าลดลงตามไปด้วยเช่นกัน และด้วยเหตุผลที่ทำให้ขนาดของโมเดลเล็กลง จึงส่งผลทำให้ค่าประสิทธิภาพความแม่นยำลดลงจาก 94.05% เหลือเพียง 71.54% อย่างต่อเนื่องตามไปด้วย

ในลำดับต่อมาได้ทำการทดลองเพื่อหาความแตกต่างจากไฮเปอร์พารามิเตอร์ความละเอียดของภาพเพื่อดูผลลัพธ์ที่แตกต่างกัน โดยนำผลลัพธ์ที่ดีที่สุดจากการทดสอบที่ได้ก่อนหน้าจากการกำหนดขนาดความละเอียดของภาพคงที่เท่ากับ 224x224 ซึ่งทำให้ได้ค่าตัวคูณความกว้างที่ดีที่สุดอยู่ที่ 1.0 จากนั้นจึงกำหนดค่าตัวคูณความกว้างคงที่อยู่ที่ 1.0 และทำการทดลองด้วยการใช้ค่าความละเอียดของภาพที่แตกต่างกัน โดยเริ่มต้นจากภาพในชุดข้อมูล PiCO_V2 ที่มีความละเอียดของภาพอยู่ที่ 224x224 จากนั้นทำการปรับขนาดของรูปภาพในชุดข้อมูล PiCO_V2 ให้มีความละเอียดของภาพน้อยลง โดยกำหนดความละเอียดของภาพอยู่ที่ 192x192, 160x160 และ 128x128 ตามลำดับ โดยการทดลองได้ใช้การฝึกโมเดลตามสถาปัตยกรรม MobileNet ตามค่าความละเอียดของภาพที่แตกต่างกัน โดยผลของความแม่นยำและขนาดพื้นที่จัดเก็บแสดงในตารางที่ 4.5

ตารางที่ 4.5 ประสิทธิภาพ MobileNet บนความละเอียดของภาพที่ต่างกัน (ความกว้างคงที่ = 1.0)

ความละเอียดของภาพ	ความแม่นยำ (%)	ขนาดพื้นที่จัดเก็บ (Mb)
224 x 224	94.05	12.52
192 x 192	93.40	12.52
160 x 160	92.56	12.52
128 x 128	91.22	12.52

จากค่าในตารางที่ 4.5 แสดงให้เห็นว่าการปรับความละเอียดของภาพนั้น มีผลกระทบต่อความแม่นยำน้อยมาก โดยค่าความแม่นยำจากการกำหนดตัวคูณความกว้างคงที่ 1.0 นั้น มีค่าความแม่นยำอยู่ระหว่าง 94.05% ถึง 91.22% ซึ่งหลังจากการทดลองการปรับขนาดความละเอียด

ของภาพที่แตกต่างกัน พบว่าความละเอียดของภาพแต่ละขนาดนั้น ไม่ส่งผลกระทบต่อขนาดของพื้นที่ในการจัดเก็บของโมเดล จากนั้นเมื่อนำค่าไฮเปอร์พารามิเตอร์ทั้ง 2 ค่า ที่ได้จากตารางที่ 4.4 และตารางที่ 4.5 มารวมกัน เพื่อหาค่าไฮเปอร์พารามิเตอร์ที่เหมาะสมในการใช้งาน ได้ดังตารางที่ 4.6

ตารางที่ 4.6 ความแม่นยำของ MobileNet กับตัวคูณความกว้างและความละเอียดของภาพ

ความละเอียดของภาพ	ตัวคูณความกว้าง			
	1.00	0.75	0.50	0.25
224 x 224	94.05	88.72	71.54	74.25
192 x 192	93.40	88.77	72.35	74.44
160 x 160	92.56	88.21	71.23	72.87
128 x 128	91.22	87.83	70.96	73.23

ซึ่งจากตารางที่ 4.6 แสดงผลลัพธ์ที่เหมาะสมสำหรับการกำหนดค่าไฮเปอร์พารามิเตอร์ของตัวคูณความกว้างที่ 1.00 และความละเอียดของภาพที่เหมาะสมอยู่ที่ความละเอียด 224 x 224 สำหรับชุดข้อมูล PiCO_V2

4.4.2 เปรียบเทียบประสิทธิภาพความแม่นยำจากการฝึกโมเดลตามสถาปัตยกรรม CNN

สำหรับการทดลองในส่วนนี้ เป็นการทดลองเพื่อเปรียบเทียบกับสถาปัตยกรรมของโมเดลที่อยู่บนพื้นฐานสถาปัตยกรรม CNN เดียวกัน โดยกำหนดตามค่าไฮเปอร์พารามิเตอร์ที่ดีที่สุด ที่ได้จากการทดลองก่อนหน้านี้ คือ ค่าตัวคูณความกว้างและค่าความละเอียดของภาพ โดยกำหนดค่าตัวคูณความกว้างเท่ากับ 1.0 และค่าความละเอียดของภาพเท่ากับ 224 x 224 จุดภาพ และสำหรับการทดสอบแบ่งเป็นการนำเข้าข้อมูลโดยไม่มีการปรับขนาด resolution 224 x 224 ซึ่งมีสถาปัตยกรรมโมเดล (Standard CNN, VGG16, VGG19, ResNet50, ResNet101 และ DenseNet121) และการปรับขนาดภาพที่ค่าความละเอียดของภาพเท่ากับ 299 x 299 ตามข้อกำหนดของข้อมูลนำเข้าของโมเดล (Xception, InceptionV3, InceptionResNetV2) โดยทำการฝึกโมเดลทั้งหมดบนชุดข้อมูล PiCO_V2 และกำหนดรอบการฝึกโมเดลที่ epoch = 15 รอบ ผลลัพธ์ที่แสดงในตารางที่ 4.7

ตารางที่ 4.7 ผลของประสิทธิภาพความแม่นยำจากการฝึกอบรมโมเดล ที่ epoch = 15 รอบ

No	model	Input resolution	Accuracy (%)	Training time (hour)	Test Time (minute)	Size Model (Mb)
1	MobileNetV2	224x224x3	**94.05**	5.13	18.20	12.52
2	Standard CNN	224x224x3	89.70	1.17	18.35	8.27
3	Xception	299x299x3	92.82	30.08	22.44	842.33
4	InceptionV3	299x299x3	93.23	15.38	20.22	96.88
5	ResNet50	224x224x3	91.65	16.04	19.27	689.54
6	ResNet101	224x224x3	90.21	25.42	19.37	795.23
7	DenseNet121	224x224x3	93.53	21.47	20.10	827.27
8	VGG16	224x224x3	91.95	8.39	18.56	689.41
9	VGG19	224x224x3	91.12	12.19	19.55	722.58
10	InceptionResNetV2	299x299x3	90.93	30.25	20.37	956.36

โดยผลลัพธ์ในตารางที่ 4.7 แสดงให้เห็นว่าสถาปัตยกรรม MobileNetsV2 นั้นมีประสิทธิภาพด้านความแม่นยำเหนือกว่าสถาปัตยกรรม CNN อื่น โดยความแม่นยำของ MobileNetV2 อยู่ที่ 94.05% ตามด้วย DenseNet121, InceptionV3, Xception, VGG16, ResNet50, VGG19, InceptionResNetV2, ResNet101 และ Standard CNN ที่ความแม่นยำที่ 93.53%, 93.23%, 92.82%, 91.95%, 91.65%, 91.12%, 90.93%, 90.21% และ 89.70% ตามลำดับ ซึ่งผลลัพธ์ที่ได้ยังแสดงให้เห็นถึงผลที่ดีกว่าของ MobileNetV2 ที่สามารถทำได้ดีกว่า Standard CNN อยู่ประมาณ 4.35 %

สำหรับการเปรียบเทียบขนาดพื้นที่ของโมเดล (Model size) ที่ได้หลังจากการฝึกโมเดลสถาปัตยกรรมโมเดลแบบ MobileNetV2 นั้นเป็นสถาปัตยกรรมโมเดลที่ต้องการขนาดของพื้นที่ในการจัดเก็บโมเดลที่เล็กที่สุด เพื่อเป็นการประหยัดทรัพยากรของเครื่องคอมพิวเตอร์หรือเป็นโมเดลที่เหมาะสำหรับอุปกรณ์ที่มีการประมวลผลข้อมูลไม่สูงมาก โดยมีขนาดของโมเดล MobileNetV2 อยู่ที่ 12.52 Mb ซึ่งน้อยกว่า DenseNet121 และ InceptionV3 อยู่ประมาณ 66 เท่า และ 7.7 เท่า ตามลำดับ ส่วนขนาดของ VGG16 และ VGG19 นั้น ค่าประสิทธิภาพความแม่นยำของ VGG16 สามารถแสดงค่าความแม่นยำที่สูงกว่าค่าประสิทธิภาพความแม่นยำของ VGG19 แต่ทั้ง VGG16 และ VGG19 นั้นยังต้องการขนาดของพื้นที่ในการจัดเก็บโมเดลที่มากกว่า MobileNetV2 ทั้งคู่ ซึ่งนับเป็นปัญหาแบบเดียวกันกับปัญหาของสถาปัตยกรรมแบบ DenseNet, Inception, Xception และ ResNet ที่มีขนาดของพื้นที่ในการจัดเก็บโมเดลที่มากกว่า แต่มีความแม่นยำของโมเดลที่น้อยกว่าสถาปัตยกรรม MobileNetV2

ส่วนผลลัพธ์ของเวลาที่ใช้ในการฝึกโมเดลในตารางที่ 4.7 เป็นผลลัพธ์ที่แสดงให้เห็นถึงเวลาที่ถูกใช้ในการฝึกโมเดลทั้งหมดที่ epoch = 15 รอบ โดยวิธีการ Standard CNN เป็นโมเดลที่ใช้

เวลาในการฝึกโมเดลน้อยที่สุด โดยใช้เวลาฝึกโมเดลประมาณ 1.17 ชั่วโมง และอันดับ 2 คือ MobileNetV2 ใช้เวลาฝึกโมเดลประมาณ 5.13 ชั่วโมง อันดับ 3 คือ VGG16 ใช้เวลาฝึกโมเดลประมาณ 8.39 ชั่วโมง อันดับ 4 คือ VGG19 ใช้เวลาฝึกโมเดลประมาณ 12.19 ชั่วโมง อันดับ 5 InceptionV3 ใช้เวลาฝึกโมเดลประมาณ 15.38 ชั่วโมง อันดับ 6 ResNet50 ใช้เวลาฝึกโมเดลประมาณ 16.04 ชั่วโมง อันดับ 7 DenseNet121 ใช้เวลาฝึกโมเดลประมาณ 21.47 ชั่วโมง อันดับ 8 ResNet101 ใช้เวลาฝึกโมเดลประมาณ 25.42 ชั่วโมง อันดับ 9 Xception ใช้เวลาฝึกโมเดลประมาณ 30.08 ชั่วโมง และอันดับสุดท้าย InceptionResNetV2 ใช้เวลาฝึกโมเดลประมาณ 30.25 ชั่วโมง โดยจากผลลัพธ์ที่แสดงในตารางที่ 4.7 แสดงให้เห็นว่า MobileNetV2 นั้น มีประสิทธิภาพเหนือกว่าโมเดลอื่นที่ได้นำเสนอมา สำหรับโมเดล MobileNetV2 นั้นสามารถจดจำและแยกประเภทของสิ่งปลอมปนในน้ำมันมะพร้าวได้ดีที่สุด

4.4.3 การวัดประสิทธิภาพด้วยตารางเมทริกซ์ความสับสน (Confusion matrix)

สำหรับการหาค่าประสิทธิภาพโดยใช้ตาราง Confusion matrix นั้น เป็นเครื่องมือสำคัญในการประเมินผลลัพธ์ของการทำนาย (Prediction) ที่ได้ค่าคำตอบจากโมเดล ที่ถูกสร้างขึ้นตามสถาปัตยกรรมต่าง ๆ ที่ได้นำเสนอ ซึ่งเป็นการวัดจากค่าคำตอบที่ได้จากโมเดล เปรียบเทียบกับคำตอบที่ถูกต้อง Confusion matrix แสดงดังตารางที่ 4.8

ตารางที่ 4.8 ตาราง Confusion matrix

Actual Values	Predicted Values	
	Positive (1)	**Negative (0)**
Positive (1)	True Positive (TP)	False Negative (FN)
Negative (0)	False Positive (FP)	True Negative (TN)

โดยค่าในตาราง confusion matrix นั้นจะถูกนำมาใช้ในการหาประสิทธิภาพของแบบโมเดลสำหรับการจัดกลุ่ม (Classification model) ซึ่งสามารถคำนวณได้จากสมการ (4.1) – (4.4)

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}} \tag{4.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{4.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{4.3}$$

$$\mathrm{F1_{Score}} = 2 \times \left[\frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}\right] \quad (4.4)$$

โดยที่

TP คือ True Positive ซึ่งหมายถึงข้อมูลที่การทำนายตรงกับข้อมูลจริงใน Class ที่กำลังพิจารณา

FP คือ False Positive ซึ่งหมายถึง ข้อมูลที่การทำนายผิดใน Class ที่กำลังพิจารณา

TN คือ True Negative หมายถึง ข้อมูลที่ทำนายตรงกับข้อมูลจริงใน Class ที่ไม่ได้พิจารณา

FN คือ False Negative ซึ่งหมายถึง ข้อมูลที่ทำนายผิดใน Class ที่ไม่ได้พิจารณา

จากสมการที่ (4.1) เป็นสมการที่ใช้ในการหาค่า Accuracy ซึ่งหมายถึง จำนวนครั้งที่ข้อมูลการทำนายตรงกับข้อมูลจริง นำมารวมกับข้อมูลการทำนายตรงกับข้อมูลจริงในกลุ่ม (Class) ที่ไม่ได้พิจารณา ซึ่งเมื่อนำมาหารด้วยจำนวนข้อมูลทั้งหมดทำให้ได้ค่าประสิทธิภาพของการทำนาย Accuracy ออกมา ค่า Precision สามารถคำนวณได้จากสมการที่ (4.2) หมายถึงจำนวนครั้งที่ข้อมูลการทำนายตรงกับข้อมูลจริง จากนั้นนำมาหารจำนวนครั้งที่ข้อมูลการทำนายตรงกับข้อมูลจริง รวมกับข้อมูลที่การทำนายผิดในกลุ่มที่กำลังพิจารณา ซึ่งค่า Precision นั้นสามารถเรียกอีกชื่อว่าค่าพยากรณ์เชิงบวกหรือ Positive Predictive Value (PPV) ค่า Recall สามารถคำนวณได้จาก สมการที่ (4.3) หมายถึง จำนวนครั้งที่ข้อมูลการทำนายตรงกับข้อมูลจริง จากนั้นนำค่าจำนวนที่ได้มาหารด้วยจำนวนครั้งที่ข้อมูลการทำนายตรงกับข้อมูลจริงรวมกับข้อมูลที่ทำนายผิดใน กลุ่มที่ไม่ได้พิจารณา และสุดท้ายค่า $\mathrm{F1_{Score}}$ สามารถคำนวณได้จากสมการที่ (4.4) หมายถึง ค่าที่ได้จากการนำค่า Precision และ Recall มาคำนวนรวมกัน ซึ่ง $\mathrm{F1_{Score}}$ ถูกสร้างขึ้นมาเพื่อเป็น Single metric ที่วัดความสามารถของโมเดล โดยคิดเป็นค่าเฉลี่ยแบบฮาร์มอนิก (Harmonic mean) ระหว่าง Precision และ Recall โดยทั้ง 4 ค่าที่ได้กล่าวมานั้น ถูกนำมาใช้ในการหาค่าประสิทธิภาพกับทุกแบบโมเดล ของสถาปัตยกรรม CNN ที่ได้นำเสนอ

สำหรับผลลัพธ์ที่ได้จากการฝึกโมเดลตามสถาปัตยกรรม CNN ที่ได้จากการทดลองก่อนหน้า ได้ถูกนำมาใช้ในการทดลองส่วนนี้เพื่อหาค่าประสิทธิภาพความแม่นยำ จากการใช้ข้อมูลภาพของชุดข้อมูล PiCO_V2 ที่มีขนาดความละเอียดของภาพที่ตรงตามข้อกำหนดของแต่ละสถาปัตยกรรมแบบ CNN โดยแบ่งออกเป็น 9 การทดลองดังต่อไปนี้

1) การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม MobileNetV2

การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม MobileNetV2 เริ่มต้นจากการนำชุดข้อมูล PiCO_V1 มาใช้ในการทดสอบ ซึ่งมีจำนวนข้อมูล 1,000 ภาพ โดยเป็นข้อมูลภาพที่ทราบ

ถึงผลลัพธ์ของคำตอบล่วงหน้าก่อนแล้ว เมื่อนำข้อมูลภาพทั้งหมดมาผ่านการทำนายข้อมูลภาพด้วยสถาปัตยกรรม MobileNetV2 ผลลัพธ์ที่ได้บันทึกด้วยตาราง Confusion matrix ดังตารางที่ 4.9

ตารางที่ 4.9 ตาราง Confusion matrix ที่ได้จากสถาปัตยกรรม MobileNetV2

Class	Air	FiberT1	FiberT2	FiberT3	FiberT4	ParticleT1	ParticleT2	ParticleT3	ParticleT4	Tissue
Air	110	3	3	0	4	0	9	1	5	0
FiberT1	3	154	10	2	2	5	6	0	4	0
FiberT2	0	1	66	0	0	1	3	0	1	0
FiberT3	4	10	3	80	4	1	6	0	2	1
FiberT4	1	9	4	3	47	3	6	0	3	0
ParticleT1	3	2	0	0	0	46	5	2	2	3
ParticleT2	3	3	2	0	0	0	79	4	0	0
ParticleT3	2	1	1	0	0	0	6	50	0	4
ParticleT4	2	0	0	0	0	0	0	0	84	0
Tissue	6	5	2	1	0	1	13	2	0	86

จากตารางที่ 4.9 ผลลัพธ์ของการทำนายพบว่า สถาปัตยกรรมโมเดลแบบ MobileNetV2 นั้นมีกลุ่มของ FiberT1 ที่มีจำนวนผิดพลาดมากที่สุด โดยทำนายผิดพลาดจำนวน 32 ภาพ จากข้อมูลภาพทั้งหมด 186 ภาพ และรองลงมาคือ กลุ่มของ Tissue ทำนายผิดพลาดจำนวน 30 ภาพ จากข้อมูลภาพจำนวน 116 ภาพ ซึ่งข้อผิดพลาดนั้นมาจากรูปทรงของวัตถุที่มีความคล้ายคลึงกันของภาพ โดยตัวอย่างของภาพผลลัพธ์ที่ให้คำตอบผิดพลาดแสดงดังรูปที่ 4.8

(ก) FiberT1 (ข) FiberT2

รูปที่ 4.8 ตัวอย่างสิ่งปลอมปนในน้ำมันมะพร้าวที่ให้คำตอบระหว่าง FiberT1 และ FiberT2

โดยจากผลในตาราง Confusion matrix ถูกนำมาคำนวณหาค่าประสิทธิภาพความแม่นยำของแบบจำลองโมเดล ในการจัดกลุ่มภาพของชุดข้อมูล PiCO_V1 ด้วยสมการที่ (4.1) - (4.4) เพื่อหาค่าความถูกต้อง (Accuracy) ค่าความแม่นยำ (Precision) ค่าความไว (Recall) และค่า $\mathbf{F1_{Score}}$ ซึ่งประสิทธิภาพของ MobileNetV2 นั้นมีค่า Accuracy 80.20% ค่า Precision 71.61% ค่า Recall 81.15% และ ค่า $\mathbf{F1_{Score}}$ 76.08% โดยผลการทดลองทั้งหมดที่ได้แสดงในตารางที่ 4.10

ตารางที่ 4.10 ผลลัพธ์ความแม่นยำของโมเดลจากการใช้ Confusion matrix บนชุดข้อมูล PiCO_V1

No	Model	Image input	Test model with Confusion matrix from PiCO_V1 dataset (%)			
			Accuracy	Precision	Recall	$F1_{Score}$
1	MobileNetV2	224x224x3	**80.20**	**71.61**	**81.15**	**76.08**
2	Standard CNN	224x224x3	54.82	45.09	61.01	51.86
3	Xception	299x299x3	72.95	64.62	76.24	69.95
4	InceptionV3	299x299x3	75.13	68.19	77.14	72.39
5	ResNet50	224x224x3	68.88	62.12	72.61	66.95
6	ResNet101	224x224x3	66.44	66.56	67.29	66.92
7	DenseNet121	224x224x3	68.55	62.75	69.38	65.90
8	VGG16	224x224x3	68.52	60.55	73.02	66.20
9	VGG19	224x224x3	70.02	69.38	71.89	68.58
10	InceptionResNetV2	299x299x3	55.55	55.25	58.79	55.44

2) การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม Xception

การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม Xception เริ่มต้นจากการนำชุดข้อมูล PiCO_V1 มาใช้ในการทดสอบ ซึ่งมีจำนวนข้อมูล 1,000 ภาพ โดยเป็นข้อมูลภาพที่ทราบถึงผลลัพธ์ของคำตอบล่วงหน้าก่อนแล้ว เมื่อนำข้อมูลภาพทั้งหมดมาผ่านการทำนายข้อมูลภาพด้วยสถาปัตยกรรม Xception ผลลัพธ์ที่ได้บันทึกด้วยตาราง Confusion matrix ดังตารางที่ 4.11

ตารางที่ 4.11 ตาราง Confusion matrix ที่ได้จากสถาปัตยกรรม Xception

Class	Air	FiberT1	FiberT2	FiberT3	FiberT4	ParticleT1	ParticleT2	ParticleT3	ParticleT4	Tissue
Air	110	0	0	0	2	2	17	0	4	0
FiberT1	3	140	12	0	0	19	4	1	5	2
FiberT2	0	2	64	0	0	0	6	0	0	0
FiberT3	7	16	8	59	0	5	11	0	5	0
FiberT4	1	25	2	1	36	1	7	0	2	1
ParticleT1	4	1	0	0	0	40	5	5	2	6
ParticleT2	6	4	1	0	0	1	75	3	0	1
ParticleT3	3	4	0	0	0	0	13	40	0	4
ParticleT4	1	0	0	1	0	0	0	0	84	0
Tissue	16	3	0	0	0	3	11	2	0	81

จากตารางที่ 4.11 แสดงผลลัพธ์ของความผิดพลาดที่เกิดขึ้นเมื่อรูปภาพถูกทำนายว่าเป็น FiberT1 แทนที่ของคำตอบ FiberT3 โดยในกรณีนี้ มีการทำนายผิดพลาดจำนวน 52 ภาพ และข้อผิดพลาดอีกส่วนหนึ่งเกิดขึ้นมาจากการทำนายวัตถุในกลุ่ม FiberT2 และ ParticleT1 แทนคำตอบที่ถูกต้องโดยเป็นกลุ่ม FiberT1 ซึ่งมีจำนวนรูปภาพที่ไม่ถูกต้องจำนวน 12 รูป จากกลุ่ม FiberT2 และรูปภาพที่ไม่ถูกต้อง 19 รูปจากกลุ่ม ParticleT1 จึงส่งผลให้ผลลัพธ์ที่ได้จากการทำนายรูปภาพผิดจำนวน 271 รูป หรืออัตราข้อผิดพลาดเท่ากับ 27.05% โดยตัวอย่างของภาพผลลัพธ์ที่ให้คำตอบผิดพลาดแสดงดังรูปที่ 4.9

(ก) FiberT1 (ข) FiberT3

รูปที่ 4.9 ตัวอย่างสิ่งปลอมปนในน้ำมันมะพร้าวที่ให้คำตอบระหว่าง FiberT1 และ FiberT3

เมื่อนำผลลัพธ์ที่ได้มาหาค่าคำตอบจากการคำนวณหาค่า Accuracy ค่า Precision ค่า Recall และ ค่า $\mathbf{F1_{Score}}$ ผลลัพธ์ที่ได้คือ 72.95%, 64.62%, 76.24%, 69.95% ตามลำดับ โดยผลที่ได้ถูกแสดงในตารางที่ 4.10

3) การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม InceptionV3

การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม InceptionV3 เริ่มต้นจากการนำชุดข้อมูล PiCO_V1 มาใช้ในการทดสอบ ซึ่งมีจำนวนข้อมูล 1,000 ภาพ โดยเป็นข้อมูลภาพที่ทราบถึงผลลัพธ์ของคำตอบล่วงหน้าก่อนแล้ว เมื่อนำข้อมูลภาพทั้งหมดมาผ่านการทำนายข้อมูลภาพด้วยสถาปัตยกรรม InceptionV3 ผลลัพธ์ที่ได้บันทึกด้วยตาราง Confusion matrix ดังตารางที่ 4.12

ตารางที่ 4.12 ตาราง Confusion matrix ที่ได้จากสถาปัตยกรรม InceptionV3

Class	Air	FiberT1	FiberT2	FiberT3	FiberT4	ParticleT1	ParticleT2	ParticleT3	ParticleT4	Tissue
Air	108	2	0	0	0	0	22	0	3	0
FiberT1	3	127	31	0	6	5	5	1	7	1
FiberT2	0	0	64	2	1	0	3	0	1	1
FiberT3	4	17	6	58	6	2	7	2	9	0
FiberT4	0	6	2	0	60	0	0	0	8	0
ParticleT1	2	2	0	0	0	42	1	3	5	8
ParticleT2	1	3	1	0	0	0	78	3	0	5
ParticleT3	3	2	0	0	0	1	3	51	0	4
ParticleT4	1	0	0	0	0	0	0	0	85	0
Tissue	19	2	1	0	1	1	11	1	2	78

จากตารางที่ 4.12 แสดงผลลัพธ์ของความผิดพลาดที่เกิดขึ้นของแบบจำลองโมเดล InceptionV3 เมื่อนำมาทำนายภาพเพื่อจัดกลุ่มรูปภาพ โดยโมเดล InceptionV3 ทำนายภาพผิดพลาดจำนวน 53 ภาพในกลุ่ม FiberT1 ซึ่งอันที่จริงแล้วควรอยู่ในกลุ่ม FiberT3 ตัวอย่างของภาพผลลัพธ์ที่ให้คำตอบผิดพลาดในรูปที่ 4.9 และโมเดล InceptionV3 ทำนายภาพผิดพลาดจำนวน 59 ภาพ ในกลุ่มของ FiberT2 ควรอยู่ในกลุ่ม FiberT1 ซึ่งเมื่อรวมกับความผิดพลาดในการจำแนกกลุ่มอื่นในชั้นอากาศ อนุภาค และเนื้อเยื่อ ได้รูปภาพทั้งหมด 249 ภาพถูกที่จำแนกประเภทผิด ซึ่งมีอัตราความผิดพลาด 24.87% เมื่อนำผลลัพธ์ที่ได้มาหาค่าคำตอบจากการคำนวณหาค่า Accuracy ค่า Precision ค่า Recall และ ค่า $\mathbf{F1}_{\mathbf{Score}}$ ผลลัพธ์ที่ได้คือ 75.13%, 68.19%, 77.14% และ 72.39% ตามลำดับ โดยผลที่ได้ถูกแสดงในตารางที่ 4.10

4) การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม ResNet50

การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม ResNet50 เริ่มต้นจากการนำชุดข้อมูล PiCO_V1 มาใช้ในการทดสอบ ซึ่งมีจำนวนข้อมูล 1,000 ภาพ โดยเป็นข้อมูลภาพที่ทราบถึงผลลัพธ์ของคำตอบล่วงหน้าก่อนแล้ว เมื่อนำข้อมูลภาพทั้งหมดมาผ่านการทำนายข้อมูลภาพด้วยสถาปัตยกรรม ResNet50 ผลลัพธ์ที่ได้บันทึกด้วยตาราง Confusion matrix ดังตารางที่ 4.13

ตารางที่ 4.13 ตาราง Confusion matrix ที่ได้จากสถาปัตยกรรม ResNet50

Class	Air	FiberT1	FiberT2	FiberT3	FiberT4	ParticleT1	ParticleT2	ParticleT3	ParticleT4	Tissue
Air	101	1	0	0	0	0	19	0	14	0
FiberT1	1	116	31	6	0	8	18	1	5	0
FiberT2	1	1	59	1	0	2	7	0	1	0
FiberT3	5	13	4	58	4	5	11	0	9	2
FiberT4	1	7	4	2	46	3	5	0	6	2
ParticleT1	3	2	0	0	1	37	10	5	3	2
ParticleT2	5	0	1	1	0	2	77	3	0	2
ParticleT3	3	0	0	0	0	0	14	46	0	1
ParticleT4	1	0	0	0	0	0	0	0	85	0
Tissue	24	4	1	0	0	2	19	3	0	63

จากตารางที่ 4.13 แสดงผลลัพธ์ของความผิดพลาดที่เกิดขึ้นของแบบจำลองโมเดล ResNet50 โดยโมเดล ResNet50 พบว่ามีการทำนายภาพผิดพลาดในกลุ่ม FiberT1 จำนวน 31 ภาพ และมีการทำนายภาพผิดพลาดในกลุ่ม FiberT2 จำนวน 18 ภาพ ซึ่งเมื่อนำมารวมกับภาพที่มีการทำนายผิดกลุ่มมีทั้งหมด 312 ภาพ คิดเป็นอัตราความผิดพลาดที่ 31.12% ซึ่งเมื่อนำผลลัพธ์ที่ได้มาหาค่าคำตอบจากการคำนวณหาค่า Accuracy ค่า Precision ค่า Recall และ ค่า $\mathbf{F1}_{\mathbf{Score}}$ ผลลัพธ์ที่ได้คือ 68.88%, 62.12%, 72.61% และ 66.95% ตามลำดับ โดยผลที่ได้ถูกแสดงในตารางที่ 4.10

5) การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม ResNet101

การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม ResNet101 เริ่มต้นจากการนำชุดข้อมูล PiCO_V1 มาใช้ในการทดสอบ ซึ่งมีจำนวนข้อมูล 1,000 ภาพ โดยเป็นข้อมูลภาพที่ทราบถึงผลลัพธ์ของคำตอบล่วงหน้าก่อนแล้ว เมื่อนำข้อมูลภาพทั้งหมดมาผ่านการทำนายข้อมูลภาพด้วยสถาปัตยกรรม ResNet101 ผลลัพธ์ที่ได้บันทึกด้วยตาราง Confusion matrix ดังตารางที่ 4.14

ตารางที่ 4.14 ตาราง Confusion matrix ที่ได้จากสถาปัตยกรรม ResNet101

Class	Air	FiberT1	FiberT2	FiberT3	FiberT4	ParticleT1	ParticleT2	ParticleT3	ParticleT4	Tissue
Air	106	3	0	0	3	0	11	1	11	0
FiberT1	3	128	16	8	7	3	12	3	6	0
FiberT2	0	7	45	10	1	1	6	0	1	1
FiberT3	2	20	4	45	16	1	10	2	10	1
FiberT4	0	19	1	1	45	1	0	0	7	2
ParticleT1	3	4	0	3	0	32	5	5	5	6
ParticleT2	5	2	1	0	0	2	68	9	0	4
ParticleT3	0	3	0	0	0	1	7	53	0	0
ParticleT4	0	0	0	0	0	0	0	2	84	0
Tissue	22	7	0	0	2	1	13	13	0	58

จากตารางที่ 4.14 แสดงผลลัพธ์ของความผิดพลาดที่เกิดขึ้นของแบบจำลองโมเดล ResNet101 โดยโมเดล ResNet101 พบว่ามีการทำนายภาพผิดพลาดในกลุ่ม FiberT3 มากที่สุด จำนวน 36 ภาพ โดยภาพที่ถูกต้องควรอยู่ในกลุ่ม FiberT1 จำนวน 20 ภาพ ตัวอย่างของภาพผลลัพธ์ที่ให้คำตอบผิดพลาดในรูปที่ 4.9 และอยู่ในกลุ่ม ParticleT4 จำนวน 16 ภาพ ซึ่งเมื่อนำมารวมกับภาพที่มีการทำนายผิดกลุ่มมีทั้งหมด 336 ภาพ ซึ่งคิดเป็นอัตราความผิดพลาดที่ 33.44% ซึ่งเมื่อนำผลลัพธ์ที่ได้มาหาค่าคำตอบจากการคำนวณหาค่า Accuracy ค่า Precision ค่า Recall และค่า $\mathbf{F1_{Score}}$ ผลลัพธ์ที่ได้คือ 66.44%, 66.56%, 67.29% และ 66.92% ตามลำดับ โดยผลที่ได้ถูกแสดงในตารางที่ 4.10

6) การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม DenseNet121

การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม DenseNet121 เริ่มต้นจากการนำชุดข้อมูล PiCO_V1 มาใช้ในการทดสอบ ซึ่งมีจำนวนข้อมูล 1,000 ภาพ โดยเป็นข้อมูลภาพที่ทราบถึงผลลัพธ์ของคำตอบล่วงหน้าก่อนแล้ว เมื่อนำข้อมูลภาพทั้งหมดมาผ่านการทำนายข้อมูลภาพด้วยสถาปัตยกรรม DenseNet121 ผลลัพธ์ที่ได้บันทึกด้วยตาราง Confusion matrix ดังตารางที่ 4.15

ตารางที่ 4.15 ตาราง Confusion matrix ที่ได้จากสถาปัตยกรรม DenseNet121

Class	Air	FiberT1	FiberT2	FiberT3	FiberT4	ParticleT1	ParticleT2	ParticleT3	ParticleT4	Tissue
Air	108	2	0	0	0	3	12	1	9	0
FiberT1	2	122	26	1	2	11	11	2	8	1
FiberT2	1	0	58	6	0	2	4	0	1	0
FiberT3	8	18	8	55	8	6	4	0	4	0
FiberT4	1	13	4	3	41	7	0	0	7	0
ParticleT1	11	1	0	0	0	39	3	6	2	1
ParticleT2	6	2	1	1	0	1	63	15	0	2
ParticleT3	2	2	2	0	0	3	7	47	0	1
ParticleT4	3	0	0	0	0	0	0	0	83	0
Tissue	18	4	0	0	0	7	7	10	1	69

จากตารางที่ 4.15 แสดงผลลัพธ์ของความผิดพลาดที่เกิดขึ้นของแบบจำลองโมเดล DenseNet121 โดยโมเดล DenseNet121 พบว่ามีการทำนายภาพผิดพลาดในกลุ่ม FiberT2 มากที่สุดจำนวน 26 ภาพ โดยภาพที่ถูกต้องควรอยู่ในกลุ่ม FiberT3 โดยตัวอย่างของภาพผลลัพธ์ที่ให้คำตอบผิดพลาดดังรูปที่ 4.10

(ก) FiberT2

(ข) FiberT3

รูปที่ 4.10 ตัวอย่างสิ่งปลอมปนในน้ำมันมะพร้าวที่ให้คำตอบระหว่าง FiberT2 และ FiberT3

เมื่อนำมารวมกับภาพที่มีการทำนายผิดกลุ่มมีทั้งหมด 315 ภาพ ซึ่งคิดเป็นอัตราความผิดพลาดที่ 31.45% ซึ่งเมื่อนำผลลัพธ์ที่ได้มาหาค่าคำตอบจากการคำนวณหาค่า Accuracy ค่า Precision ค่า Recall และ ค่า $\mathbf{F1}_{\mathbf{score}}$ ผลลัพธ์ที่ได้คือ 68.55%, 62.75%, 69.38% และ 65.90% ตามลำดับ โดยผลที่ได้ถูกแสดงในตารางที่ 4.10

7) การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม VGG16

การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม VGG16 เริ่มต้นจากการนำชุดข้อมูล PiCO_V1 มาใช้ในการทดสอบ ซึ่งมีจำนวนข้อมูล 1,000 ภาพ โดยเป็นข้อมูลภาพที่ทราบถึงผลลัพธ์ของคำตอบล่วงหน้าก่อนแล้ว เมื่อนำข้อมูลภาพทั้งหมดมาผ่านการทำนายข้อมูลภาพด้วยสถาปัตยกรรม VGG16 ผลลัพธ์ที่ได้บันทึกด้วยตาราง Confusion matrix ดังตารางที่ 4.16

ตารางที่ 4.16 ตาราง Confusion matrix ที่ได้จากสถาปัตยกรรม VGG16

Class	Air	FiberT1	FiberT2	FiberT3	FiberT4	ParticleT1	ParticleT2	ParticleT3	ParticleT4	Tissue
Air	104	1	1	0	3	1	15	0	9	1
FiberT1	2	139	6	1	0	14	15	0	9	0
FiberT2	0	6	56	1	0	4	3	0	2	0
FiberT3	6	15	6	57	0	8	10	2	7	0
FiberT4	1	12	2	1	42	10	6	0	2	0
ParticleT1	5	2	0	2	0	34	10	1	5	4
ParticleT2	6	2	0	0	0	2	74	5	0	2
ParticleT3	0	4	0	0	0	0	15	42	2	1
ParticleT4	2	0	0	0	0	0	1	0	83	0
Tissue	26	2	0	0	0	5	25	3	1	54

จากตารางที่ 4.16 แสดงผลลัพธ์ของความผิดพลาดที่เกิดขึ้นของแบบจำลองโมเดล VGG16 โดยโมเดล VGG16 พบว่ามีการทำนายภาพผิดพลาดในกลุ่ม FiberT3 มากที่สุดจำนวน 15 ภาพ โดยภาพที่ถูกต้องควรอยู่ในกลุ่ม FiberT1 ซึ่งเมื่อนำมารวมกับภาพที่มีการทำนายผิดกลุ่มมีทั้งหมด 315 ภาพ คิดเป็นอัตราความผิดพลาดที่ 31.48% เมื่อนำผลลัพธ์ที่ได้มาหาค่าคำตอบจากการคำนวณหาค่า Accuracy ค่า Precision ค่า Recall และ ค่า $\mathbf{F1_{Score}}$ ผลลัพธ์ที่ได้คือ 68.52%, 60.54%, 73.02% และ 66.20% ตามลำดับ โดยผลที่ได้ถูกแสดงในตารางที่ 4.10

8) การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม VGG19

การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม VGG19 เริ่มต้นจากการนำชุดข้อมูล PiCO_V1 มาใช้ในการทดสอบ มีจำนวนข้อมูล 1,000 ภาพ โดยเป็นข้อมูลภาพที่ทราบถึงผลลัพธ์ของคำตอบล่วงหน้าก่อนแล้ว เมื่อนำข้อมูลภาพทั้งหมดมาผ่านการทำนายข้อมูลภาพด้วยสถาปัตยกรรม VGG19 ผลลัพธ์ที่ได้บันทึกด้วยตาราง Confusion matrix ดังตารางที่ 4.17

ตารางที่ 4.17 ตาราง Confusion matrix ที่ได้จากสถาปัตยกรรม VGG19

Class	Air	FiberT1	FiberT2	FiberT3	FiberT4	ParticleT1	ParticleT2	ParticleT3	ParticleT4	Tissue
Air	91	2	3	0	5	0	18	0	9	7
FiberT1	10	146	10	1	2	2	3	0	12	0
FiberT2	2	4	47	7	1	1	9	0	1	0
FiberT3	10	20	4	50	8	1	9	0	9	0
FiberT4	1	19	4	1	40	2	3	0	6	0
ParticleT1	7	6	0	0	1	30	9	1	4	5
ParticleT2	5	0	3	0	1	0	76	5	0	1
ParticleT3	1	0	1	0	0	0	11	51	0	0
ParticleT4	3	0	0	0	0	0	0	0	83	0
Tissue	17	1	1	0	1	2	16	2	1	75

จากตารางที่ 4.17 แสดงผลลัพธ์ของความผิดพลาดที่เกิดขึ้นของแบบจำลองโมเดล VGG19 โดยโมเดล VGG19 พบว่ามีการทำนายภาพผิดพลาดในกลุ่ม FiberT3 มากที่สุดจำนวน 20 ภาพ โดยภาพที่ถูกต้องควรอยู่ในกลุ่ม FiberT1 ตัวอย่างของภาพผลลัพธ์ที่ให้คำตอบผิดพลาดในรูปที่ 4.9 โดยเมื่อนำมารวมกับภาพที่มีการทำนายผิดกลุ่มมีทั้งหมด 299 ภาพ คิดเป็นอัตราความผิดพลาดที่ 29.98% เมื่อนำผลลัพธ์ที่ได้มาหาค่าคำตอบจากการคำนวณหาค่า Accuracy ค่า Precision ค่า Recall และค่า $\mathbf{F1_{Score}}$ ผลลัพธ์ที่ได้คือ 70.02%, 69.38%, 71.89% และ 68.58% ตามลำดับ ผลลัพธ์ที่ได้ถูกแสดงในตารางที่ 4.10

9) การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม InceptionResNetV2

การทดลองหาค่าประสิทธิภาพจากสถาปัตยกรรม InceptionResNetV2 เริ่มต้นจากการนำชุดข้อมูล PiCO_V1 มาใช้ในการทดสอบ มีจำนวนข้อมูล 1,000 ภาพ โดยเป็นข้อมูลภาพที่ทราบถึงผลลัพธ์ของคำตอบล่วงหน้าก่อนแล้ว เมื่อนำข้อมูลภาพทั้งหมดมาผ่านการทำนายข้อมูลภาพด้วยสถาปัตยกรรม InceptionResNetV2 ผลลัพธ์ที่ได้บันทึกด้วยตาราง Confusion matrix ดังตารางที่ 4.18

ตารางที่ 4.18 ตาราง Confusion matrix ที่ได้จากสถาปัตยกรรม InceptionResNetV2

Class	Air	FiberT1	FiberT2	FiberT3	FiberT4	ParticleT1	ParticleT2	ParticleT3	ParticleT4	Tissue
Air	54	1	5	0	3	1	33	2	13	23
FiberT1	4	73	59	3	10	12	7	3	13	2
FiberT2	1	0	64	1	1	2	0	1	1	1
FiberT3	5	4	16	42	10	10	8	3	10	3
FiberT4	1	4	9	0	46	1	3	3	6	3
ParticleT1	5	1	2	0	1	26	8	6	2	12
ParticleT2	5	0	2	1	0	0	63	14	0	6
ParticleT3	2	2	1	0	0	3	17	36	0	3
ParticleT4	3	0	0	0	0	0	0	0	83	0
Tissue	17	2	6	0	0	5	13	5	0	68

จากตารางที่ 4.18 แสดงผลลัพธ์ของความผิดพลาดที่เกิดขึ้นของแบบจำลองโมเดล InceptionResNetV2 พบว่ามีการทำนายภาพผิดพลาดในกลุ่ม FiberT3 มากที่สุดจำนวน 20 ภาพ โดยภาพที่ถูกต้องควรอยู่ในกลุ่ม FiberT3 ซึ่งเมื่อนำมารวมกับภาพที่มีการทำนายผิดกลุ่มมีทั้งหมด 445 ภาพ คิดเป็นอัตราความผิดพลาดที่ 44.5% เมื่อนำผลลัพธ์ที่ได้มาหาค่าคำตอบจากการคำนวณหาค่า Accuracy ค่า Precision ค่า Recall และค่า $\mathbf{F1}_{\mathbf{Score}}$ ผลลัพธ์ที่ได้คือ 55.55%, 55.25%, 58.79% และ 55.44% ตามลำดับ โดยผลที่ได้ถูกแสดงในตารางที่ 4.10

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

สำหรับการวัดค่าความขุ่นและการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าวนั้นอาจมีได้หลายวิธี ซึ่งวิธีการที่ผู้วิจัยได้นำเสนอก็เป็นวิธีการหนึ่งที่ช่วยสร้างความเชื่อมั่นให้กับกระบวนการผลิตน้ำมันมะพร้าวของอุตสาหกรรมน้ำมันมะพร้าวของประเทศไทยได้ โดยผู้วิจัยได้พัฒนาระบบการวัดค่าความขุ่นของน้ำมันมะพร้าวขึ้นเพื่อให้สามารถทราบถึงค่าความขุ่นของน้ำมันมะพร้าวได้ทันที ก่อนการบรรจุน้ำมันมะพร้าวเป็นผลิตภัณฑ์เพื่อวางจำหน่าย โดยจากการทดลองพบว่าขั้นตอนวิธีการ MAMoH นั้นให้ผลลัพธ์ของการทดลองดีที่สุด โดยคิดเป็น 99.0% ที่สามารถหาค่าความขุ่นของน้ำมันมะพร้าวตามค่าระดับความขุ่นที่กำหนดได้อย่างถูกต้อง และสำหรับข้อผิดพลาด 1% ที่เกิดขึ้นนั้น เกิดขึ้นจากช่วงเริ่มต้นของการทำงานของโปรแกรมที่ไม่ได้มีการนำค่ามารวมในการหาค่าเฉลี่ยเคลื่อนที่ จึงทำให้เกิดข้อผิดพลาดของผลลัพธ์คำตอบที่ได้ สำหรับการเปรียบเทียบกับขั้นตอนวิธีการ MoGS ที่หาความขุ่นจากภาพระดับเทานั้น มีค่าประสิทธิภาพของขั้นตอนวิธีการคิดเป็น 15.30% เมื่อเทียบกับวิธีการ MAMoH ส่วนวิธีการ MoH และวิธีการ RPMoH ที่เป็นการหาความขุ่นของน้ำมันมะพร้าวจากการอาศัยค่าความแข็งแรงของสีนั้น มีค่าประสิทธิภาพคิดเป็น 88.04% และ 85.91% ตามลำดับเมื่อเทียบกับวิธีการ MAMoH ซึ่งสำหรับขั้นตอนวิธี MAMoH นั้นสามารถแบ่งระดับความขุ่นของน้ำมันมะพร้าวได้ชัดเจนที่สุด เมื่อเปรียบเทียบกับวิธีการอื่นที่นำเสนอ และจากการทดลองจากการนำไปใช้งานจริง พบว่าทั้ง 3 รอบ ของการผลิตน้ำมันมะพร้าวในโรงงาน ขั้นตอนวิธีการ MAMoH มีความคงทนต่อการเปลี่ยนแปลงของข้อมูลที่นำเข้ามาประมวลผลได้ โดยหลังจากที่มีการปรับเปลี่ยนขั้นตอนของกระบวนการผลิตน้ำมันมะพร้าวขึ้นมาใหม่นั้น ขั้นตอนวิธีการ MAMoH ยังคงสามารถแสดงผลลัพธ์ของคำตอบได้อย่างถูกต้อง ส่วนการตรวจจับสิ่งปลอมปนในน้ำมันมะพร้าวได้ใช้แบบจำลองการเรียนรู้เชิงลึก เพื่อแก้ไขปัญหาในการจัดกลุ่มประเภทวัตถุแปลกปลอมในน้ำมันมะพร้าว จำนวน 10 กลุ่มโดยใช้โครงข่ายประสาทเทียมแบบ CNN ซึ่งจากการทดลองเปรียบเทียบประสิทธิภาพของแต่ละสถาปัตยกรรม พบว่าสถาปัตยกรรม MobileNetV2 นั้นให้ผลลัพธ์ค่าความแม่นยำในการจดจำวัตถุแปลกปลอมในน้ำมันมะพร้าวดีที่สุดเมื่อทำการเปรียบเทียบกับสถาปัตยกรรมอื่นที่ได้นำเสนอ ผลลัพธ์ที่ได้จากสถาปัตยกรรม MobileNetV2 นั้นผ่านการปรับปรุงค่าพารามิเตอร์ 2 ค่า คือของค่าพารามิเตอร์ของความกว้างและค่าพารามิเตอร์ของตัวคูณความละเอียด ซึ่งทั้ง 2 ค่าพารามิเตอร์นั้นมีส่วนสำคัญที่ส่งผลกระทบโดยตรงต่อขนาดโมเดลได้ และจากผลลัพธ์จากการทดลองแสดงให้เห็นว่าสถาปัตยกรรม MobileNetV2 ให้ความแม่นยำอยู่ที่ 80.20% สำหรับชุดข้อมูล PiCO_V1 ซึ่งให้ผลลัพธ์ของความแม่นยำที่มากกว่าสถาปัตยกรรมแบบอื่นที่ได้ทำการทดลอง โดยสถาปัตยกรรม

InceptionV3 ให้ความแม่นยำอยู่ที่ 75.13% และสถาปัตยกรรม Xception ให้ความแม่นยำอยู่ที่ 72.95%

5.2 ข้อเสนอแนะ

5.2.1 ข้อเสนอแนะเพื่อการพัฒนา

1) การพัฒนาเพื่อควบคุมระบบสายการผลิตด้วยการประมวลผลภาพน้ำมันมะพร้าว สามารถช่วยพัฒนาการควบคุมในการผลิตได้ดียิ่งขึ้น
2) การพัฒนาระบบตรวจสิ่งปลอมปนและระบุสิ่งปลอมปนในน้ำมันมะพร้าวสำหรับการบริโภค สามารถช่วยสร้างความมั่นใจให้กับผู้บริโภค และช่วยเสริมมูลค่าของผลิตภัณฑ์ให้มีมูลค่าเพิ่มขึ้น

5.2.2 ข้อเสนอแนะอื่นๆ

1) สามารถนำไปใช้งานร่วมกับการควบคุมสายการผลิตโดยใช้เครื่องประมวลผลด้านกราฟิก(GPU) ในการทำงาน
2) สามารถพัฒนาร่วมกับระบบแจ้งเตือนหรือสัญญาณเตือนอื่นๆ เมื่อพบสิ่งผิดปรกติในกระบวนการผลิตน้ำมันมะพร้าว

5.2.3 ข้อเสนอแนะในการวิจัยครั้งต่อไป

1) ควรศึกษาในเรื่องการประมวลผลภาพแบบปรับตัวเองได้ตามสภาพแสงและปัจจัยภายนอก
2) ควรศึกษาประเภทของสิ่งปลอมปนอื่นที่ส่งผลกระทบทั้งทางตรงและทางอ้อมต่อน้ำมันมะพร้าว

5.3 ข้อจำกัด

1) การทำงานควรอยู่ภายในสภาพแวดล้อมแบบปิดหรืออยู่ภายใต้การควบคุมสภาพแสง

2) จากชุดข้อมูลภาพที่ได้มาพบว่า กลุ่มของข้อมูลภาพวัตถุปลอมปนในน้ำมันมะพร้าวบางกลุ่มมีความคล้ายคลึงกันและมีลักษณะของภาพใกล้เคียงกันมาก ส่งผลทำให้ค่าประสิทธิภาพของโมเดลในการระบุภาพวัตถุนั้นลดลง

เอกสารอ้างอิง

[1] Gonzalez R.C. 2021. Digital Image Processing Basics. [Online]. Available : https://www.geeksforgeeks.org/digital-image-processing-basics/.

[2] Stefanoa, L.D. Empinottib, V. Schmidtc, L. Jacobid, P. R. Ferreirac, J.G and Guerra, J. 2016. "Measuring Information Transparency in the Water Sector: What Story Do Indicators Tell?." *International Journal of Water Governance*. 1 : 1–22.

[3] Forughi, A.F. Stoeber, B. Green, S.I. 2017. "Transparency measurement of thin films with one-sided optical access using fluorescence imaging." *National Center for Biotechnology Information*. 56(12) : 3359-3364.

[4] Mayerhofer, T.G. Pahlow, S. and Popp, J. 2020. "The Bouguer-Beer-Lambert Law: Shining Light on the Obscure." *Europe Chemistry Societies*. 21 : 2029-2046.

[5] อรรถพล พลานนท์, สรร รัตนสัญญา และ สมชาติ รุ่งเรืองสรการ. 2011. "การนับจำนวนห่อกระดาษด้วยการใช้ประโยชน์จากวิธีการฉายภาพ." หน้า 51-56. ใน The 8th joint Conference on Computer Science and Software Engineering (JCSSE 2011) ปีที่ 8. นครปฐม.

[6] Sebe, N. and Lew, M. S. 2003. **Robust Computer Vision: Theory and Applications**. 2003rd ed. Springer.

[7] Szeliski, R. 2010. **Computer Vision: Algorithms and Applications**. Springer.

[8] Deswal, M. and Sharma, N. 2014. "A fast HSV image color and texture detection and image conversion algorithm." *Int. J. Sci. Res.* 3(6) : 1279-1284.

[9] Abdullah, A. S. 2015. "Text hiding based on hue content in HSV color space." *Int. J. Emerg. Trends Technol. Comput. Sci.* 4(2) : 170-173.

[10] อรรถพล พลานนท์, อวยไชย อินทรสมบัติ และบุญหทัย เครือแก้ว. 2559. "การหาตำแหน่งการไขหมุดประดับล้อแม็กรถยนต์สำหรับแขนกลไขน็อตแบบ 3 แกน." หน้า 433-440. ใน การประชุมวิชาการระดับชาติ ครั้งที่ 8 มหาวิทยาลัยราชภัฏนครปฐม. นครปฐม : มหาวิทยาลัยราชภัฏนครปฐม.

[11] Kumar, T. and Verma, K. 2010. "A theory based on conversion of RGB image to gray image." *Int. J. Comput. Appl.* 7(2) : 7-10.

[12] Vijaya, R. Prudvi, K. Ravi, L. and Jogendra, M. 2016. "Grey level to RGB using YCbCr color space technique." *Int. J. Comput. Appl.* 147(7) : 25-28.

เอกสารอ้างอิง (ต่อ)

[13] Cheong, K. H. Poeschmann, S. Lai, J. W. Koh, J. M. Acharya,U. R. Yu, S. C. M. and Tang, K. J. W. 2019. "Practical automated video analytics for crowd monitoring and counting." IEEE Access. 7 : 183252-183261.

[14] Sim, K. S. Tso, C. P. and Tan, Y. Y. 2007. "Recursive sub-image histogram equalization applied to gray scale images." *Pattern Recognition Lett.* 28(10) : 1209-1221.

[15] Stark, J. A. 2000. "Adaptive image contrast enhancement using generalizations of histogram equalization." *IEEE Trans. Image Process.* 9(5) : 889-896.

[16] Ali, M. M. N. and Abdullah-Al-Wadud, M. 2012. "Image enhancement using a modified histogram equalization." pp. 17-24. in Kim, T. Mohammed, S. Ramos, C. Abawajy, J. Kang, B.H. Slezak, D. (eds). **Computer Applications for Web, Human Computer Interaction, Signal and Image Processing, and Pattern Recognition. ICHCI 2012, WSE 2012, SIP 2012. Communications in Computer and Information Science**. Vol 342. Heidelberg : Springer.

[17] Lee, J. Pant, S. R. and Lee, H.-S. 2015. "An adaptive histogram equalization based local technique for contrast preserving image enhancement." *Int. J. Fuzzy Log. Intelligent Syst.* 15(1) : 35-44.

[18] Chen, H.C. and Chen, S.W. 2003. "A Moving Average based Filtering System with its Application to Real time QRS Detection." *Computers in Cardiology*. 585_588.

[19] Smith, S. W. 1997. "Moving average filters." pp. 277-284. in **The Scientist and Engineer's Guide to Digital Signal Processing**. USA : California Technical Publishing.

[20] Al-Odienat, A. I. and Al-Mbaideen, A. A. 2015. "Optimal length determination of the moving average filter for power system applications." *Int. J. Innov. Comput., Inf. Control*. 11(2) : 691-705.

[21] Tyagi, T. and Mishra, V. 2016. "2D Gaussian filter for image processing: A study." *Int. J. Sci. Technol. Eng.* 3(6) : 22-24.

[22] Agarwal, R. 2012. "Bit plane average filtering to remove Gaussian noise from high contrast images" pp. 1-5. in **Proceedings of International Conference Computer Communication Information**.

เอกสารอ้างอิง (ต่อ)

[23] Amoako-Yirenkyi, P. Appati, J. K. and Dontwi, I. K. 2016. "Performance analysis of image smoothing techniques on a new fractional convolution mask for image edge detection." *Open J. Appl. Sci.* 6(7) : 478-488.

[24] Asmaidi, A. Putra, D.S. Risky, M.M. and Fitria, Ulfa, R. 2019. "Implementation of Sobel Method Based Edge Detection for Flower Image Segmentation." *Sinkron : Jurnal Dan Penelitian Teknik Informatika.* 3(2) : 161-166.

[25] OR. Vincent and O. Folorunso "A Descriptive Algorithm for Sobel Image Edge Detection," Informing Science & IT Education Conference (InSITE), pp.97-106, 2009

[26] LeCun, Y. Boser, B. Denker, J. S. Henderson, D. Howard, R. E. Hubbard, W. and Jackel, L. D. 1989. "Back-propagation applied to handwritten zip code recognition." *Neural Computation.* 1 : 541–551.

[27] Ranzato, M. A. Huang, F. Boureau, Y. and LeCun, Y. 2007. "Unsupervised learning of invariant feature hierarchies with applications to object recognition." pp. 1-8. in **Proceedings of 2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Minneapolis : IEEE.

[28] Brinker, T.J. Hekler, A. Utikal, J. Grabe, N. Schadendorf, D. Klode, J. Berking, C. Steeb, T. Enk, A. and Kalle, C.V. 2018. "Skin Cancer Classification Using Convolutional Neural Networks: Systematic Review." *J. Med. Internet Res.* 20(10) : e11936, 1-8.

[29] Krizhevsky, A. Sutskever, I. and Hinton, G.E. 2017. "ImageNet classification with deep convolutional neural networks." *Communications of the ACM.* 60 : 84 - 90.

[30] Simonyan, K. Vedaldi, A. and Zisserman, A. 2014. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps." *CoRR.* arXiv:1312.6034.

[31] Simonyan, K. and Zisserman, A. 2015. "Very Deep Convolutional Networks for Large-Scale Image Recognition." in **Proceedings of the 2015 International Conference on Learning Representations (ICLR 2015).** abs/1409.1556.

เอกสารอ้างอิง (ต่อ)

[32] Howard, A.G. Zhu, M. Chen, B. Kalenichenko, D. Wang, W. Weyand, T. Andreetto, M. and Adam, H. 2017. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." ArXiv abs/1704.04861.

[33] Pan, H. Pang, Z. Wang, Y. Wang, Y. and Chen, L. 2020. "A New Image Recognition and Classification Method Combining Transfer Learning Algorithm and MobileNet Model for Welding Defects." *IEEE Access*. 8 : 119951-119960.

[34] Palananda, A. and Kimpan, W. 2021. "Turbidity of Coconut Oil Determination Using the MAMoH Method in Image Processing." *IEEE Access.* 9 : 41494-41505.

[35] Szegedy, C. Liu, W. Jia, Y. Sermanet, P. Reed, S.E. Anguelov, D. Erhan, D. Vanhoucke, V. and Rabinovich, A. 2015. "Going deeper with convolutions." pp. 1-9. in **Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),** Boston, MA, USA : IEEE.

[36] He, K. Zhang, X. Ren, S. and Sun, J. 2016. "Deep residual learning for image recognition" pp. 770–778. in **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**, Las Vegas, NV, USA : IEEE.

[37] Aparna, Bhatia, Y. Rai, R. Gupta, V. Aggarwal, N. and Akula, A. "Convolutional neural networks based potholes detection using thermal imaging." *J. King Saud Univ. - Comput. Inf. Sci.* (in press).

[38] Huang, G. Liu, Z. Pleiss, G. Maaten, L.V. and Weinberger, K.Q. 2019. "Convolutional Networks with Dense Connectivity." *IEEE Trans. Pattern. Anal. Mach. Intell.* 1-12.

[39] Tao, Y. Xu, M. Lu, Z. and Zhong, Y. 2018. "DenseNet-Based Depth-Width Double Reinforced Deep Learning Neural Network for High-Resolution Remote Sensing Image Per-Pixel Classification." *Remote. Sens.* 10 : 779.

[40] Too, E.C. Yujian, L. Njuki, S. and Yingchun, L. 2019. "A comparative study of fine-tuning deep learning models for plant disease identification." *Comput. Electron. Agric*. 161 : 272-279.

[41] Tammina, S. 2019. "Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images." *International Journal of Scientific and Research Publications (IJSRP).* 9 : 143-150.

ภาคผนวก

ภาคผนวก
งานวิจัยที่ตีพิมพ์

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 5, 2021, accepted February 28, 2021, date of publication March 9, 2021, date of current version March 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3065004

Turbidity of Coconut Oil Determination Using the MAMoH Method in Image Processing

ATTAPON PALANANDA AND WARANGKHANA KIMPAN, (Member, IEEE)
Department of Computer Science, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand

Corresponding author: Warangkhana Kimpan (warangkhana.ki@kmitl.ac.th)

This research was supported in testing part by Tropicana Oil Co., Ltd. and funding part by Faculty of Science, King Mongkut's Institute of Technology Ladkrabang.

ABSTRACT In general, considering standard production, as well as coconut oil production, in oil consumption industries is an important factor. Oil color is an important element, as it is an important factor for consumers or buyers in selecting coconut oil. In the process of producing coconut oil, the cold-pressed method has been chosen to maintain the essential quality of coconut oil. The quality of the coconut oil is inspected from the production process by means of light passing through the coconut oil. Then, the production staff compares the turbidity of coconut oil with the master sample. The turbidity of coconut oil in every production must be compared with a master sample to maintain standards control. According to previous studies, there are many methods for determining coconut oil turbidity. One method that has been utilized is determining turbidity from light passing through the medium in which the transmitted light can be absorbed through the turbidity of the variable medium. This process is applied together with image processing to determine the coconut oil turbidity. In this research, we propose a method for measuring coconut oil turbidity by the Moving Average Median of Hue (MAMoH), which is better in detecting the coconut oil turbidity than the Median of Gray Scale (MoGS) method, Median of Hue (MoH) method, and Random Position Median of Hue (RPMoH) method. In terms of the percentage accuracy of the efficiency test; the MAMoH method has 99 percent accuracy, while the MoGS method is not applicable, the MoH method has 88.04 percent accuracy, and the RPMoH method has 85.91 percent accuracy. Thus, the MAMoH method is considered an appropriate method for measuring coconut oil turbidity.

INDEX TERMS Image processing, turbidity level, moving average, coconut oil, computer vision.

I. INTRODUCTION

In the coconut oil manufacturing industry, oil transparency is the first priority, because consumers can see this with their naked eye. Therefore, the coconut oil manufacturing and inspection process is essential.

The two methods for producing coconut oil are hot-pressed and cold-pressed [1]. The cold-pressed production method is produced by compressing the coconut pulp into oil, which helps to maintain nutritional value. The coconut oil production processes consist of coconut meat preparation, coconut meat processing step, coconut meat compression, coconut oil filtration, and the last step is coconut oil packing. When coconut meat is compressed, coconut oil is extracted. The resulting coconut oil may contain impurities. Normally, the coconut oil production quality lies in the process of filtering the oil obtained from the previous step. In the filtration process, the coconut oil is sent through a centrifugation method to filter impurities or coconut meat fiber residue through a 0.25-micron fine filter until non-turbid coconut oil is removed. In the coconut oil production process, each production cycle has a different turbidity. Therefore, the degree of turbidity of the coconut oil obtained in each production cycle has to be examined by the staff using light shining through the sample oil bottles to compare turbidity. The turbidity of coconut oil is related to its color, similar to olive oil color [2], which can be compared or observed with the naked eye. Different colors of olive oil are caused by different turbidities, which occur from the fibers or rinds of the olives mixed during the oil production process. Too much sediment or impure oil is not healthy, and the oil inspection process is therefore very important.

Likewise, image processing is used to check the turbidity of the water flowing through the pipe, examine the sludge, dust or debris in the water, and find the floating oil layer that flows with the water inside the pipe [3]–[5]. Therefore, this image processing technique helps to measure the oil turbidity to a level that can be consumed. The research

The associate editor coordinating the review of this manuscript and approving it for publication was You Yang.

method proposes that the coconut oil turbidity measurement presented here is based on only one feature; that is, it considers the brightness of the light passing through the coconut oil, which enables rapid processing to determine the coconut oil turbidity level. In doing so, a developed turbidity measurement device is installed in a position after the coconut oil filtration process. It is connected to the coconut oil conveying pipe before the coconut oil flows into the oil storage tank waiting for packing. The coconut oil turbidity value obtained from the proposed algorithm can be used to determine the coconut oil turbidity levels in each production cycle, which makes it more convenient and faster in routine working processes of industrial employees.

The paper is structured as follows. Section II presents related research, and the proposed methods are described in detail in Section III. Experimental results are shown in Section IV. Finally, Section V presents a conclusion.

II. RELATED RESEARCHES

The process of coconut oil turbidity analysis generally consists of conversion of the HSV hue, saturation, and value color model, image segmentation, image enhancement, and determination of turbidity values from images, which is summarized below.

A. COLOR SPACE

A standard color system for computers can represent each pixel with a three-dimensional vector. Each pixel can represent 3 different color values: red, green, and blue. This is called the RGB color system. RGB is an international standard, and there is another standard called the HSV color model. HSV relies on the separation of brightness from the color of the pixel. The value in the HSV color model is represented by the three-dimensional vector. This HSV color space describes colors (hue or tint) in terms of their shade (saturation or amount of gray) and their brightness value. In color systems, a global standard can convert color systems such as RGB to HSV or RGB to grayscale conversions to help reduce the data processing size.

1) COLOR SPACE CONVERSION

Converting RGB to HSV is possible because the HSV color system conversion is similar to human color information processing. Humans can understand HSV color better than the RGB color system. The RGB color defines a color space in terms of three components: red, green, and blue, which range from 0-25. The RGB and HSV values can be converted as shown in Fig. 1.

Fig. 1 can determine the color or color appearance parameters H from (1) where θ is approximately the angle of the vector obtained from the RGB color. It is calculated by(2). S and V are calculated according to (3) and (4), respectively.

$$H = \begin{cases} \theta, & B \le G \\ 2\pi - \theta, & otherwise \end{cases} \quad (1)$$

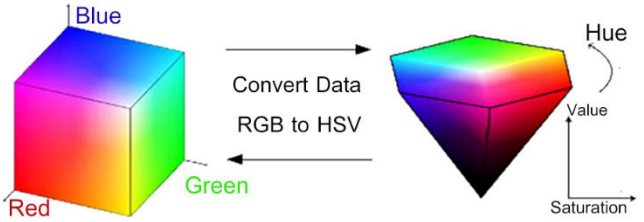


FIGURE 1. Relationship between the RGB color and the HSV color systems.

$$\theta = cos^{-1}\left(\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)\cdot(G-B)}}\right) \quad (2)$$

$$S = 1 - 3\cdot\frac{Min\,(R,G,B)}{R+G+B} \quad (3)$$

$$V = \frac{R+G+B}{3} \quad (4)$$

Many researchers have used RGB to HSV conversion; for example, in 2010, Chang *et al.* [6] proposed HSV-based color texture image classification using wavelet transform and motif patterns. Later, in 2014, Deswal *et al.* [7] proposed a method called "A Fast HSV Image Color and Texture Detection Algorithm" based on color intensity using artificial intelligence. They used the HSV-based color model since it greatly decreases the image color and grayscale information size. Hue filtering was used to segment the specified color. A year later, Abdullah [8] presented a new method for hiding information that relied on the HSV color space. He used the color image conversion from RGB color space to HSV and converted the color space from HSV to RGB. Chernov *et al.* [9] introduced an integer-based algorithm to convert color representation from RGB to HSV color spaces and vice versa. In 2016, Saravanan *et al.* [10] presented new generic color space conversion hardware. They explained the RGB to HSV/HSI/HSL conversion architectures. The transformation can be performed by the algorithm to convert the color space in (1) to (4).

2) GRAY SCALE CONVERSION

This converts the RGB color image to a grayscale image to reduce the number of color layers from RGB color spaces to only a grayscale color space. The image retains the outline of the image and the color saturation or brightness value for each pixel between 0- 255 levels. There are several methods to convert a color image to a grayscale image [11], [12]. The methods used to convert a color image to a grayscale image can be obtained using the average method as in (5).

$$I\_Gray = \frac{R+G+B}{3} \quad (5)$$

After the previous step, the brightness cannot be compensated. When the original image is very opaque, the color of the grayscale image becomes rendered as a black image, which is different from a human's vision. Therefore, to obtain a grayscale image that is similar to the visibility of the human eye, we studied grayscale image transformation by taking the

weight value (w) [11] combined with the new grayscale image calculation, which can be found using (6).

$$\text{I\_Gray} = (\text{w}_1 \times \text{R}) + (\text{w}_2 \times \text{G}) + (\text{w}_3 \times \text{G}) \quad (6)$$

where $w_1 + w_2 + w_3 = 1$ and $w_1 + w_2 + w_3 > 0$

According to the experimental results of Kumar and Verma [11], the suitable weights in each color are $w_1 = 0.33$, $w_2 = 0.50$, and $w_3 = 0.166$. When the pixel values of the readable RGB image are 18, 17, 17 using (6), the grayscale image is equal to I_Gray = (0.333∗187) + (0.50∗179) + (0.166∗176), which makes the value of the image spot equal to 180.98. After that, all pixels in the color image are processed and converted to a grayscale image.

B. IMAGE SEGMENTATION

There are a variety of image segmentation techniques. One of the image processing methods is region of interest (ROI) based image segmentation. This method requires exactly both undesired and desired position information. The method frames a square around the desired area to process the specific squared part of the image. Each image can be specified in many parts, as shown in Fig. 2. Another method is to select the desired area in the image by image subtraction, which removes the undesired background image from the desired object. In 2019, Cheong *et al.* [13] introduced a method of counting the number of people in the group using the image subtraction technique. After that, CNN classify, which is an AI algorithm, is applied to segment objects for verification and automatic tracking.

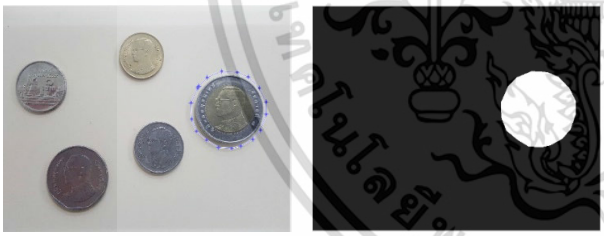


FIGURE 2. Sample images of a region of interest (ROI) based image segmentation.

C. IMAGE ENHANCEMENT

Image enhancement is a method for improving image quality. This method causes the image change to obtain a clear image for processing or to reduce noise in the image for quality improvement. Image enhancement can be processed in many ways, such as histogram equalization and Gaussian filter.

1) HISTOGRAM EQUALIZATION

Histogram equalization is a method in which the image processing of contrast adjustment uses the image histogram. Histogram equalization is another nonlinear contrast enhancement technique. In this technique, the histogram of the original image is redistributed to produce a uniform population density. This is obtained by grouping certain adjacent

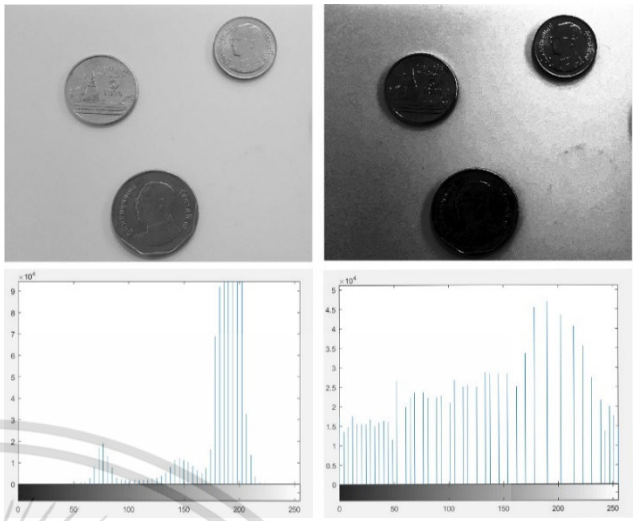


FIGURE 3. Image enhancement using the histogram equalization method.

gray values. Thus, the number of gray levels in the enhanced image is less than the number of gray levels in the original image, as shown in Fig. 3.

In 2000, Stark [14] proposed an image sharpness adjustment using the adaptive histogram, which allows adjusting the sharpness of the image by itself using the brightness values in the image. This method can increase the sharpness level in many images.

In 2007, Sim *et al.* [15] developed a histogram method using recursive sub-image histogram equalization (RSIHE) compared with the traditional method of histogram equalization. The RSIHE method compensated for the brightness portion of an image by finding the average and adjusting each of the subimages. The results of the image using this method are sharper than those of the traditional method.

In 2012, Naushad Ali and Abdullah-Al-Wadud [16] proposed detailed improvements of a separated image and combined it with the updated image using the weighted function to improve the quality of the image. In 2015, Lee *et al.* [17] proposed a method for preserving the details and sharpness of images that were missing from the use of the histogram equalization method by calculating the current density of nearby pixels. Then obtained values were used to evaluate the difference of pixels. This method can help analyze pixels that need further improvement.

2) GAUSSIAN FILTER

A Gaussian filter is used to blur images while maintaining color intensity. The Gaussian blur method in 2D with Gaussian filters can be represented in (7).

$$G_\sigma(x_i, y_i) = e^{-\frac{1}{2\pi\sigma^2} e^{\frac{x_i^2+y_i^2}{2\sigma^2}}} \quad (7)$$

where σ is the standard deviation of the width from the image expansion, including the results of a smoother image. If the σ value is large, the Gaussian filter will be wider. When

the center of the image is the highest for the Gaussian filter weight value, the weight value decreases. When the reference position is far from the center, the size of the Gaussian filter affects image blurring, which helps reduce the disturbance that occurs in the image. The results of the image filtering using the Gaussian method are shown in Fig. 4.

FIGURE 4. Sample of images using the Gaussian filter method.

In 2012, Agarwal [18] presented the Gaussian distribution method to remove Gaussian noise from high-contrast images, and it was used to adopt the moving average to help improve the color of an image.

A year later, Chandel and Gupta [19] studied various image filtering techniques that affected pixels. From this experiment, it was found that the median filtering method performed well when an image histogram was used. Moreover, it was found that the median filtering method is better than the average filter algorithm to eliminate image noise. In addition, this method can keep the edges of the image and the same edge as the specified original window size.

In 2015, Codevilla *et al.* [20] improved the quality of turbidity levels of a seabed scene captured using the simulated TURBID dataset with different turbidity degradation. The images obtained from the dataset go through a Gaussian filter process and a Hessian-Laplace process to detect what is happening in the TURBID dataset image.

In 2016, Tyagi and Mishra [21] applied Gaussian filters and 2D images to use real-time filters for increased smoothing. The results showed that using 2D-Gaussian filters can eliminate noise from images even with varying image sizes and can compare values between fixed points and floating points through the operation of the CPU, GPU, and FPGA.

In the same year, Amoako-Yirenkyi *et al.* [22] discussed border detection by creating a window to detect borders and describing the use of structural similarity index values and comparing them. This research proposed a comparison of the median filter, Gaussian filter, and B-spline filter from the same original image. The results showed that the median filter method outperformed the other methods. This method is used to avoid interference by a smoothing filter to find the contour of the image. The test results showed that this method can reduce noise in images, which clearly displays the contours.

D. MOVING AVERAGE FILTER

The moving average filter helps improve uneven and unstable data or noise. It takes the average of input samples to produce a single output point. As the length of the filter increases, the smoothness of the output also increases, whereas the sharp modulations in the data are made increasingly blunt [23]. The moving average is calculated by (8).

$$\bar{p}_{sm} = \frac{p_m + p_{m-1} + p_{m-2} + \ldots + p_{m(n-1)}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} p_{m-i} \tag{8}$$

where $\bar{p}_{sm}$ is the value of the moving average
p_m is the value of data
n is the number of datasets

When using the orange line data to filter the data in (8), the signal can be filtered using the moving average method. The result is the dotted line as shown in Fig. 5.

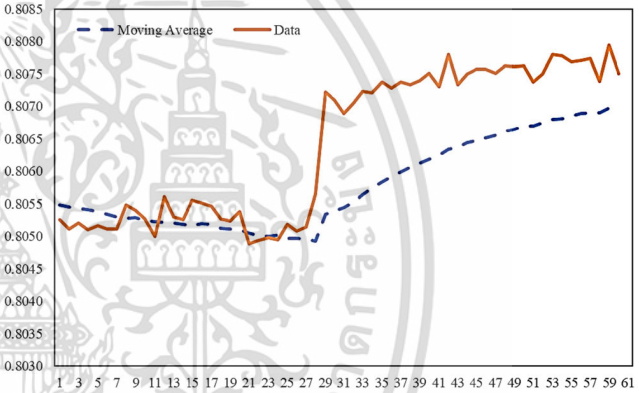


FIGURE 5. Results of image filtering using the moving average method.

In 2010, Morales and Shmaliy [24] introduced the novel technique for the moving average filter method for signals from ultrasound images by taking the value obtained from the filter to create a new image called the Moving Average Hybrid (MAH) method. The development was carried out through the finite impulse response filter (FIR) and the Taylor polynomial approximation. The results showed that the general format of the MAH filter can be used to improve ultrasound images.

In 2012, Azami *et al.* [25] explained how to compare the moving average and Zavitzky-Golay methods to help filter frequency to make the signal height and frequency smoother. The proposed method was compared with the standard method for filtering high pass filter noise. In the experiment, the proposed method was better than the standard high pass filter method. Luo *et al.* [26] used the moving average method to improve an ultraviolet (UV) communication system by creating a noise model to study the effect on the developed communication signal. The results showed that the moving average method helps the system endure noise better and perform noticeably better.

In 2015, Al-Odienat and Al-Mbaideen [27] introduced the moving average method to help filter low and

high frequencies by offering signal processing. In 2017, Isufi *et al.* [28] proposed the autoregressive moving average (ARMA) method to increase the efficiency of the moving average in filtering analog signals and reducing the error term until the signal is reliable. This method has been used to forecast the current signal value, which means it is related to the signal value obtained in the past.

A year later, Kolkova [29] adopted a moving average calculation method by forecasting the exponential smoothing graph to help reduce the severity of the swing of the signal and make the graph smoother. It is used as a technical analysis indicator to predict prices in food industries.

In the same year, Raudys and Pabarškaitė [30] presented a moving average method used to prevent financial signal swing. The determination of the moving average that is appropriate for the financial signal increases the efficiency in the analysis of the stock market. Gonzalez and Catania [31] adopted a moving average calculation method used with inexpensive sensors and compared the data values obtained from expensive sensors. The experimental results indicated that adjusting the moving average used with the inexpensive sensors had similar answer values as the expensive sensors.

III. PROPOSED METHODS

The process of the proposed methods begins with opening the oil pipe through the process of squeezing oil from coconuts. When the coconut oil flows through the pipeline to the oil turbidity detection position, the oil transparency is checked. The oil transparency checking process consists of 3 items: lighting shield box, transparency tube, and lighting inside. The lighting shield box protects the light from outside. The transparency tube is made from glass and stainless-steel pipe, which are used in the food industry. It has a hole to allow light to pass through. The lighting shield box is drilled on all 4 sides and closed with glass and silicon. An overview of the process is shown in Fig. 6.

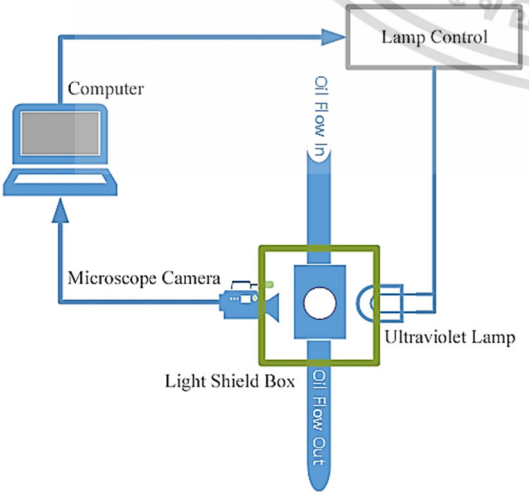


FIGURE 6. Framework of turbidity detection.

Fig. 6 describes the operation of the system as follows. Coconut oil is taken in a position to detect oil turbidity by a USB digital microscope at 50-100x magnification. The images are taken by light passing through the coconut oil at the front of the microscope lens. A black-light at wavelength 410 nm. (nanometer) is used. The algorithms run using Microsoft Visual Studio C # with Library Emgu CV and AForge.net.

A. EXPERIMENTAL DATA

At the beginning stage of data analysis, a total of 1,409 coconut oil images from three different production periods were collected and recorded in a bitmap color image (bmp) with an image size of 1,280 × 1,024 pixels to be used as reference images. This dataset is called the coconut oil determine turbidity (CCODT). It was composed of coconut oil images that were taken when light was emitted by an oil imaging device through the camera. The coconut oil images obtained at various turbidity levels from the highest to the lowest level are shown in Fig. 7(a) to Fig. 7(b), respectively. Other images of the CCODT dataset are displayed in Fig 7(c).

(a) Coconut oil image at the highest turbidity level.

(b) Coconut oil image at the lowest turbidity level.

(c) Example of the CCODT image dataset.

FIGURE 7. Example images of coconut oil used in the experiment.

B. EXPERIMENTAL METHODS

The images obtained from the initial reference data were used as samples of the experimental images for comparison in the proposed methods to determine the brightness of the sample coconut oil, which was sequenced on a linear scale to find the maximum and minimum values of light that passed through the oil to the front of the camera lens. We performed the experiments to compare the following 4 methods: 1) Median of Gray Scale method (MoGS), 2) Median of Hue method (MoH), 3) Random Position Median of Hue method (RPMoH), and 4) Moving Average Median of Hue method (MAMoH). The details of each method can be defined as follows:

1) METHOD 1: MEDIAN OF GRAY SCALE METHOD (MoGS)

This method is available for converting RGB to grayscale images. All pixels of the grayscale image are taken to find the average value of the center of the image. After that, the average value of the grayscale image is recorded. The pseudocode of the MoGS method is shown in Algorithm 1.

Algorithm 1 : MoGS Method

```
    Input: Im = coconut oil image
    // image with values from R = 0 to 255
      G = 0 to 255 and B = 0 to 255 for each pixel
1   FOR i = 1 to 1,409 images
2       image = Im (i)
          // convert RGB image to grayscale
             image numberofshades are between 0-255
3       FOR all the RGB image pixels
          ConvertFactor = 255 / (Numberofshades - 1)
          Avg_Value = (Red + Green + Blue) / 3
          Im_Gray = Integer ((Avg_Value/ConvertFactor)
               + 0.5) * ConvertFactor
        END FOR
          // generate (Im_Gray) Matrix of grayscale image
from the value obtained
4       M_Gray <– Median (Im_Gray)
          // find the median for value in Im_Gray
5       store [i] = M_Gray // save to array
6   END FOR
```

The average record of all grayscale images is used to find boundary values of the data, which can be obtained from the minimum and maximum values of the data to be distributed to find the data separation layer from the CCODT dataset. The boundary values derived from the MoGS method were 0.8260 and 0.5263 for the maximum and minimum values, respectively. The boundary values in the MoGS method are shown in Table 1.

2) METHOD 2: MEDIAN OF HUE METHOD (MoH)

The MoH method is a method for applying hue color in the HSV domain to find the median value of an image. Abdullah [8] discovered that hue in the HSV domain color is better resistant to noise than an RGB color domain. The MoH method has the following steps. First, the CCODT dataset is processed by blurring the image using the Gaussian method at 70% to reduce noise in the image. Second, the color domain is changed from RGB to HSV, and the HSV image obtained from the image conversion RGB to HSV is taken. After that, only the hue layer in the HSV domain is selected. Then, in the third step, the value obtained from all pixels of the hue image is taken to find the median value and is recorded. The MoH method can be explained as shown in Algorithm 2.

TABLE 1. The comparison of data boundary values with the maximum and minimum values from the reference image in each method.

Value	MoGS method	MoH method	RPMoH method	MAMoH method
Minimum	0.5263	0.8045	0.8031	0.8047
Maximum	0.8260	0.8337	0.8352	0.8336

Algorithm 2 : MoH Method

```
    Input: Im = coconut oil image
      // image with values from R = 0 to 255
        G = 0 to 255 and B = 0 to 255 for each pixel
1   FOR i = 1 to 1,409 images
2       image = Im (i)
3       Blur_Im <– Gaussian (image,70)
          // blur image for reduce noise at 70%
4       Im_HSV = HSV <– RGB(Blur_Im(i))
          // convert domain color RGB to HSV
5       H_Im = Select H from Im_HSV
          // split H color from Im_HSV
        M_H_Im <– Median(H_Im)
          // find the median for value in H_Im
7       store [i] = M_H_Im
          // save value to array
8   END FOR
```

After that, all images in the dataset are processed by the MoH method to record all values. The boundary values of the data that can be found are the minimum value of 0.8045 and the maximum value of 0.8337. The results of the data according to the MoH method are shown in Table 1.

3) METHOD 3: RANDOM POSITION MEDIAN OF HUE METHOD (RPMoH)

The RPMoH method considers sampling some positions in the image. This method is improved from considering the entire images to some random positions, resulting in less processing time. The RPMoH method has the following procedures. In the first step, the CCODT dataset is processed by blurring the image using Gaussian. This value is used to reduce noise in the image in the same way as in themes method. In the second step, the image is converted from an RGB to an HSV image, and the hue layer of the HSV domain is taken to determine 10 random positions of pixels in the

image. Next, the hue values from pixels are taken to find the median value that is derived from 10 random positions. The RPMoH method is explained in Algorithm 3.

Algorithm 3 : RPMoH Method

Input: Im = coconut oil image
// image with values from R = 0 to 255
G = 0 to 255 and B = 0 to 255 for each pixel
FOR i = 1 to 1,409 images
  image = Im (i)
  Blur_Im <– Gaussian (image,70)
    // blur image RGB with value 70 % of image
  Im_HSV = HSV <– RGB(Blur_Im(i))
    // convert Blur_Im from domain color RGB to HSV
  H_Im = Select H from Im_HSV
    // split H color from Im_HSV
    // random position (x,y) for select pixel in image
  [XIm,YIm] = size Of H_Im // find size Image
  FOR 10 Positions
    X[i] = Random [0-XIm]
    Y[i] = Random [0-YIm]
    STORE RH_Im[i] <– Read Data H in Position X[i], Y[i]
  END FOR
  M_RH_Im <– Median(RH_Im)
    // find the median for value in RH_Im
  store [i] <– M_RH_Im // save value to array
END FOR

After that, all images in the CCODT dataset are processed by the RPMoH method to record all values. The boundary values of the data that can be found are the minimum value of 0.801 and the maximum value of 0.832. The results of the data according to the RPMoH method are shown in Table 1.

4) METHOD 4: MOVING AVERAGE MEDIAN OF HUE METHOD (MAMoH)

The MAMoH method is an improved method based on the RPMoH method to determine the median value of the hue layer. The MAMoH method has a process for calculating the median value from the hue layer in the HSV domain color. Then, the moving average method is used by taking the average value obtained from the median value of the hue layer, which is the average of the calculation of the previous median hue value with the current value. In the moving average method, the window size of the data must be determined for consideration. In the experiment of different sizes of moving windows, sizes of 3, 7, 9, 11, and 21 were specified. The comparison results showed the moving window size at 7 data ranges that are suitable for the underlying reference of this dataset. If the specified size of the moving window was more than 9 data ranges, the results obtained from the moving window method were lower than the reality of the data. The comparison of the results of using different window sizes is shown in Fig. 8.

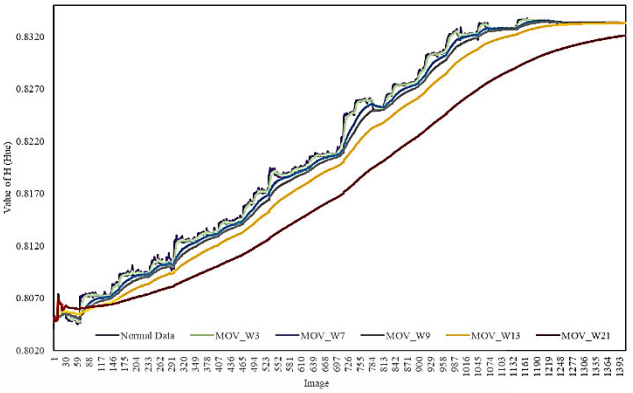


FIGURE 8. Comparison results using different window sizes.

When the values obtained by the moving window are used with a total of different sizes, it was found that the optimum size is at 7-value of data ranges of the moving window. After the data were passed through the moving average method, the result was still within the range of the original data. This experiment found that the new graph is smoother than the original graph when using the MAMoH method. The MAMoH method can be explained in Algorithm 4.

Algorithm 4 : MAMoH Method

Input: Im = coconut oil image
// image with values from R = 0 to 255
G = 0 to 255 and B = 0 to 255 for each pixel
FOR i = 1 to 1,409 images
  image = Im (i)
  Blur_Im <– Gaussian (image,70)
    // blur image for reduce noise at 70%
  Im_HSV = HSV <– RGB(Blur_Im(i))
    // convert domain color RGB to HSV
  H_Im = Select H from Im_HSV
    // split H color from Im_HSV
  HIM <– Median (H_Im)
    // find the median for value in H_Im
  store[i] = (HIM(i-1) + HIM(i-2)+ HIM(i-3) + · · · + HIM(i-n))/n
    // moving average value
END FOR

The experiment was carried out with simulated data from different coconut oil turbidity levels with a total of 1,409 images. The image data from the CCODT dataset were tested by four methods. The results are shown in Table 1.

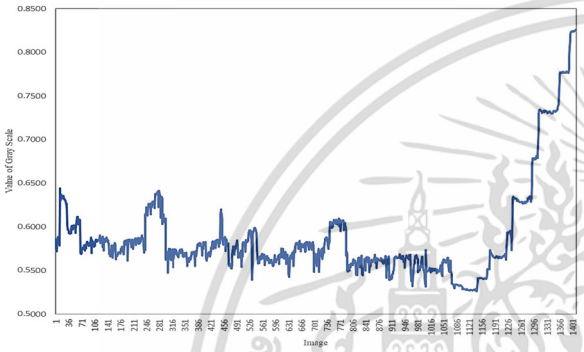
IV. EXPERIMENTAL RESULTS

The experiments were classified into 2 experiments. The first was an experiment focused on performance measurements of the 4 proposed algorithms to consider the most appropriate method in solving this problem. The second experiment focuses on applying the best method derived from the first

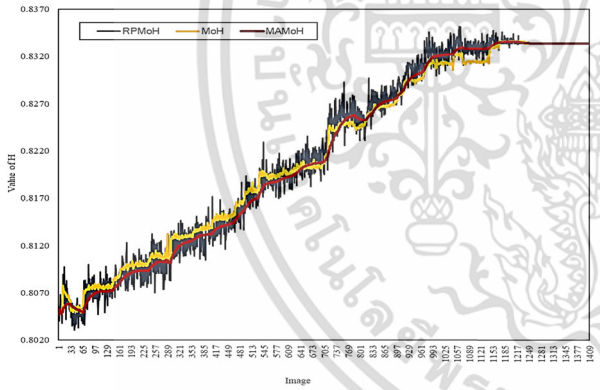
experiment to measure the turbidity of coconut oil in actual usage.

A. PERFORMANCE COMPARISON OF THE 4 PROPOSED METHODS

The CCODT dataset is used to compare all 4 proposed methods; the MoGS method examines the grayscale domain, and the MoH, RPMoH, and MAMoH methods examine the intensity of color from the HSV domain. The comparison results of all 4 methods are shown in Fig. 9, and the data range is classified into 5 turbidity levels from all 4 proposed methods, as shown in Table 2.



(a) The grayscale images domain.



(b) Intensity of color from the HSV domain.

FIGURE 9. Sample of images using Gaussian filtering method.

Table 2 describes the division of the data range in each image of the CCODT dataset derived from the 4 proposed methods for determining coconut oil turbidity. The results show the minimum and maximum values of each method and 5 turbidity levels: level 1 is worse, level 2 is bad, level 3 is fair, level 4 is good, and level 5 is excellent.

Fig. 9(a) shows the results derived from the MoGS method. The graph is a result of grayscale images from the CCODT dataset. Considering this graph, it can be indicated that the graph fluctuated and the line was not smooth. Therefore, this graph cannot be used further. The experiments of the MoGS method found that the minimum and maximum values of grayscale images in the CCODT dataset are 0.5263 and 0.8260, respectively.

Fig. 9(b) shows the comparison of data from the color intensity in the H domain using the MoH, RPMoH, and MAMoH methods in the same HSV color domain. According to the CCODT dataset, when using the MoH method, the result is displayed as a yellow line graph, as shown in Fig. 9(b). The experimental results of the MoH method based on the CCODT dataset show that the minimum and maximum values are 0.8045 and 0.8337, respectively.

According to the CCODT dataset, when using the RPMoH method, the result is displayed as a blue line graph, as shown in Fig. 9(b). The blue line graph is not continuous. The experimental results of the RPMoH method based on the CCODT dataset show that the minimum and maximum values are 0.8031 and 0.8352, respectively. Similar to the RPMoH method, the last method is the MAMoH method, which is an improvement over the MoH method. The MAMoH method used the moving average from the center of the intensity of the image. The result is displayed in a red line graph, as shown in Fig. 9(b). It can be seen that this graph is continuous. The experimental results of the MAMoH method based on the CCODT dataset show that the minimum and maximum values are 0.8047 and 0.8336, respectively.

Fig. 10(a) shows that the results from the CCODT dataset were processed using the MoGS method. Each image was compared with the data division according to the turbidity level of coconut oil in Table 2. The comparison results show that the MoGS method is unable to classify the correct turbidity level value of coconut oil.

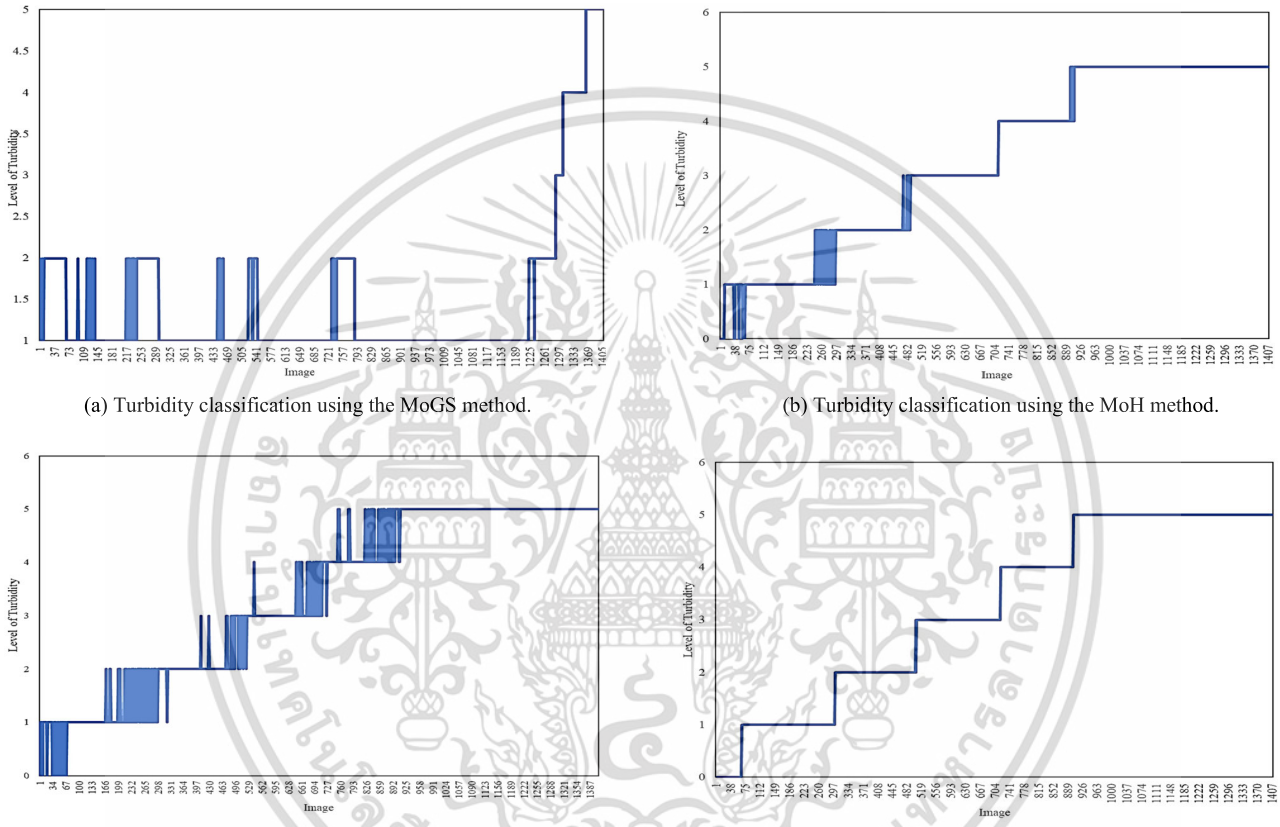
Fig. 10(b) shows the results from the CCODT dataset processed using the MoH method. Each image was compared with the data division according to the coconut oil turbidity level in Table 2. The comparison results show that the MoH method can classify only some parts of the data range. After 5 iterations, it was found that around 166 – 169 coconut oil images were not clearly classified in each iteration. Therefore, the MoH method has 88.04 percent accuracy.

Fig. 10(c) shows that the results from the CCODT dataset were processed using the RPMoH method. The RPMoH method is an improvement of the MoH method in terms of improving image processing time. The comparison results of the coconut oil turbidity level values are shown in Table 2. The results from the CCODT dataset show that using the RPMoH method, the coconut oil turbidity level value achieved lower accuracy than the MoH method even though it performed well in less processing time. After 5 iterations, it was found that around 199 – 202 coconut oil images were not clearly classified in each iteration. Therefore, the RPMoH method has 85.91 percent accuracy.

Fig. 10(d) shows the results from the CCODT image dataset that were processed using the MAMoH method. The MAMoH method is an improvement of the MoH method as well as the RPMoH method, but it performs better. After 5 iterations, it was found that only 1 or 2 coconut oil images were not clearly classified in each iteration. The MAMoH method has 99 percent accuracy.

TABLE 2. The turbidity level of coconut oil.

Value	MoGS method		MoH method		RPMoH method		MAMoH method	
	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX
5 excellent	0.7661	0.8260	0.8279	0.8337	0.8287	0.8352	0.8278	0.8336
4 Good	0.7061	0.7661	0.8220	0.8279	0.8223	0.8287	0.8220	0.8278
3 fair	0.6462	0.7061	0.8162	0.8220	0.8159	0.8223	0.8162	0.8220
2 communicable	0.5862	0.6462	0.8104	0.8162	0.8095	0.8159	0.8105	0.8162
1 Bad	0.5263	0.5862	0.8045	0.8104	0.8031	0.8095	0.8047	0.8105



(a) Turbidity classification using the MoGS method.

(b) Turbidity classification using the MoH method.

(c) Turbidity classification using the RPMoH method.

(d) Turbidity classification using the MAMoH method.

FIGURE 10. Turbidity classification using the four proposed methods.

It can be concluded in the experiments that the MAMoH method has the best accuracy value of the turbidity levels when compared to all 4 methods. Therefore, the MAMoH method was chosen for the first experiment to be used in the second experiment.

B. THE ACTUAL USE OF THE MAMoH METHOD IN OIL FILTERING PRODUCTION

In the second experiment, a light transmission device and oil pipeline from the coconut oil centrifuge machine were installed. After that, the impurities in the oil were filtered to 0.25 microns to obtain pure oil without contamination before packing. The equipment installation is shown in Fig. 11. Prior to each experiment, samples of coconut oil were collected for comparison in each production cycle, as shown in Fig. 12.

In the experiments, we analyzed coconut oil that was processed by centrifugation to extract coconut oil for production cycles. Each production cycle was from different dates and times. The first round of trial usage was on January 21, 2019, the second round was on January 25, 2019, and the third round was on February 27, 2019. The coconut oil samples for each day are shown in Table 3.

According to the comparison results obtained from the simulated data for testing in the 4 proposed methods, the MAMoH method outperformed the others, with the best results for the turbidity levels in the separation of coconut oil. Therefore, in this experiment, the MAMoH with coconut oil flowed through the oil pipeline to the tank to wait for packing into containers. The program was developed for use in the turbidity determination of coconut oil, which can

TABLE 3. Coconut oil production information for each production round.

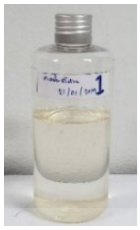
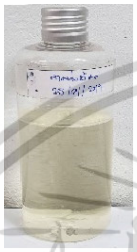
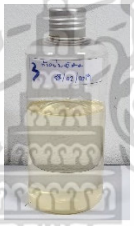
Round	Oil sample	Descriptions
Round 1 21 January 2019		Coconut oil obtained from this production cycle has a clear color like normal water because in the process of producing coconut oil, there are only a few parts of the coconut shell combined with the coconut oil. Therefore, when using coconut oil that passed through the finest filtering at 1.5 microns, coconut oil is clear like water.
Round 2 25 January 2019		Coconut oil obtained from this production cycle has a yellow color due to changing the production process. Large machinery was used to help crack the coconut shells resulting in more scraps of coconut shells found in the production process. However, the amount of coconut oil production also increased. After that, the coconut oil is passed through the hot air dehydration process and compressed until the coconut oil is obtained.
Round 3 27 February 2019		Coconut oil obtained from this production cycle is more yellow than the second cycle. It is caused by the process of heating to expel the moisture of the coconut. In this production cycle, it took longer than in the second cycle, causing the color of the coconut oil to be more yellow than the color of the second cycle.

FIGURE 11. Sample of coconut oil obtained after each centrifuge process.

FIGURE 12. Location of the device installation which coconut oil passing through before entering the tank.

be measured in numerical values. The program interface is shown in Fig. 13.

The operation of the program begins by sending a command to the camera to take photos every seconds. Then, the images are processed according to the MAMoH method. The time required for each experiment was 14 minutes, which caused the coconut oil to flow through the pipes into the tank until all coconut oil was loaded into the product packages. The image processing data according to the MAMoH method, when used to test the coconut oil turbidity in all 3 production cycles, obtained the results shown in Table 4. According to Table 4, it can be found from the coconut oil turbidity measurement test for production cycles that the measured values in all production cycles had turbidity levels at level or excellent compared to the results in Table 2. In the first round,

FIGURE 13. The turbidity of coconut oil interface.

TABLE 4. Turbidity measurement values of coconut oil from actual use by the MAMoH method.

Round	Minimum value	Maximum value
1	0.8320	0.8321
2	0.8311	0.8315
3	0.8300	0.8302

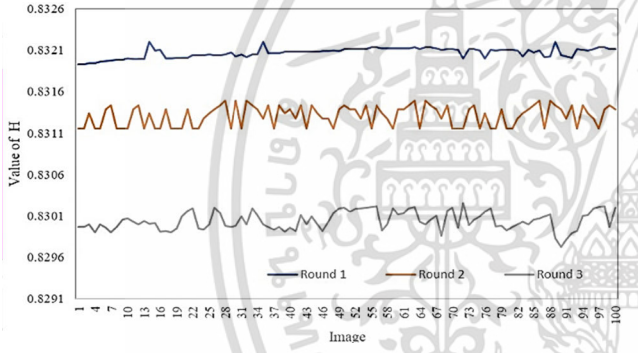


FIGURE 14. Comparison of the measurement results of coconut oil in 3 rounds.

the coconut oil turbidity was between 0.8320 and 0.8321, which is as clear as that of normal water. In the production process, contaminants such as the inner shell of the coconut shell or fiber from the coconut shell mixed with the coconut were present in small amounts; therefore, the coconut oil had less turbidity. In the second round, the coconut oil turbidity was between 0.8311 and 0.8315. The results showed that the turbidity was lower than that in the first round. The cause was the change in machinery to increase production capacity in the factory resulting in the cracking process. There were more contaminants than were found in the first round. Moreover, the moisture repellency of coconut also increased. When considering the color of coconut oil with the naked eye, it can be found that the coconut oil started to have a light-yellow color.

In the third round, the coconut oil turbidity was between 0.8300 and 0.8302. In this production cycle, the oil is more yellow than in the second cycle due to the moisture content of the coconut or drying process. Excessive heat with the coconut causes the coconut to turn yellow. The production process of squeezing the oil from the coconut makes the coconut oil more yellow than the oil in the second round. The comparison of the turbidity values of each production cycle is shown in Fig. 14.

According to Fig. 14, the result of the determination of the coconut oil turbidity using the MAMoH method in all 3 experiments showed that the coconut oil has different colors but still has the same turbidity level. The results of the coconut oil turbidity measurement are shown in Table 2. Each round of coconut oil was tested under the quality inspection department of the coconut oil factory; it was found that the coconut oil passed the quality criteria set by the factory.

V. CONCLUSION

Among the many methods used to determine coconut oil turbidity, the proposed methods can help in the production process of coconut oil industries. It can be used for the transparency level of coconut oil measurement in each production cycle, which increases the efficiency of the manufacturing process. In the proposed method, the MAMoH algorithm achieved the best experimental result in classifying the degree of coconut oil turbidity at 99%. An error of 1% was found due to the effect of incomplete data processing when the moving window method was used in the MAMoH algorithm. According to the factory's production experimental results, the MAMoH algorithm could clearly classify the transparency level of coconut oil when compared with other methods from the 3 cycles of the experiments. This algorithm was stable to the variation in the imported data when the new coconut oil production process was modified. The MAMoH algorithm could still classify the correct degree of coconut oil turbidity. The precaution in working with oil photography equipment requires a control circuit and a good quality built-in cooling lamp. If the temperature of the control circuit and the lamp used are too high, it may cause the light of the lamp to flicker. All of the proposed methods rely on the constant light of a bulb that is emitted through the oil. Therefore, inspection and maintenance of oil imaging equipment are essential in determining coconut oil turbidity. In addition, this research can be further developed to investigate contaminants such as dust or coir particles in coconut oil.

ACKNOWLEDGMENT

This research was supported in testing part by Tropicana Oil Co., Ltd. and funding part by Faculty of Science, King Mongkut's Institute of Technology Ladkrabang.

REFERENCES

[1] D. D. Bawalan and K. R. Chapman, "Virgin coconut oil," in *Production Manual for Micro and Village-Scale Processing*, 1st ed. Bangkok, Thailand: Thammada Press, 2006.
[2] B. Gordillo, L. Ciaccheri, A. G. Mignani, M. L. Gonzalez-Miret, and F. J. Heredia, "Influence of turbidity grade on color and appearance of virgin olive oil," *J. Amer. Oil Chemists Soc.*, vol. 88, no. 9, pp. 1317–1327, Sep. 2011.
[3] C. En, Z. Rong-Xin, and Y. Fei, "Automatic detection and assessment system of water turbidity based on image processing," *Telkomnika Indonesian J. Electr. Eng.*, vol. 11, no. 3, pp. 1506–1513, Mar. 2013.

[4] A. B. Riaño, I. H. Rodriguez, A. C. Bannwart, and O. M. H. Rodriguez, "Film thickness measurement in oil–water pipe flow using image processing technique," *Exp. Therm. Fluid. Sci.*, vol. 68, pp. 330–338, Nov. 2015.
[5] Y. Liu, Y. Chen, and X. Fang, "A review of turbidity detection based on computer vision," *IEEE Access*, vol. 6, pp. 60586–60604, 2018.
[6] J. D. Chang, S. S. Yu, H. H. Chen, and C. S. Tsai, "HSV-based color texture image classification using wavelet transform and motif patterns," *J. Comput.*, vol. 20, no. 4, pp. 63–69, Jan. 2010.
[7] M. Deswal and N. Sharma, "A fast HSV image color and texture detection and image conversion algorithm," *Int. J. Sci. Res.*, vol. 3, no. 6, pp. 1279–1284, Jun. 2014.
[8] A. S. Abdullah, "Text hiding based on hue content in HSV color space," *Int. J. Emerg. Trends Technol. Comput. Sci.*, vol. 4, no. 2, pp. 170–173, Mar./Apr. 2015.
[9] V. Chernov, J. Alander, and V. Bochko, "Integer-based accurate conversion between RGB and HSV color spaces," *Comput. Electr. Eng.*, vol. 46, pp. 328–337, Aug. 2015.
[10] G. Saravanan, G. Yamuna, and S. Nandhini, "Real time implementation of RGB to HSV/HSI/HSL and its reverse color space models," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Melmaruvathur, India, Apr. 2016, pp. 0462–0466.
[11] T. Kumar and K. Verma, "A theory based on conversion of RGB image to gray image," *Int. J. Comput. Appl.*, vol. 7, no. 2, pp. 7–10, Sep. 2010.
[12] R. Vijaya, K. Prudvi, L. Ravi, and M. Jogendra, "Grey level to RGB using YCbCr color space technique," *Int. J. Comput. Appl.*, vol. 147, no. 7, pp. 25–28, Aug. 2016.
[13] K. H. Cheong, S. Poeschmann, J. W. Lai, J. M. Koh, U. R. Acharya, S. C. M. Yu, and K. J. W. Tang, "Practical automated video analytics for crowd monitoring and counting," *IEEE Access*, vol. 7, pp. 183252–183261, 2019, doi: 10.1109/ACCESS.2019.2958255.
[14] J. A. Stark, "Adaptive image contrast enhancement using generalizations of histogram equalization," *IEEE Trans. Image Process.*, vol. 9, no. 5, pp. 889–896, May 2000.
[15] K. S. Sim, C. P. Tso, and Y. Y. Tan, "Recursive sub-image histogram equalization applied to gray scale images," *Pattern Recognit. Lett.*, vol. 28, no. 10, pp. 1209–1221, Jul. 2007.
[16] M. M. N. Ali and M. Abdullah-Al-Wadud, "Image enhancement using a modified histogram equalization," in *Computer Applications for Web, Human Computer Interaction, Signal and Image Processing, and Pattern Recognition*. Berlin, Germany: Springer, 2012, pp. 17–24.
[17] J. Lee, S. R. Pant, and H.-S. Lee, "An adaptive histogram equalization based local technique for contrast preserving image enhancement," *Int. J. Fuzzy Log. Intell. Syst.*, vol. 15, no. 1, pp. 35–44, Mar. 2015.
[18] R. Agarwal, "Bit plane average filtering to remove Gaussian noise from high contrast images," in *Proc. Int. Conf. Comput. Commun. Informat.*, Jan. 2012, pp. 1–5.
[19] R. Chandel and G. Gupta, "Image filtering algorithms and techniques: A review," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 10, pp. 198–202, Oct. 2013.
[20] F. Codevilla, J. D. O. Gaya, N. D. Filho, and S. S. C. C. Botelho, "Achieving turbidity robustness on underwater images local feature detection," in *Proc. Brit. Mach. Vis. Conf.* Swansea, U.K.: BMVA Press, Sep. 2015, p. 154, doi: 10.5244/c.29.154.
[21] T. Tyagi and V. Mishra, "2D Gaussian filter for image processing: A study," *Int. J. Sci. Technol. Eng.*, vol. 3, no. 6, pp. 22–24, Dec. 2016.
[22] P. Amoako-Yirenkyi, J. K. Appati, and I. K. Dontwi, "Performance analysis of image smoothing techniques on a new fractional convolution mask for image edge detection," *Open J. Appl. Sci.*, vol. 6, no. 7, pp. 478–488, Jan. 2016.
[23] S. W. Smith, "Moving average filters," in *The Scientist and Engineer's Guide to Digital Signal Processing*. San Diego, CA, USA: California Technical Publishing, 1997, ch. 15, pp. 277–284.
[24] L. J. Morales and Y. Shmaliy, "Moving average hybrid filter to the enhancing ultrasound image processing," *IEEE Latin Amer. Trans.*, vol. 8, no. 1, pp. 9–16, Mar. 2010, doi: 10.1109/TLA.2010.5453940.
[25] H. Azami, K. Mohammadi, and B. Bozorgtabar, "An improved signal segmentation using moving average and Savitzky-Golay filter," *J. Signal Inf. Process.*, vol. 3, no. 1, pp. 39–44, Feb. 2012, doi: 10.4236/jsip.2012.31006.
[26] P. Luo, M. Zhang, Y. Liu, D. Han, and Q. Li, "A moving average filter based method of performance improvement for ultraviolet communication system," in *Proc. 8th Int. Symp. Commun. Syst., Netw. Digit. Signal Process. (CSNDSP)*, Warsaw, Poland, Jul. 2012, pp. 18–20, doi: 10.1109/CSNDSP.2012.6292672.
[27] A. I. Al-Odienat and A. A. Al-Mbaideen, "Optimal length determination of the moving average filter for power system applications," *Int. J. Innov. Comput., Inf. Control*, vol. 11, no. 2, pp. 691–705, Apr. 2015.
[28] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 274–288, Oct. 2017.
[29] A. Kolkova, "Indicators of technical analysis on the basis of moving averages as prognostic methods in the food industry," *J. Competitiveness*, vol. 10, no. 4, pp. 102–119, Dec. 2018, doi: 10.7441/joc.2018.04.07.
[30] A. Raudys and Ž. Pabarškaitė, "Optimising the smoothness and accuracy of moving average for stock price data," *Technol. Econ. Develop. Economy*, vol. 24, no. 3, pp. 984–1003, May 2018, doi: 10.3846/20294913.2016.1216906.
[31] R. Gonzalez and C. A. Catania, "A statistical approach for optimal order adjustment of a moving average filter," in *Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS)*, Monterey, CA, USA, Apr. 2018, pp. 1542–1546, doi: 10.1109/PLANS.2018.8373549.

ATTAPON PALANANDA received the B.S. degree in technology education (computer engineering) from the Rajamangala University of Technology Thanyaburi, Thailand, and the master's degree from the King Mongkut's University of Technology North Bangkok, Bangkok, Thailand. His main research interests include computer vision, image processing, and artificial intelligence.

WARANGKHANA KIMPAN (Member, IEEE) received the Ph.D. degree in system information engineering from Kagoshima University, Japan. She is currently an Assistant Professor with the Department of Computer Science, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. Her main research interests include swarm intelligence, biomedical engineering, big data, data science and analytics, cloud computing, and the Internet of Things.

applied sciences

MDPI

Article

Classification of Adulterated Particle Images in Coconut Oil Using Deep Learning Approaches

Attapon Palananda and Warangkhana Kimpan *

Department of Computer Science, School of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand; 57605017@kmitl.ac.th

* Correspondence: warangkhana.ki@kmitl.ac.th

Abstract: In the production of coconut oil for consumption, cleanliness and safety are the first priorities for meeting the standard in Thailand. The presence of color, sediment, or impurities is an important element that affects consumers' or buyers' decision to buy coconut oil. Coconut oil contains impurities that are revealed during the process of compressing the coconut pulp to extract the oil. Therefore, the oil must be filtered by centrifugation and passed through a fine filter. When the oil filtration process is finished, staff inspect the turbidity of coconut oil by examining the color with the naked eye and should detect only the color of the coconut oil. However, this method cannot detect small impurities, suspended particles that take time to settle and become sediment. Studies have shown that the turbidity of coconut oil can be measured by passing light through the oil and applying image processing techniques. This method makes it possible to detect impurities using a microscopic camera that photographs the coconut oil. This study proposes a method for detecting impurities that cause the turbidity in coconut oil using a deep learning approach called a convolutional neural network (CNN) to solve the problem of impurity identification and image analysis. In the experiments, this paper used two coconut oil impurity datasets, PiCO_V1 and PiCO_V2, containing 1000 and 6861 images, respectively. A total of 10 CNN architectures were tested on these two datasets to determine the accuracy of the best architecture. The experimental results indicated that the MobileNetV2 architecture had the best performance, with the highest training accuracy rate, 94.05%, and testing accuracy rate, 80.20%.

Keywords: coconut oil; deep learning; convolutional neural networks; image recognition; object detection

Citation: Palananda, A.; Kimpan, W. Classification of Adulterated Particle Images in Coconut Oil Using Deep Learning Approaches. *Appl. Sci.* **2022**, *12*, 656. https://doi.org/10.3390/app12020656

Academic Editor: Byung-Gyu Kim

Received: 10 December 2021
Accepted: 7 January 2022
Published: 10 January 2022

1. Introduction

Coconut oil is a fatty acid-rich oil that is healthful for human consumption. The key to production is maintaining the benefits of the coconut oil. The production process that maintains the highest value of coconut oil is cold pressed, which produces virgin coconut oil. The process starts with the dehumidification of coconut meat at approximately 65–70 degrees Celsius; the meat is then compressed to extract the oil. To remove impurities, a centrifuge is used to circulate the oil through a particle filter. These impurities contribute to the turbidity of the coconut oil. Examples of such impurities are coconut scraps, coconut shell scraps, coconut coir, and dust particles that are retained after the oil compression process. Impurities that remain after the impurity filtering process will become suspended sediment that will sink to the bottom of the container until it can be seen with the naked eye. Large amounts of sediment in the coconut oil can cause turbidity. Turbidity can be checked in coconut oil in the same way as in water. Karnawat and Patil [1] photographed water and applied image processing. Many researchers have also applied image processing methods to help analyze other problems, such as digital image analysis of coarse dust particles or aerosols transmitted by laser light [2], seafloor plankton analysis using image processing and machine learning [3], automatic screening classification of diabetic retinopathy from

fuzzy image processing combined with machine learning [4], photographic analysis of airborne PM2.5 (particulate matter) and PM1.0 adhering to a microscopic glass surface [5], and three-step real-time vehicle tracking method on the road using the Adaboost algorithm together with the image segmentation method [6]. Consequently, this paper uses deep learning methods to identify adulterating objects in coconut oil is proposed. As coconut oil flowed through the pipeline to the storage tank, a microscope was used to photograph the oil in a closed environment. The front lens of the microscope inside receives light from a black-light lamp that shines through the coconut oil. The black light does not reflect light onto objects, making the objects distinct. On the basis of the steps mentioned above, microscopic photographs of the coconut oil were recorded at different times of the production process, resulting in 4725 photographs of the coconut oil.

The next step involved the process of finding and identifying impurities in the coconut oil images. A learning model with the standard neural network architecture was built when the image of the adulterating object in the coconut oil was taken. In this paper, the performances of a standard convolutional neural network (CNN), MobileNetV2, VGGNet16, VGGNet19, GoogLeNet (Inception V3, Xception, and InceptionResNetV2), ResNet50, ResNet101, and DenseNet121 were compared in terms of image processing speed and accuracy. The best-performing model was then used to detect impurities in the coconut oil.

The rest of this paper is structured as follows: Section 2 discusses the literature review. Then, the research methodology is shown in Section 3. Finally, Section 4 presents the research conclusions and future work.

2. Literature Review

2.1. Deep Learning

Deep learning is a subset of machine learning (ML) within artificial intelligence (AI). Based on artificial neural networks (ANNs), deep learning simulates the process of information processing in the human brain. An ANN consists of interconnected cells as the nervous system used for data processing. Deep learning was first presented through the implementation of ML by Rina Dechter [7,8]. It is a method that does not use data manipulation to work through predefined equations; rather, it uses basic data-related parameter settings and teaches computers to experience self-learning by processing to identify the strengths and differences in the data. Deep learning is filtered by layers according to the nature of the neural networks to process the output answers and to examine whether the output is right or wrong. If the examined results of the output data do not match reality, the learning settings will be adjusted and the data reprocessed to increase the accuracy of the output by increasing the number of samples or adding layer depth to consider more details. Therefore, deep learning can consider differences in the data and provide output answers without a need for human suggestions. Therefore, the order of the number of layers is the depth in deep learning.

2.2. Convolutional Neural Networks

The CNN method was introduced by Fukushima which is a neural network with a convolutional layer added. It performs as a filter, defining the attributes of an input image and collecting various features from each image for classification. In 2019, Fukushima [9] developed neocognitron. Unlike standard CNN, neocognitron was used to develop recognition of partly occluded patterns via the mechanism of selective attention. The backpropagation neural network was first stated by Rumelhart to help optimize prediction of data classification by internal restructuring using hidden layers. LeCun [10] tested backpropagation with a CNN on the Modified National Institute of Standards and Technology (MNIST) dataset, a handwritten numeric dataset. Then in 2019, they proposed deep learning hardware: past, present, and future which described the evolution of neural networks and deep learning affected by hardware and software improvements. In 2020, Li et al. [11] proposed improvements of backpropagation to be used for deep transfer learning. They used a

pretrained model to transfer knowledge which learned from larger datasets to the target task. In 2007, Ranzato et al. [12] first presented the use of backpropagation in collaboration with max-pooling CNNs (MPCNNs). Then, in 2018, Brinker et al. [13] proposed a CNN approach to skin cancer screenings. In 2020, Patil and Bellary [14] presented a method for identifying the growth of melanoma cancer cells using a CNN approach.

Newby et al. [15] proposed ANN approach in particle automatic tracking to analyze particle localization and track particle motion from 2D and 3D collected video images obtained through a microscope.

Khairi et al. [16] proposed a method for determining the turbidity level of water samples from the application of X-ray imaging systems. An ANN was then used to help determine the turbidity level. This proposed method was able to predict the change in turbidity level. It helped classify water quality levels and benefited industries that require high water quality.

Rong et al. [17] proposed two different convolutional neural networks using deep learning for automatically segmenting images and detecting different sizes of foreign objects that occur both naturally and man-made, such as dry leaves, scrap of papers, plastic scraps, and metal parts. This proposed structure was applied to walnut images. The experimental results had an accuracy of 99.5% in object segmentation and it was able to correctly classify 95% of the foreign objects and found that the segmentation and detection processing time of each image was less than 50 ms. Ferreira et al. [18] proposed a method for weed detection from aerial photographs of soybean fields with drones. CNN was used for weed classification in four classes: soil, soybean, broadleaf, and grass weeds. They used the CaffeNet framework based on the AlexNet architecture model. The experimental results showed that CNN had an accuracy of 99% which was better than SVM, AdaBoost, and random forest.

The core structure of a CNN consists of the three following principles: first is the input, which is used to receive information or object images as well as human vision. Second is the hidden layer, which is used to process the input data as well as the human brain and can learn to classify things. Last is the output, which shows the results obtained from the processing of the second part. The standard structure of a CNN is shown in Figure 1.

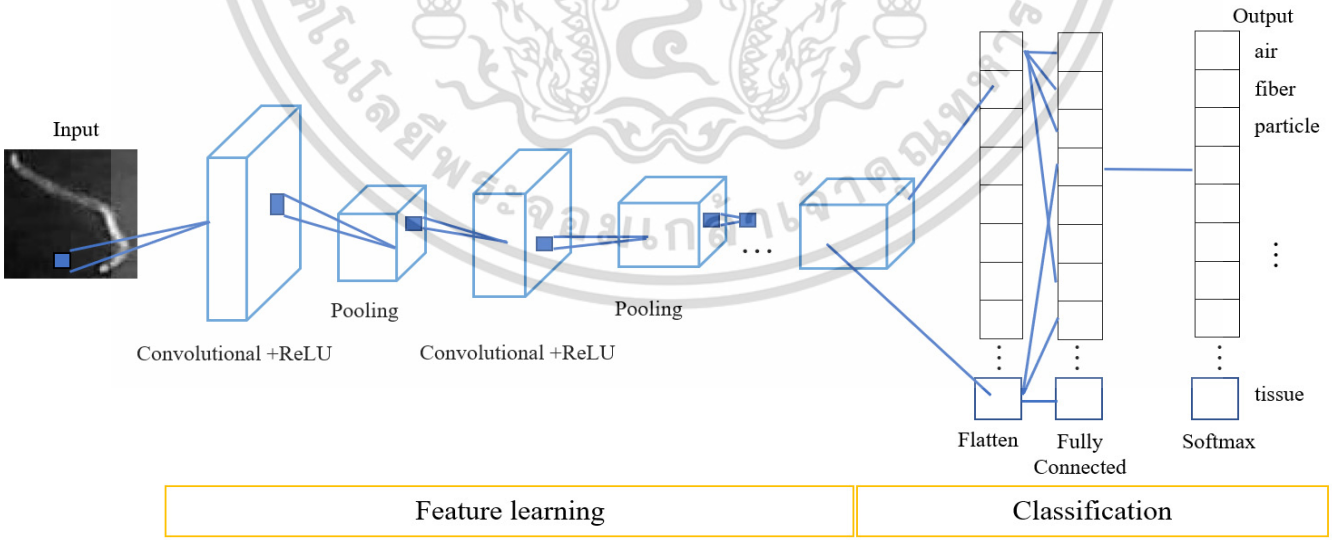


Figure 1. CNN architecture.

Figure 1 describes the general structure of a CNN, which consists of the following parts:

1. Input layer: reads the image input data prior to passing it to the neural network.
2. Convolutional layer: filters the image features from the analysis of each pixel of the image that has been read. The result is a convolutional feature map.
3. Rectified linear unit (ReLU): performs a nonlinear activation function.

4. Pooling layer: provides a subsample rectified feature map to reduce linear dimensions and create a feature representation.
5. Softmax layer: configures the output to display it in the form of a multiclass logistic classifier.
6. Output layer: displays the results of the classification.

The CNN concept was further developed. One method called AlexNet [19], which has eight convolutional layers, competed in the ImageNet Large-Scale Visual Recognition Challenge in 2010 (ILSVRC2010). After that achievement, researchers used the CNN concept to develop various models. For example, in 2014, Simonyan and Zisserman [20,21] presented the VGGNet approach. From 2014 to 2016, the Google team proposed the GoogLeNet, or inception method. Then, in 2017, Howard et al. [22] presented the MobileNet approach, which is a small model that can work quickly and uses fewer processing resources. This has made the MobileNet architecture popular for mobile devices, and we will focus on it in this paper.

2.3. MobileNet

MobileNet is a CNN approach introduced by Howard et al. [22]. It aims to reduce the size of the standard CNN model for use in mobile devices. It can also maintain a performance level close to that of a large-scale deep learning neural network. MobileNet takes an input image with dimensions of $224 \times 224 \times 3$ and passes it through the convolution layers. It demonstrates the ability of small artificial neural networks to recognize objects quickly without the use of a GPU. The structure of MobileNet is shown in Figure 2. In 2020, Pan et al. [23] proposed a MobileNet approach combined with a transfer learning algorithm (TL-MobileNet) for image analysis and recognition. This solved the problem of the new image classification accuracy. DropBlock and global average were added to the original MobileNet layer. The results showed that the model had 97.69% prediction accuracy on the MNIST dataset. Kerf et al. [24] presented detection of oil spills in water using MobileNet architecture which helped decision-making process and provide solutions of oil spills on the surface of water. Drone was used to explore and photograph with an infrared camera which can be used at night. Both RGB and IR (Infrared) images were processed. In the initial process, RGB images encountered with the process of extract water region press, quantify oil in water, and create mask and image resize, while IR images undergo only resize. RGB and IR images were then synchronized and calibrated. The resulting IR images and RGB masks were then used for data augmentation process to train the CNN model. For the testing method, began with the IR images were resized and then the images were taken in a pre-learning CNN on interface device process to detect oil spills on the water surface. As a result, when an oil leak was found, a GPS alerted the oil leak position to the monitoring system.

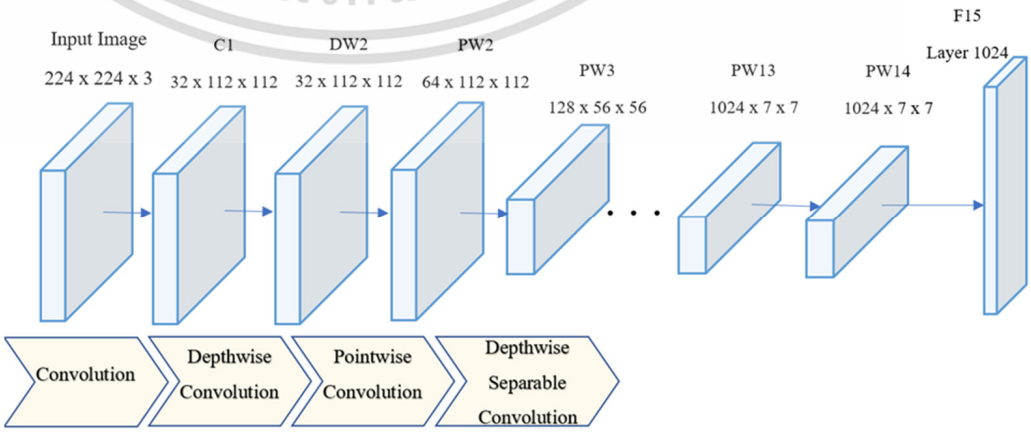


Figure 2. MobileNet architecture.

In 2021, Iqbal et al. [25] presented blockage classification of culverts by remotely applying deep learning models. They used three datasets: images of culvert openings and blockage (ICOB), visual hydrology-lab dataset (VHD), and synthetic images of culverts (SIC), to predict the blockage in the image. The performances of CNN algorithms (DarkNet53, DenseNet121, InceptionResNetV2, InceptionV3, MobileNet, ResNet50, VGG16, EfficientNetB3, NASNet) were compared in terms of image processing response times and accuracy. The results showed that the MobileNet is effective in classification performance and response times.

2.4. VGGNet

VGGNet was developed by the visual geometry group (VGG) at the University of Oxford. In 2014, Simonyan and Zisserman [20,21] proposed a VGG network architecture. This model implemented a 3 × 3 convolutional layer that added a step above the standard CNN model (max-pooling, fully connected, and softmax layers). The experimental results of VGG11, VGG11 (local response normalization (LRN)), VGG13, VGG16 (Conv 1), VGG16, and VGG19 showed that VGG16 and VGG19 had the lowest error values. The numbers 16 and 19 refer to the number of layers. The error value of VGG19 with 19 layers is higher than that of VGG16 with 16 layers, which means that a higher number of layers cannot reduce the error values. In addition, the size of the VGG19 model is larger than that of the VGG16 model. In 2019, Tammina [26] proposed a solution to a limited number of image classifications by training the VGG16 model with the minimum number of samples. The experiment showed the accuracy and loss after the CNN parameter was adjusted, with the lowest accuracy being 79.20%. The number of images was then increased to train the model, and the new accuracy value was 95.40%.

2.5. GoogLeNet

GoogLeNet has 22 stacked convolutional layers. It was developed by Google researchers as a form of inception network. Beginning in 2015, Inception V1 was developed by Szegedy et al. [27] to increase the ability to examine and classify individual objects. This method includes different convolutional (1 × 1, 3 × 3, and 5 × 5) and 3 × 3 max-pooling node sizes, resulting in better performance than that of VGGNet in the Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). After this achievement in the competition, GoogLeNet improved its training performance with a batch normalization technique, known as Inception V2, and later upgraded to Inception V3. It factored the convolutional node into the inception module and made it smaller. Inception V4 improved Inception V3 by adding wider inception modules than those used in Inception V3. Performance tests showed that Inception V4 had the lowest error rate and spent the most time training the model compared to the previous 3 models.

2.6. ResNet

ResNet, or residual neural network, was first presented by He et al. [28]. In this study, deep residual learning for image recognition was used to solve vanishing gradient problems. The authors used residual block learning methods instead of learning certain properties of the image. This approach removes the properties learned from the input of that layer. If the number of layers reaches 152, a shortcut module design technique is applied to the network to cross layers to reduce errors in training the deep layer CNN model. In training ResNet models, the number of parameters is the layer used for naming, such as ResNet50 or ResNet101. The structure of ResNet50 has dimensions (3,4,6,3), which means (3 + 4 + 6 + 3) × 3 = 48 layers + 2 layers, or 50 layers, as shown in Figure 3.

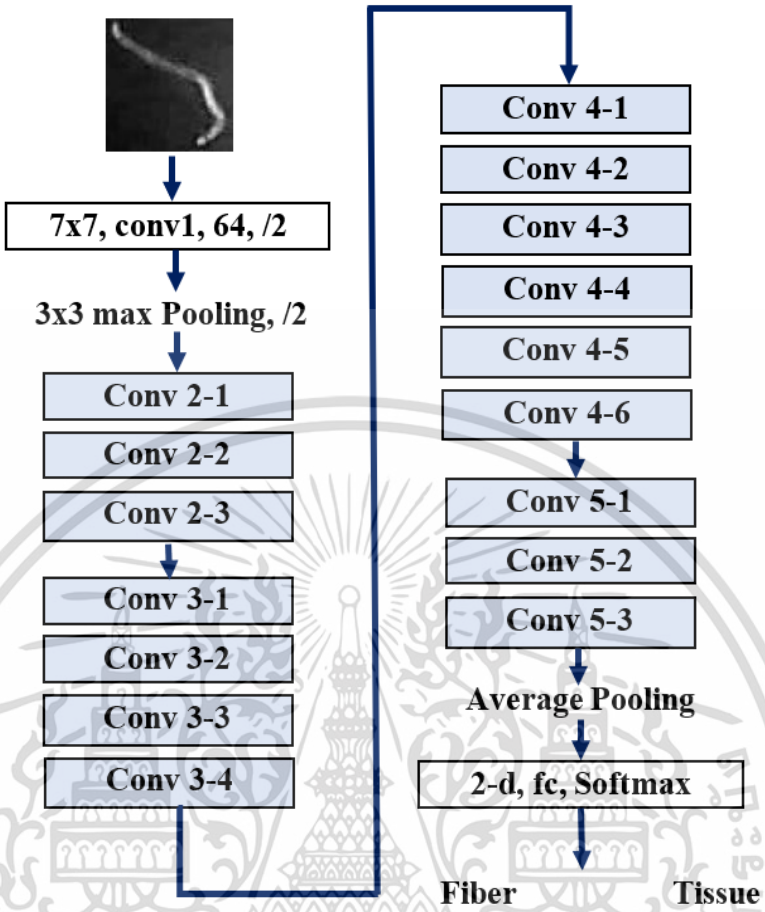


Figure 3. ResNet50 architecture.

In 2020, Aparna et al. [29] proposed the detection of holes from thermal images using convolutional neural networks and the comparison of each ResNet model. The results obtained from the confusion matrix table showed that ResNet101 had an accuracy of 97.08% with thermal images obtained from the FLIR ONE thermal camera.

2.7. DenseNet

DenseNet, or dense convolutional network, was proposed in 2019 by Huang et al. [30] as an innovation codeveloped by Cornwell University and Tsinghua University. In DenseNet, each layer receives input from all previous layers and passes the attributes of its own layer to the following layer. This connection allows each layer to receive data from all previous layers. The connection between the DenseNet data layers is shown in Figure 4.

In 2018, Tao et al. [31] proposed a solution to the problem of ambiguity in the classification of sensitive data that enhanced the depth and width of the deep neural network (DNN) through multiple filter sizes to create a wider neural network. Moreover, the proposed process enables the network to work more smoothly.

Later, Too et al. [32] proposed a method to classify plant diseases into 38 different groups based on images of diseased and unaffected plant leaves of 14 healthy species by comparing VGG16, InceptionV4, ResNet50, ResNet101, ResNet152, and DenseNet121. The results showed that the DenseNet architecture achieved an accuracy of 99.75%, which was the highest accuracy of all the architectures that were tested.

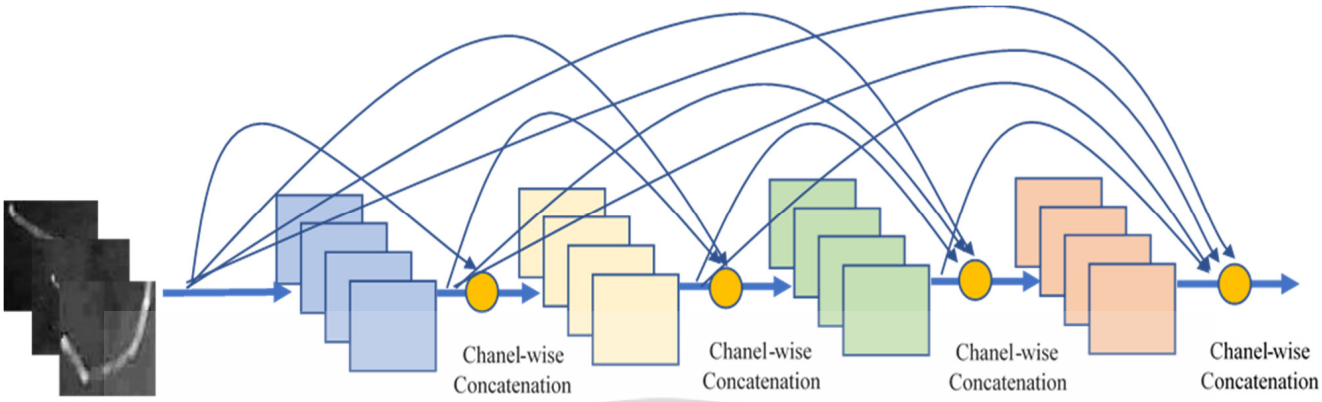


Figure 4. Connection in DenseNet.

3. Research Methodology

This section explains how to obtain impurities from coconut oil images flowing through a microscope. This method was proposed in a previous study by Palananda and Kimpan [33] that focused on determining coconut oil turbidity by image processing. In this paper, we relied on the same method of obtaining images used in our previous study. Coconut oil images were recorded at different time intervals, and ultimately, 4725 images were obtained. All the obtained coconut oil images underwent an image processing technique to find impurities according to the process of finding objects in coconut oil, as shown in Figure 5. The process started with converting the images from RGB (red, green, blue) to BGR (blue, green, red). Then, the images were converted from BGR to grayscale. This reduced 3 layers to 1 layer. Next, the images were converted from grayscale to black-and-white or binary images, which involves converting the image pixel value from 0–255 levels to 0 or 1. Next, the image enhancement process was performed to reduce noise with the closing method, which is one of the commonly used morphological methods. Then, we quantified the total pixels, which meant gathering more than 30 pixels in an image and creating a line around the group of pixels or objects using the Sobel method, which searches for differences in the pixel intensity. This avoids the image noise left over from the image enhancement process. Finally, we used the segmentation method to crop the resulting object, meaning the adulterating object in the coconut oil image that we were interested in.

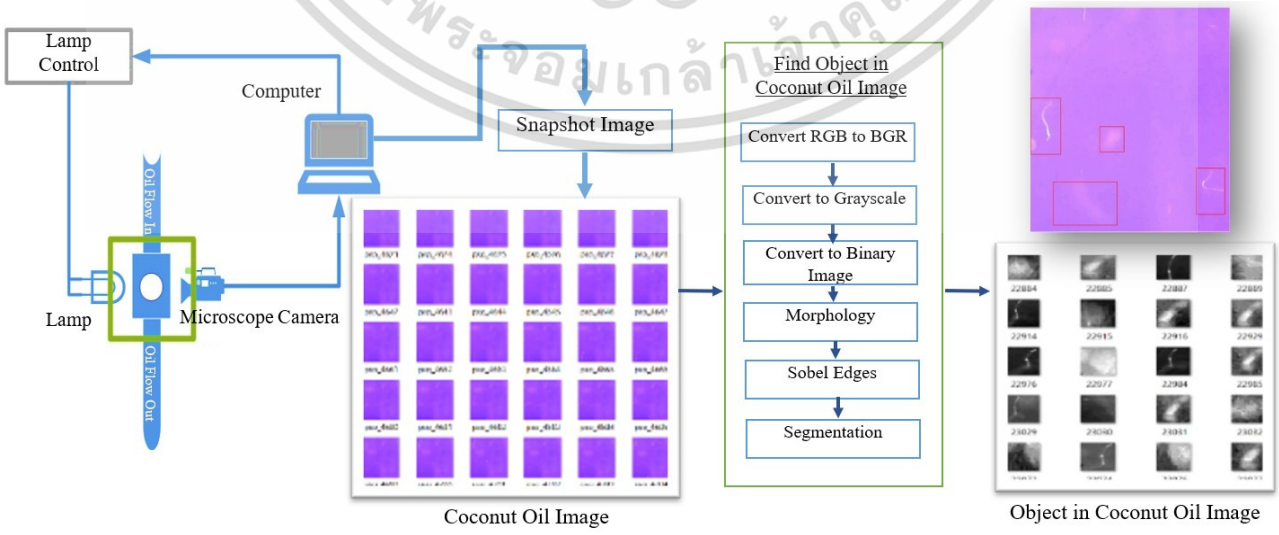


Figure 5. Object image acquisition in coconut oil.

As shown in Figure 5, after the coconut oil images were taken through the process of finding adulterants in coconut oil, a total of 14,784 images were captured, some of which were apparently incomplete. These images were not filtered by experts. Individual object images and indistinguishable or incomplete object images were filtered out, resulting in 7861 images of adulterating objects in the coconut oil.

After that, we classified the image of the impurity objects to new 10 categories as: (1) FiberT1 represents the group of impurity objects derived from yarn fibers from bags of grated coconut pulp in the process of compressing coconut oil, (2) FiberT2 represents representing the group of impurity object derived from the fibers of the coconut pulp, (3) FiberT3 represents the fiber images of coconut coir fibers, (4) FiberT4 represents the fiber images of the inner shell of the coconut shell, (5) ParticleT1 represents the group of impurity object derived from coconut coir, (6) ParticleT2 represents the group of impurity object derived from coconut shell fragments, (7) ParticleT3 represents the group of impurity object from the coconut pulp, (8) ParticleT4 represents the group of impurity object derived from coconut shell dust, (9) Air, and (10) Tissue.

All images in these datasets are the images each type of objects in coconut oil with different motions of objects and different perspectives of the coconut oil images according to the actual conditions arising from each production cycle of coconut oil. As a result, the images from these datasets are similar to the images which used the data augmentation process, including horizontal flip, random shift, rotation, zoom, and brightness of the images.

Next, the data were separated into two parts: the first part was used for the training model, and the other part was used for the test model to determine the model performance after the training model was completed.

3.1. Datasets

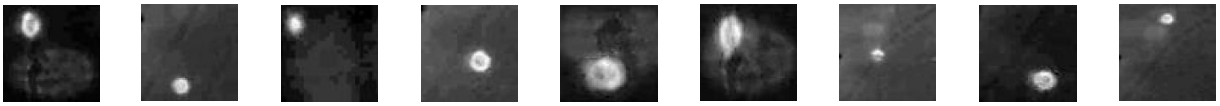
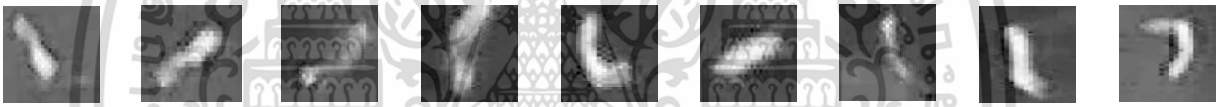
From the data of 7861 images of adulterating object particles obtained from the previous process, the image data of the impurities were divided into 10 classes: FiberT1, FiberT2, FiberT3, FiberT4, ParticleT1, ParticleT2, ParticleT3, ParticleT4, Air, and Tissue, as shown in Table 1. The classes of the impurity images in each group were separated into two parts, which were used to create datasets to train the model and test its performance. Each dataset is described as follows.

The first dataset, "Particles in Coconut Oil_V1 (PiCO_V1)", was used for model performance testing. The total of 1000 particle images was determined to consist of 135 air images, 186 FiberT1 images, 72 FiberT2 images, 111 FiberT3 images, 76 FiberT4 images, 63 ParticleT1 images, 91 ParticleT2 images, 64 ParticleT3 images, 86 ParticleT4 images, and 116 Tissue images, as shown in Table 1.

The second dataset, "Particles in Coconut Oil_V2 (PiCO_V2)", was used for the training model. The total number of particle images was 6861, consisting of 1529 air images, 1107 FiberT1 images, 311 FiberT2 images, 398 FiberT3 images, 391 FiberT4 images, 516 ParticleT1 images, 661 ParticleT2 images, 483 ParticleT3 images, 898 ParticleT4 images, and 567 tissue images. Table 1 shows sample images of some particle objects.

The two datasets, PiCO_V1 and PiCO_V2, contain grayscale images that are scaled to meet the training model requirements. In the first step, we set the image resolution to 224 × 224 pixels for use with the standard CNN, MobileNetV2, ResNet50, ResNet101, DenseNet121, VGG16, and VGG19 architectures. In the second step, we set the image resolution to 299 × 299 pixels for use with the Xception, InceptionV3, and InceptionResNetV2 architectures, which were later used in the experimental section.

Table 1. 10 classes of particles in coconut oil.

Class	Example of Particle Image for Model Training
Air	
FiberT1	
FiberT2	
FiberT3	
FiberT4	
ParticleT1	
ParticleT2	
ParticleT3	
ParticleT4	
Tissue	

3.2. Experimental Settings and Results

The CNN modeling experiments were conducted on a laptop with a 2.60 GHz Intel i7-9750H processor and 16 GB of main memory with a Jupyter Notebook and Python 3.7 kernel with TensorFlow, Keras, and OpenCV libraries. After the proposed model was developed, it was tested with the PiCO_V1 and PiCO_V2 datasets. The results were evaluated using a confusion matrix for the correctness measurement. The values in the confusion matrix were used to determine the efficiency of the classification model, which can be calculated from Equations (1)–(4).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1}_{\text{Score}} = 2 \times \left[\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \right] \tag{4}$$

where: *TP* is true positive, which means that the predicted data match the actual data in the class under consideration; *FP* is false positive, which means that the predicted data are do not match the actual data in the class under consideration; *TN* is true negative, which means that the predicted data match the actual data in a class that is not considered; and *FN* is false negative, which means mispredicted data in an unconsidered class.

Equation (1) is used to determine accuracy, meaning the number of times the prediction data match the actual data and, combined with the prediction data, match the actual data in a class that is not considered. When that number is divided by the total number of data, the efficiency of accuracy prediction is obtained.

Precision value can be calculated from Equation (2), which means the number of times the prediction data match the actual data. Then, it is divided by the number of times the prediction data match the actual data plus the incorrect predictive data in the class under consideration. The precision value can also be called a positive predictive value (PPV).

The recall value, which means the number of times the prediction data match the actual data, can be calculated from Equation (3). Then, the number obtained is divided by the number of times that the predicted data match the actual data plus the wrong predicted data in the class that is not considered. Finally, the F1_{Score} value, meaning the value obtained by combining precision and recall values, can be calculated from Equation (4). It is the harmonic mean between the precision and recall values. These four values were used to determine the performance of all models of the CNN architectures. The test results were based on the PiCO_V1 dataset, which will be discussed in the next section.

For the training model, 10 standard CNNs were used. The preliminary testing of the model's accuracy began with the input of images from the PiCO_V2 dataset that had a resolution size that met the requirements of each standard CNN architecture. The process of training the model started by importing one image at a time until every image of the PiCO_V2 dataset had been imported. The experiment started with the MobileNetV2 architecture, and the results were then compared with those of other CNNs.

3.2.1. Experiments with MobileNetV2

In the first part of our experiment, MobileNetV2 was performed to test the differences between two hyperparameters: width multipliers and fixed resolution. The MobileNetV2 performance obtained from the PiCO_V2 dataset showed that the width multiplier values affected the accuracy and size of the resulting models, as is clearly illustrated in Table 2. There is a relation between the width and the size of the model. When the width multipliers are reduced in value, the size of the model is also reduced. Because the size of the model is smaller, the accuracy performance continuously decreases from 94.05% to only 71.54%.

Table 2. Performance of MobileNets on different width multipliers (fixed resolution = 224).

Width Multipliers	Accuracy (%)	Size Model (Mb)
1.00	94.05	12.52
0.75	88.72	8.38
0.50	71.54	5.60
0.25	74.25	3.88

A test was then conducted to differentiate one hyperparameter from another, in terms of resolution, to see how the findings differed from those of the prior test. In this experiment, the best results were obtained from the previous step with a resolution of 244 × 244 pixels,

which was set to a width multiplier value of 1.0 as a constant throughout the testing process. Moreover, different image resolution values were used. Initially, the test was performed with the same image resolution size of 224 × 224 pixels. The image resolution was scaled down three times in the PiCO_V2 dataset, to 192 × 192, 160 × 160, and 128 × 128, and retrained to a new training model. The results of the image resolution scaling test are shown in Table 3. Scaling the resolution had little effect on the accuracy performance. The accuracy ranged from 94.05% to 91.22%, and changes in the resolution scaling had no effect on the model size.

Table 3. Performance of MobileNets on different resolution multipliers (fixed width = 1.0).

Resolution	Accuracy (%)	Size Model (Mb)
224 × 224	94.05	12.52
192 × 192	93.40	12.52
160 × 160	92.56	12.52
128 × 128	91.22	12.52

The hyperparameters affecting the experiments on MobilenetV2 from Tables 2 and 3 were combined, and the results are presented in Table 4. The resolution of the image at 224 × 224 pixels and a width multiplier value of 1.0 were optimized for the MobileNetV2 network layer for the PiCO_V2 dataset.

Table 4. Accuracy of MobileNets on every combination of resolution and width multipliers.

Resolution	Width Multipliers			
	1.00	**0.75**	**0.50**	**0.25**
224 × 224	94.05	88.72	71.54	74.25
192 × 192	93.40	88.77	72.35	74.44
160 × 160	92.56	88.21	71.23	72.87
128 × 128	91.22	87.83	70.96	73.23

The performance of the MobileNetV2 model was then investigated using the 1000 images PiCO_V1 dataset, the results of which were already known. First, the images were exported into MobileNetV2 for testing. The confusion matrix values were used to describe the model's accuracy performance. The test results are shown in Table 5.

Table 5. Confusion matrix of the MobileNetV2 architecture.

		Predicted Class (Accuracy)									
	Class	**Air**	**FiberT1**	**FiberT2**	**FiberT3**	**FiberT4**	**ParticleT1**	**ParticleT2**	**ParticleT3**	**ParticleT4**	**Tissue**
Actual Class	Air	**0.81**	0.02	0.02	0.00	0.03	0.00	0.07	0.01	0.04	0.00
	FiberT1	0.02	**0.83**	0.05	0.01	0.01	0.03	0.03	0.00	0.02	0.00
	FiberT2	0.00	0.01	**0.92**	0.00	0.00	0.01	0.04	0.00	0.01	0.00
	FiberT3	0.04	0.09	0.03	**0.72**	0.04	0.01	0.05	0.00	0.02	0.01
	FiberT4	0.01	0.12	0.05	0.04	**0.62**	0.04	0.08	0.00	0.04	0.00
	ParticleT1	0.05	0.03	0.00	0.00	0.00	**0.73**	0.08	0.03	0.03	0.05
	ParticleT2	0.03	0.03	0.02	0.00	0.00	0.00	**0.87**	0.04	0.00	0.00
	ParticleT3	0.03	0.02	0.02	0.00	0.00	0.00	0.09	**0.78**	0.00	0.06
	ParticleT4	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.98**	0.00
	Tissue	0.05	0.04	0.02	0.01	0.00	0.01	0.11	0.02	0.00	**0.74**

According to the prediction results, MobileNetV2 had the highest number of errors in the FiberT1 class, with 32 false predictions out of 186 images, followed by the tissue class, with 30 prediction errors out of 116 images. The errors occur because the shape of the object is similar to the image. The results in the confusion matrix were used to calculate the efficiency of the accuracy of the image classification of the PiCO_V1 dataset using Equations (1)–(4). The results indicated that the accuracy, precision, recall, and $F1_{Score}$ were 80.20%, 71.61%, 81.15%, and 76.08%, respectively.

3.2.2. Comparison of the MobileNet and Other CNN Architectures

In this section of the experiments, we compared the results of MobileNet with those of other CNN architectures based on the best width multiplier and resolution parameters obtained from the previous experiment with the MobileNetV2 architecture. The resolution was set to 224 × 224 pixels, and the width multiplier was 1.0. We tested two types of imported data: (1) data without resizing the resolution, which was 224 × 224; the standard CNN, VGG16, VGG19, ResNet50, ResNet101, and DenseNet121 were tested; and (2) the data with a scalable resolution of 299 × 299 based on the input model specification; Xception, InceptionV3, and InceptionResNetV2 were tested. We trained the entire model on the PiCO_V2 dataset with 15 epochs. The results are shown in Table 6.

Table 6. Evaluation of the runtime performances of CNN networks on our dataset at epochs = 15.

No	Model	Input Resolution	Accuracy (%) Model	Training Time (Hour)	Test Time (Minute)	Size Model (Mb)
1	MobileNetV2	224 × 224 × 3	94.05	5.13	18.20	12.52
2	Standard CNN	224 × 224 × 3	89.70	1.17	18.35	8.27
3	Xception	299 × 299 × 3	92.82	30.08	22.44	842.33
4	InceptionV3	299 × 299 × 3	93.23	15.38	20.22	96.88
5	ResNet50	224 × 224 × 3	91.65	16.04	19.27	689.54
6	ResNet101	224 × 224 × 3	90.21	25.42	19.37	795.23
7	DenseNet121	224 × 224 × 3	93.53	21.47	20.10	827.27
8	VGG16	224 × 224 × 3	91.95	8.39	18.56	689.41
9	VGG19	224 × 224 × 3	91.12	12.19	19.55	722.58
10	InceptionResNetV2	299 × 299 × 3	90.93	30.25	20.37	956.36

The results shown in Table 6 indicate that MobileNetV2 significantly outperformed all CNN models. The accuracy of MobileNetV2 was 94.05%, followed by DenseNet121, InceptionV3, Xception, VGG16, ResNet50, VGG19, InceptionResNetV2, ResNet101, and Standard CNN with accuracies of 93.53%, 93.23%, 92.82%, 91.95%, 91.65%, 91.12%, 90.93%, 90.21%, and 89.70%, respectively. The results also show that MobileNetV2 outperformed the standard CNN by approximately 4.35%.

In terms of the model size, MobileNetV2 was still the smallest model architecture. The total size of the MobileNetV2 network was only 12.52 Mb, making it approximately 6.6 and 7.7times smaller than DenseNet121 and InceptionV3, respectively. In terms of network size, VGG16 outperformed VGG19. VGG16 and VGG19, on the other hand, required enormous network sizes. The same problem occurred for the DenseNet, Inception, Xception, and ResNet networks with larger model sizes, and the accuracy was less than that of MobileNetV2. As a result, those models were not used in our research to solve the problem.

Table 6 illustrates the time it took to train all models at 15 epochs. The model that took the least amount of time in training was the standard CNN approach, which took 1.17 h. MobileNetV2, VGG16, VGG19, InceptionV3, ResNet50, DenseNet121, ResNet101, Xception, and InceptionResNetV2 took approximately 5.13, 8.39, 12.19, 15.38, 16.04, 21.47, 25.42, 30.08, and 30.25 h, respectively. The results are shown in Table 6. MobileNetV2 outperformed the other models and performed well in recognizing and classifying impurities in coconut oil.

3.2.3. Experiments with Data Augmentation

In this experiment, we compared the MobileNetV2 model's performance to the performance of each of the other models using the confusion matrix. The accuracy, precision, recall, and $F1_{Score}$ can be determined using Equations (1)–(4). The models used the PiCO_V1 dataset with 1000 images, and the test results are shown in Table 7.

Table 7. Results of model accuracy from the confusion matrix of the PiCO_V1 dataset.

No	Model	Image Input	Test Model with Confusion Matrix Form PiCO_V1 Dataset (%)			
			Accuracy	Precision	Recall	$F1_{Score}$
1	MobileNetV2	224 × 224 × 3	80.20	71.61	81.15	76.08
2	Standard CNN	224 × 224 × 3	54.82	45.09	61.01	51.86
3	Xception	299 × 299 × 3	72.95	64.62	76.24	69.95
4	InceptionV3	299 × 299 × 3	75.13	68.19	77.14	72.39
5	ResNet50	224 × 224 × 3	68.88	62.12	72.61	66.95
6	ResNet101	224 × 224 × 3	66.44	66.56	67.29	66.92
7	DenseNet121	224 × 224 × 3	68.55	62.75	69.38	65.90
8	VGG16	224 × 224 × 3	68.52	60.55	73.02	66.20
9	VGG19	224 × 224 × 3	70.02	69.38	71.89	68.58
10	InceptionResNetV2	299 × 299 × 3	55.55	55.25	58.79	55.44

In the first experiment, we measured the standard CNN, and the accuracy, precision, recall, and $F1_{Score}$ were 54.82%, 45.09%, 61.01%, and 51.86%, respectively. The results of the accuracy testing showed that the errors in this model were due largely to incorrect selections. The model selected ParticleT4 instead of FiberT3, and ParticleT4 was selected instead of FiberT4, which represented a large number of incorrect selections. There were 452 misclassified images, or a 45.18% error rate.

In the second experiment, we measured the Xception model, and the accuracy, precision, recall, and $F1_{Score}$ were 72.95%, 64.62%, 76.24%, and 69.95%, respectively. Errors in the model occurred when images were classified as FiberT1 instead of FiberT3. In this case, 52 images were mistakenly selected. Another mistake occurred when images were classified as FiberT2 and ParticleT1 instead of FiberT1. There were 12 incorrect images from FiberT2 and 19 incorrect images from ParticleT1, resulting in 271 misclassified images, or a 27.05% error rate.

In the third experiment, we measured the InceptionV3 model, and the accuracy, precision, recall, and $F1_{Score}$ were 75.13%, 68.19%, 77.14%, and 72.39%, respectively. Errors in the model occurred when images were classified in the incorrect fiber class. The InceptionV3 model incorrectly predicted 53 images in the FiberT1 class that should actually have been in the FiberT3 class. Furthermore, the model predicted that 59 images belonged to FiberT2; however, the validation revealed that FiberT1 was the proper class. In the air, particle, and tissue classes, a total of 249 images were misclassified for a 24.87% error rate.

In the fourth experiment, we measured the ResNet50 model, and the accuracy, precision, recall, and $F1_{Score}$ were 68.88%, 62.12%, 72.61%, and 66.95%, respectively. The most misclassified images were those from FiberT1. The model determined that 31 and 18 images should be classified as FiberT2 and ParticleT2, respectively. A total of 312 images were misclassified for a 31.12% error rate.

In the fifth experiment, we measured the ResNet101 model, and the accuracy, precision, recall, and $F1_{Score}$ were 66.44%, 66.56%, 67.29%, and 66.92%, respectively. The most misclassified class was FiberT3. The model determined that 20 and 16 images should be FiberT1 and ParticleT4, respectively. A total of 336 images were misclassified for a 33.44% error rate.

In the sixth experiment, we measured the DenseNet121 model, and the accuracy, precision, recall, and $F1_{Score}$ were 68.55%, 62.75%, 69.38%, and 65.90%, respectively. The

most misclassified class was FiberT2. The model determined that 26 images should be FiberT3. A total of 315 images were misclassified for a 31.45% error rate.

In the seventh experiment, we measured the VGG16 model, and the accuracy, precision, recall, and $F1_{Score}$ were 68.52%, 60.54%, 73.02%, and 66.20%, respectively. The most misclassified class was FiberT3. The model determined that 15 images should be FiberT1. A total of 315 images were misclassified for a 31.48% error rate.

In the eighth experiment, we measured the VGG19 model, and the accuracy, precision, recall, and $F1_{Score}$ were 70.02%, 69.38%, 71.89%, and 68.58%, respectively. The most misclassified class was FiberT3. The model determined that 20 images should be FiberT1. A total of 299 images were misclassified, representing an error rate of 29.98%.

Finally, we examined the InceptionResNetV2 model. The accuracy, precision, recall, and $F1_{Score}$ were 55.55%, 55.25%, 58.79%, and 55.44%, respectively. The most misclassified class was FiberT3. The model determined that 16 and 110 images should be FiberT2 and ParticleT4, respectively. A total of 445 images were misclassified, representing an error rate of 44.5%. The results of the confusion matrix for each model are shown in Table 7.

3.3. Discussion

The overall results of this research demonstrate the high efficiency of the MobileNetV2 architecture in recognizing classes of impurities contained in coconut oil. MobileNetV2 spent the second-lowest amount of time training the model on both datasets, PiCO_V1 and PiCO_V2, while the standard CNN consumes the shortest time. By comparing the accuracy with that of each CNN model, the models were trained from the beginning without any preconfigured parameters, and only 15 training epochs were used for speed comparison. This problem may affect accuracy values in the future because the number of epochs affects the model performance. If the number of epochs is too small, it can lead to incorrect grouping errors and a lack of resemblance to the actual answer, a condition known as underfitting. In contrast, if the number of epochs is too high, it can cause problems in training the model. Then, even though the model can precisely classify objects, it is impractical to use, a problem that is called overfitting. This problem can be solved by adjusting the parameter dropout, so that the data is distributed among each class. To obtain a good compromise, the parameter adjustments should be made accordingly.

4. Conclusions

In this paper, experiments were conducted using a deep learning model to solve the problem of classifying adulterating objects in coconut oil. We selected ten CNN models, MobileNetV2, Standard CNN, Xception, InceptionV3, ResNet50, ResNet101, DenseNet121, VGG16, VGG19, and InceptionResNetV2, to benchmark the performance of their architectures. The experimental results indicated that MobileNetV2 was the best method for recognizing adulterating objects in coconut oil. It had the best accuracy compared to other architectures. The FiberT3 test results were the worst in the classification due to the similarity in the shape of FiberT3 objects to those of FiberT4 and FiberT2 objects. As a result of this classification, the performance values were reduced. Moreover, the MobileNetV2 architecture is suitable for all platforms, including mobile devices with low processing speed and low memory.

When MobileNetV2 was tested against the PiCO_V1 dataset, which is a previously existing dataset, changing the width and resolution parameters had no effect on the model accuracy. However, such changes can greatly reduce the size of the model. The MobileNetV2 architecture has a model size of 12.54 Mb., which is smaller than that of other models. Additionally, MobileNetV2 achieved an accuracy of 80.20% in testing on the PiCO_V1 dataset, which was better than the results of the other CNN models. Finally, we utilized MobileNetV2 with the coconut oil impurity detection program for further integration with the coconut oil production line.

Precaution for working in coconut oil images is the heat of the lamp used to shine through the coconut oil. If the lamp is overheating, this causes the light from the lamp

to flicker. As a result, the coconut oil images are unevenly bright and unable to show the objects contained in the coconut oil. Therefore, cooling of the lamp is a necessary factor to detect impurities in the coconut oil.

Author Contributions: Conceptualization, A.P. and W.K.; Methodology, A.P. and W.K.; Software, A.P.; Validation, A.P. and W.K.; Formal analysis, A.P. and W.K.; Investigation, A.P.; Resources, A.P.; Data curation, A.P.; Writing—original draft preparation, A.P.; Writing—review and editing, A.P. and W.K.; Visualization, A.P.; Supervision, W.K.; Project administration, W.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by School of Science, King Mongkut's Institute of Technology Ladkrabang, Thailand.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The authors would like to thank Tropicana Oil Co., Ltd. for providing resources and supported in testing part.

Conflicts of Interest: The authors declare no potential conflict of interest.

References

1. Karnawat, V.; Patil, S. Turbidity detection using image processing. In Proceedings of the 2016 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 29–30 April 2016.
2. Berg, M.; Videen, G. Digital holographic imaging of aerosol particles in flight. *J. Quant. Spectrosc. Radiat. Transf.* **2011**, *112*, 1776–1783. [CrossRef]
3. Pastore, V.P.; Zimmerman, T.; Biswas, S.; Bianco, S. Annotation-free learning of plankton for classification and anomaly detection. *Sci. Rep.* **2020**, *10*, 12142. [CrossRef] [PubMed]
4. Versaci, M.; Calcagno, S.; Jia, Y.; Morabito, F.C. Fuzzy Geometrical Approach Based on Unit Hyper-Cubes for Image Contrast Enhancement. In Proceedings of the 2015 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), Kuala Lumpur, Malaysia, 19–21 October 2015.
5. amirberAgain, Python and openCV to Analyze Microscope Slide Images of Airborne Particles. Available online: https://publiclab.org/notes/amirberAgain/01-12-2018/python-and-opencv-to-analyze-microscope-slide-images-of-airborne-particles (accessed on 25 May 2021).
6. Oheka, O.; Chunling, T. Fast and Improved Real-Time Vehicle Anti-Tracking System. *Appl. Sci.* **2020**, *10*, 5928. [CrossRef]
7. Dechter, R. Learning while searching in constraint-satisfaction-problems. In Proceedings of the Fifth AAAI National Conference on Artificial Intelligence (AAI'86), Philadelphia, PA, USA, 11–15 August 1986.
8. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
9. Fukushima, K. Recent advances in the deep CNN neocognitron. *Nonlinear Theory Appl. IEICE* **2019**, *10*, 304–321. [CrossRef]
10. LeCun, Y. 1.1 Deep Learning Hardware: Past, Present, and Future. In Proceedings of the 2019 IEEE International Solid-State Circuits Conference—(ISSCC), San Francisco, CA, USA, 17–21 February 2019.
11. Li, X.; Xiong, H.; An, H.; Xu, C.; Dou, D. RIFLE: Backpropagation in Depth for Deep Transfer Learning through Re-Initializing the Fully-connected LayEr. In Proceedings of the 37th International Conference on Machine Learning (ICML2020), Vienna, Austria, 12–18 July 2020.
12. Ranzato, M.A.; Huang, F.; Boureau, Y.; LeCun, Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Minneapolis, MN, USA, 17–22 June 2007.
13. Brinker, T.J.; Hekler, A.; Utikal, J.; Grabe, N.; Schadendorf, D.; Klode, J.; Berking, C.; Steeb, T.; Enk, A.; Kalle, C.V. Skin Cancer Classification Using Convolutional Neural Networks: Systematic Review. *J. Med. Internet Res.* **2018**, *20*, e11936. [CrossRef]
14. Patil, R.; Bellary, S. Machine learning approach in melanoma cancer stage detection. *J. King Saud Univ. Comput. Inf. Sci.* **2020**, *in press*. [CrossRef]
15. Khairi, M.T.M.; Ibrahim, S.; Yunus, M.A.M.; Faramarzi, M.; Yusuf, Z. Artificial Neural Network Approach for Predicting the Water Turbidity Level Using Optical Tomography. *Arab. J. Sci. Eng.* **2016**, *41*, 3369–3379. [CrossRef]
16. Newby, J.M.; Schaefer, A.M.; Lee, P.T.; Forest, M.G.; Lai, S.K. Convolutional neural networks automate detection for tracking of submicron-scale particles in 2D and 3D. *Proc. Natl. Acad. Sci. USA* **2018**, *15*, 9026–9031. [CrossRef] [PubMed]
17. Rong, D.; Xie, L.; Ying, Y. Computer vision detection of foreign objects in walnuts using deep learning. *Comput. Electron. Agric.* **2019**, *162*, 1001–1010. [CrossRef]
18. Ferreira, A.D.S.; Freitas, D.M.; Silva, G.G.D.; Pistori, H.; Folhes, M.T. Weed detection in soybean crops using ConvNets. *Comput. Electron. Agric.* **2017**, *143*, 314–324. [CrossRef]

19. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
20. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv* **2014**, arXiv:1312.6034.
21. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 2015 International Conference on Learning Representations (ICLR 2015). *arXiv* **2014**, arXiv:1409.1556.
22. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
23. Pan, H.; Pang, Z.; Wang, Y.; Wang, Y.; Chen, L. A New Image Recognition and Classification Method Combining Transfer Learning Algorithm and MobileNet Model for Welding Defects. *IEEE Access.* **2020**, *8*, 119951–119960. [CrossRef]
24. Kerf, T.D.; Gladines, J.; Sels, S.; Vanlanduit, S. Oil Spill Detection Using Machine Learning and Infrared Images. *Remote Sens.* **2020**, *12*, 4090. [CrossRef]
25. Iqbal, U.; Barthelemy, J.; Li, W.; Perez, P. Automating Visual Blockage Classification of Culverts with Deep Learning. *Appl. Sci.* **2021**, *11*, 7561. [CrossRef]
26. Tammina, S. Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. *Int. J. Sci. Res. Publ.* **2019**, *9*, 143–150. [CrossRef]
27. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
29. Aparna; Bhatia, Y.; Rai, R.; Gupta, V.; Aggarwal, N.; Akula, A. Convolutional neural networks based potholes detection using thermal imaging. *J. King Saud Univ. Comput. Inf. Sci.* **2019**, *in press*. [CrossRef]
30. Huang, G.; Liu, Z.; Pleiss, G.; Maaten, L.V.; Weinberger, K.Q. Convolutional Networks with Dense Connectivity. *IEEE Trans. Pattern. Anal. Mach. Intell.* **2019**, 1–12. [CrossRef] [PubMed]
31. Tao, Y.; Xu, M.; Lu, Z.; Zhong, Y. DenseNet-Based Depth-Width Double Reinforced Deep Learning Neural Network for High-Resolution Remote Sensing Image Per-Pixel Classification. *Remote. Sens.* **2018**, *10*, 779. [CrossRef]
32. Too, E.C.; Yujian, L.; Njuki, S.; Yingchun, L. A comparative study of fine-tuning deep learning models for plant disease identification. *Comput. Electron. Agric.* **2019**, *161*, 272–279. [CrossRef]
33. Palananda, A.; Kimpan, W. Turbidity of Coconut Oil Determination Using the MAMoH Method in Image Processing. *IEEE Access* **2021**, *9*, 41494–41505. [CrossRef]

ประวัติผู้เขียน

ชื่อ	นาย อรรถพล พลานนท์
วัน เดือน ปีเกิด	1 มิถุนายน พ.ศ. 2524
ที่อยู่ปัจจุบัน	4/1 หมู่ 8 ตำบล วังศาลา อำเภอ ท่าม่วง จังหวัด กาญจนบุรี 71110
ประวัติการศึกษา	(2548) ครุศาสตร์อุตสาหกรรมบัณฑิต สาขา วิศวกรรมคอมพิวเตอร์ เกรดเฉลี่ย 2.66 (มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี)
	(2554) วิทยาศาสตรมหาบัณฑิต สาขา วิทยาการคอมพิวเตอร์ เกรดเฉลี่ย 3.22 (มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ)
ทุนการศึกษาที่ได้รับ	ทุนมหาวิทยาลัยราชภัฏนครปฐม
ผลงานทางวิชาการ	1. Palananda, A. and Kimpan, W. 2021. “Turbidity of Coconut Oil Determination Using the MAMoH Method in Image Processing.” *IEEE Access.* Vol 9 : 41494-41505. 2021, Doi: 10.1109/ACCESS.2021.3065004 (Journal Publication)
	2. Palananda, A. and Kimpan, W. 2022. “Classification of Adulterated Particle Images in Coconut Oil Using Deep Learning Approaches” *Appl. Sci.* Vol 12 656 : 1-16. 2022, Doi: https://doi.org/10.3390/app12020656 (Journal Publication)