การตรวจหาลายเส้นบนฝ่ามือและเส้นขอบนิ้วมือ เพื่อการระบุตัวบุคคลโดยใช้ FIR System

PALM LINES AND FINGER CONTOURS EXTRACTION FOR PERSONAL IDENTIFICATION USING FIR SYSTEM

สวการย์ หิริวัฒนวงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

การตรวจหาลายเส้นบนฝ่ามือและเส้นขอบนิ้วมือ เพื่อการระบุตัวบุคคลโดยใช้ FIR System

PALM LINES AND FINGER CONTOURS EXTRACTION FOR PERSONAL IDENTIFICATION USING FIR SYSTEM

สวการย์ หิริวัฒนวงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

PALM LINES AND FINGER CONTOURS EXTRACTION FOR PERSONAL IDENTIFICATION USING FIR SYSTEM

SAWAKARN HIRIWATANAWONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERRING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUE OF TECHNOLOGY LARDKRABANG
ACADEMIC YEAR 2018

ปริญญานิพนธ์ปีการศึกษา 2561
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์ การตรวจหาลายเส้นบนฝ่ามือและเส้นขอบนิ้วมือเพื่อการระบุตัวบุคคลโดยใช้ FIR System

PALM LINES AND FINGER CONTOURS EXTRACTION FOR PERSONAL IDENTIFICATION USING FIR SYSTEM

นักศึกษาผู้จัดทำ นายสวการย์ หิริวัฒนวงศ์ **รหัสนักศึกษา** 58011280

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2561

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รองศาสตราจารย์.ดร.เกษตร์ ศิริสันติสัมฤทธิ์	

หัวข้อปริญญานิพนธ์	การตรวจหาลายเส้นบนฝ่ามือและเส้นขอบนิ้วมือเพื่อการระบุตัวบุคคลโดยใช้ FIR System PALM LINES AND FINGER CONTOURS EXTRACTION FOR PERSONAL IDENTIFICATION USING FIR SYSTEM
นักศึกษาผู้จัดทำ	นายสวการย์ หิริวัฒนวงศ์ รหัสนักศึกษา 58011280
อาจารย์ที่ปรึกษา	รองศาสตราจารย์.ดร.เกษตร์ ศิริสันติสัมฤทธิ์
ปีการศึกษา	2561

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอการระบุตัวบุคคลโดยใช้ FIR (Finite Impulse Response) System ภาพมือถูกถ่ายด้วยกล้องดิจิตอลหรือสแกนเนอร์ซึ่งเป็นภาพสี RGB ถูกแปลงเป็นภาพระดับสีเทา จากนั้นผ่านกระบวนการแยกภาพมือออกเป็นสองส่วนคือ 1) นิ้วมือ และ 2) ฝ่ามือ ในส่วนของนิ้วมือจะทำการหาเส้นขอบนิ้วมือและทำให้เส้นขอบบางลงเหลือเพียงหนึ่งจุดภาพ จากนั้นหาจุดบนสุดและล่างสุดของแต่ละนิ้วมือ เพื่อแยกนิ้วมือแต่ละนิ้วและหาพื้นที่ของ Palm Print หาเส้นขอบของ Palm Lines และทำให้บางเหลือหนึ่งจุดภาพแล้วหา Coordinates ในแนวแกน x และ y ของนิ้วมือทั้งสี่นิ้วและ Palm Lines ทั้งสามเส้น จากนั้นทำการ Normalize คำนวณค่า DCT Coefficients และ Impulse Response ของ FIR System ของนิ้วมือทั้งสี่นิ้วและ Palm Lines ทั้งสามเส้น ซึ่ง Impulse Response ของ FIR System จะถูกใช้เพื่อบ่งชี้ตัวบุคคล สำหรับการบ่งชี้ตัวบุคคลจะใช้วิธีการตรวจสอบความผิดพลาดน้อยที่สุดของ Impulse Response ของ FIR System ของบุคคลที่ไม่ทราบกับ Impulse Response ในฐานข้อมูล โดยจะใช้ฐานข้อมูลของ KMITL ใช้บุคคลจำนวน 7 บุคคล บุคคลละ 4 ภาพ รวมทั้งหมด 28 ภาพในการทดลอง ผลการทดลองการบ่งชี้ตัวบุคคลโดยใช้เส้นขอบนิ้วมือให้ความถูกต้องร้อยละ 100 ผลการทดลองการบ่งชี้ตัวบุคคลโดยใช้ลายเส้นบนฝ่ามือให้ความถูกต้องร้อยละ 78.57 และผลการทดลองการบ่งชี้ตัวบุคคลโดยใช้ลายเส้นบนฝ่ามือและเส้นขอบนิ้วมือร่วมกันให้ความถูกต้องร้อยละ 92.86

Thesis Title	PALM LINES AND FINGER CONTOURS EXTRACTION FOR PERSONAL IDENTIFICATION USING FIR SYSTEM
Authors	Mr. Sawakarn Hiriwatanawong
Thesis Advisor	Assoc. Prof. Dr. Kaset Sirisantisamrid
Year	2018

ABSTRACT

This thesis proposes a personal identification using FIR (Finite Impulse Response) system. The hand image is captured by a digital camera or scanner which is RGB color image and transformed to gray scale image. Then, it passes to a process to separate the hand image into two parts: 1) fingers and 2) palm print. In the part of fingers, the edge of each finger is extracted, and its thickness is reduced to one pixel by thinning algorithm. Then, the tip and valley points of each finger is extracted to separate each finger and palm print from the hand image. The palm print image is processed to find the edge of palm lines and thinning them to one pixel same as each finger. Then, find the coordinates in x and y axes, normalized these coordinates, compute the DCT coefficients and impulse response of FIR system of four fingers and three palm lines. These impulse response of FIR systems are used to identify unknown person. If an error between the impulse response of FIR systems of unknown person and their database is minimum, that person can identified. On experiments, the proposed method tested on KMITL database, which use 7 person and each person have 4 images, total image are 28 images. The identification using hand contours correction is 100%, identification using palm lines correction is 78.57% and identification using both palm lines and finger contours correction is 92.86%.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีด้วยความกรุณา รศ.ดร.เกษตร์ ศิริสันติสัมฤทธิ์ อาจารย์ที่ปรึกษา ที่ได้ให้คำแนะนำ ให้คำปรึกษาในการแก้ไขปัญหาที่พบระหว่างการทำปริญญานิพนธ์ จนสามารถผ่านอุปสรรคต่างๆมาได้ ตลอดจนให้ความรู้ ข้อคิดเห็น และประสบการณ์ต่างๆ อันเป็นประโยชน์แก่ข้าพเจ้าเป็นอย่างมาก อีกทั้งยังทำให้ข้าพเจ้าได้มีความรู้ในด้านอื่นๆเพิ่มมากขึ้นนอกเหนือจากเนื้อหาวิชาการที่ได้เรียนในห้องเรียน รวมถึงขอขอบคุณอาจารย์ที่เอื้อเฟื้อห้องที่ใช้ในการทำงานวิจัยจนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จด้วยดี

ขอขอบพระคุณคณาจารย์หลักสูตรวิศวกรรมการวัดคุมทุกท่านที่ช่วยให้การสั่งสอน ให้ความรู้ในเนื้อหาวิชาการต่างๆ อันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

สุดท้ายนี้ข้าพเจ้าขอขอบพระคุณบิดา มารดา และครอบครัวที่ช่วยสนับสนุน ตลอดจนการช่วยเหลือและให้กำลังใจมาโดยตลอดจนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยดี

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

ผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

ในปัจจุบันได้มีการนำการประมวลผลภาพดิจิตอล (Digital Image Processing) มาใช้กับงานในด้านต่างๆหลายด้าน อาทิเช่น ทางด้านสถิติ ในเรื่องของการวิเคราะห์กลุ่มเป้าหมายในการเลือกซื้อสินค้าและบริการจากเพศและช่วงอายุ การใช้งานร่วมกับระบบอัตโนมัติ เช่น การอ่านเลขโค้ดไปรษณีย์เพื่อคัดแยกจดหมาย การตรวจสอบเลขทะเบียนรถยนต์ หรือการทำงานของหุ่นยนต์ในโรงงานอุตสาหกรรมหรือแม้แต่ในสำนักงานต่างๆ เช่น หุ่นยนต์ที่ใช้รับส่งของ เป็นต้น การใช้งานทางด้านการรักษาความปลอดภัยและป้องกันอาชญากรรม การระบุตัวบุคคลจากใบหน้า การระบุสิ่งของต่างๆ รวมถึงการประยุกต์ใช้งานรวมกับระบบปัญญาประดิษฐ์ (Artificial Intelligence) เพื่อพัฒนาประสิทธิภาพให้ดีขึ้นและใช้งานได้หลายด้านมากขึ้น เช่น ใช้การประมวลภาพในรถยนต์ไร้คนขับ เป็นต้น ด้วยเหตุผลดังกล่าวจึงทำให้มีความสนใจในเรื่องของการศึกษาการประมวลผลภาพดิจิตอลเพื่อให้มีความรู้พื้นฐานทั้งในด้านทฤษฎีและการใช้งานโปรแกรมที่เกี่ยวข้อง

ในปริญญานิพนธ์ฉบับนี้จะเป็นการศึกษาการระบุตัวบุคคลที่เกี่ยวข้องกับการรักษาความปลอดภัย โดยการรักษาความปลอดภัยโดยการระบุตัวบุคคลมีหลากหลายวิธี เช่น การใช้รหัสผ่าน การใช้บัตรผ่าน แต่อาจยังไม่มีความปลอดภัยเพียงพอจึงมีการนำลักษณะทางกายภาพของแต่ละบุคคลซึ่งมีความแตกต่างกันมาช่วยในการระบุตัวบุคคลให้มีความปลอดภัยมากขึ้น ได้แก่ ลายนิ้วมือ เส้นลายมือ ฝ่ามือ นิ้วมือ เล็บมือ ม่านตา ใบหู และใบหน้า [1] โดยการระบุตัวบุคคลทางกายภาพที่มีการใช้งานมากคือการใช้ลายนิ้วมือและม่านตา เนื่องจากมีความปลอดภัยสูง แต่ก็มีค่าใช้จ่ายที่สูงตามไปด้วยเช่นกัน ดังนั้นปริญญานิพนธ์ฉบับนี้จึงจะได้เสนอทางเลือกใหม่ที่ไม่ยุ่งยากซับซ้อน แต่มีความน่าเชื่อถืออยู่ในระดับที่ยอมรับได้ เพื่อเป็นทางเลือกในการใช้งานระบุตัวบุคคล เช่น การนำไปใช้ในการควบคุมพื้นที่ที่ให้เพียงบุคคลที่ได้รับอนุญาตเข้าพื้นที่ได้เท่านั้น ปริญญานิพนธ์ฉบับนี้จะเป็นการศึกษาต่อยอดจากวิทยานิพนธ์ของ ณัฐภัทร ธีรเบญจกุล “การระบุตัวบุคคลด้วย FIR SYSTEM” [1] ที่ใช้เส้นขอบนิ้วมือในการระบุตัวบุคคล โดยคาดหวังว่าจะสามารถเพิ่มประสิทธิภาพในการระบุตัวบุคคลให้มากขึ้น ด้วยการนำลายเส้นมือบนฝ่ามือมาใช้ระบุตัวบุคคลร่วมกับเส้นขอบนิ้วมือ

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ศึกษาทฤษฎีและหลักการที่เกี่ยวข้องกับการประมวลผลภาพ
2. ศึกษาสมการทางคณิตศาสตร์ที่ใช้ในการวิเคราะห์และคำนวณภาพถ่าย
3. ศึกษาการใช้โปรแกรม MATLAB
4. ศึกษาขั้นตอนและวิธีการในการแยกส่วนประกอบของมือจากภาพถ่าย
5. สามารถระบุตัวบุคคลโดยใช้เส้นขอบนิ้วมือและลายเส้นมือบนฝ่ามือ

1.3 ขอบเขตของปริญญานิพนธ์

1. สามารถแยกส่วนฝ่ามือออกจากภาพได้
2. สามารถหาเส้น 3 เส้นบนฝ่ามือ และกำหนดจุดพิกัดแนวแกน x, y
3. ระบุตัวบุคคลด้วยวิธีการ FIR ได้
4. นำไปประยุกต์ใช้งานร่วมกับการระบุตัวบุคคลด้วยเส้นขอบนิ้วมือ

1.4 ขั้นตอนการศึกษา

1. ศึกษาข้อมูลเกี่ยวกับการประมวลผลภาพความเป็นมาและความสำคัญของปัญหา ความจำเป็นในการศึกษา ทฤษฎีและเนื้อหาที่เกี่ยวข้อง
2. ศึกษาปริญญานิพนธ์ที่เกี่ยวข้อง รวมถึงงานวิจัยที่ใช้วิธีการแยกส่วนของมือออกจากภาพถ่าย แยกส่วนของนิ้วมือและส่วนของฝ่ามือออกภาพ
3. ศึกษาหลักการและทฤษฎีที่เกี่ยวข้องกับการประมวลผลภาพ การปรับปรุงภาพถ่ายให้สามารถแยกส่วนที่ต้องการได้ และเลือกวิธีการรวมถึงทฤษฎีที่เหมาะสมที่ได้จากการศึกษา
4. ศึกษาวิธีการใช้งานโปรแกรม MATLAB การเขียนโปรแกรมเพื่อประมวลผลภาพ การเขียนสมการทางคณิตศาสตร์และเงื่อนไขต่างๆ ลงในโปรแกรม MATLAB
5. ศึกษาวิธีการที่จะระบุเอกลักษณ์ของแต่ละนิ้วมือและลายเส้นมือบนฝ่ามือจากงานวิจัยที่เกี่ยวข้อง รวมถึงการเปรียบเทียบกับบุคคลในฐานข้อมูล
6. วิเคราะห์เอกลักษณ์ของเส้นขอบนิ้วมือและเส้นลายมือบนฝ่ามือของแต่ละบุคคล แล้วนำข้อมูลที่ได้จัดเก็บเป็นฐานข้อมูล
7. แสดงผลการทดลองรวมถึงความผิดพลาดที่เกิดขึ้น
8. สรุปผลการทดลอง อภิปรายผลการทดลอง ปัญหาที่พบและข้อเสนอแนะ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้เกี่ยวกับการประมวลผลภาพ
2. สามารถใช้งาน MATLAB ในการวิเคราะห์ภาพได้
3. ได้เรียนรู้เกี่ยวกับทฤษฎีและสมการทางคณิตศาสตร์ที่ใช้ในการวิเคราะห์ภาพ
4. สามารถนำไปประยุกต์ใช้ในการระบุตัวบุคคลได้
5. สามารถนำไปใช้ในการรักษาความปลอดภัยอาคารหรือบริเวณที่จำกัดเฉพาะผู้ได้รับอนุญาต
6. สามารถนำความรู้ที่ได้ไปประยุกต์ใช้ในเรื่องอื่นๆที่เกี่ยวข้องกับ *Image Processing* ไม่เพียงเฉพาะการระบุอัตลักษณ์บุคคล แต่รวมไปถึงระบบควบคุมระบบอัตโนมัติต่างๆ ตลอดจนเป็นพื้นฐานในการประยุกต์ใช้งานในอนาคต

บทที่ 2
หลักการประมวลผลภาพและ FIR System

ในบทนี้จะกล่าวถึงทฤษฎีและหลักการที่เกี่ยวข้องกับการประมวลผลภาพดิจิตอล โดยทฤษฎีและหลักการที่ถูกนำมาใช้ในปริญญานิพนธ์ฉบับนี้ได้แก่ การหาขอบภาพ การทำขอบภาพให้บาง การหาตำแหน่งจุดพิกัด การนอร์มอลไลเซชัน การอินเตอร์โพเลชัน การหาค่าสัมประสิทธิ์ดิสครีตโคไซน์ และทฤษฎีระบบผลตอบสนองอิมพัลส์ที่จำกัดจำนวน (FIR System) ซึ่งจะใช้ในการระบุเอกลักษณ์ของแต่ละบุคคล

2.1 ทฤษฎีการประมวลผลภาพดิจิตอล (Digital Image Processing)

ภาพโมโนโครม (Monochrome Image) หมายถึง ภาพที่แสดงอยู่ในลักษณะสองมิติ คือมีมิติของความกว้าง และความยาว สามารถแสดงได้ในฟังก์ชั่น f(x, y) เมื่อ x และ y คือพิกัดในแนวนอนและแนวตั้งตามลำดับ และแอมพลิจูดของ f ที่จุดพิกัด (x, y) เป็นค่าความเข้มของแสง (Light Intensity) หมายถึงค่าความสว่างของแสงหรือค่าระดับสีเทา (Gray Level) ของภาพ ณ ตำแหน่ง (x, y) หากค่าแอมพลิจูดของ f มีค่าสูง ความสว่างในจุดดังกล่าวของภาพก็จะมีค่าสูงไปด้วย โดยภาพโมโนโครมนั้นจะเป็นภาพที่มีสีเพียงสีเดียวแต่จะมีความสว่างในแต่ละจุดของภาพที่แตกต่างกันไป หรือหมายถึงการมีความเข้มของสีที่แตกต่างกัน เช่น ภาพขาวดำที่จะใช้เพียงสีดำในการแสดงผลแต่จะมีความเข้มหรือเฉดสีที่แตกต่างกัน หรือภาพที่ใช้เพียงสีน้ำเงินในการแสดงผลก็จะมีความเข้มของสีน้ำเงินที่แตกต่างกัน เป็นต้น

ภาพดิจิตอล (Digital Image) หมายถึง ภาพที่เก็บอยู่ในรูปแบบดิจิตอล จะมีลักษณะเป็นภาพสองมิติ คือมีมิติของความกว้าง และความยาว สามารถแสดงเป็นฟังก์ชันได้ คือรูปภาพ f(x, y) เมื่อ x และ y คือพิกัดในแนวนอนและแนวตั้งตามลำดับ และแอมพลิจูดของ f ที่จุดพิกัด (x, y) หมายถึงความสว่าง และเมื่อค่า x, y และ แอมพลิจูดของ f เป็นค่าที่จำกัด (Finite Value) จะเรียกภาพดังกล่าวว่าภาพดิจิตอล โดยภาพดิจิตอลสามารถที่จะแสดงเป็นภาพสี ภาพระดับสีเทา ภาพสองระดับ หรือเป็นภาพโมโนโครมก็ได้ สำหรับส่วนประกอบที่อยู่รูปภาพจะเรียกว่าพิกเซล (Pixel) เทียบได้กับจุดภาพ 1 จุดภาพ โดยภาพๆหนึ่งจะประกอบด้วยพิกเซลหลายๆพิกเซล สำหรับความแตกต่างระหว่างภาพโมโนโครมกับภาพดิจิตอลคือ ภาพโมโนโครมเป็นภาพที่ใช้สีเพียงสีเดียวในการแสดงผลแต่จะมีความเข้มของสีหรือแอมพลิจูดของ f ที่แตกต่างกัน และภาพโมโนโครมสามารถที่จะเป็นภาพดิจิตอลได้โดยค่าแอมพลิจูดของ f นั้นจะต้องเป็นค่าที่จำกัด และลักษณะของภาพดิจิตอลคือเป็นภาพที่ในแต่ละพิกเซลจะเก็บข้อมูลแอมพลิจูดของ f ในรูปแบบตัวเลขหรือบิตในระบบคอมพิวเตอร์

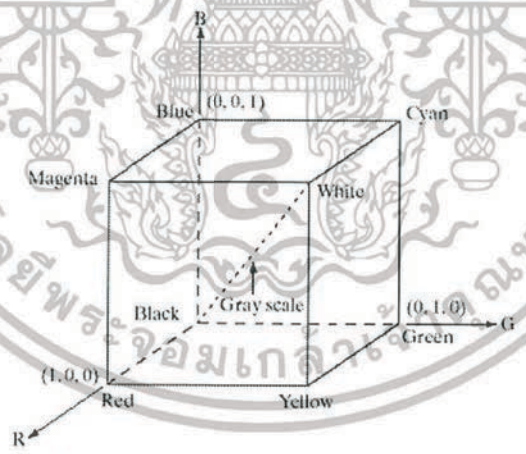
การประมวลผลภาพ (Image Processing) คือการใช้ขั้นตอนหรือกระบวนการต่างๆมากระทำกับภาพดิจิตอล เพื่อปรับปรุงคุณภาพของภาพให้มีคุณสมบัติตามที่ต้องการและมีความเหมาะสมในการประมวลผล เช่น ทำให้ภาพมีความคมชัดมากขึ้น การทำให้ภาพมีความสว่างมากขึ้นหรือลดความสว่างลง การย่อหรือขยายรูปภาพ การตัดภาพเฉพาะในส่วนที่ต้องการ การทำให้ภาพสีกลายเป็นภาพระดับสีเทาหรือการทำให้รูปภาพเป็นภาพสองระดับ เป็นต้น โดยภาพที่จะนำมาประมวลผลสามารถแยกชนิดของภาพได้ดังนี้

2.1.1 ภาพสี

รูปแบบของสีคือลักษณะเฉพาะที่แสดงได้โดยระบบพิกัด 3 มิติ และที่จุดใดจุดหนึ่งในระบบพิกัด 3 มิตินั้นจะแสดงลักษณะเฉพาะของแต่ละสี โดยรูปแบบของสีที่ใช้กันทุกวันนี้จะขึ้นอยู่กับฮาร์ดแวร์ (เช่น หน้าจอคอมพิวเตอร์หรือเครื่องพิมพ์เอกสาร) หรือซอฟต์แวร์ที่ใช้งาน (เช่น สีกราฟฟิกสำหรับแอนิเมชั่น) สำหรับรูปแบบสีที่ใช้จะมีรูปแบบสี RGB (RGB Model) ซึ่งจะประกอบไปด้วยสีแดง (Red) สีเขียว (Green) และสีน้ำเงิน (Blue) รูปแบบสี CMY (Cyan, Magenta, Yellow) รูปแบบสี YIQ (Y จะเกี่ยวข้องกับ Luminance, I คือ Inphase, Q คือ Quadrature) และรูปแบบสี HSI (Hue, Saturation, Intensity) โดยรูปแบบสีที่มักจะใช้ในการประมวลผลคือ RGB, YIQ และ HSI ซึ่งมีรายละเอียดดังนี้

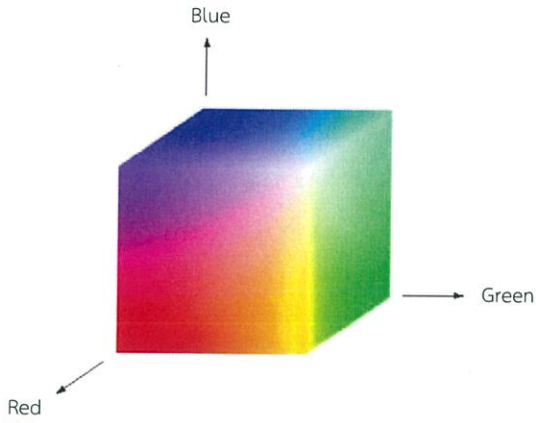
2.1.1.1 รูปแบบสี RGB (RGB Color Model)

ในรูปแบบสี RGB ซึ่งประกอบด้วยสีแดง (Red) สีเขียว (Green) และสีน้ำเงิน (Blue) จะเป็นรูปแบบสีที่ใช้ในการแสดงผลของจอคอมพิวเตอร์ เป็นรูปแบบที่อยู่บนพื้นฐานของระบบพิกัดคาร์ทีเชี้ยน (Cartesian) ดังแสดงในรูปที่ 2.1 สีแต่ละสีคือค่าที่อยู่ ณ ตำแหน่งมุมของลูกบาศก์ โดยสีแดงจะอยู่ที่ (1, 0, 0) สีเขียวจะอยู่ที่ (0, 1, 0) และสีแดงจะอยู่ที่ (0, 0, 1) ส่วนสีดำจะเป็นสีที่อยู่ที่จุดกำเนิด (0, 0, 0) และสีขาวจะอยู่ที่มุมที่ไกลที่สุดจากจุดกำเนิด (1, 1, 1) เมื่อนำสีแต่ละสีมาผสมกันก็จะได้สีอื่นๆที่แตกต่างกันไป เช่น สีแดงผสมสีเขียวจะได้สีเหลือง (Yellow) สีน้ำเงินผสมกับสีเขียวได้สีฟ้า (Cyan) สีน้ำเงินผสมกับสีแดงได้สีแดงอมม่วง (Magenta) หรือหากนำทุกสีมาผสมกันจะได้สีขาว เป็นต้น



รูปที่ 2.1 รูปแบบสี RGB ในรูปลูกบาศก์ [8]

โดยระบบสีที่ใช้ในระบบคอมพิวเตอร์จะเป็นระบบสีแบบ 8 บิต คือการนำหน่วยความจำดิจิตอลขนาด 8 บิตมาแทนเฉดสีแต่ละสีในเชิงคณิตศาสตร์ แต่ละบิตจะมีค่าเป็น 0 และ 1 หมายความว่าจะมีเฉดสีที่เป็นไปได้ทั้งหมด 2^8 หรือสามารถเขียนในรูปเลขฐานสองได้ดังนี้ 00000000_2 (มีค่าเท่ากับ 0 ในเลขฐานสิบ) จนถึง 11111111_2 (มีค่าเท่ากับ 255 ในเลขฐานสิบ) หรือกล่าวได้ว่าในแต่ละสีจะมีเฉดสีแตกต่างกันทั้งหมด 256 เฉดสี



รูปที่ 2.2 แบบจำลองรูปแบบสี RGB [8]

2.1.1.2 รูปแบบสี CMY (CMY Color Model)

สีฟ้า (Cyan) สีแดงอมม่วง (Magenta) และสีเหลือง (Yellow) เป็นสีลำดับที่สองหรือสีทางเลือกของแสง ตัวอย่างเช่น เมื่อพื้นผิวถูกทาด้วยสีฟ้า เมื่อฉายแสงสีขาวลงบนพื้นผิว จะไม่มีแสงสีแดงสะท้อนกลับออกมาจากพื้นผิว หรือหมายถึงสีฟ้า จะลบแสงสีแดงออกจากการสะท้อนของแสงสีขาว แม้ว่าแสงสีขาวที่ฉายลงบนพื้นผิวจะประกอบไปด้วยแสงสีแดง สีเขียว และสีน้ำเงินในจำนวนที่เท่ากัน

อุปกรณ์ที่จะใช้รูปแบบสี CMY มักจะเป็นเครื่องปริ้นท์และเครื่องถ่ายเอกสาร ถ้าสีที่เข้ามาเป็นรูปแบบสี RGB ก็จะต้องมีการเปลี่ยนเป็นรูปแบบสี CMY โดยใช้สมการที่ (2.1)

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.1}$$

สมการที่ (2.1) เป็นสมการที่ผ่านการนอร์มอลไลซ์ให้มีค่า 0 ถึง 1 แล้ว จากสมการแสดงให้เห็นว่าถ้ามีพื้นผิวถูกเคลือบด้วยสีฟ้าจะไม่มีแสงสีแดงสะท้อนออกมา ($C = 1 - R$) ถ้าพื้นผิวเคลือบด้วยสีแดงอมม่วงจะไม่มีแสงสีเขียวสะท้อนออกมา และถ้าพื้นผิวเคลือบด้วยสีเหลืองจะไม่มีสีน้ำเงินสะท้อนออกมา สำหรับการประมวลผลภาพรูปแบบสี CMY จะถูกใช้เมื่อจะพิมพ์เอกสารออกมา (Hard Copy)

2.1.1.3 รูปแบบสี YIQ (YIQ Color Model)

รูปแบบสี YIQ ถูกใช้ในการออกอากาศของโทรทัศน์ โดยจะประกอบไปด้วย Y คือส่วนที่บรรจุค่าความสว่าง (Luminance) และเฉดสี (Hue) ของภาพไว้, I คือ Inphase โดยจะบรรจุค่าสีที่ใช้สร้างสีเนื้อผิว (Flesh-Tone Shading) และ Q คือ Quadrature จะบรรจุค่าสีของสีเขียวและสีแดงอมม่วง [9] รูปแบบสี YIQ ถูกใช้ในการออกอากาศของโทรทัศน์ โดยพื้นฐานแล้วสี YIQ จะถูกแปลงมาจากรูปแบบสี RGB เพื่อการส่งอย่างมีประสิทธิภาพและการแก้ไขให้มีความเหมาะสมกับมาตรฐานของโทรทัศน์แบบโมโนโครม ในความเป็นจริงแล้ว Y ใน YIQ จะเก็บข้อมูลเกือบทั้งหมดของวิดิโอที่โทรทัศน์แบบโมโนโครมต้องการ หรือกล่าวได้ว่าโทรทัศน์แบบโมโนโครมสามารถใช้เพียงค่า Y ค่าเดียวในการแสดงผลภาพได้ สำหรับการแปลงรูปสี RGB เป็นรูปแบบสี YIQ ทำได้ดังสมการที่ (2.2)

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2.2)$$

รูปแบบสี YIQ จะมีการใช้ Bandwidth ที่มาก (หรือใช้หลายบิตในกรณีของดิจิตอล) สำหรับการแสดงผลของ Y และใช้ Bandwidth ที่น้อยสำหรับ I และ Q (หรือใช้บิตที่น้อยในกรณีของดิจิตอล)

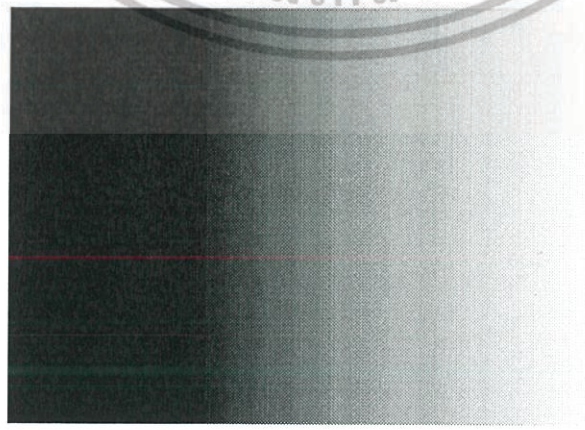
สำหรับการประมวลผลภาพ ความสว่าง (Y) และข้อมูลสี (I และ Q) จะถูกแยกออกจากกัน สิ่งสำคัญของการแยกส่วนของความสว่างออกมาคือการที่ความสว่างของรูปภาพจะถูกประมวลผล โดยไม่มีผลกระทบกับข้อมูลสี

2.1.1.4 รูปแบบสี HSI (HSI Color Model)

รูปแบบสี HSI เป็นระบบสีที่ใกล้เคียงกับการรับรู้สีของมนุษย์ โดย H คือ Hue หมายถึงคุณลักษณะของสีที่อธิบายสีบริสุทธิ์ (Pure Color) (สีเหลือง สีส้ม และสีแดง) S คือ Saturation หมายถึง ค่าองศาที่สีบริสุทธิ์จะถูกเจือจาง (Dilute) ด้วยแสงสีขาว หรือค่าความอิ่มตัวของสีเมื่อผสมกับแสงสีขาว I คือ Intensity หมายถึงค่าความเข้มของแสงหรือความสว่าง รูปแบบสี HSI จะใช้ประโยชน์จากสองส่วน อย่างแรก ส่วนของความสว่าง (Intensity) ถูกแยกออกมาจากข้อมูลสีในรูปภาพ อย่างที่สอง ส่วนของค่าสีบริสุทธิ์ (Hue) และความอิ่มตัวของสี (Saturate) จะใกล้เคียงกับการรับรู้สีของมนุษย์ จากลักษณะทั้งสองอย่างนี้ทำให้โมเดลสี HSI เป็นเครื่องมือในอุดมคติสำหรับการพัฒนาอัลกอริทึมของการประมวลผลภาพที่อยู่บนพื้นฐานของการรับรู้สีของมนุษย์ สำหรับการแปลงรูปแบบสี RGB เป็นรูปแบบสี HSI จะมีความยุ่งยากและซับซ้อนมาก ดังนั้นหากต้องการศึกษาข้อมูลเพิ่มเติมแนะนำให้ศึกษาจากหนังสือ “Digital Image Processing” ซึ่งเขียนโดย Rafael C. Gonzalez, and Richard E. Woods [2]

2.1.2 ภาพระดับสีเทา (Gray Scale Image)

จากรูปที่ 2.1 ภาพระดับสีเทาหมายถึงภาพที่มีค่าเริ่มต้นจากสีดำจนถึงสีขาว หรือหมายถึงค่าสีดำที่เริ่มต้นจากจุดกำเนิด (0, 0, 0) ไล่ระดับสีไปจนถึงค่าสีขาวที่อยู่ที่มุมที่ไกลที่สุดจากจุดกำเนิด (1, 1, 1) โดยกำหนดให้สีดำมีค่าเท่ากับ 0 และกำหนดให้ค่าสีขาวมีค่าเท่ากับ 255 หรือหมายถึงความเข้มแสงของภาพที่มีค่าตั้งแต่ 0 ถึง 255 สามารถแสดงการไล่เฉดสีระดับสีเทาได้ดังรูปที่ 2.3



รูปที่ 2.3 ภาพการไล่เฉดสีระดับสีเทา [1]

ภาพสีรูปแบบ RGB เป็นการนำสีสามสีคือ สีแดง สีเขียว และสีน้ำเงินมาผสมรวมกัน ทำให้เกิดเป็นภาพสีขึ้นมา การนำภาพสี RGB มาประมวลผลเลยจึงเป็นเรื่องยาก ดังนั้นเพื่อความสะดวกและง่ายในการประมวลผลจึงจะทำภาพสี RGB ให้กลายเป็นภาพระดับสีเทาเสียก่อน ซึ่งการทำให้ภาพสี RGB เป็นภาพระดับสีเทา คือการผสมความเข้มของสีทั้งสามสีเข้าด้วยกัน สามารถแสดงการคำนวณได้ดังสมการที่ (2.3)

$$\text{Gray Scale} = (\text{Red} \times 0.299) + (\text{Green} \times 0.587) + (\text{Blue} \times 0.114) \qquad (2.3)$$

เมื่อ

Gray Scale คือ ค่าระดับสีเทาในแต่ละจุดภาพของภาพสี RGB
Red คือ ค่าความเข้มของสีแดง
Green คือ ค่าความเข้มของสีเขียว
Blue คือ ค่าความเข้มของสีน้ำเงิน

จากสมการที่ (2.3) ค่าระดับสีเทาที่ได้ออกมาคือภาพระดับสีเทาของภาพสี RGB โดยจะมีค่าตั้งแต่ 0 ถึง 255 โดยที่ 0 หมายถึงสีดำ และ 255 หมายถึงสีขาว

2.1.3 ภาพสองระดับ (Binary Image)

ภาพสองระดับหมายถึงภาพที่มีความเข้มของสีเพียง 2 ระดับเท่านั้นคือ 0 และ 255 โดย 0 หมายถึงสีดำ และ 255 หมายถึงสีขาว ตามปกติแล้วการประมวลภาพที่เป็นภาพสี RGB จะนิยมเปลี่ยนภาพสีดังกล่าวเป็นภาพระดับสีเทา แล้วจึงเปลี่ยนภาพระดับสีเทาเป็นภาพสองระดับเพื่อลดข้อมูลของภาพลง ทำให้สามารถประมวลผลได้รวดเร็วขึ้นและยังลดพื้นที่ในการจัดเก็บข้อมูลด้วย

การสร้างภาพสองระดับสามารถทำได้โดยการใช้การตัดค่าเทรชโฮล (Threshold) ในภาพระดับสีเทาให้กลายเป็นภาพสองระดับ โดยค่าเทรชโฮลจะอยู่ในช่วง 0 ถึง 255 สมมติค่าเทรชโฮลที่เราต้องการคือ 127 หากค่าพิกเซลใดในภาพระดับสีเทามีค่าต่ำกว่า 127 จะกำหนดให้พิกเซลดังกล่าวมีค่า 0 หรือสีดำ แต่หากค่าพิกเซลใดมีค่ามากกว่า 127 ก็จะกำหนดให้มีค่า 255 หรือสีขาว สามารถแสดงได้ดังสมการที่ (2.4)

$$g(x,\ y) = \begin{cases} 255 & \text{ถ้า } f(x,\ y) \geq T \\ 0 & \text{ถ้า } f(x,\ y) < T \end{cases} \qquad (2.4)$$

เมื่อ

$g(x,\ y)$ คือ ภาพสองระดับ
$f(x,\ y)$ คือ ภาพระดับสีเทา
T คือ ค่าเทรชโฮล

การเลือกค่าเทรชโฮลขึ้นอยู่กับว่าต้องการให้พิกเซลจุดใดของภาพเป็นสีขาวหรือสีดำ ต้องการส่วนไหนของภาพในการประมวลผล จึงจำเป็นต้องเลือกค่าให้มีความเหมาะสม ให้ภาพในส่วนที่ต้องการมีความชัดมากที่สุด รูปที่ 2.4 แสดงตัวอย่างการตัดเทรชโฮลที่เหมาะสม

รูปที่ 2.4 ภาพแสดงการตัดค่าเทรชโฮลที่เหมาะสม
(ก) ภาพระดับสีเทา
(ข) ภาพสองระดับที่ได้จากการตัดเทรชโฮลที่เหมาะสม
(ค) ภาพสองระดับที่ได้จากการเลือกค่าเทรชโฮลที่ต่ำเกินไป
(ง) ภาพสองระดับที่ได้จากการเลือกค่าเทรชโฮลที่สูงเกินไป

2.2 ทฤษฎีการหาขอบภาพ (Edge Detection)

การหาขอบภาพคือการหาขอบของวัตถุที่อยู่ในภาพ โดยขอบภาพจะแสดงถึงรูปร่างของวัตถุ การหาขอบภาพสามารถหาได้จากการวัดการเปลี่ยนแปลงความเข้มของแสง ที่ถูกสะท้อนออกมาจากการที่แสงกระทบกับวัตถุในตำแหน่งต่างๆ โดยหากตำแหน่งหรือจุดพิกเซลใดเมื่อเปรียบเทียบกับตำแหน่งหรือจุดพิกเซลใกล้เคียงแล้วพบว่ามีความเข้มของแสงที่แตกต่างกันมาก แสดงว่าตำแหน่งดังกล่าวเป็นขอบภาพ

2.2.1 การหาขอบภาพด้วยวิธีโซเบล (Sobel Edge Detection)

การหาขอบภาพด้วยวิธีโซเบล เป็นการคำนวณในกลุ่มของเกรเดี้ยน (Gradient) คือการหาขอบภาพด้วยสมการอนุพันธุ์อันดับหนึ่ง โดยแยกหาผลความแตกต่างของความเข้มแสงของภาพในแนวแกนนอนกับแนวแกนตั้ง แล้วหาผลรวมความแตกต่างของทั้งสองแกนด้วยวิธีเวคเตอร์ โดยเกรเดี้ยนของรูปภาพ f(x, y) ณ ตำแหน่ง (x, y) จะอยู่ในรูปเวคเตอร์ ∇f ดังสมการที่ (2.5) เมื่อ G_x และ G_y คือผลของการหาอนุพันธุ์อันดับหนึ่งในแนวแกน x และ y ตามลำดับ

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.5}$$

ลักษณะของภาพจะเป็นเมทริกซ์ 2 มิติ การดำเนินการหาขอบภาพจะทำโดยการหาความแตกต่างของแต่ละจุดโดยนำหน้าต่างขนาด 3 x 3 ในรูปที่ 2.5 วางทาบบนรูปภาพ f(x, y) แล้วนำค่าน้ำหนักในหน้าต่างคูณกับจุดภาพ และนำมาค่าที่ได้มารวมกันเรียกว่าการคอนโวลูชั่น (Convolution) โดยทำเช่นนี้กับทุกๆจุดของภาพ

-1	-2	-1
0	0	0
1	2	1

(ก)

-1	0	1
-2	0	2
-1	0	1

(ข)

รูปที่ 2.5 หน้าต่างโซเบลสำหรับหาขอบภาพในแนวนอนและแนวตั้ง
(ก) หน้าต่างสำหรับหาขอบภาพในแนวนอน
(ข) หน้าต่างสำหรับหาขอบภาพในแนวตั้ง

การทำคอนโวลูชั่นจะแยกทำแกนนอนกับแกนตั้ง ได้ผลลัพธ์ออกมาเป็น G_x และ G_y ตามลำดับ จึงต้องนำภาพในแกนนอนและแกนตั้งมารวมกันทางเวคเตอร์ โดยขนาด (Magnitude) ของเกรเดี้ยนหาได้จากสมการที่ (2.6)

$$\nabla f = mag(\nabla \boldsymbol{f}) = \left[G_x^2 + G_y^2\right]^{1/2} \tag{2.6}$$

หรือหาขนาดของเกรเดี้ยนได้จากค่าสัมบูรณ์ (Absolute Values) ดังแสดงในสมการที่ (2.7)

$$\nabla f \approx |G_x| + |G_y| \tag{2.7}$$

และหาขนาดมุมหรือทิศทางของเกรเดี้ยนได้จากสมการที่ (2.8)

$$\alpha(x,y) = tan^{-1}\left(\frac{G_y}{G_x}\right) \quad (2.8)$$

เมื่อหาขนาดของเกรเดี้ยนออกมาแล้ว ผลลัพธ์ที่ได้จะเป็นการรวมกันของภาพในแกนนอนและแกนตั้ง ส่วนขนาดมุมของเกรเดี้ยน เป็นการหาเส้นขอบโดยใช้ความแตกต่างของข้อมูลสี จากนั้นจะนำภาพที่หาขนาดของเกรเดี้ยนออกมาแล้ว ไปหาขอบภาพโดยเลือกค่าเทรชโฮลที่เหมาะสมเพื่อทำเป็นภาพสองระดับต่อไป

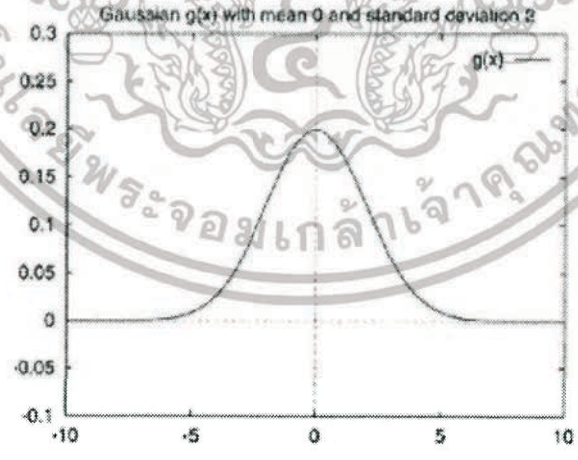
2.2.2 การหาขอบภาพด้วยวิธี Canny (Canny Edge Detection)

ทฤษฎีการหาขอบภาพด้วยวิธี Canny ได้ศึกษาจากงานวิจัยของ John Canny, "A Computational Approach to Edge Detection", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6, NOVEMBER 1986, pp. 679-698. [4] โดยสามารถอธิบายอัลกอริทึมในการหาขอบภาพดังนี้

ในขั้นตอนแรกของการหาขอบภาพด้วยอัลกอริทึมของ Canny จะเริ่มจากการสร้างหน้าต่างตัวตรวจจับเส้นขอบซึ่งใช้สมการเก๊าเซี้ยน (Gaussian) และสมการอนุพันธ์อันดับที่หนึ่งของเก๊าเซี้ยน โดยสามารถแสดงสมการเก๊าเซี้ยน (G_σ) หนึ่งมิติ (One-Dimension) ได้ในสมการที่ (2.9) เมื่อ σ คือค่าซิกม่าซึ่งจะควบคุมการกระจายของสมการเก๊าเซี้ยน

$$G_\sigma = exp\left(-\frac{x^2}{2\sigma^2}\right) \quad (2.9)$$

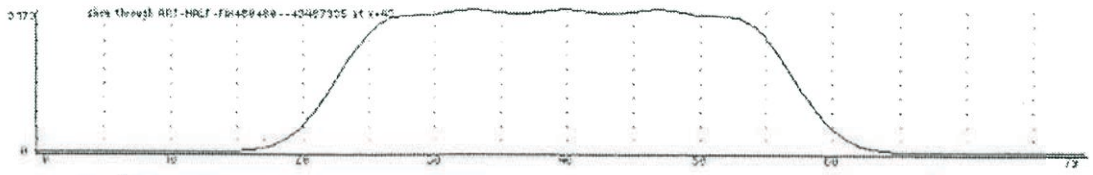
ลักษณะการกระจายของสมการเก๊าเซี้ยนสามารถแสดงได้ดังรูปที่ 2.6



รูปที่ 2.6 การกระจายของสมการเก๊าเซี้ยน [10]

จากนั้นแทรกค่าสูงสุด (Maximum Value) ของค่าการกระจายของสมการเก๊าเซี้ยน ลงไปในผลลัพธ์ของการกระจายของสมการเก๊าเซี้ยน ตามปกติแล้วการกระจายของเก๊าเซี้ยนจะมีค่าสูงสุดเพียงค่าเดียว แต่การแทรกค่าสูงสุดนี้จะทำให้มีค่าสูงสุดมากกว่าหนึ่งค่า การแทรกค่าสูงสุดนี้ทำเพื่อเพิ่ม

ประสิทธิภาพในการกรองสัญญาณรบกวนคือสิ่งที่ไม่ใช่เส้นขอบ และเพิ่มความชัดเจนของเส้นขอบ แต่ผลจากการแทรกค่าสูงสุดลงในผลของการกระจายของสมการเก๊าเซี้ยน คือจะทำให้เส้นขอบที่มีลักษณะโค้งถูกกรองออกไปด้วย สามารถแสดงตัวอย่างผลการแทรกค่าสูงสุดได้ในรูปที่ 2.7 ซึ่งจะเห็นได้ว่ามีค่าสูงสุดมากกว่าหนึ่งค่า และกำหนดให้ผลของการกระจายของสมการเก๊าเซี้ยนที่ผ่านการแทรกค่าสูงสุดแล้วคือ G_{σ}^{new}



รูปที่ 2.7 ตัวอย่างผลของการแทรกค่าสูงสุดของการกระจายของสมการเก๊าเซี้ยน [4]

จากนั้นหาสมการอนุพันธุ์อันดับที่หนึ่งของสมการเก๊าเซี้ยน (G'_{σ}) ได้ดังสมการที่ (2.10)

$$G'_{\sigma} = -\frac{x}{\sigma^2} exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{2.10}$$

ลักษณะของสมการอนุพันธุ์อันดับที่หนึ่งของเก๊าเซี้ยนสามารถแสดงได้ดังรูปที่ 2.8

รูปที่ 2.8 การกระจายของสมการอนุพันธุ์อันดับที่หนึ่งของเก๊าเซี้ยน [10]

จากนั้นนำสมการอนุพันธุ์อันดับหนึ่งของสมการเก๊าเซี้ยนมาทรานสโพส (Transpose) ซึ่งเป็นการจัดเรียงในทิศทางแนวตั้ง หรือคือการสลับแถวและหลัก จากนั้นนำสมการอนุพันธุ์อันดับหนึ่งของสมการเก๊าเซี้ยนที่สลับแถวและหลักแล้ว มาคูณกับสมการเก๊าเซี้ยนที่ผ่านการแทรกค่าสูงสุดแล้ว (G_{σ}^{new}) แสดงได้ดังสมการที่ (2.11)

$$H_{\sigma} = G_{\sigma}^{new} \times (G'_{\sigma})^{T} \tag{2.11}$$

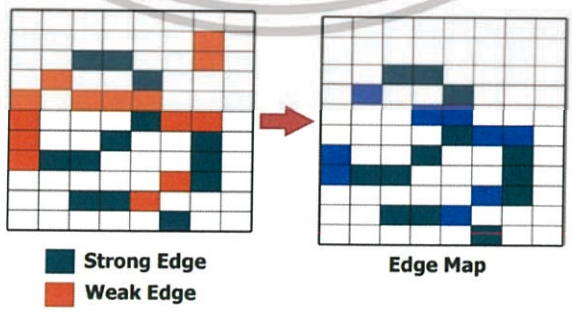
เมื่อ T คือทรานสโพส และ H_σ คือหน้าต่างที่เกิดจากการคูณกันของสมการเก๊าเซี้ยนที่ผ่านการแทรกค่าสูงสุดกับสมการอนุพันธุ์อันดับหนึ่งของสมการเก๊าเซี้ยนที่สลับแถวและหลัก หรือคือหน้าต่างตรวจจับเส้นขอบ โดยหน้าต่างนี้จะตรวจจับเส้นขอบในทิศทาง 0 องศาหรือทิศทางแนวนอนเท่านั้น จากนั้นเราจะนำหน้าต่าง H ไปคอนโวลูชั่นกับภาพ I แสดงได้ดังสมการที่ (2.12)

$$S = I * H_\sigma \tag{2.12}$$

เมื่อ S คือภาพ I ที่ผ่านการคอนโวลูชั่นกับหน้าต่าง H แล้ว แต่ภาพ S ที่หามาได้นั้นจะเป็นการหาเส้นขอบในทิศทางแนวนอน (Horizontal) หรือทิศทาง 0 องศา เท่านั้น ซึ่งการหาขอบภาพจะต้องหาขอบภาพในทิศทางอื่นด้วย เพื่อให้ขอบภาพมีความสมบูรณ์เนื่องจากขอบภาพสามารถมีได้หลายทิศทาง ดังนั้นจะต้องหาขอบภาพในทิศทางอื่นๆอีก 5 ทิศทาง คือทิศทาง 30, 60, 90, 120 และ 150 องศา รวมทั้งหมด 6 ทิศทาง หรือเพิ่มขึ้นครั้งละ 30 องศา วิธีการหาขอบภาพในทิศทางอื่นๆทำได้โดยการหมุนหน้าต่าง H ไปในทิศทาง 30, 60, 90, 120 และ 150 องศา

เมื่อหาขอบภาพมาได้ครบ 6 ทิศทางแล้ว จากนั้นจะนำขอบภาพจากทั้ง 6 ทิศทางนี้มารวมกันโดยการเลือกค่าสูงสูง (Maximum Value) ของแต่ละทิศทางในแต่ละจุดภาพมาใช้งาน จะได้เส้นขอบภาพที่มีสูงสุดของแต่ละทิศทางมาใช้งาน แต่เนื่องจากวิธีการนี้อาจยังเหลือขอบภาพที่ไม่ใช่ขอบจริงอยู่ ดังนั้นจึงจะตัดเทรชโฮลเพื่อให้เหลือเพียงขอบภาพจริงเท่านั้น

โดยการตัดเทรชโฮลนี้จะการตัดค่าเทรชโฮลสองระดับ หรือคือการใช้ฮีสเตอร์รีซิสเทรชโฮล (Hysteresis Threshold) เพื่อหาขอบภาพที่แท้จริง โดยการใช้เทรชโฮลสองระดับนั้น จะเป็นการกำหนดเทรชโฮลที่แตกต่างกันสองค่าคือ เทรชโฮลค่าสูง (High Threshold) และ เทรชโฮลค่าต่ำ (Low Threshold) สำหรับวิธีการเลือกค่าเทรชโฮลทั้งสองนั้น อาจจะใช้ค่าเทรชโฮลสูงซึ่งมีค่าเป็นสองเท่าของเทรชโฮลค่าต่ำ โดยจุดพิกเซลใดที่มีขนาดต่ำกว่าเทรชโฮลค่าต่ำ จะถือว่าไม่ใช่ขอบ แต่ถ้าพิกเซลใดมีขนาดสูงกว่าเทรชโฮลค่าสูง จะถือว่าเป็นขอบ (Strong Edge) แต่ถ้าพิกเซลใดมีขนาดอยู่ระหว่างเทรชโฮลค่าต่ำและค่าสูง (Weak Edge) จะต้องพิจารณาว่าพิกเซลดังกล่าวเชื่อมต่อกับขอบหรือไม่ ถ้ามีการเชื่อมต่อกับขอบ (Strong Edge) จะถือว่าเป็นขอบ โดยสามารถแสดงตัวอย่างได้ในรูปที่ 2.9



รูปที่ 2.9 ตัวอย่างการพิจารณาเส้นขอบ [13]

จากรูปที่ 2.9 ด้านซ้ายจะเห็นว่าพิกเซลที่เป็นขอบคือพิกเซลสีเขียว และพิกเซลที่มีขนาดอยู่ระหว่างเทรชโฮลค่าต่ำและค่าสูง ซึ่งอยู่ระหว่างพิจารณาว่าเป็นขอบหรือไม่จะมีสีส้ม โดยหลักการคือหากพิกเซลสีส้มเชื่อมติดอยู่กับพิกเซลสีเขียวซึ่งเป็นขอบ จะให้ถือว่าพิกเซลนั้นเป็นขอบดังแสดงในรูปขวาที่จะเปลี่ยนสีฟ้าแทน และหากพิกเซลสีส้มใด ไม่เชื่อมติดกับพิกเซลที่เป็นขอบ (สีเขียว) พิกเซลนั้นก็จะถูกลบทิ้งไป กลายเป็นช่องว่างดังแสดงในรูปที่ 2.9 ด้านขวา

2.3 ทฤษฎีการหาแกนกลาง (Skeleton Detection)

การหาแกนกลางของภาพคือการทำให้รูปภาพที่มีความหนา เหลือความหนาเพียงหนึ่งพิกเซล โดยภาพที่จะนำไปหาแกนกลางนั้นจะต้องเป็นภาพสองระดับเท่านั้น หลักการหาแกนกลางของภาพจะประกอบไปด้วยขั้นตอนสองขั้นตอน โดยขั้นตอนแรกจะเป็นการลบจุดภาพทางด้านขวามือ ด้านล่าง และมุมซ้ายบน และขั้นตอนที่สองจะเป็นการลบจุดภาพทางด้านซ้ายมือ ด้านบน และมุมขวาล่าง จนกระทั่งเหลือเพียงแกนกลางของรูปภาพที่มีความหนา 1 พิกเซล

การหาแกนกลางของภาพจะเริ่มต้นจากการหาจุดพิกเซลที่มีค่า 1 แล้วนำหน้าต่างขนาด 3 x 3 ในรูปที่ 2.10 ไปวางทาบโดยให้จุด p1 อยู่บนพิกเซลที่มีค่า 1

p9	p2	p3
p8	p1	p4
p7	p6	p5

รูปที่ 2.10 หน้าต่างขนาด 3 x 3

การหาแกนกลางของภาพจะมีทั้งหมดสองขั้นตอนคือ

<u>ขั้นตอนที่หนึ่ง</u>

จะเป็นการลบจุดภาพทางด้านขวามือ ด้านล่าง และมุมซ้ายบน มีเงื่อนไขในการตรวจสอบดังนี้

a) $2 \le N(p1) \le 6$

b) $S(p1) = 1$

c) $p2 \cdot p4 \cdot p6 = 0$

d) $p4 \cdot p6 \cdot p8 = 0$

เมื่อ

N(p1) คือ ผลรวมของจุดภาพทั้งหมดที่มีค่า 1 ที่อยู่ล้อมรอบจุด p1

S(p1) คือ จำนวนครั้งที่มีการเปลี่ยนแปลงค่าจาก 0 ไปเป็นค่า 1 รอบจุด p1

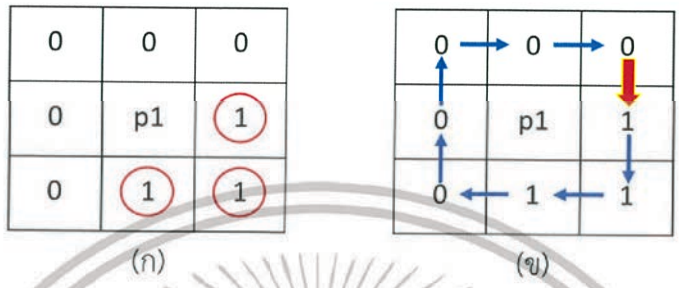
" • " คือ เครื่องหมาย AND

มีคำอธิบายเงื่อนไขดังนี้

เงื่อนไข a) เป็นการนับจำนวนจุดรอบจุด p1 ที่มีค่าเป็น 1 โดยจุดภาพที่ล้อมรอบอยู่จะต้องมีจำนวนตั้งแต่ 2 ถึง 6 ค่า หรือ N(p1) = p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9

เงื่อนไข b) เป็นการนับจำนวนครั้ง ที่จุดภาพที่มีค่าเป็น 0 แล้วตัวต่อไปเปลี่ยนค่าเป็น 1 รอบจุด p1

สามารถแสดงตัวอย่างการทำงานของเงื่อนไข a) และ b) ได้ในรูปที่ 2.11



รูปที่ 2.11 ตัวอย่างการทำงานของเงื่อนไข a) และ b)
(ก) เงื่อนไข a) ตัวอย่างจุดรอบจุด p1 ที่มีค่าเป็น 1
(ข) เงื่อนไข b) ตัวอย่างการเปลี่ยนค่าจาก 0 เป็น 1

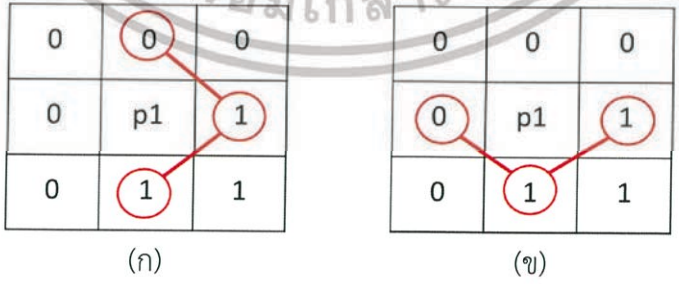
จากรูปที่ 2.11 (ก) เป็นการแสดงเงื่อนไข a) จะเห็นว่ามีจุดภาพที่มีค่าเป็น 1 รอบจุด p1 มีจำนวนทั้งหมด 6 จุด ดังนั้น N(p1) มีค่าเท่ากับ 3

จากรูปที่ 2.11 (ข) เป็นการแสดงเงื่อนไข b) จะเห็นว่ามีจุดภาพที่มีค่าเป็น 0 แล้วตัวต่อไปเปลี่ยนค่าเป็น 1 รอบจุด p1 จำนวนทั้งหมด 1 ครั้ง เพราะฉะนั้นจะได้ S(p1) มีค่าเท่ากับ 1

เงื่อนไข c) เป็นการ AND กันทางลอจิกของจุดภาพ p2, p4 และ p6 โดยต้องมีค่าเท่ากับ 0

เงื่อนไข d) เป็นการ AND กันทางลอจิกของจุดภาพ p4, p6 และ p8 โดยต้องมีค่าเท่ากับ 0

สามารถแสดงตัวอย่างการทำงานของเงื่อนไข c) และ d) ได้ในรูปที่ 2.12

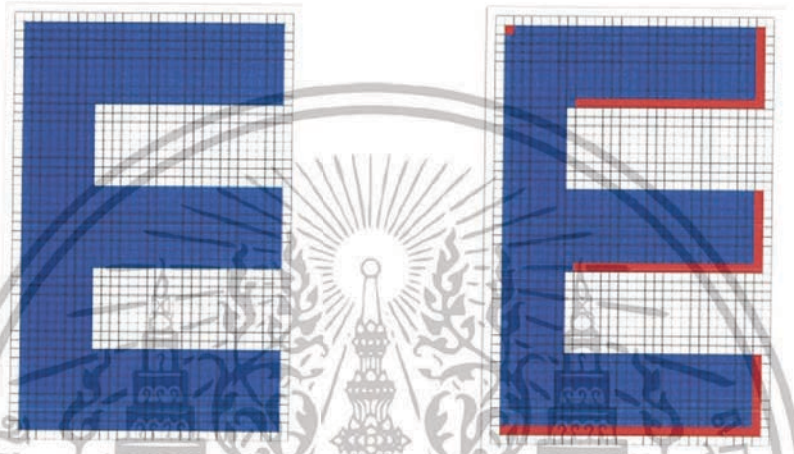


รูปที่ 2.12 ตัวอย่างการทำงานของเงื่อนไข c) และ d)
(ก) เงื่อนไข c) ตัวอย่างการ AND กันของจุดภาพ
(ข) เงื่อนไข d) ตัวอย่างการ AND กันของจุดภาพ

จากรูปที่ 2.12 (ก) เป็นการแสดงเงื่อนไข c) จะเห็นว่าจุดภาพ p2, p4 และ p6 มีค่า 0, 1 และ 1 ตามลำดับ เมื่อ AND กันจะมีค่าเท่ากับ 0

จากรูปที่ 2.12 (ข) เป็นการแสดงเงื่อนไข d) จะเห็นว่าจุดภาพ p4, p6 และ p8 มีค่า 1, 1 และ 0 ตามลำดับ เมื่อ AND กันจะมีค่าเท่ากับ 0

หากจุดใดเข้าเงื่อนไขทั้ง 4 ข้อ จะเก็บค่าตำแหน่งนั้นไว้ และดำเนินการตรวจสอบจุดภาพทุกจุดจนหมดทั้งภาพ แล้วจึงลบจุดภาพทั้งหมด แสดงตัวอย่างการหาแกนกลางในขั้นตอนที่หนึ่งในรูปที่ 2.13 โดยสีแดงในรูปที่ 2.13 (ข) คือจุดที่ถูกลบออกไปในขั้นตอนที่หนึ่ง



(ก) (ข)

รูปที่ 2.13 ตัวอย่างการลบจุดภาพตามขั้นตอนที่หนึ่ง เพื่อหาแกนกลางของภาพ
(ก) ภาพตัวอย่างที่จะทำการหาแกนกลาง
(ข) ขั้นตอนที่หนึ่งจะลบจุดภาพทางด้านขวามือ ด้านล่าง และมุมซ้ายบน

<u>ขั้นตอนที่สอง</u>

จะเป็นการลบจุดภาพทางด้านซ้ายมือ ด้านบน และมุมขวาล่าง มีเงื่อนไขดังนี้

a) $2 \le N(p1) \le 6$

b) $S(p1) = 1$

c*) $p2 \cdot p4 \cdot p8 = 0$

d*) $p2 \cdot p6 \cdot p8 = 0$

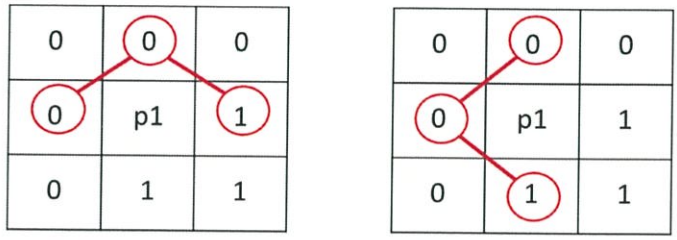
โดยมีคำอธิบายเงื่อนไขดังนี้

เงื่อนไข a) และ b) จะเหมือนกับคำอธิบายในขั้นตอนที่หนึ่ง

เงื่อนไข c*) ของขั้นตอนที่สองเป็นการ AND กันทางลอจิกของจุดภาพ p2, p4 และ p8 โดยต้องมีค่าเท่ากับ 0

เงื่อนไข d*) ของขั้นตอนที่สองเป็นการ AND กันทางลอจิกของจุดภาพ p2, p6 และ p8 โดยต้องมีค่าเท่ากับ 0

สามารถแสดงตัวอย่างการทำงานของเงื่อนไข c*) และ d*) ได้ในรูปที่ 2.14

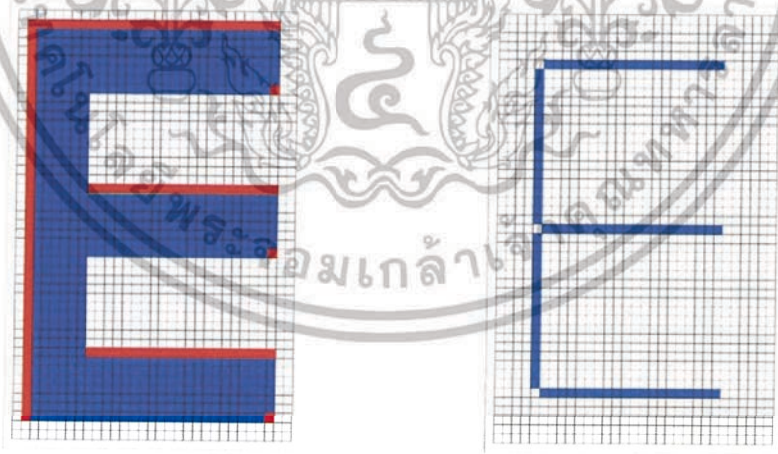


รูปที่ 2.14 ตัวอย่างการทำงานของเงื่อนไข c*) และ d*)
(ก) เงื่อนไข c*) ตัวอย่างการ AND กันของจุดภาพ
(ข) เงื่อนไข d*) ตัวอย่างการ AND กันของจุดภาพ

จากรูปที่ 2.14 (ก) เป็นการแสดงเงื่อนไข c*) จะเห็นว่าจุดภาพ p2, p4 และ p8 มีค่า 0, 1 และ 0 ตามลำดับ เมื่อ AND กันจะมีค่าเท่ากับ 0

จากรูปที่ 2.14 (ข) เป็นการแสดงเงื่อนไข d*) จะเห็นว่าจุดภาพ p2, p6 และ p8 มีค่า 0, 1 และ 0 ตามลำดับ เมื่อ AND กันจะมีค่าเท่ากับ 0

หากจุดใดเข้าเงื่อนไขทั้ง 4 ข้อ จะเก็บค่าตำแหน่งนั้นไว้ และดำเนินการตรวจสอบจุดภาพทุกจุดจนหมดทั้งภาพ แล้วจึงลบจุดภาพทั้งหมด แสดงตัวอย่างการหาแกนกลางในขั้นตอนที่สองในรูปที่ 2.15 (ก) ซึ่งเป็นขั้นตอนต่อจากรูปที่ 2.13 (ข) โดยสีแดงในรูปที่ 2.15 (ก) คือจุดที่ถูกลบออกไปในขั้นตอนที่หนึ่ง

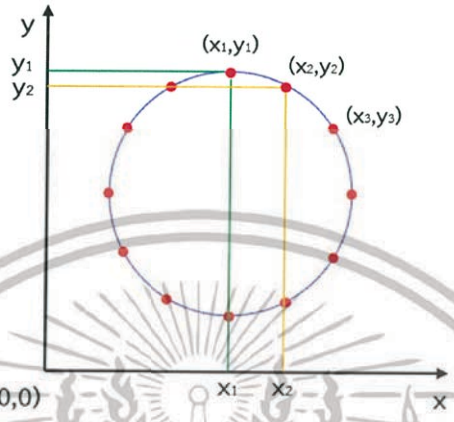


รูปที่ 2.15 ตัวอย่างการลบจุดภาพตามขั้นตอนที่สองเพื่อหาแกนกลางของภาพ
(ก) ขั้นตอนที่สองจะลบจุดภาพทางด้านซ้ายมือ ด้านบน และมุมขวาล่าง
(ข) ผลลัพธ์ที่ได้จากการหาแกนกลางของภาพ

จากนั้นตรวจสอบภาพโดยใช้ขั้นตอนที่หนึ่งและขั้นตอนที่สองวนซ้ำไปเรื่อยๆ จนกว่าจะเหลือเพียงแกนกลางของภาพความหนา 1 พิกเซล ดังแสดงในรูปที่ 2.15 (ข)

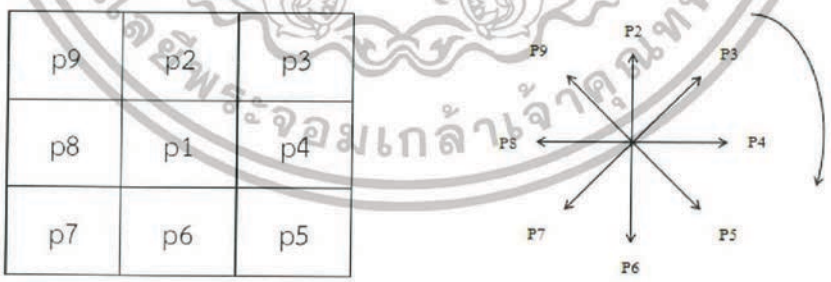
2.4 ทฤษฎีการหาตำแหน่งพิกัด (Coordinate Detection)

การหาตำแหน่งพิกัดคือการค้นหาตำแหน่งของจุดภาพแล้วเก็บตำแหน่งพิกัดของจุดภาพดังกล่าวแยกในแนวแกน x และแนวแกน y โดยรูปภาพที่ใช้ในการหาจุดตำแหน่งพิกัดของภาพ จะต้องเป็นภาพสองระดับและผ่านการทำให้บางหรือหาแกนกลางจนเหลือความหนาของภาพเพียง 1 พิกเซลมาแล้ว เพื่อให้สามารถจัดเก็บตำแหน่งพิกัดได้อย่างถูกต้อง



รูปที่ 2.16 พิกัดจุด (x_1, y_1) และ (x_2, y_2) ของรูปวงกลม

จากรูปที่ 2.16 เป็นการแสดงจุดพิกัดรูปวงกลม ในระนาบ x และ y โดยสมมติให้มีจุดพิกัดจำนวน n จุด กำหนดให้จุดที่อยู่บริเวณเหนือสุดของวงกลมเป็นจุด (x_1, y_1) และเป็นจุดเริ่มต้นของจุดพิกัดรูปวงกลม จากนั้นกำหนดให้จุดพิกัดต่อไปของรูปวงกลมคือจุดพิกัด (x_2, y_2), (x_3, y_3) ไปเรื่อยๆ จนถึงจุดพิกัดสุดท้ายที่ (x_n, y_n) โดยจุดพิกัดเหล่านี้จะถูกจัดเก็บแยกกันในแนวแกน x คือจุด x_u และแนวแกน y คือจุด y_u เมื่อ u คือตำแหน่งที่ใช้ในการจัดเก็บจุดพิกัด กำหนดให้ u มีค่าเท่ากับ 1,2,3 ไปจนถึงตัวที่ N



รูปที่ 2.17 วิธีการหาจุดพิกัดรอบจุด p1 [1]

การหาจุดพิกัดเริ่มจากการหาจุดพิกเซลที่มีค่า 1 จุดแรก จากนั้นนำหน้าต่างขนาด 3 x 3 มาวางทาบโดยให้จุด p1 อยู่บนพิกเซลที่มีค่า 1 นั้น แล้วเก็บค่าพิกัดในแนวแกน x และ y เป็นจุดที่ (x_1, y_1) เมื่อ x และ y คือค่าพิกัดที่จัดเก็บในแนวแกน x และ y จากนั้นพิจารณาทิศทาง 8 ทิศทางรอบจุด p1 ซึ่งการพิจารณาทิศทางรอบจุด p1 จะขึ้นอยู่กับความเหมาะสมและเงื่อนไขในการจัดเก็บจุดพิกัดของวัตถุที่ต้องการ ในที่นี้ต้องการจัดเก็บจุดพิกัดของรูปวงกลมในรูปที่ 2.16 ดังนั้นจะตรวจสอบจุด p2,

p3, p4, p5, p6, p7, p8 และ p9 ตามลำดับ เมื่อพบว่าจุดใดมีค่าเป็น 1 ก็จะเลื่อนหน้าต่าง 3 x 3 โดยให้จุด p1 ไปอยู่บนพิกเซลที่มีค่า 1 นั้นและเก็บค่าพิกัดในแนวแกน x และ y เช่นกัน เป็นจุดที่ (x_2, y_2) และเมื่อจัดเก็บค่าพิกัดของพิกเซลใดๆแล้ว ให้ลบพิกเซลนั้นทิ้งไปเพื่อป้องกันการจัดเก็บจุดพิกัดที่ซ้ำซ้อนกัน การดำเนินการจะทำซ้ำไปเรื่อยๆ จนกระทั่งเก็บค่าพิกัดได้ครบหมดทุกจุดหรือถึงจุดที่ (x_n, y_n) จากนั้นนำจุดพิกัดมาจัดเก็บแยกกันในแนวแกน x เป็น x_u และแนวแกน y คือ y_u เมื่อกำหนดให้ u มีค่าเท่ากับ 1,2,3 ไปจนถึงตัวที่ N โดย u คือตำแหน่งที่ใช้จัดเก็บจุดพิกัด

2.5 ทฤษฎีการนอร์มอลไลเซชัน (Normalization)

การนอร์มอลไลเซชัน คือการจัดเรียงและหมุนจุดพิกัดให้มาอยู่ในแนวแกนเดียวกันและหาจุดศูนย์กลางของของจุดพิกัด การนอร์มอลไลเซชันจะเป็นการจัดเรียงจุดพิกัดใหม่ โดยเริ่มจากการหาค่าเฉลี่ยของตำแหน่งพิกัดดังสมการที่ (2.13) และ (2.14)

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.13}$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{2.14}$$

เมื่อ

$\bar{x}$ และ $\bar{y}$ เป็น ค่าเฉลี่ยของจุดพิกัดในแนวแกน x และ y ตามลำดับ
x_i และ y_i เป็นจุดพิกัดของวัตถุในแนวแกน x และ y ตามลำดับ
i คือ ตำแหน่งจุดพิกัดมีค่าตั้งแต่ 1, 2, 3 จนถึง N
N คือ จำนวนจุดพิกัดทั้งหมดของวัตถุที่ถูกคำนวณ

จากนั้นหาค่าเบี่ยงเบน (x_i, y_i) จากจุดศูนย์กลางหรือค่าเฉลี่ยของจุดพิกัดของวัตถุได้จากสมการที่ (2.15)

$$\mathrm{x}_i = x_i - \bar{x} \quad \text{และ} \quad \mathrm{y}_i = y_i - \bar{y} \tag{2.15}$$

และหาค่าเมทริกซ์ความแปรปรวนร่วม (Covariant Matrix : C) ดังสมการที่ (2.16)

$$C = \frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix}(\mathrm{x}_i, \mathrm{x}_i) & (\mathrm{x}_i, \mathrm{y}_i)\\ (\mathrm{y}_i, \mathrm{x}_i) & (\mathrm{y}_i, \mathrm{y}_i)\end{bmatrix} \tag{2.16}$$

หาค่าไอเกน (Eigen Value) และค่าไอเกนเวคเตอร์ (Eigen Vector) เพื่อนำมาใช้ในการปรับตำแหน่งของจุดพิกัดของวัตถุ หรือตำแหน่งของรูปภาพให้หมุนมาอยู่ในแนวแกนเดียวกัน ค่าไอเกนและค่าไอเกนเวคเตอร์สามารถหาได้จากสมการที่ (2.17)

$$Cv = \lambda v \tag{2.17}$$

เมื่อ

λ คือ ค่าไอเกน

v คือ ค่าไอเกนเวคเตอร์

เมื่อได้ค่าไอเกน และ ค่าไอเกนเวคเตอร์มาแล้ว จะนำค่าเบี่ยงเบนจากจุดศูนย์กลางของจุดพิกัดของวัตถุทั้งในแนวแกน x และ y ที่ได้จากสมการที่ (2.15) มาคูณกับค่าไอเกนเวคเตอร์ (Eigen Vector) เพื่อคำนวณหาจุดพิกัดอันใหม่ของของวัตถุหรือภาพในแนวแกน x และ y ได้ดังสมการที่ (2.18)

$$\mathbb{X}_i = \mathrm{x}_i v \quad \text{และ} \quad \mathbb{Y}_i = \mathrm{y}_i v \tag{2.18}$$

เมื่อ

$\mathbb{X}_i$และ $\mathbb{Y}_i$ คือ จุดพิกัดอันใหม่ของวัตถุหรือจุดพิกัดที่ถูกนอร์มอลไลซ์แล้วในแนวแกน x และ y

2.6 ทฤษฎีการอินเตอร์โพเลชัน (Interpolation)

การอินเตอร์โพเลชันคือกระบวนการแทรกจุดเข้าไประหว่างจุดภาพสองจุดที่มีอยู่เดิม เพื่อเพิ่มหรือลดจำนวนจุดพิกัด ในการแทรกจุดพิกัดของภาพจะใช้คำสั่ง interp1 ซึ่งเป็นเครื่องมือที่มีอยู่ในโปรแกรม MATLAB หมายถึง 1-D Interpolation คือการประมาณค่าโดยกำหนดจำนวนจุดพิกัดที่ต้องการจากอินพุตเพียงตัวแปรเดียว มีรูปแบบการใช้งานคำสั่งดังนี้

```
Var_Interp = interp1(Sam,Var,Samq,'Option')
```

มีคำอธิบายดังนี้

Sam แทนจำนวนจุดพิกัดของตัวแปร Var มีรูปแบบดังเช่น Sam = 1:2:600 หมายความว่ามีจุดที่ต้องการให้ Sampling เริ่มตั้งแต่จุดที่ 1 ถึงจุดที่ 600 โดยจะมีจุดที่ถูก Sampling จำนวน 600/2 = 300 จุด

Var แทนตัวแปรที่ต้องการแทรกจุดพิกัดลงไป เช่น จุดพิกัดในแนวแกน x หรือ y

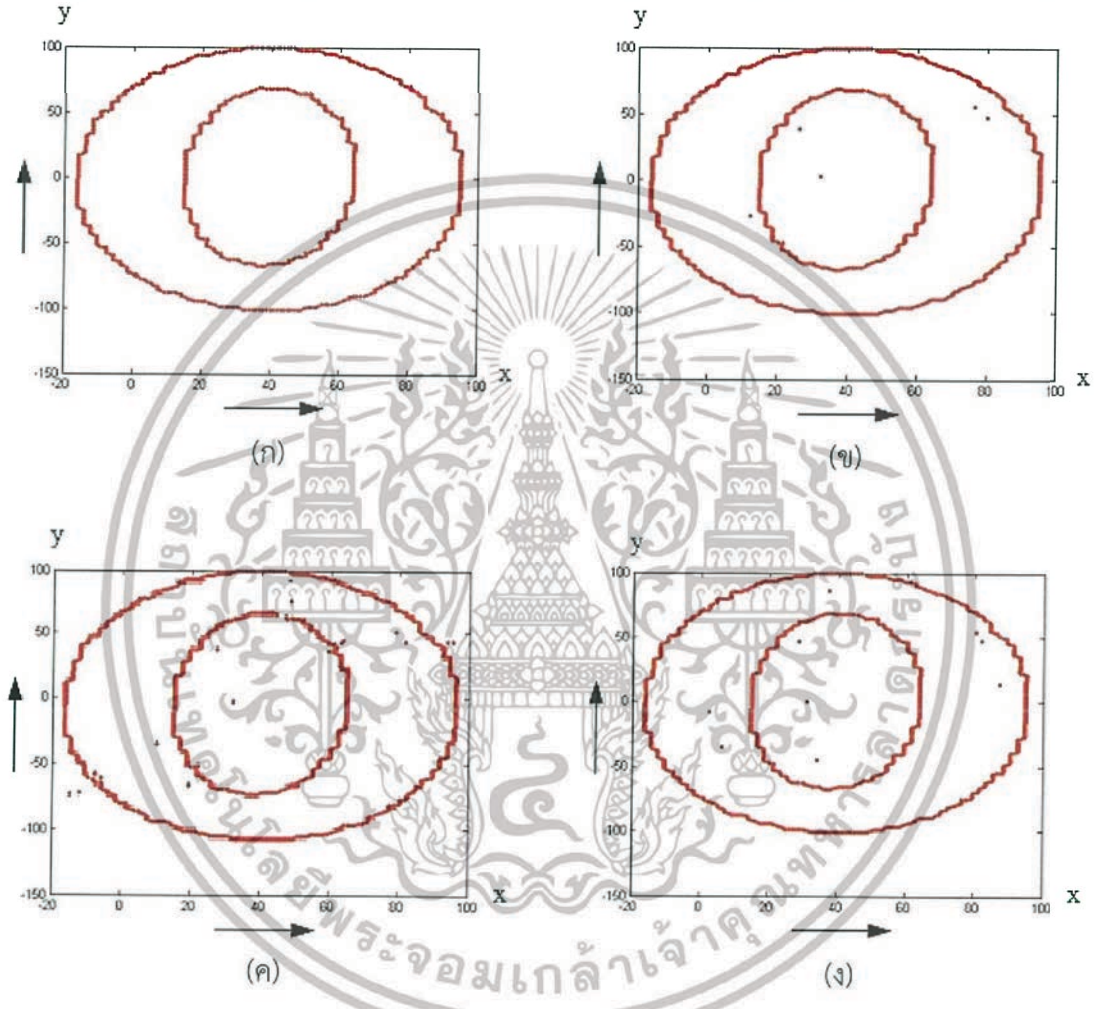
Samq แทนจำนวนจุดที่ต้องการแทรกลงไป มีรูปแบบดังเช่น Samq = 1:4:600 หมายความว่ามีจุดที่ต้องการให้ Sampling เริ่มตั้งแต่จุดที่ 1 ถึงจุดที่ 600 และให้มีการ Sampling จำนวน 600/4 = 150 จุด หรือ Samq = 1:0.5:600 ก็จะมีการ Sampling จำนวน 600/0.5 = 1200 จุด หมายความว่ามีการแทรกจุดลงไปจนครบ 1200 จุด

Option หมายถึง ตัวเลือกหรือวิธีการที่จะใช้ในการแทรกจุดพิกัด เป็นตัวเลือกหรือวิธีการที่มีให้เลือกในโปรแกรม MATLAB ในที่นี้จะยกตัวอย่างวิธีอินเตอร์โพเลชัน 4 วิธีคือ

1. วิธี 'nearest' คือ วิธีการแทรกแบบแทรกจุดข้างเคียง
2. วิธี 'linear' คือ วิธีการแทรกแบบเส้นตรง
3. วิธี 'spline' คือ วิธีการแทรกแบบเส้นโค้ง
4. วิธี 'cubic' คือ วิธีการแทรกแบบต่อเนื่องเป็นช่วง [1]

โดยจะสามารถแสดงตัวอย่างผลลัพธ์ของการอินเตอร์โพเลชันทั้ง 4 แบบได้ดังรูปที่ 2.18

จากตัวอย่างการอินเตอร์โพเลชันทั้ง 4 วิธีในรูปที่ 2.18 จะพบว่าการอินเตอร์โพเลชันด้วยวิธี Nearest จะให้ค่าผลลัพธ์ที่ดีโดยไม่มีจุดที่อยู่นอกเส้นที่ต้องการ แต่การอินเตอร์โพเลชันด้วยวิธี Linear, Spline และ Cubic จะพบจุดที่กระจายตัวหรือจุดไม่อยู่ในแนวเส้นที่ต้องการ ดังนั้นการอินเตอร์โพเลชันด้วยวิธี Nearest จะให้ผลลัพธ์ที่ดีกว่าการอินเตอร์โพเลชันด้วยวิธี Linear, Spline และ Cubic



รูปที่ 2.18 การอินเตอร์โพเลชันด้วยวิธี Nearest, Linear, Spline และ Cubic [1]
(ก) การอินเตอร์โพเลชันด้วยวิธี Nearest
(ข) การอินเตอร์โพเลชันด้วยวิธี Linear
(ค) การอินเตอร์โพเลชันด้วยวิธี Spline
(ง) การอินเตอร์โพเลชันด้วยวิธี Cubic

เมื่อแทรกจุดเรียบร้อยแล้วต่อมาจะทำการลดจำนวนจุดพิกัดลง (Down Sampling) เพื่อลดจำนวนจุดพิกัดให้มีความเหมาะสมก่อนนำไปประมวลผล ซึ่งจะช่วยลดจำนวนข้อมูลที่จะใช้ในการประมวลผลลงและทำให้ประมวลผลได้รวดเร็วขึ้น การลดจำนวนจุดพิกัดจะใช้คำสั่ง downsample เป็นคำสั่งที่อยู่ในเครื่องมือของ MATLAB เช่นดียวกัน มีรูปแบบคำสั่งดังนี้

Var_Downsamp = downsample(Var_Interp,Downsamp)

มีคำอธิบายดังนี้

Var_Interp หมายถึง ตัวแปรที่ผ่านการอินเตอร์โพเลชันมาแล้ว

Downsamp หมายถึง จำนวนจุดภาพที่ต้องการลดของตัวแปร เช่น ถ้า Downsamp เท่ากับ 2 และจำนวนจุดของตัวแปร Var_Interp มีค่า 600 ดังนั้นจำนวนตัวแปรจะถูกลดลงเหลือ 600/2 = 300 จุด

Var_Downsamp หมายถึง ตัวแปรของจำนวนจุดพิกัดที่ถูกลดลงแล้ว

2.7 ทฤษฎีการหาค่าสัมประสิทธิ์ดิสครีตโคไซน์ (Discrete Cosine Transform)

ดีสครีตโคไซน์ทรานส์ฟอร์ม (Discrete Cosine Transform : DCT) เป็นเทคนิคการบีบอัดข้อมูลหรือลดจำนวนข้อมูลลง โดยเป็นการบีบอัดข้อมูลออกมารวมกันอยู่ที่บริเวณมุมด้านซ้ายหรือช่วงต้นของการบีบอัดข้อมูล ซึ่งจะช่วยลดจำนวนข้อมูลที่ใช้ในการคำนวณหรือประมวลผลลงไปโดยยังคงไว้ซึ่งคุณสมบัติเดิมของวัตถุหรือชุดข้อมูล โดยข้อมูลที่ถูกบีบอัดหรือค่าสัมประสิทธิ์ที่คำนวณได้จากจุดพิกัดของวัตถุหรือภาพจะมีค่าที่ไม่เหมือนกัน จึงเป็นการแสดงถึงความเป็นเอกลักษณ์ของแต่ละวัตถุหรือภาพ

การหาค่าสัมประสิทธิ์ของวัตถุหรือภาพจะหาได้จากสมการดีสครีตโคไซน์ทรานสฟอร์มดังสมการที่ (2.19)

$$c_1(u) = w(u) \sum_{n=0}^{D-1} f(x) \cos \frac{(2n+1)u\pi}{2D} \qquad (2.19)$$

เมื่อ

$$w(u) = \begin{cases} \sqrt{\frac{1}{D}}, & u = 0 \\ \sqrt{\frac{2}{D}}, & 1 \le u \le D-1 \end{cases}$$

เมื่อ

u คือ ลำดับของจุดพิกัดที่นำเข้ามาคำนวณ กำหนดให้ u = 0, 1, 2, 3, ..., D-1

D คือ จำนวนสูงสุดของจุดพิกัดที่นำเข้ามาคำนวณ

n คือ ลำดับของจุดพิกัด

$f(x)$ คือ $\tilde{x}_n$ และ $\tilde{y}_n$ เป็นตำแหน่งจุดพิกัดในแนวแกน x และ y ที่ผ่านการดาวน์แซมปลิ้ง

$c_1(u)$ และ $c_2(u)$ คือ ค่าสัมประสิทธิ์โคไซน์ของแนวแกน x และ y

$w(u)$ คือ ค่าคงที่ที่เปลี่ยนแปลงตามเงื่อนไขที่กำหนด

ค่าที่ได้ออกมาคือ $c_1(u)$ และ $c_2(u)$ เป็นค่าสัมประสิทธิ์โคไซน์ในแนวแกน x และ y ตามลำดับ จากนั้นจะนำค่าสัมประสิทธิ์นี้มาวิเคราะห์ด้วยระบบผลตอบสนองอิมพัลส์ที่จำกัดจำนวน (FIR System) เพื่อหาค่าความเป็นเอกลักษณ์ของวัตถุหรือภาพหรือชุดข้อมูลนั้นๆ ดังนั้นกำหนดตัวแปรใหม่ดังสมการที่ (2.20) เพื่อนำไปเป็นอินพุตและเอาท์พุตของระบบ FIR

$$\tilde{X}_n = c_1(u) \quad \text{และ} \quad \tilde{Y}_n = c_2(u) \tag{2.20}$$

2.8 ทฤษฎีระบบผลตอบสนองอิมพัลส์ที่จำกัดจำนวน (Finite Impulse Response)

ระบบผลตอบสนองอิมพัลส์ที่จำกัดจำนวน (Finite Impulse Response System) เป็นระบบที่ข้อมูลมีความสัมพันธ์กันระหว่างค่าอินพุตและเอาท์พุต โดยมีค่าสัมประสิทธิ์ผลตอบสนองของอิมพัลส์ (δ) เป็นตัวบ่งบอกคุณลักษณะหรือเอกลักษณ์ของข้อมูล ดังแสดงในรูปที่ 2.19

รูปที่ 2.19 ความสัมพันธ์ของ FIR system

สามารถแสดงได้ดังสมการที่ (2.21)

$$\tilde{Y}_n = \sum_{m=0}^{S} \delta_m \tilde{X}_{n-m} \tag{2.21}$$

เมื่อ

$\tilde{X}_{n-m}$ คือ ค่าอินพุทระบบ FIR
δ_m คือ ค่าสัมประสิทธิ์ผลตอบสนองของอิมพัลส์ของระบบ FIR
$\tilde{Y}_n$ คือ ค่าเอาท์พุตระบบ FIR
S คือ จำนวนลำดับของระบบ FIR

เนื่องจากอินพุต $\tilde{X}_{n-m}$ ของระบบ FIR เป็นเวคเตอร์จึงไม่สามารถคูณกับเวคเตอร์เอกลักษณ์ของข้อมูล δ_m ดังนั้นการที่จะทำให้ค่าทั้งสองสามารถคูณกันได้ในระบบ FIR จึงต้องจัดเรียงค่าอินพุต $\tilde{X}_{n-m}$ ใหม่ให้อยู่ในรูปเมทริกซ์สามเหลี่ยมล่าง (Lower Triangular Matrix : T) ดังสมการที่ (2.22)

$$T = \begin{bmatrix} \tilde{X}_0 & 0 & \cdots & 0 \\ \tilde{X}_1 & \tilde{X}_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{X}_D & \tilde{X}_{D-1} & \cdots & \tilde{X}_0 \end{bmatrix} \tag{2.22}$$

จากนั้นนำมาเขียนเป็นสมการใหม่ได้เป็นสมการที่ (2.23)

$$\begin{bmatrix} \tilde{Y}_1 \\ \tilde{Y}_2 \\ \vdots \\ \tilde{Y}_D \end{bmatrix} = \begin{bmatrix} \tilde{X}_0 & 0 & \cdots & 0 \\ \tilde{X}_1 & \tilde{X}_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{X}_D & \tilde{X}_{D-1} & \cdots & \tilde{X}_0 \end{bmatrix} \begin{bmatrix} \delta_0 \\ \delta_1 \\ \vdots \\ \delta_S \end{bmatrix} \tag{2.23}$$

เมื่อ D คือจำนวนสัมประสิทธิ์ที่ผ่านการแปลง DCT และสามารถเขียนได้ดังสมการที่ (2.24)

$$T * \delta_m = \tilde{Y}_n \tag{2.24}$$

เนื่องจากทราบค่าอินพุต $\tilde{X}_{n-m}$ และเวกเตอร์ค่าเอาท์พุต $\tilde{Y}_n$ แล้ว นำทั้ง 2 ค่าดังกล่าวมาทำการหาเวกเตอร์ค่าสัมประสิทธิ์ผลตอบสนองของอิมพัลส์ δ_m ของระบบ FIR ซึ่งเป็นตัวบ่งบอกคุณลักษณะหรือเอกลักษณ์ของข้อมูล โดยสามารถค่าสัมประสิทธิ์ผลตอบสนองของอิมพัลส์ δ_m ได้จากสมการที่ (2.25)

$$\begin{aligned} (T^T * T) * \delta_m &= T^T * \tilde{Y}_n \\ \delta_m &= (T^T * T)^{-1} * T^T * \tilde{Y}_n^T \end{aligned} \tag{2.25}$$

บทที่ 3
การแยกเส้นขอบนิ้วมือและเส้นลายมือบนฝ่ามือ และการใช้ FIR System

ในบทนี้จะกล่าวถึงขั้นตอนและวิธีการแยกเส้นขอบนิ้วมือและเส้นลายมือบนฝ่ามือออกจากภาพมือที่ถูกถ่ายจากกล้องถ่ายภาพดิจิตอล จากนั้นก็จะทำการเก็บค่าจุดพิกัดของเส้นขอบนิ้วมือและเส้นลายมือบนฝ่ามือ เพื่อนำไปหาเอกลักษณ์ของเส้นขอบนิ้วมือและเส้นลายมือบนฝ่ามือโดยใช้ FIR System และใช้ในการระบุตัวบุคคล

3.1 การหาเส้นขอบมือ (Hand Contour)

3.1.1 การหาเส้นขอบมือด้วยวิธีโซเบลโดยการเขียนโปรแกรมด้วยตนเอง

การหาเส้นขอบมือจะเป็นการใช้ทฤษฎีการหาขอบภาพ (Edge Detection) ที่ได้กล่าวไว้ในบทที่ 2 โดยจะเริ่มจากการนำภาพที่ได้จัดเก็บไว้เข้าในโปรแกรม MATLAB ด้วยการใช้คำสั่ง imread ซึ่งเป็นเครื่องมือ (Tool) ของโปรแกรม MATLAB ภาพที่จะนำเข้ามาใช้ในโปรแกรมจะเป็นไฟล์ภาพนามสกุล JPEG, TIF, BMP, PNG เป็นต้น สำหรับไฟล์ภาพที่ใช้งานในปริญญานิพนธ์ฉบับนี้จะใช้ไฟล์ภาพที่เป็นนามสกุล JPEG ตัวอย่างภาพสี RGB ที่ใช้แสดงได้ในรูปที่ 3.1

รูปที่ 3.1 ตัวอย่างภาพสี RGB ที่ใช้ในโปรแกรม

เมื่อนำเข้าภาพได้จัดเก็บไว้เข้าในโปรแกรมแล้ว ต่อมาจะทำการลดขนาดรูปภาพเพื่อให้มีขนาดที่เหมาะสมการในประมวลผล เนื่องจากรูปภาพที่มีขนาดใหญ่จะใช้เวลาในการประมวลผลมาก โดยใช้คำสั่ง imresize ซึ่งเป็นเครื่องมือในโปรแกรม MATLAB ในการลดขนาดภาพ จากนั้นจะเปลี่ยนจากภาพสี RGB เป็นรูปภาพระดับสีเทา โดยใช้เครื่องมือที่มีอยู่แล้วในโปรแกรมคือ rgb2gray หลักการคือการผสมความเข้มของสีแดง สีเขียว และสีน้ำเงินเข้าด้วยกัน สามารถแสดงการคำนวณได้ดังสมการที่ (3.1)

$$\text{Gray Scale} = (\text{Red} \times 0.299) + (\text{Green} \times 0.587) + (\text{Blue} \times 0.114) \qquad (3.1)$$

เมื่อ

Gray Scale คือ ค่าระดับสีเทาในแต่ละจุดภาพของภาพสี RGB
Red คือ ค่าความเข้มของสีแดง
Green คือ ค่าความเข้มของสีเขียว
Blue คือ ค่าความเข้มของสีน้ำเงิน

สามารถแสดงการใช้คำสั่ง imread, imresize และ rgb2gray ได้ดังนี้

```
% เก่าเข้าภาพจากโฟลเดอร์ที่จัดเก็บไฟล์ไว้
Image_Input = imread('D:\ที่อยู่ของไฟล์ภาพ\ชื่อภาพ.jpg');

% ลดขนาดรูปภาพลง ในที่นี้คือลดขนาดเหลือร้อยละ 45
Image_Resize = imresize(Image_Input,0.45);

% เปลี่ยนจากภาพสี RGB เป็นภาพระดับสีเทา
Image_Gray = rgb2gray(Image_Resize);
```

สาเหตุที่ต้องเปลี่ยนเป็นภาพระดับสีเทาเนื่องจากการหาขอบภาพจะทำได้กับภาพที่เป็นระดับสีเทา หรือภาพสองระดับเท่านั้น โดยรูปภาพระดับสีเทาสามารถแสดงได้ในรูปที่ 3.2

รูปที่ 3.2 ภาพระดับสีเทา

นำรูปภาพระดับสีเทามาหาขอบภาพ เริ่มจากการนำหน้าต่างโซเบลในแนวตั้งและแนวนอนของรูปที่ 3.3 (ก) และ (ข) มาทำการคอนโวลูชั่น (Convolution) กับรูปภาพระดับสีเทา

-1	-2	-1
0	0	0
1	2	1

(ก)

-1	0	1
-2	0	2
-1	0	1

(ข)

รูปที่ 3.3 หน้าต่างโซเบลสำหรับหาขอบภาพในแนวนอนและแนวตั้ง
(ก) หน้าต่างสำหรับหาขอบภาพในแนวนอน
(ข) หน้าต่างสำหรับหาขอบภาพในแนวตั้ง

การคอนโวลูชั่นจะแยกกันในแนวแกนนอนและแนวแกนตั้ง ดังแสดงในสมการที่ (3.2) และ (3.3) เมื่อ $I(x,y)$ คือรูปภาพระดับสีเทา

$$G_x(x,y) = I(x,y) * Sobel\ Mask\ X\ direction \tag{3.2}$$

$$G_y(x,y) = I(x,y) * Sobel\ Mask\ Y\ direction \tag{3.3}$$

เมื่อ "$*$" คือการคอนโวลูชั่น ได้ผลออกมาเป็น $G_x(x,y)$ และ $G_y(x,y)$ คือภาพที่ผ่านการคอนโวลูชั่นกับหน้าต่างโซเบลในแนวแกนนอนและแกนตั้งตามลำดับ

```
% การคอนโวลูชั่นภาพกับหน้าต่างโซเบลในแนวแกนนอนและแนวแกนตั้ง
K = Image_Gray; % กำหนดตัวแปรใหม่
[m,n] = size(K); % หาขนาดรูปภาพในทิศทางแกน x และ y
for x = 2: m-1 % สแกนภาพจากซ้ายไปขวา
    for y = 2: n-1 % สแกนภาพจากบนลงล่าง
        % นำภาพคอนโวลูชั่นกับหน้าต่างโซเบลในแนวแกน x
        Gx(x,y) = (-1)*K(x-1,y-1) + (-2)*K(x-1,y) + (-1)*K(x-1,y+1) +
                  (1)*K(x+1,y-1)  + (2)*K(x+1,y)  + (1)*K(x+1,y+1);
        % นำภาพคอนโวลูชั่นกับหน้าต่างโซเบลในแนวแกน y
        Gy(x,y) = (1)*K(x-1,y-1)  + (2)*K(x,y-1)  + (1)*K(x+1,y-1)  +
                  (-1)*K(x-1,y+1) + (-2)*K(x,y+1) + (-1)*K(x+1,y+1);
    end
end
```

ค่าของผลลัพธ์ที่ได้ออกมาจะมีค่าที่ติดลบและค่าที่เป็นบวก แต่โปรแกรมจะแสดงผลได้ในช่วง 0 ถึง 255 ดังนั้นต้องทำการสเกล (Scale) ค่าใหม่โดยใช้สมการที่ (3.4) และ (3.5) เพื่อให้ค่าอยู่ในช่วงที่โปรแกรมแสดงผลได้

$$G_1(x,y) = \frac{G_x(x,y) - Min}{Max - Min} \times 255 \tag{3.4}$$

$$G_2(x,y) = \frac{G_y(x,y)-Min}{Max-Min} \times 255 \quad (3.5)$$

```
% สเกล Gx และ Gy ให้มีค่าตั้งแต่ 0 ถึง 255
Max1 = max(max(Gx)); % หาค่าสูงสุดของ Gx
Min1 = min(min(Gx)); % หาค่าต่ำสุดของ Gx
Max2 = max(max(Gy)); % หาค่าสูงสุดของ Gy
Min2 = min(min(Gy)); % หาค่าต่ำสุดของ Gy
for x = 1: m
    for y = 1: n
            G1(x,y) = (((Gx(x,y)-Min1)/(Max1-Min1))); % สเกลค่าของ Gx ได้ G1
            G2(x,y) = (((Gy(x,y)-Min2)/(Max2-Min2))); % สเกลค่าของ Gy ได้ G2
    end
end
```

เมื่อ $G_1(x,y)$ และ $G_2(x,y)$ คือภาพที่ผ่านการสเกลลิ่ง (Scaling) ของภาพ $G_x(x,y)$ และ $G_y(x,y)$ ตามลำดับ จะได้ผลออกมาดังรูปที่ 3.4 เป็นภาพที่ได้จากการคอนโวลูชั่นในแนวนอนและแนวตั้งที่ผ่านการสเกลลิ่งแล้ว

(ก) (ข)

รูปที่ 3.4 ภาพที่ได้จากการคอนโวลูชั่นในแนวนอนและแนวตั้ง
(ก) ภาพที่ได้จากการคอนโวลูชั่นในแนวนอน
(ข) ภาพที่ได้จากการคอนโวลูชั่นในแนวตั้ง

ภาพที่ได้จากการคอนโวลูชั่นจะแยกกันในแนวแกนนอนและแกนตั้ง ดังนั้นจึงต้องรวมภาพจากทั้งสองแกนเข้าด้วยกันทางเวคเตอร์ สามารถคำนวณขนาดของเกรเดี้ยน (Gradient) ได้ดังสมการที่ (3.6) หรือ (3.7)

$$\nabla f = mag(\nabla \boldsymbol{f}) = \left[G_x^2 + G_y^2\right]^{1/2} \quad (3.6)$$

$$\nabla f \approx |G_x| + |G_y| \quad (3.7)$$

```
% รวม G1 และ G2 เข้าด้วยกัน ได้ภาพระดับสีเทาที่รวมภาพจากแกน x และ y ไว้ด้วยกัน
Image_Gray_Combine = sqrt((G1.^2) + (G2.^2));
```

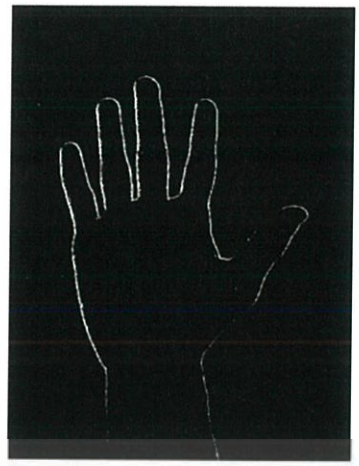
แต่หลังจากถอดรากที่ 2 ออกมาแล้ว จะมีค่าที่เกิน 1 ออกมาด้วย จึงต้องสเกลค่าใหม่อีกครั้งให้มีเพียง 0 ถึง 1 สามารถแสดงผลของการรวมภาพจากทั้งสองแกนเข้าด้วยกันทางเวคเตอร์และผ่านการสเกลลิ่งแล้วได้ดังรูปที่ 3.5

รูปที่ 3.5 ภาพระดับสีเทาที่รวมภาพจากทั้งสองแกนเข้าด้วยกันทางเวคเตอร์

จากนั้นนำภาพที่ผ่านการสเกลค่าและแสดงผลได้ถูกต้องแล้วมาทำให้เป็นภาพสองระดับโดยการเลือกค่าเทรชโฮล (Threshold Value) ซึ่งจะใช้คำสั่ง graythresh ในโปรแกรม MATLAB มาช่วยในการเลือกค่าเทรชโฮลออกมาโดยอัตโนมัติ แต่ค่าที่ได้อาจจะยังไม่เหมาะสมกล่าวคือค่าเทรชโฮลอาจจะมีค่าที่สูงหรือต่ำเกินไป ทำให้ภาพเส้นขอบไม่มีความชัดเจนและไม่สามารถแยกความแตกต่างระหว่างเส้นขอบมือกับส่วนอื่นๆได้ โดยภาพเส้นขอบที่ต้องการเมื่อตัดระดับค่าเทรชโฮลออกมาจะต้องมีสีขาวหรือมีค่า 255 และภาพส่วนอื่นๆจะต้องมีสีดำหรือมีค่า 0 ทำให้ต้องมีการปรับค่าเพิ่มเติมด้วยการเพิ่มหรือลดค่าเทรชโฮลจากค่าเทรชโฮลที่โปรแกรมคำนวณออกมาอีกครั้ง จากนั้นใช้คำสั่ง im2bw เพื่อปรับให้ภาพระดับสีเทาเป็นภาพสองระดับโดยใช้ค่าเทรชโฮลที่ได้ปรับค่าแล้ว สามารถแสดงการเขียนโปรแกรมได้ดังนี้

```
% คำสั่งหาค่าเทรชโฮลโดยอัตโนมัติ
Threshold_Value = graythresh(Image_Gray_Combine);

% คำสั่งเปลี่ยนจากภาพระดับสีเทาไปเป็นภาพสองระดับ
Image_Binary = im2bw(Image_Gray_Combine, Threshold_Value-0.2);
% "-0.2" คือ ลดค่าเทรชโฮลให้มีความเหมาะสม
```

ภาพสองระดับที่แสดงภาพเส้นขอบมือสามารถแสดงได้ดังรูปที่ 3.6



รูปที่ 3.6 ภาพเส้นขอบมือ

ผลลัพธ์ที่ได้จากการเขียนอัลกิริทึมตามวิธีการหาขอบภาพของโซเบลในรูปที่ 3.6 จะเห็นว่าบริเวณยอดนิ้วและบริเวณนิ้วหัวแม่มือจะมีรูปร่างที่ไม่สมบูรณ์ เส้นขอบนิ้วไม่ปรากฏ ดังนั้นจึงจะเสนอวิธีหาขอบภาพด้วยวิธีโซเบลโดยการใช้เครื่องมือที่มีอยู่ในโปรแกรม MATLAB

3.1.2 การหาเส้นขอบนิ้วมือโดยใช้คำสั่ง edge ของโปรแกรม MATLAB

การหาเส้นขอบมือโดยใช้คำสั่ง edge ซึ่งเป็นเครื่องมือที่มีอยู่แล้วใน MATLAB จะใช้วิธีการเดียวกับการหาเส้นขอบนิ้วมือด้วยโปรแกรมที่เขียนขึ้นมาเอง โดยเริ่มจากการนำภาพเข้าในโปรแกรม MATLAB จากนั้นลดขนาดรูปภาพให้มีความเหมาะสม แล้วเปลี่ยนจากรูปภาพสี RGB ให้กลายเป็นภาพระดับสีเทา จากนั้นใช้คำสั่ง graythresh เพื่อหาค่าเทรชโฮลโดยอัตโนมัติ และปรับแต่งด้วยการเพิ่มหรือลดค่าเทรชโฮลให้มีความเหมาะสม จากนั้นใช้คำสั่ง im2bw เพื่อปรับให้ภาพระดับสีเทา เป็นภาพสองระดับโดยใช้ค่าเทรชโฮลที่ได้ปรับแต่งแล้ว จากนั้นใช้คำสั่ง edge วิธี Sobel ในการหาเส้นขอบภาพ โดยสามารถแสดงโปรแกรมได้ดังนี้

```
% เข้าเข้าภาพจากโฟลเดอร์ที่จัดเก็บไฟล์ไว้
Image_Input = imread('D:\ที่อยู่ของไฟล์ภาพ\ชื่อภาพ.jpg');

% ลดขนาดรูปภาพลง ในที่นี้คือลดขนาดเหลือร้อยละ 45
Image_Resize = imresize(Image_Input,0.45);

% เปลี่ยนจากภาพสี RGB เป็นภาพระดับสีเทา
Image_Gray = rgb2gray(Image_Resize);

% คำสั่งหาค่าเทรชโฮลโดยอัตโนมัติ
Threshold_Value = graythresh(Image_Gray);

% คำสั่งเปลี่ยนจากภาพระดับสีเทาไปเป็นภาพสองระดับ
Image_Binary = im2bw(Image_Gray, Threshold_Value-0.2);
% "-0.2" คือ ลดค่าเทรชโฮลให้มีความเหมาะสม
```

```
% การหาขอบภาพด้วยวิธีโซเบล
Image_Edge_Sobel = edge(Image_Binary,'sobel');
```

ภาพฝ่ามือที่เป็นภาพสองระดับแสดงผลได้ดังรูปที่ 3.7 และแสดงภาพเส้นขอบมือที่หาได้จากคำสั่ง edge วิธี Sobel ได้ดังรูปที่ 3.8

รูปที่ 3.7 ภาพฝ่ามือที่เป็นภาพสองระดับ

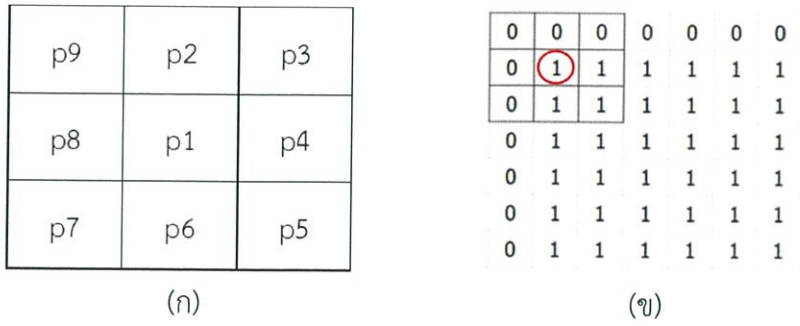
รูปที่ 3.8 ภาพเส้นขอบนิ้วมือ

เมื่อนำผลลัพธ์ในรูปที่ 3.6 ซึ่งเป็นเส้นขอบนิ้วมือที่หาได้จากโปรแกรมที่เขียนขึ้นมาตามอัลกอริทึมของการหาขอบภาพด้วยวิธีโซเบล เปรียบเทียบกับเส้นขอบนิ้วมือที่หาได้จากคำสั่ง edge ใน MATLAB ในรูปที่ 3.8 พบว่ารูปที่ 3.8 สามารถหาเส้นขอบภาพได้สมบูรณ์ ชัดเจน และครบถ้วนกว่า เช่น บริเวณยอดนิ้วมือของแต่ละนิ้ว หรือเส้นขอบของนิ้วหัวแม่มือ ดังนั้นในปริญญานิพนธ์ฉบับนี้จะใช้คำสั่ง edge ซึ่งเป็นเครื่องมือในโปรแกรม MATLAB ในการหาเส้นขอบนิ้วมือ

3.2 การทำเส้นขอบนิ้วให้บาง

การทำเส้นขอบนิ้วให้บาง คือการทำให้เส้นขอบนิ้วมือที่มีความหนาเหลือความหนาเพียง 1 พิกเซล เนื่องจากเส้นขอบที่หามาได้อาจจะยังมีความหนาอยู่ การทำให้เส้นขอบนิ้วมีความหนา 1 พิกเซล จะทำให้สามารถหาค่าตำแหน่งพิกัด (Coordinate) ได้ถูกต้องและสมบูรณ์ หากเส้นขอบนิ้วที่นำไปหาตำแหน่งพิกัดยังมีความหนาอยู่จะทำให้การเก็บค่าไม่เป็นระเบียบและบิดเบี้ยวได้

การทำเส้นขอบนิ้วให้บางจะเริ่มจากการหาพิกเซลที่มีค่า 1 โดยการสแกนภาพจากซ้ายไปขวาและจากบนลงล่าง เมื่อพบพิกเซลที่มีค่า 1 แล้ว จากนั้นนำหน้าต่างขนาด 3 x 3 ในรูปที่ 3.9 (ก) ไปวางทาบโดยให้จุด p1 อยู่บนพิกเซลที่มีค่า 1 ดังรูปที่ 3.9 (ข)



รูปที่ 3.9 การวางหน้าต่างขนาด 3 x 3 ลงบนภาพ
(ก) หน้าต่างขนาด 3 x 3
(ข) วางหน้าต่าง 3 x 3 ลงบนจุดภาพแรกที่ตรวจพบ

จากนั้นตรวจสอบเงื่อนไขในขั้นตอนที่หนึ่งซึ่งเป็นการลบจุดภาพทางด้านขวามือ ด้านล่าง และมุมซ้ายบน มีเงื่อนไขในการตรวจสอบดังนี้

a) $2 \le N(p1) \le 6$
b) $S(p1) = 1$
c) $p2 \cdot p4 \cdot p6 = 0$
d) $p4 \cdot p6 \cdot p8 = 0$

เมื่อ

N(p1) คือ ผลรวมของจุดภาพทั้งหมดที่มีค่า 1 ที่อยู่ล้อมรอบจุด p1
S(p1) คือ จำนวนครั้งที่มีการเปลี่ยนแปลงค่าจาก 0 ไปเป็นค่า 1 รอบจุด p1
" • " คือ เครื่องหมาย AND

หากจุดภาพใดเข้าเงื่อนไขทั้งหมดให้ทำการเก็บค่าตำแหน่งจุดภาพที่ผ่านการตรวจสอบไว้ และดำเนินการตรวจสอบจุดภาพทุกจุดจนหมดทั้งภาพ แล้วจึงลบจุดภาพทั้งหมด สามารถแสดงการทำงานของโปรแกรมได้ดังนี้

```
% การทำขอบภาพให้บาง
A = Image_Edge_Sobel; % กำหนดตัวแปรใหม่
% กำหนดจำนวนครั้งที่จะตรวจสอบเพื่อทำให้เส้นขอบภาพบางลง ในที่นี้กำหนด 10 ครั้ง
 for c = 1:10
% การทำเส้นขอบภาพให้บางขั้นตอนที่ 1
    % หาจุดภาพแรกที่มีค่า 1 หรือคือ A(x,y) == 1
    for x = 2: m-1 % สแกนภาพจากบนลงล่าง
         for y = 2: n-1 % สแกนภาพจากซ้ายไปขวา
            if A(x,y) == 1
                p9 = A(x-1,y-1);   p2 = A(x-1,y);   p3 = A(x-1,y+1);
                p8 = A(x,y-1);     p1 = A(x,y);     p4 = A(x,y+1);
                p7 = A(x+1,y-1);   p6 = A(x+1,y);   p5 = A(x+1,y+1);
                % เงื่อนไขข้อ a) 2 <= N <= 6
                N = p2 + p3 + p4  + p5 + p6 + p7 + p8 + p9;
```

```
                % เงื่อนไขข้อ b) 0->1 around p1
                S = 0;
                if p9 == 0 && p2 ~= 0
                   S = S + 1;
                end
                if p2 == 0 && p3 ~= 0
                   S = S + 1;
                end
                if p3 == 0 && p4 ~= 0
                   S = S + 1;
                end
                if p4 == 0 && p5 ~= 0
                   S = S + 1;
                end
                if p5 == 0 && p6 ~= 0
                   S = S + 1;
                end
                if p6 == 0 && p7 ~= 0
                   S = S + 1;
                end
                if p7 == 0 && p8 ~= 0
                   S = S + 1;
                end
                if p8 == 0 && p9 ~= 0
                   S = S + 1;
                end
                % ตรวจสอบเงื่อนไขทั้งหมด a) b) c) และ d)
                % เงื่อนไข a) 2 <= N <= 6
                % เงื่อนไข b) 0->1 around p1
                % เงื่อนไข c) p2 & p4 & p6 = 0
                % เงื่อนไข d) p4 & p6 & p8 = 0
                if (N >= 2) && (N <= 6) && (S == 1) && (p2*p4*p6 == 0) &&
                   (p4*p6*p8 == 0)
                   % กำเข้าเงื่อนไขให้จัดเก็บจุดภาพที่ผ่านการตรวจสอบไว้
                   B(x,y) = 10; % จัดเก็บจุดภาพที่เข้าเงื่อนไขโดยใช้ตัวแปร B
                else                % และกำหนดให้มีค่า 10
                   B(x,y) = 0;
                end
             end
          end
       end
% เมื่อตรวจสอบเงื่อนไขครบทุกจุดในภาพเรียบร้อยแล้ว จะลบจุดภาพที่จัดเก็บไว้ (จุดที่มีค่า 10)
[m_B,n_B] = size(B);
for x = 1: m_B
    for y = 1: n_B
        if B(x,y) == 10
           A(x,y) = 0;
        end
    end
end
```

จากนั้นตรวจสอบเงื่อนไขที่สอง จะเป็นการลบจุดภาพทางด้านซ้ายมือ ด้านบน และมุมขวาล่าง มีเงื่อนไขดังนี้

a) $2 \le N(p1) \le 6$

b) $S(p1) = 1$

c*) $p2 \cdot p4 \cdot p8 = 0$

d*) $p2 \cdot p6 \cdot p8 = 0$

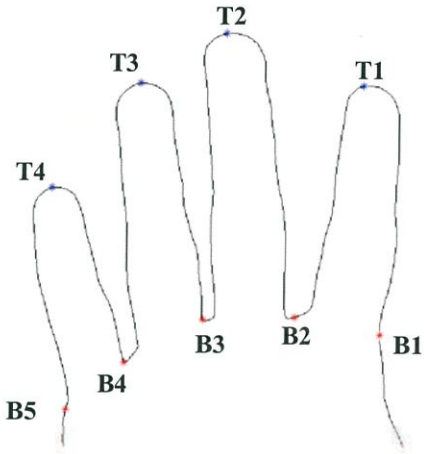
หากจุดภาพใดเข้าเงื่อนไขทั้งหมดให้ทำการเก็บค่าตำแหน่งจุดภาพที่ผ่านการตรวจสอบไว้ และดำเนินการตรวจสอบจุดภาพทุกจุดจนหมดทั้งภาพ แล้วจึงลบจุดภาพทั้งหมดเช่นเดียวกัน

จากนั้นตรวจสอบภาพของเส้นขอบนิ้วมือโดยใช้ขั้นตอนที่หนึ่ง และขั้นตอนที่สองวนซ้ำไปเรื่อยๆ จนกว่าจะเหลือเพียงเส้นขอบนิ้วมือที่มีความหนา 1 พิกเซล แล้วนำไปตรวจหาจุดพิกัดต่อไป สำหรับตัวอย่างภาพเส้นขอบที่ผ่านการทำให้บางเหลือ 1 พิกเซลสามารถแสดงได้ในรูปที่ 3.10

```
% สำหรับเงื่อนไขในขั้นตอนที่สอง การเขียนโปรแกรมจะเหมือนกับเงื่อนไขในขั้นตอนแรก และอยู่ในลูป
% เดียวกับขั้นตอนที่หนึ่งคือลูป for c = 1:10 แตจะตางกัน ในเงื่อนไขขอ c*) และ d*)
% ที่จะต้องเปลี่ยนตามเงื่อนไขที่กำหนด เป็นการตรวจสอบตามเงื่อนไขนี้
  % ตรวจสอบเงื่อนไขทั้งหมด a) b) c) และ d)
  % เงื่อนไข a) 2 <= N <= 6
  % เงื่อนไข b) 0->1 around p1
  % เงื่อนไข c*) p2 & p4 & p8 = 0
  % เงื่อนไข d*) p2 & p6 & p8 = 0
  if (N >= 2) && (N <= 6) && (S == 1) && (p2*p4*p8 == 0) && (p2*p6*p8 == 0)
%  เมื่อทำเส้นขอบภาพให้บางเหลือเพียง 1 พิกเซลได้แล้ว จะเปลี่ยนเชื่อตัวแปรใหม่ดังนี้
Image_Edge_Skeleton = A ; % เป็นภาพเส้นขอบที่ผ่านการทำให้บางเหลือ 1 พิกเซลแล้ว
```

3.3 การหาตำแหน่งพิกัดเพื่อแยกนิ้วมือออกจากภาพมือ

การหาตำแหน่งพิกัดเป็นการหาพิกัดตามแนวแกน x และ y ของเส้นขอบมือ โดยจะต้องเป็นรูปภาพที่ผ่านการหาขอบภาพและการทำให้บางเหลือความหนา 1 พิกเซล แล้วดังในรูปที่ 3.10 โดยการเก็บตำแหน่งพิกัดจะเก็บเฉพาะนิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อยเท่านั้น ส่วนนิ้วหัวแม่มือนั้นจะไม่ทำการเก็บค่าเพราะด้วยลักษณะของนิ้วจะทำให้ถ่ายรูปออกมาได้เพียงบางส่วนของนิ้ว ทำให้ไม่สามารถระบุลักษณะขอบของนิ้วหัวแม่มือได้อย่างถูกต้อง ในการเก็บตำแหน่งพิกัดจะกำหนดจุดสูงสุดหรือจุดยอดของนิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อย เป็น T1, T2, T3 และ T4 ตามลำดับ และกำหนดจุดต่ำสุดของนิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อย เป็น B1, B2, B3, B4 และ B5 ได้ดังรูปที่ 3.10



รูปที่ 3.10 ตำแหน่งพิกัดนิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อย

การหาตำแหน่งพิกัดเริ่มต้นจะทำการหาจุดพิกเซลที่มีค่าเป็น 1 โดยการตรวจสอบจุดภาพจากซ้ายไปขวาและจากบนลงล่าง เมื่อพบแล้วจึงนำหน้าต่างขนาด 3 x 3 ในรูปที่ 3.9 (ก) ไปวางทาบโดยให้จุด p1 อยู่บนพิกเซลที่มีค่าเป็น 1 ที่ตรวจพบ

เนื่องจากนิ้วกลางจะเป็นนิ้วที่ยาวที่สุด ดังนั้นจุดพิกเซลที่มีค่าเป็น 1 ตำแหน่งแรกที่ตรวจพบคือจุดสูงสุดหรือจุดยอดของนิ้วกลางให้เป็นจุด T2 แล้วเก็บจุดพิกัดที่ตำแหน่ง T2 โดยแยกเก็บพิกัดในแนวแกน x และ y จากนั้นเริ่มเก็บพิกัดทางด้านขวาของนิ้วกลางด้วยการตรวจหาจุดพิกเซลที่มีค่าเป็น 1 ที่อยู่รอบตำแหน่ง p1 ที่ตำแหน่ง p3, p4, p5, p6 และ p7 ตามลำดับ เนื่องจากการตรวจสอบตำแหน่งดังกล่าวคือการตรวจสอบบริเวณด้านขวาของจุด T2 สาเหตุที่ต้องเก็บพิกัดจุด p3 ซึ่งเป็นจุดที่อยู่บนขวาของจุด p1 เนื่องจากในบางครั้งตำแหน่งพิกัดบริเวณยอดนิ้วจะไม่เรียบเสมอกัน ดังตัวอย่างในรูปที่ 3.11 แสดงรูปจุดสูงสุดของนิ้วก้อย ซึ่งจะมีจุดสูงสุดอยู่ 2 จุดตามที่ลูกศรชี้ไว้ จึงต้องตรวจสอบจุดที่อยู่เหนือกว่าจุด p1 เพื่อให้สามารถเก็บจุดพิกัดต่อไปได้

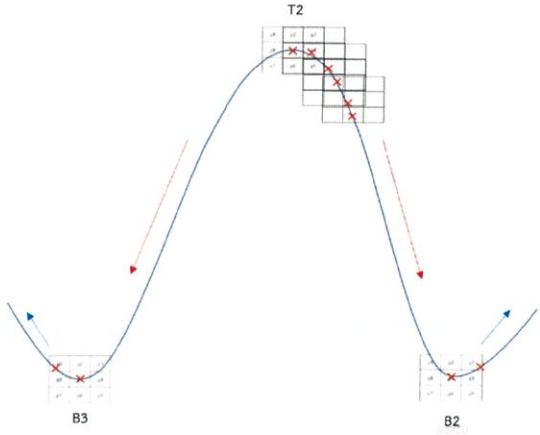
รูปที่ 3.11 จุดสูงสุดของนิ้วก้อย ซึ่งจะมีจุดสูงสุดอยู่ 2 จุด

และการที่ต้องเก็บจุด p7 แม้ว่าจุดดังกล่าวจะเป็นจุดที่อยู่ทางด้านซ้ายล่างของจุด p1 เพราะในบางครั้งลักษณะการวางมือจะมีการเอียง ทำให้จุดพิกัดที่ต้องการเก็บค่อนไปทางด้านซ้ายแม้จะเป็น

การเก็บพิกัดทางด้านขวาของนิ้วก็ตาม การตรวจสอบตำแหน่งพิกัดจะเรียงตามจุดดังกล่าว โดยถ้าหากตำแหน่งใดมีค่าเป็น 1 ให้เคลื่อนจุด p1 ของหน้าต่าง 3 x 3 ในรูปที่ 3.9 (ก) ไปที่ตำแหน่งนั้นแล้วเก็บค่าตำแหน่งพิกัดนั้นไว้ แล้วดำเนินการตรวจสอบด้วยวิธีการดังกล่าววนซ้ำจนกว่าจะลงมาจนถึงจุดต่ำสุดทางด้านขวาของนิ้วกลางคือจุด B2 เป็นการเก็บพิกัดด้านขวาของนิ้วมือ เราจะทราบได้ว่าจุด B2 เป็นจุดต่ำสุดเมื่อจุดพิกัดอยู่ที่ตำแหน่ง p9, p2 หรือ p3 คือทิศทางของจุดภาพเริ่มเป็นขาขึ้น ดังรูปที่ 3.12 แสดงการเขียนโปรแกรมหาจุดพิกัดจากตำแหน่ง T2 ไปตำแหน่ง B2 ได้ดังนี้

```
% โปรแกรมจัดเก็บจุดพิกัดน้ำกลาง
false_detect_1st_pixel = 10; % กรณีเจอจุดพิกัดที่ไม่ใช่เส้นขอบน้ำมือให้เริ่มหาค่า 1 ใหม่
                             % กำหนดให้ทำซ้ำเพียง 10 ครั้ง เนื่องจากภาพเส้นขอบน้ำมือ
                             % มีจุดพิกัดที่ไม่ใช่เส้นขอบน้ำมือเพียงเล็กน้อย
false_detect_coordinate = 100; % จำนวนจุดพิกัดขั้นต่ำของแต่ละน้ำมือ หากน้อยกว่า
                               % ที่กำหนดแสดงว่าจุดพิกัดนั้นไม่ใช่เส้นขอบน้ำมือ
% กรณีเจอจุดพิกัดที่ไม่ใช่เส้นขอบน้ำมือให้เริ่มหาค่า 1 ใหม่
for find = 1 : false_detect_1st_pixel
    END = 1;  % กำหนดค่าเริ่มต้นเพื่อใช้ในการตรวจสอบว่าได้จัดเก็บจุดพิกัดน้ำได้ถูกต้อง
%   หาจุดพิกเซลที่มีค่า 1 จุดแรกโดยการสแกนภาพจากซ้ายไปขวาและจากบนลงล่าง
    for x = 2: m-1 % สแกนภาพในทิศทางบนลงล่าง
        for y = 2: n-1 % สแกนภาพในทิศทางซ้ายไปขวา
            if Image_Edge_Skeleton(x,y) == 1 % ถ้าเจอพิกเซลที่มีค่า 1
               break % พบจุดพิกัดที่มีค่า 1 แล้วออกจากลูป y = 2: n-1
            end
        end
        if Image_Edge_Skeleton(x,y) == 1 % ถ้าเจอพิกเซลที่มีค่า 1
           break % พบจุดพิกัดที่มีค่า 1 แล้วออกจากลูป x = 2: m-1
        end       % ซึ่งคือการออกจากลูปเพราะจุดพิกัดที่มีค่า 1 จุดแรก
    end
%   โปรแกรมตรวจสอบจุดพิกัดน้ำกลาง T2 ไป B2
    T2(x,y) = Image_Edge_Skeleton(x,y); % จุดแรกที่พบจะเป็นจุด T2, จัดเก็บตำแหน่ง T2
                                        % ในรูปของเมตริกซ์ที่มีขนาด x คูณ y
    a = 1;  % เริ่มเก็บจุดพิกัดจุดแรก
    p3 = 0; % กำหนดค่าตอนเริ่มเก็บจำนวนจุด p3 ที่ตรวจพบ
    % จำนวนจุดพิกัดของน้ำมือจะมีจำนวนสูงสุดไม่เกิน 1000 จุด จึงให้ทำซ้ำจำนวน 1000 ครั้ง
    for c = 1 : 1000
        C(x,y) = Image_Edge_Skeleton(x,y); % ใช้ตัวแปร C ในการจัดเก็บจุดพิกัดน้ำกลาง
        Image_Edge_Skeleton(x,y) = 10; % ลบจุดพิกัดน้ำกลางที่จัดเก็บไปแล้ว
                                       % โดยเปลี่ยนค่าจาก 1 เป็น 10
        xET2(a) = x; % จัดเก็บตำแหน่งจุดพิกัด x ที่อยู่ด้านขวาของน้ำกลาง
        yET2(a) = y; % จัดเก็บตำแหน่งจุดพิกัด y ที่อยู่ด้านขวาของน้ำกลาง
        a = a+1; % เพิ่ม a ทีละ 1 เมื่อจัดเก็บจุดพิกัดแล้ว
        if Image_Edge_Skeleton(x-1,y+1) == 1 % ตรวจสอบตำแหน่ง p3
           % เนื่องจากจุด p3 เป็นจุดที่อยู่ในทิศทางขวาบนซึ่งจะใช้ในการหยุดการจัดเก็บจุดพิกัด
           % แต่จำเป็นต้องใช้ในการจัดเก็บจุดพิกัดด้วยเช่นกัน จึงต้องกำหนดให้มีการจัดเก็บจุดพิกัด
           % อย่างน้อยหนึ่งครั้ง และเมื่อพบ p3 ให้นับ p3 เพิ่มขึ้น 1 ครั้ง
           p3 = p3 + 1;
           if p3 > 1 % สิ้นสุดการจัดเก็บเมื่อเจอ p3 เกินหนึ่งครั้ง
```

```
            % ถ้าจำนวนจุดพิกัดน้อยกว่าที่กำหนด แสดงว่าไม่ใช่จุดพิกัดน้ำกลาง
            if a < false_detect_coordinate
               clear C xET2 yET2 x y T2 % ลบค่าตัวแปรที่ใช้ในการจัดเก็บจุดพิกัด
                                        % ป้องกันข้อมูลซ้ำซ้อนเมื่อจัดเก็บจุดพิกัดใหม่
               break % ออกจากลูปจัดเก็บจุดพิกัดแล้วเริ่มหาจุดที่มีค่า 1 ใหม่
            end
            a = a-1; % ลดค่า a ลง 1 ในกรณีสิ้นสุดการจัดเก็บ เพราะเพิ่มค่าเกินจำเป็น
            B2(x,y) = C(x,y); % จัดเก็บตำแหน่ง B2 ในรูปเมตริกซ์เช่นเดียวกับ T2
            disp(' END collected Middle finger T2->B2 ')
            END = 1000; % ใช้ในการตรวจสอบว่าได้จัดเก็บจุดพิกัดน้ำได้ถูกต้อง
            break  % ออกจากลูปจัดเก็บจุดพิกัด
         end
            x = x-1;
            y = y+1;
      elseif Image_Edge_Skeleton(x,y+1) == 1 % ตรวจสอบตำแหน่ง p4
            x = x;
            y = y+1;
      elseif Image_Edge_Skeleton(x+1,y+1) == 1 % ตรวจสอบตำแหน่ง p5
            x = x+1;
            y = y+1;
      elseif Image_Edge_Skeleton(x+1,y) == 1 % ตรวจสอบตำแหน่ง p6
            x = x+1;
            y = y;
      elseif Image_Edge_Skeleton(x+1,y-1) == 1 % ตรวจสอบตำแหน่ง p7
            x = x+1;
            y = y-1;
      % สิ้นสุดการจัดเก็บตำแหน่งพิกัดเมื่อมีทิศทางขึ้น  ตรวจสอบตำแหน่ง p2 และ p9
      elseif (Image_Edge_Skeleton(x-1,y) == 1)    || % ตรวจสอบตำแหน่ง p2
            (Image_Edge_Skeleton(x-1,y-1) == 1)     % ตรวจสอบตำแหน่ง p9
            % ถ้าจำนวนจุดพิกัดน้อยกว่าที่กำหนด แสดงว่าไม่ใช่จุดพิกัดน้ำกลาง
            if a < false_detect_coordinate
               clear C xET2 yET2 x y T2 % ลบค่าตัวแปรที่ใช้ในการจัดเก็บจุดพิกัด
                                        % ป้องกันข้อมูลซ้ำซ้อนเมื่อจัดเก็บจุดพิกัดใหม่
               break % ออกจากลูปจัดเก็บจุดพิกัดแล้วเริ่มหาจุดที่มีค่า 1 ใหม่
            end
            a = a-1; % ลดค่า a ลง 1 ในกรณีสิ้นสุดการจัดเก็บ เพราะเพิ่มค่าเกินจำเป็น
            B2(x,y) = C(x,y); % จัดเก็บตำแหน่ง B2 ในรูปเมตริกซ์เช่นเดียวกับ T2
            disp(' END collected Middle finger T2->B2 ')
            END = 1000; % ใช้ในการตรวจสอบว่าได้จัดเก็บจุดพิกัดน้ำได้ถูกต้อง
            break % ออกจากลูปจัดเก็บจุดพิกัด
      end
   end
   if END == 1000 % ถ้า END คือ 1000 แสดงว่าจัดเก็บจุดพิกัดน้ำกลางถูกต้อง เข้าเงื่อนไข
                  % ถ้า END มีค่า 1 แสดงว่าไม่ใช่จุดพิกัดน้ำกลาง เริ่มหาจุดที่มีค่า 1 ใหม่
      break       % ออกจากโปรแกรมจัดเก็บจุดพิกัดน้ำกลาง
   end
end
```

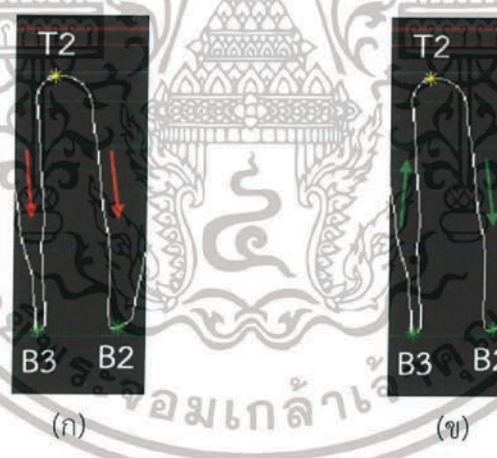


รูปที่ 3.12 วิธีการเก็บพิกัดของนิ้วมือ

จากนั้นกลับมาเริ่มที่จุด T2 อีกครั้ง แล้วเริ่มตรวจสอบตำแหน่งพิกัดที่พิกเซลมีค่า 1 ทางด้านซ้ายของนิ้วกลาง โดยตรวจสอบจุด p9, p8, p7, p6 และ p5 ตามลำดับ ซึ่งเป็นตำแหน่งที่อยู่ทางด้านซ้ายของจุด p1 สาเหตุที่ต้องตรวจสอบจุด p5 และ p9 เป็นเหตุผลเดียวกับการตรวจสอบจุด p3 และ p7 ทางด้านขวา ดำเนินการตรวจสอบด้วยวิธีการเช่นเดียวกับการเก็บพิกัดทางด้านขวามือของนิ้วกลาง จนถึงจุดต่ำสุดทางด้านซ้ายของนิ้วกลางคือจุด B3 เราจะทราบได้ว่าจุด B3 เป็นจุดต่ำสุดเมื่อจุดพิกัดอยู่ที่ตำแหน่ง p9, p2 หรือ p3 คือทิศทางของจุดภาพเริ่มเป็นขาขึ้นดังรูปที่ 3.12 การเก็บพิกัดของนิ้วกลางแสดงได้ดังรูปที่ 3.13 (ก) เมื่อจัดเก็บเสร็จแล้วเรียงพิกัดใหม่โดยเรียงจากจุด B3 ไปจุด T2 และเรียงต่อจนถึงจุด B2 ดังรูปที่ 3.13 (ข)

```
% โปรแกรมตรวจสอบจุดพิกัดนิ้วกลาง T2 ไป B3
  [x,y] = size(T2); % จุดพิกัดของ T2 ที่จัดเก็บในรูปของเมตริกซ์ที่มีขนาด x คูณ y
% เริ่มจากจุด T2
  a1 = 1; % เริ่มนับจุดพิกัดจุดแรก
  p9 = 0; % กำหนดค่าก่อนเริ่มนับจำนวนจุด p9 ที่ตรวจพบ
  for c = 1 : 1000
      if c ~= 1 % หลีกเลี่ยงการจัดเก็บจุด T2 ซึ่งคือจุดแรกที่ถูกนับ สำหรับจุดอื่นๆเก็บจุดพิกัดปกติ
         C(x,y) = Image_Edge_Skeleton(x,y); %ใช้ตัวแปร C ในการจัดเก็บจุดพิกัดนิ้วกลาง
         Image_Edge_Skeleton(x,y) = 10; % ลบจุดพิกัดนิ้วกลางที่จัดเก็บไปแล้ว
                                        % โดยเปลี่ยนค่าจาก 1 เป็น 10
         xWT2(a1) = x; % จัดเก็บตำแหน่งจุดพิกัด x ที่อยู่ด้านซ้ายของนิ้วกลาง
         yWT2(a1) = y; % จัดเก็บตำแหน่งจุดพิกัด y ที่อยู่ด้านซ้ายของนิ้วกลาง
         a1 = a1+1; % เพิ่ม a1 ทีละ 1 เมื่อจัดเก็บจุดพิกัดแล้ว
      end
      if  Image_Edge_Skeleton(x-1,y-1) == 1 % ตรวจสอบตำแหน่ง p9
        % เนื่องจากจุด p9 เป็นจุดที่อยู่ในทิศทางซ้ายบนซึ่งจะใช้ในการหยุดการจัดเก็บจุดพิกัด
        % แต่จำเป็นต้องใช้ในการจัดเก็บจุดพิกัดด้วยเช่นกัน จึงต้องกำหนดให้มีการจัดเก็บจุดพิกัด
        % อย่างน้อยหนึ่งครั้ง และเมื่อพบ p9 ให้นับ p9 เพิ่มขึ้น 1 ครั้ง
         p9 = p9 + 1;
         if p9 > 1 % สิ้นสุดการจัดเก็บเมื่อเจอ p9 เกินหนึ่งครั้ง
            B3(x,y) = C(x,y); % จัดเก็บตำแหน่ง B3 ในรูปเมตริกซ์
            a1 = a1-1; %ลดค่า a1 ลง 1 ในกรณีสิ้นสุดการจัดเก็บ เพราะเพิ่มค่าเกินจำเป็น
            disp(' END collected Middle finger T2->B3 ')
```

```
            break  % ออกจากโปรแกรมจัดเก็บจุดพิกัด
        end
        x = x-1;
        y = y-1;
    elseif Image_Edge_Skeleton(x,y-1) == 1 % ตรวจสอบตำแหน่ง p8
        x = x;
        y = y-1;
    elseif Image_Edge_Skeleton(x+1,y-1) == 1 % ตรวจสอบตำแหน่ง p7
        x = x+1;
        y = y-1;
    elseif Image_Edge_Skeleton(x+1,y) == 1 % ตรวจสอบตำแหน่ง p6
        x = x+1;
        y = y;
    elseif Image_Edge_Skeleton(x+1,y+1) == 1 % ตรวจสอบตำแหน่ง p5
        x = x+1;
        y = y+1;
% สิ้นสุดการจัดเก็บตำแหน่งพิกัดเมื่อมีทิศทางขึ้น  ตรวจสอบตำแหน่ง p2 และ p3
    elseif (Image_Edge_Skeleton(x-1,y) == 1)   || % ตรวจสอบตำแหน่ง p2
        (Image_Edge_Skeleton(x-1,y+1) == 1)     % ตรวจสอบตำแหน่ง p3
        B3(x,y) = C(x,y); % จัดเก็บจุด B3 ในรูปเมตริกซ์
        a1 = a1-1; %ลดค่า a1 ลง 1 ในกรณีสิ้นสุดการจัดเก็บ เพราะเพิ่มค่าเกินจำเป็น
        disp(' END collected Middle finger T2->B3 ')
        break % ออกจากโปรแกรมจัดเก็บจุดพิกัด
    end
end
```

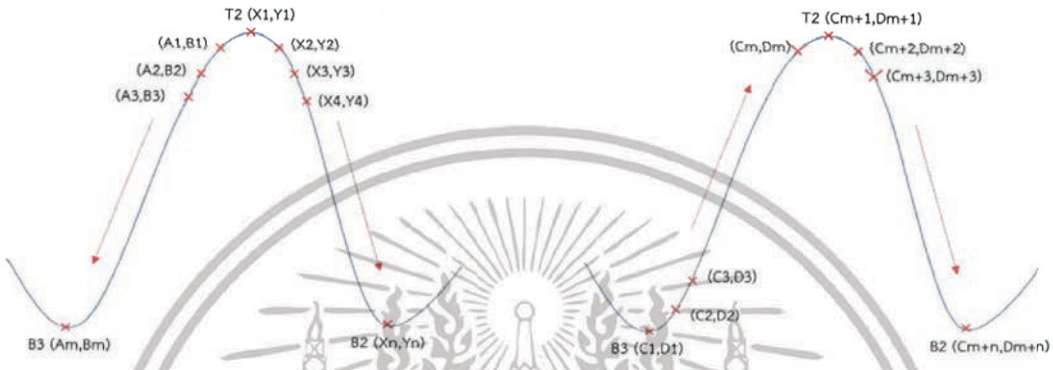


รูปที่ 3.13 ทิศทางการเก็บและเรียงจุดพิกัดนิ้วกลาง
(ก) การเก็บจุดพิกัดนิ้วกลาง
(ข) การเรียงจุดพิกัดนิ้วกลาง

การเก็บค่าพิกัดจะเก็บอยู่ในรูป X_n และ Y_n เมื่อ X และ Y คือพิกัดในแนวแกน x และ y ตามลำดับ และ n คือจำนวนจุดพิกัดที่จัดเก็บ กำหนดให้จุด T2 มีพิกัด (X_1,Y_1) พิกัดทางด้านขวาจุดต่อมาให้มีพิกัด (X_2, Y_2), (X_3, Y_3) เรียงไปจนถึงจุด B2 มีพิกัด (X_n, Y_n) ดังรูปที่ 3.14 (ก) สำหรับตำแหน่งพิกัดด้านซ้ายที่ต่อจากจุด T2 ให้มีพิกัด (A_1, B_1) โดยให้ A และ B คือพิกัดในแนวแกน x และ y พิกัดทาง

ด้านซ้ายจุดต่อมาให้มีพิกัด (A_2, B_2), (A_3, B_3) เรียงไปจนถึงจุด B3 ที่พิกัด (A_m, B_m) เมื่อ m คือจำนวนจุดพิกัดที่จัดเก็บด้านซ้าย

เมื่อเรียงพิกัดใหม่โดยเริ่มจากจุด B3 จะให้มีค่าพิกัด (C_1, D_1) เมื่อ C และ D คือพิกัดในแนวแกน x และ y และ C_1 มีค่าพิกัดเท่ากับ A_m ส่วน D_1 มีค่าพิกัดเท่ากับ B_m ดังนั้นตำแหน่งพิกัด (C_2, D_2) จะมีค่าเท่ากับพิกัด (A_{m-1}, B_{m-1}) เรียงไปเรื่อยๆ จนถึงตำแหน่ง T2 จะนับพิกัดต่อโดยมีค่าตำแหน่งพิกัด (C_{m+1}, D_{m+1}) เท่ากับพิกัด (X_1, Y_1) จนกระทั่งถึงจุด B2 ตำแหน่งพิกัด (C_{m+n}, D_{m+n}) จะมีค่าเท่ากับพิกัด (X_n, Y_n) ดังรูปที่ 3.14 (ข) สามารถพล๊อต (Plot) กราฟออกมาในรูปที่ 3.19 (ก)



(ก) (ข)

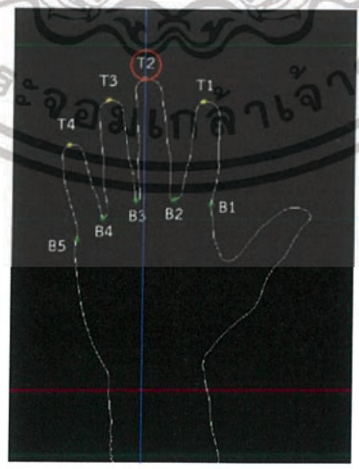
รูปที่ 3.14 วิธีเก็บและเรียงจุดพิกัดของนิ้ว
(ก) วิธีเก็บจุดพิกัดของนิ้ว
(ข) วิธีเรียงจุดพิกัดของนิ้ว

```
% เรียงจุดพิกัด x, y ของจุดพิกัดเส้นแบ่งน้ำกลางใหม่จาก B3 ไป T2 และจาก T2 ไป B2
% 'a' คือ จำนวนจุดพิกัดทางด้านขวาของน้ำกลาง
% 'a1' คือ จำนวนจุดพิกัดทางด้านซ้ายของน้ำกลาง
  b = 1; % กำหนดค่าเริ่มต้นที่ใช้เก็บจุดพิกัดด้านขวาของน้ำกลาง
  b1 = a1; % คงค่าตัวแปร 'a1' และใช้ 'b1' ในการเก็บจุดพิกัดด้านซ้ายของน้ำกลาง
% ในการเรียงจุดพิกัด จะเรียงจุดพิกัดทางด้านซ้ายและขวาของน้ำกลางพร้อมๆกัน
% คือเรียงจุดพิกัดจาก B3 ไป T2 และจาก T2 ไป B2 พร้อมกัน
  for c = 1 : a+a1 % a+a1 คือจำนวนจุดพิกัดน้ำกลางทั้งหมด
      if b1 >= 1 % เรียงจุดพิกัด B3 ไป T2 (ด้านซ้ายของน้ำกลาง)
         RxT2(c) = xWT2(b1); % จุดพิกัดในแนวแกน x ที่ถูกเรียงใหม่
         RyT2(c) = yWT2(b1); % จุดพิกัดในแนวแกน y ที่ถูกเรียงใหม่
         b1 = b1-1; % ใช้เก็บจำนวนจุดพิกัดทางด้านซ้ายของน้ำกลาง
      end
      if b <= a % เรียงจุดพิกัด T2 ไป B2 (ด้านขวาของน้ำกลาง)
         RxT2(c+a1) = xET2(b); % จุดพิกัดในแนวแกน x ที่ถูกเรียงใหม่
         RyT2(c+a1) = yET2(b); % จุดพิกัดในแนวแกน y ที่ถูกเรียงใหม่
         b = b+1; % ใช้เก็บจำนวนจุดพิกัดทางด้านขวาของน้ำกลาง
      end
      if (a == b) && (b1 == 0) % สิ้นสุดการเรียงจุดพิกัดด้านขวาและด้านซ้ายของน้ำกลาง
         break
      end
  end
```

เมื่อเก็บพิกัดของนิ้วกลางครบทุกจุดแล้ว จะทำการลบจุดพิกัดของนิ้วกลางที่ได้จัดเก็บเอาไว้ทิ้งไป เนื่องจากการหาจุดพิกัดของนิ้วอื่นๆได้แก่ นิ้วนาง นิ้วก้อย และนิ้วชี้ จะใช้หลักการเดียวกับนิ้วกลาง คือหาจุดสูงสุดของนิ้วก่อน หากไม่ลบจุดภาพบนภาพที่ได้จัดเก็บไปแล้ว เมื่อทำการหาจุดยอดของนิ้วถัดไป จะทำให้เจอกับจุดยอดของนิ้วเดิมที่ได้จัดเก็บไปแล้ว

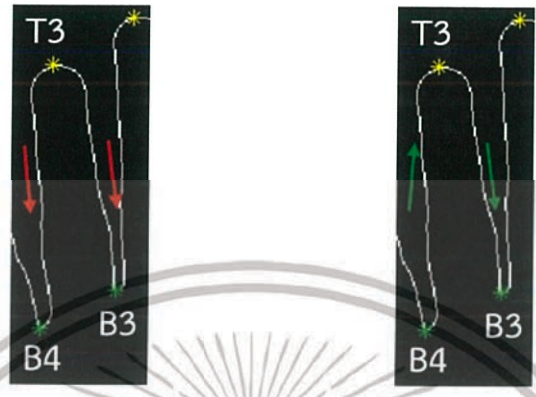
ต่อมาเป็นการหาจุดพิกัดของนิ้วนางโดยจะใช้หลักการเดียวกันกับการหาจุดพิกัดของนิ้วกลาง คือหาจุดพิกัดที่มีค่า 1 โดยการตรวจสอบภาพจากซ้ายไปขวาและจากบนลงล่าง แต่เนื่องจากความยาวของนิ้วนางและนิ้วชี้มีความใกล้เคียงกัน ลักษณะการวางมือขณะถ่ายภาพอาจมีการเอียงมือทำให้จุดยอดนิ้วชี้ขึ้นมาอยู่เหนือนิ้วนาง รวมถึงการที่แต่ละบุคคลมีความยาวนิ้วชี้และนิ้วนางที่แตกต่างกัน เช่น บุคคลที่มีนิ้วชี้ยาวกว่านิ้วนางหรือบุคคลที่มีนิ้วนางยาวกว่านิ้วชี้ ทำให้การเก็บค่าพิกัดอาจไปเก็บค่านิ้วชี้ก่อนนิ้วนาง จึงต้องกำหนดขอบเขตในการตรวจสอบโดยให้ตรวจสอบเฉพาะบริเวณด้านซ้ายของภาพโดยใช้จุดยอดของนิ้วกลางหรือจุด T2 เป็นจุดกำหนดขอบเขต ดังรูปที่ 3.15

```
% หาจุดพิกเซลที่มีค่า 1 จุดแรกของเส้นทางโดยการสแกนภาพจากซ้ายไปขวาและจากบนลงล่าง
[tx,ty] = size(T2); % จุดพิกัดของ T2 ที่จัดเก็บในรูปของเมตริกซ์ที่มีขนาด x คูณ y
for x = 2: m-1 % สแกนภาพในทิศทางบนลงล่าง
    for y = 2: ty % สแกนภาพในทิศทางซ้ายไปขวาจนถึงจุดพิกัดในแนวแกน y ของ T2
        if Image_Edge_Skeleton(x,y) == 1 % ถ้าเจอพิกเซลที่มีค่า 1
            break % พบจุดพิกัดที่มีค่า 1 แล้วออกจากลูป y = 2: ty
        end
    end
    if Image_Edge_Skeleton(x,y) == 1 % ถ้าเจอพิกเซลที่มีค่า 1
        break % พบจุดพิกัดที่มีค่า 1 แล้วออกจากลูป x = 2: m-1
    end       % ซึ่งคือการออกจากลูปค้นหาจุดพิกัดที่มีค่า 1 จุดแรก
end
% ---------------------------------------------------------------------
% การจัดเก็บจุดพิกัดของเส้นทาง การเขียนโปรแกรมจะใช้หลักการเดียวกับการจัดเก็บจุดพิกัดของเส้นกลาง
% ---------------------------------------------------------------------
```



รูปที่ 3.15 จุด T2 ที่เป็นจุดกำหนดขอบเขต

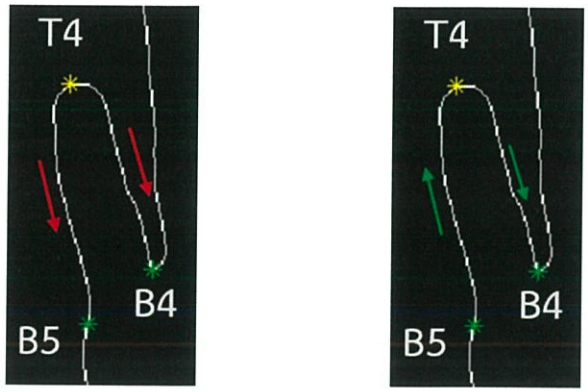
ดังนั้นจุดแรกที่พบจะเป็นจุดสูงสุดของนิ้วนางคือจุด T3 การเก็บพิกัดใช้วิธีเดียวกับนิ้วกลางทุกประการ คือแยกเก็บพิกัด x และ y เก็บพิกัดทางด้านขวาของนิ้วนางจนถึงจุด B3 กลับไปเริ่มที่จุด T3 และเก็บพิกัดทางด้านซ้ายของนิ้วจนถึงจุด B4 ดังรูปที่ 3.16 (ก) เมื่อเสร็จแล้วเรียงจุดพิกัดใหม่จากจุด B4 ไปจุด T3 และเรียงจนถึงจุด B3 ดังรูปที่ 3.16 (ข) จากนั้นจึงลบจุดพิกัดในรูปที่ 3.10 เช่นเดียวกัน



(ก) (ข)

รูปที่ 3.16 ทิศทางการเก็บและเรียงจุดพิกัดนิ้วนาง
(ก) การเก็บจุดพิกัดนิ้วนาง
(ข) การเรียงจุดพิกัดนิ้วนาง

นิ้วต่อมาที่จะทำการเก็บพิกัดคือนิ้วก้อย โดยจะใช้หลักการเดียวกับนิ้วนางคือตรวจสอบภาพเพียงด้านซ้ายของภาพเท่านั้น โดยจุดสูงสุดของนิ้วก้อยคือจุด T4 หรือจุดยอดของนิ้วก้อย จากนั้นเก็บพิกัดด้วยวิธีเดียวกับนิ้วกลาง คือเก็บพิกัดทางด้านขวาของนิ้วก้อยจนถึงจุด B4 แต่ระหว่างนั้นให้นับจำนวนจุดภาพที่จัดเก็บด้วย เนื่องจากการจัดเก็บพิกัดจะหยุดเก็บพิกัดเมื่อจุดภาพที่ขอบนิ้วเริ่มมีทิศทางขึ้นเพราะผ่านจุดต่ำสุดของนิ้วมือแล้ว แต่ที่ด้านซ้ายของนิ้วก้อยจะเป็นเส้นขอบนิ้วที่เชื่อมกับเส้นของบริเวณฝ่ามือจึงมีเพียงทิศทางลงอย่างเดียวจึงต้องใช้การนับจำนวนจุดพิกัดในการกำหนดจำนวนจุดที่จะจัดเก็บแทน จากนั้นกลับมาที่จุด T4 และเก็บพิกัดด้านซ้ายของนิ้วก้อยโดยให้จำนวนจุดภาพที่จัดเก็บให้มีจำนวนจุดที่เท่ากับจำนวนจุดภาพด้านขวาของนิ้วก้อยที่และบวกไปอีกร้อยละ 20 ของจุดภาพด้านขวาของนิ้วก้อย เมื่อครบแล้วจุดดังกล่าวจะเป็นจุด B5 ดังรูปที่ 3.17 (ก) สาเหตุที่ต้องบวกจำนวนจุดที่จัดเก็บไปอีกร้อยละ 20 เนื่องจากจุดดังกล่าวนี้จะถูกนำไปกำหนดจุดพิกัดในการแยกภาพฝ่ามือต่อไป หากกำหนดให้มีการจัดเก็บจุดภาพในจำนวนที่เท่ากันจุด B5 จะมีตำแหน่งที่สูงเกินไปและทำให้ภาพฝ่ามือที่ถูกแยกออกมามีภาพที่อยู่เหนือบริเวณฝ่ามือและทำให้ไม่สามารถเก็บภาพฝ่ามือจนถึงบริเวณด้านล่างของฝ่ามือได้ หลังจากเก็บจุดพิกัดครบแล้วให้เรียงจุดพิกัดใหม่โดยเรียงจากจุด B5 ไปจุด T4 และเรียงไปจนถึงจุด B4 ดังรูปที่ 3.17 (ข) จากนั้นจึงลบจุดพิกัดในรูปที่ 1 เช่นเดียวกัน



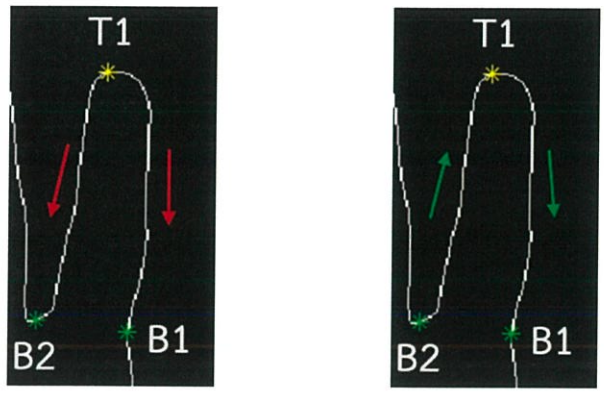
(ก) (ข)

รูปที่ 3.17 ทิศทางการเก็บและเรียงจุดพิกัดนิ้วก้อย
(ก) การเก็บจุดพิกัดนิ้วก้อย
(ข) การเรียงจุดพิกัดนิ้วก้อย

นิ้วสุดท้ายที่จะทำการเก็บจุดพิกัดคือนิ้วชี้ หลักการที่หาจุดสูงสุดของนิ้วจะเหมือนกับการหาจุดสูงสุดของนิ้วนางคือหาเพียงบางส่วนของภาพ แต่ต่างกันตรงที่การหาจุดสูงสุดของนิ้วชี้จะหาบริเวณด้านขวาของภาพของรูปที่ 3.15 โดยใช้จุด T2 เป็นจุดกำหนดขอบเขตเช่นเดียวกัน

```
% หาจุดพิกเซลที่มีค่า 1 จุดแรกของนิ้วชี้โดยการสแกนภาพจากซ้ายไปขวาและจากบนลงล่าง
[tx,ty] = size(T2); % จุดพิกัดของ T2 ที่จัดเก็บในรูปของเมตริกซ์ทั้งขนาด x คูณ y
for x = 2: m-1 % สแกนภาพในทิศทางบนลงล่าง
    % สแกนภาพในทิศทางซ้ายไปขวา เริ่มจากจุดพิกัดในแนวแกน y ของ T2 จนถึงขอบภาพ
    for y = ty: n-1
        if Image_Edge_Skeleton(x,y) == 1 % ถ้าเจอพิกเซลที่มีค่า 1
            break % พบจุดพิกัดที่มีค่า 1 แล้วออกจากลูป y = ty: n-1
        end
    end
    if Image_Edge_Skeleton(x,y) == 1
       break % พบจุดพิกัดที่มีค่า 1 แล้วออกจากลูป x = 2: m-1
    end       % ซึ่งคือการออกจากลูปค้นหาจุดพิกัดที่มีค่า 1 จุดแรก
end
% ---------------------------------------------------------------------
% การจัดเก็บจุดพิกัดของนิ้วชี้ การเขียนโปรแกรมจะใช้หลักการเดียวกับการจัดเก็บจุดพิกัดของนิ้วก้อย
% ---------------------------------------------------------------------
```

โดยจุดสูงสุดของนิ้วชี้เป็นจุด T1 หลักการเก็บจุดพิกัดจะใช้หลักการเดียวกับการเก็บจุดพิกัดของนิ้วก้อย แต่ต่างกันตรงที่จะเริ่มเก็บจุดพิกัดจากทางด้านซ้ายของนิ้วชี้ก่อน และนับจำนวนจุดภาพที่จัดเก็บจนถึงจุด B2 จากนั้นกลับมาที่จุด T1 แล้วเก็บจุดพิกัดด้านขวาของนิ้วชี้โดยจัดเก็บจำนวนจุดภาพที่ให้มีจำนวนเท่ากับจำนวนจุดภาพด้านซ้ายของนิ้วชี้และบวกไปอีกร้อยละ 20 ของจุดภาพด้านซ้ายของนิ้วชี้ ด้วยสาเหตุเดียวกับการจัดเก็บจุดพิกัดของนิ้วก้อย เมื่อครบแล้วกำหนดจุดดังกล่าวให้เป็นจุด B1 ดังรูปที่ 3.18 (ก) จากนั้นเรียงจุดพิกัดใหม่โดยเรียงจากจุด B2 ไปจุด T1 และเรียงไปจนถึงจุด B1 ดังรูปที่ 3.18 (ข)



(ก) (ข)

รูปที่ 3.18 ทิศทางการเก็บและเรียงจุดพิกัดนิ้วชี้
(ก) การเก็บจุดพิกัดนิ้วชี้
(ข) การเรียงจุดพิกัดนิ้วชี้

เมื่อสั่งพล๊อท (Plot) กราฟของแต่ละนิ้วที่จัดเก็บค่าไว้ออกมาจะได้ดังรูปที่ 3.19 (ก) (ข) (ค) และ (ง) คือนิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อยตามลำดับ

(ก) (ข)

(ค) (ง)

รูปที่ 3.19 การพล๊อทกราฟนิ้วมือจากจุดพิกัดที่จัดเก็บไว้
(ก) การพล๊อทกราฟนิ้วชี้
(ข) การพล๊อทกราฟนิ้วกลาง
(ค) การพล๊อทกราฟนิ้วนาง
(ง) การพล๊อทกราฟนิ้วก้อย

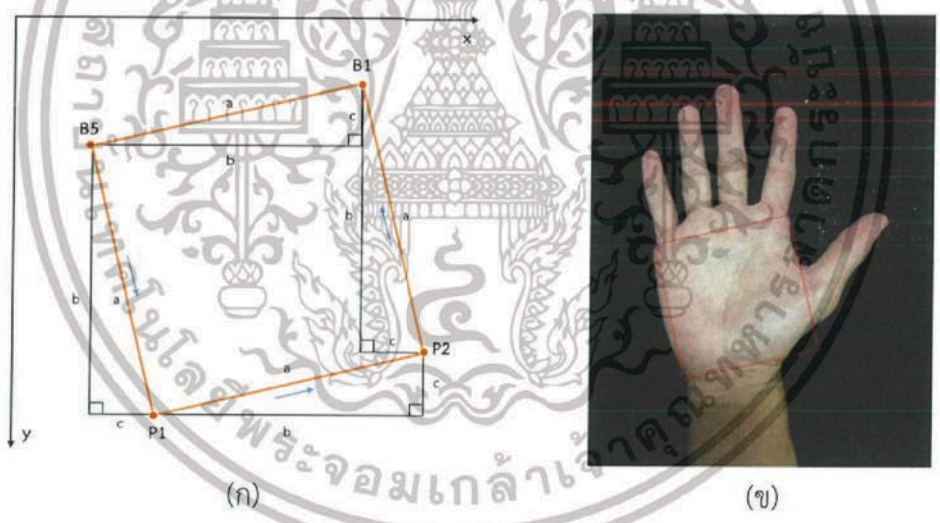
3.4 การแยกฝ่ามือออกจากมือ (Palm Print Extraction)

การแยกฝ่ามือออกจากมือในปริญญานิพนธ์ฉบับนี้ได้นำเสนอแนวทางในการแยกฝ่ามือทั้งหมด 2 แนวทาง

ในการหาภาพฝ่ามือด้วยแนวทางแรกนั้น จะใช้จุดพิกัดที่หาได้จากขั้นตอนเก็บจุดพิกัดของแต่ละนิ้วมือ ที่แสดงในรูปที่ 3.10 โดยใช้จุดพิกัด B1 และ B5 เป็นตัวกำหนดขนาดฝ่ามือที่ต้องการ เริ่มจากหาความยาวระหว่างจุดทั้งสองและตีเส้น ในที่นี้ใช้ทฤษฎีพีทาโกรัสดังสมการที่ (3.8)

$$a^2 = b^2 + c^2 \tag{3.8}$$

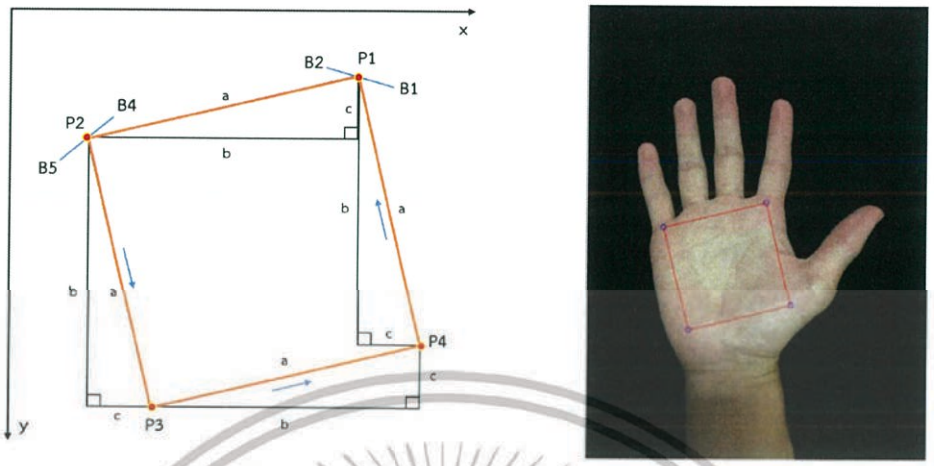
โดยให้ a คือความยาวระหว่าง B1 และ B5, b และ c คือด้านประกอบมุมฉากในรูปที่ 3.19 (ก) จากจุด B5 วัดความยาวลงมาขนานแกน y ขนาด b หน่วย แล้ววัดความยาวไปด้านขวาขนานแกน x ขนาด c หน่วย จะได้จุดพิกัดที่ต้องการ สมมติเป็นจุด P1 ทำการบันทึกจุดพิกัดดังกล่าวไว้ จากนั้นวัดความยาวต่อไปด้านขวาอีก b หน่วย แล้ววัดความยาวขึ้นไปด้านบนขนานแกน y ขนาด c หน่วย จะได้จุดพิกัดที่ต้องการ สมมติเป็นจุด P2 ทำการบันทึกจุดพิกัดดังกล่าวไว้ ดังรูปที่ 3.20 (ก) จุดพิกัดดังกล่าวทั้ง 4 จุดได้แก่ จุด B1, B5, P1 และ P2 จะได้กรอบสี่เหลี่ยมที่จะตัดภาพฝ่ามือออกมา ดังรูปที่ 3.20 (ข)



รูปที่ 3.20 การหาภาพฝ่ามือด้วยแนวทางที่หนึ่ง
(ก) วิธีกำหนดขนาดฝ่ามือ
(ข) กรอบที่จะตัดภาพฝ่ามือออกมา

สำหรับแนวทางที่สอง จะใช้จุดพิกัด B1, B2, B4 และ B5 เริ่มจากหาความยาวระหว่างจุด B1 และ B2 แล้วกำหนดจุดกึ่งกลางระหว่างความยาวทั้งสองนั้นให้เป็น P1 จากนั้นหาความยาวระหว่างจุด B4 และ B5 แล้วกำหนดจุดกึ่งกลางระหว่างความยาวทั้งสองนั้นให้เป็น P2 จากนั้นใช้วิธีการเดียวกับแนวทางแรก จะได้ผลดังรูปที่ 3.21 (ก) จุดพิกัดดังกล่าวทั้ง 4 จุดได้แก่ จุด P1, P2, P3 และ P4 ได้กรอบสี่เหลี่ยมที่จะตัดภาพฝ่ามือออกมา ดังรูปที่ 3.21 (ข)

จากรูปที่ 3.20 (ข) และรูปที่ 3.21 (ข) จะเห็นได้ว่ากรอบสี่เหลี่ยมที่จะตัดภาพฝ่ามือออกมาจะมีขนาดไม่เท่ากันจะทำให้ภาพฝ่ามือที่ตัดออกมาได้ภาพที่แตกต่างกันด้วย



(ก) (ข)

รูปที่ 3.21 การหาภาพฝ่ามือด้วยแนวทางที่สอง
(ก) วิธีกำหนดขนาดฝ่ามือ
(ข) กรอบที่จะตัดภาพฝ่ามือออกมา

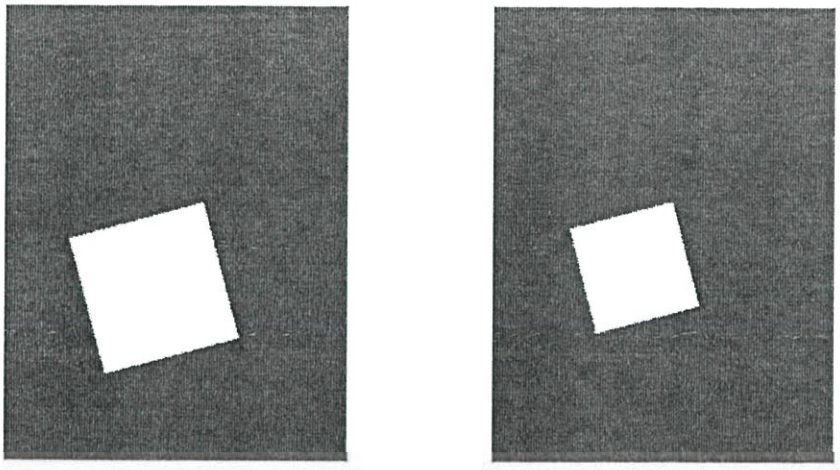
```
% โปรแกรมตัดภาพฝ่ามือแนวทางที่สอง
% หาจุดพิกัดของ B1 B2 B4 B5
[B1x,B1y] = size(B1); % จุดพิกัดของ B1 ที่จัดเก็บในรูปของเมตริกซ์ที่มีขนาด x คูณ y
[B2x,B2y] = size(B2); % จุดพิกัดของ B2 ที่จัดเก็บในรูปของเมตริกซ์ที่มีขนาด x คูณ y
[B4x,B4y] = size(B4); % จุดพิกัดของ B4 ที่จัดเก็บในรูปของเมตริกซ์ที่มีขนาด x คูณ y
[B5x,B5y] = size(B5); % จุดพิกัดของ B5 ที่จัดเก็บในรูปของเมตริกซ์ที่มีขนาด x คูณ y
% หาความยาวระหว่างจุด B1 และ B2
ppLB12x = B1x-B2x; % แนวแกน x
ppLB12x = abs(ppLB12x);
ppLB12y = B1y-B2y; % แนวแกน y
ppLB12y = abs(ppLB12y);
% กำหนดจุดที่จะตัด Palm Print จุดที่ 1 เป็นจุดที่อยู่ระหว่าง B1 และ B2 กำหนดเป็นจุด P1
palmprintx(1) = B1x-(ppLB12x/2);
palmprinty(1) = B1y-(ppLB12y/2);
% หาความยาวระหว่างจุด B4 และ B5
ppLB45x = B4x-B5x;
ppLB45x = abs(ppLB45x);
ppLB45y = B5y-B4y;
ppLB45y = abs(ppLB45y);
% กำหนดจุดที่จะตัด Palm Print จุดที่ 2 เป็นจุดที่อยู่ระหว่าง B4 และ B5 กำหนดเป็นจุด P2
palmprintx(2) = B5x+(ppLB45x/2);
palmprinty(2) = B5y-(ppLB45y/2);
% หาความยาวสามเหลี่ยมระหว่างจุด P1 และ P2
ppLx = palmprintx(1)-palmprintx(2);
ppLx = abs(ppLx);
ppLy = palmprinty(2)-palmprinty(1);
ppLy = abs(ppLy);
```

```
% กำหนดจุดชั่วคราวจุดที่ 1
up1x = palmprintx(2);
up1y = palmprinty(2)+ppLx;
% กำหนดจุดที่จะตัด Palm Print จุดที่ 3 กำหนดเป็นจุด P3
palmprintx(3) = up1x+ppLy;
palmprinty(3) = up1y;
% กำหนดจุดชั่วคราวจุดที่ 2
up2x = palmprintx(3)+ppLx;
up2y = palmprinty(3);
% กำหนดจุดที่จะตัด Palm Print จุดที่ 4 กำหนดเป็นจุด P4
palmprintx(4) = up2x;
palmprinty(4) = up2y-ppLy;
% กำหนดจุดที่จะตัด palmprint จุดที่ 5 (จุดเดียวกับจุดที่ 1)
palmprintx(5) = palmprintx(1);
palmprinty(5) = palmprinty(1);
%-------------------------------------------------------------
% ตัวแปร palmprintx และ palmprinty ใช้เก็บจุดพิกัดที่จะใช้ตัด Palm Print
%-------------------------------------------------------------
```

เมื่อได้จุดพิกัดที่จะใช้งานแล้ว ต่อมาจะกำหนดพื้นที่ที่ต้องการ โดยใช้คำสั่ง roipoly เป็นเครื่องมือในโปรแกรม MATLAB ที่จะกำหนดพื้นที่ที่สนใจ (Region of Interest : ROI) แสดงการใช้งานคำสั่งได้ดังนี้

```
% การใช้งานคำสั่ง roipoly
Palmprint_ROI = roipoly(Image_Gray,palmprintx,palmprinty);
% palmprintx คือจุดพิกัดในแนวแกน x ของจุดพิกัด B1, B5, P1 และ P2 ของแนวทางที่หนึ่ง
%                                    P1, P2, P3 และ P4 ของแนวทางที่สอง
% palmprinty คือจุดพิกัดในแนวแกน y ของจุดพิกัด B1, B5, P1 และ P2 ของแนวทางที่หนึ่ง
%                                    P1, P2, P3 และ P4 ของแนวทางที่สอง
```

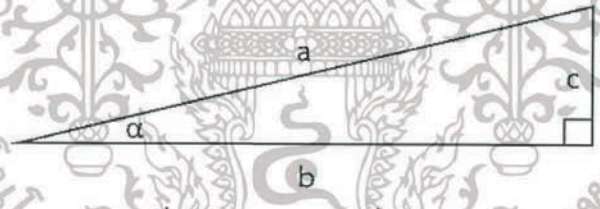
โดยพื้นที่ภายในจุดพิกัดที่กำหนดจะมีค่าเป็น 255 หรือสีขาว และนอกพื้นที่ที่สนใจหรือนอกจุดพิกัดจะค่าเป็น 0 หรือสีดำแสดงได้ดังรูปที่ 3.22 (ก) และ 3.22 (ข) เป็นพื้นที่ฝ่ามือจากแนวทางแรกและแนวทางที่สองตามลำดับ



(ก) (ข)

รูปที่ 3.22 พื้นที่ฝ่ามือแนวทางที่หนึ่งและสอง
(ก) พื้นที่ฝ่ามือแนวทางที่หนึ่ง
(ข) พื้นที่ฝ่ามือแนวทางที่สอง

ขั้นตอนต่อมาคือการหมุนภาพ เนื่องจากโปรแกรม MATLAB สามารถตัดภาพ (Crop) ได้ในแนวแกน x และ y เท่านั้น จึงต้องหมุนภาพหรือพื้นที่ที่เราต้องการให้อยู่ในแนวแกนเสียก่อน จากรูปที่ 3.20 (ก) และรูปที่ 3.21 (ก) เมื่อเราทราบความยาวด้านของสามเหลี่ยมแล้วเราจะสามารถหามุมของพื้นที่ที่เราต้องการหมุนได้ แสดงรูปสามเหลี่ยมได้ดังรูปที่ 3.23

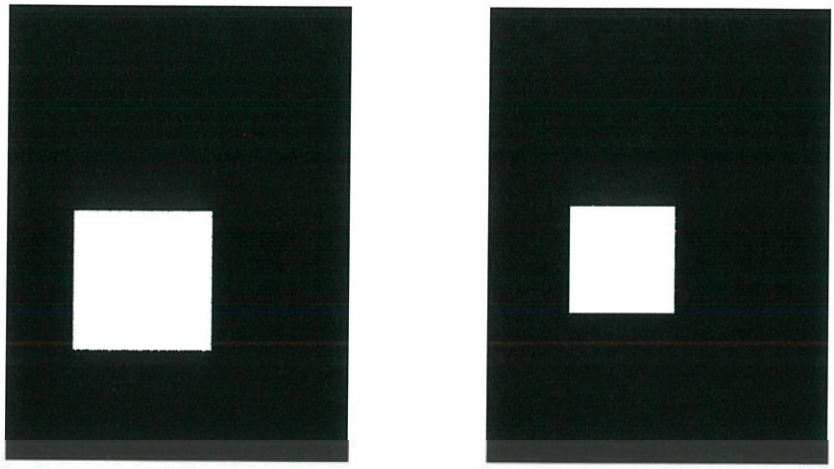


รูปที่ 3.23 รูปสามเหลี่ยมมุมฉาก

จากรูปที่ 3.23 สามารถหามุม α ได้จากสมการที่ (3.9)

$$\alpha = tan^{-1}\left(\frac{c}{b}\right) \qquad (3.9)$$

เมื่อทราบขนาดมุม α แล้วจะใช้คำสั่ง imrotate ในโปรแกรม MATLAB หมุนรูปที่ 3.22 (ก) และ (ข) โดยจะหมุนไปขนาด α องศา ได้ผลลัพธ์ดังรูปที่ 3.24 (ก) และ (ข) เป็นภาพจากแนวทางแรกและแนวทางที่สองตามลำดับ



(ก) (ข)

รูปที่ 3.24 พื้นที่ฝ่ามือแนวทางที่หนึ่งและสองที่ถูกหมุน
(ก) พื้นที่ฝ่ามือแนวทางที่หนึ่งที่ถูกหมุน
(ข) พื้นที่ฝ่ามือแนวทางที่สองที่ถูกหมุน

สำหรับรูปภาพที่จะใช้ดำเนินการในขั้นตอนถัดไปจำเป็นที่จะต้องใช้ภาพที่เป็นภาพระดับสีเทา ดังนั้น รูปภาพฝ่ามือที่เราต้องการแยกออกมาจึงต้องใช้ภาพระดับสีเทาดังรูปที่ 3.25 และต้องหมุนภาพดังกล่าวเช่นเดียวกัน โดยใช้คำสั่ง imrotate และหมุนไปขนาด α องศา ได้ผลดังรูปที่ 3.26

รูปที่ 3.25 ภาพฝ่ามือระดับสีเทาก่อนหมุน

รูปที่ 3.26 ภาพฝ่ามือระดับสีเทาที่ถูกหมุนแล้ว

```
% หามุมที่จะหมุนภาพ
angle_rotate = atand(ppLy/ppLx);
% หมุนภาพ
Palmprint_ROI_Rotate = imrotate(Palmprint_ROI,angle_rotate);
Hand_rotate = imrotate(Image_gray,angle_rotate);
```

เมื่อหมุนภาพเสร็จแล้วต่อมาจะเป็นการตัดภาพเฉพาะบริเวณที่ต้องการ โดยใช้คำสั่ง regionprops กับรูปที่ 3.24 เพื่อหาจุดพิกัดและขนาดของบริเวณที่สนใจคือกล่องสี่เหลี่ยมสีขาว จากนั้นนำพิกัดและขนาดที่ได้มาใช้ร่วมกับคำสั่ง imcrop ในการตัดภาพตัดภาพที่ 3.26

```
% การใช้งานคำสั่ง regionprops และ imcrop ร่วมกันเพื่อตัดภาพฝ่ามือ
  Palmprint_ROI_Rotate_2 = regionprops(Palmprint_ROI_Rotate,'BoundingBox');
  Palmprint_Crop = imcrop(Hand_rotate,Palmprint_ROI_Rotate_2.BoundingBox);
% 'BoundingBox' คือคำสั่งที่จะหาจุดพิกัดและขนาดของบริเวณที่เป็นกล่องสี่เหลี่ยมสีขาวในภาพ
% Palmprint_ROI_Rotate แล้วนำพิกัดและขนาดที่ได้มาตัดภาพ Hand_rotate
% อ้างอิงการใช้งานคำสั่งจาก https://www.mathworks.com/matlabcentral/answers/3584
%                         42-how-to-crop-and-save-plot-square-on-an-image-usin
%                         g-4-points-coordinates [17]
```

สามารถตัดภาพฝ่ามือออกมาได้ดังรูปที่ 3.27 (ก) และ (ข) เป็นภาพฝ่ามือแนวทางแรกและแนวทางที่สองตามลำดับ

การเลือกแนวทางในการแยกภาพฝ่ามือที่เหมาะสมจะเลือกใช้แนวทางที่สองในการแยกภาพฝ่ามือออกจากมือ เนื่องจากได้พื้นที่ฝ่ามือที่อยู่ในขอบเขตของฝ่ามือ ส่วนภาพที่ได้จากแนวทางแรกแม้จะได้พื้นที่ฝ่ามือที่มากกว่าแนวทางที่สอง แต่จะได้ภาพที่อยู่นอกฝ่ามือมาด้วยในที่นี้คือภาพพื้นหลังเป็นผลจากการกำหนดพื้นที่ในการตัดภาพฝ่ามือที่ใหญ่เกินไป โดยภาพฝ่ามือที่ได้ออกมาจะนำไปปรับปรุงภาพให้มีความเหมาะสมเพื่อนำไปแยกเส้นลายมือบนฝ่ามือต่อไป

(ก) (ข)

รูปที่ 3.27 ผลที่ได้จากการตัดภาพฝ่ามือด้วยแนวทางที่หนึ่งและสอง
(ก) ภาพฝ่ามือแนวทางที่หนึ่ง
(ข) ภาพฝ่ามือแนวทางที่สอง

3.5 การแยกเส้นลายมือออกจากฝ่ามือ (Palm Lines Extraction)

หลังจากที่ตัดภาพฝ่ามือออกมาจากภาพมือแล้ว ต่อมาจะทำการแยกเส้นลายมือออกจากภาพฝ่ามือ ปริญญานิพนธ์ฉบับนี้จะกล่าวถึงวิธีการที่ได้ทำตามขั้นตอนของงานวิจัยต่างๆที่ได้ศึกษา โดยสามารถแยกเป็นแนวทางได้ 3 แนวทางดังนี้

3.5.1 ศึกษาข้อมูลจากงานวิจัยเรื่อง Palmprint based Verification System Robust to Occlusion using Low-order Zernike Moments of Sub-images

งานวิจัยนี้เขียนโดย Badrinath G. S., Naresh Kumar Kachhi and Phalguni Gupta, BMVC, 2009, pp. 1-11. [5] จะกล่าวถึงขั้นตอนและวิธีการในการตัดภาพฝ่ามือออกจากภาพมือ การปรับปรุงภาพฝ่ามือ (Palm Print Enhancement) การแยกเส้นลายมือออกจากภาพฝ่ามือ และการระบุตัวบุคคล โดยเนื้อหาที่ปริญญานิพนธ์นี้จะนำมาใช้ในการศึกษาคือขั้นตอนการปรับปรุงภาพฝ่ามือ

เนื่องจากปริญญานิพนธ์ฉบับนี้จะใช้เส้นขอบนิ้วมือและเส้นลายมือในการยืนยันตัวบุคคล จึงต้องมีการแยกเส้นลายมือจำนวน 3 เส้นออกจากภาพมือเสียก่อน ดังนั้นจึงต้องมีการปรับปรุงภาพฝ่ามือให้มีความเหมาะสมในการแยกเส้นลายมือ การปรับปรุงภาพเริ่มจากการนำภาพฝ่ามือในรูปที่ 3.28 (ก) มาแบ่งเป็นบล็อก (Block) จำนวน 32 x 32 บล็อก จากนั้นหาค่าเฉลี่ยในแต่ละบล็อก แล้วเปลี่ยนค่าของทุกจุดภาพในซับบล็อกนั้นให้เป็นค่าเฉลี่ยที่หามาได้ของซับบล็อกนั้นๆ ดังรูปที่ 3.28 (ข)

```
% การแบ่งภาพเป็นบล็อกจำนวน 32 x 32 บล็อก จะเริ่มจากการหาขนาดความยาวของแต่ละบล็อกก่อน
[palmprint_crop_x,palmprint_crop_y] = size(Palmprint_Crop); %หาขนาดภาพฝ่ามือ
subblock = 32 % จำนวนบล็อกที่ต้องการจะแบ่ง
palmprint_crop_x_length = palmprint_crop_x/subblock;%ขนาดของบล็อกในแนวแกน x
palmprint_crop_y_length = palmprint_crop_y/subblock;%ขนาดของบล็อกในแนวแกน y
PCpixel_x = 1; % เริ่มแบ่งพิกเซลที่ 1 ในแนวแกน x
PCpixel_y = 1; % เริ่มแบ่งพิกเซลที่ 1 ในแนวแกน y
for SBcountx = 1: subblock
    for SBcounty = 1: subblock
        % หาจุดเริ่มของแต่ละบล็อกในแนวแกน x ที่ต้องการจะ Crop
        % หลักการคือจะนำบล็อกที่ต้องการหาจุดเริ่มต้น เช่น บล็อกที่ 2 (SBcountx = 2) มาลบด้วย 1
        % เนื่องจากการหาจุดเริ่มต้นของแต่ละบล็อกจะต้องแบ่งพิกเซลออกจากบล็อกก่อนหน้า (บล็อกที่ 1)
        % ซึ่งบล็อกก่อนหน้าจะมีความยาวของบล็อกคือ palmprint_crop_x_length
        % ดังนั้นจะนำ SBcountx ที่ลบ 1 แล้ว (SBcountx - 1) มาคูณกับความยาวของแต่ละบล็อก
        % (palmprint_crop_x_length) จะได้จำนวนของพิกเซลทั้งหมดของบล็อกก่อนหน้า
        % จากนั้นนำจำนวนพิกเซลทั้งหมดของบล็อกก่อนหน้า
        % ((SBcountx-1)*(palmprint_crop_x_length))
        % มาบวกด้วย 1 เพื่อหาจุดเริ่มต้นของบล็อกที่ต้องการจะ Crop ได้ PCpixel_x
        % คือจุดเริ่มต้นของบล็อกที่ 2
        PCpixel_x = 1+((SBcountx-1)*(palmprint_crop_x_length));
        % หาจุดเริ่มของแต่ละบล็อกในแนวแกน y ที่ต้องการจะ Crop
        % ใช้หลักการเดียวกับการหาจุดเริ่มต้นของบล็อกในแนวแกน x
        PCpixel_y = 1+((SBcounty-1)*(palmprint_crop_y_length));
        % กำหนดชื่อตัวแปรใหม่
        PCxL = palmprint_crop_x_length; % ขนาดของบล็อกในแนวแกน x
        PCyL = palmprint_crop_y_length; % ขนาดของบล็อกในแนวแกน y
        % Crop แต่ละบล็อกที่ต้องการหาค่าเฉลี่ย
        palmprint_subblock = imcrop(Palmprint_Crop,[PCpixel_x PCpixel_y PCxL PCyL]);
        % หาค่าเฉลี่ยของแต่ละบล็อก
        palmprint_subblock_mean = mean(mean(palmprint_subblock));
        % สร้างบล็อกจำนวน 32 x 32 บล็อก
        for countx = PCpixel_x: ((SBcountx)*(palmprint_crop_x_length))
            for county = PCpixel_y: ((SBcounty)*(palmprint_crop_y_length))
                palmprint_crop_blocks(county,countx) =  palmprint_subblock_mean;
            end
        end
    end
end
```

เนื่องจากค่าเฉลี่ยของแต่ละบล็อกจะมีค่าที่ไม่เท่ากัน จะทำให้ภาพมีรอยต่อระหว่างบล็อก จึงจะทำให้ภาพมีความราบเรียบ ไม่มีรอยต่อระหว่างบล็อก โดยใช้ Bi-cubic Interpolation ซึ่งเป็นวิธีการที่งานวิจัยฉบับนี้ใช้ โดยใช้คำสั่งดังกล่าวกระทำกับภาพที่ 3.28 (ข) จะได้ผลดังรูปที่ 3.28 (ค)

```
% ทำให้ภาพมีความราบเรียบ ไม่มีรอยต่อระหว่างบล็อก โดยใช้ Bi-cubic Interpolation
% หาขนาดของภาพที่ผ่านการสร้างบล็อกจำนวน 32 x 32 บล็อก
[pp_cb_y,pp_cb_x] = size(palmprint_crop_blocks)
% กำหนดว่าจะทำให้ภาพราบเรียบตั้งแต่พิกเซลใดจนถึงพิกเซลใด ในที่นี้คือทำให้ภาพราบเรียบทั้งภาพ
x = (1:pp_cb_x); % กำหนดทำให้ภาพราบเรียบตั้งแต่พิกเซลแรกจนถึงพิกเซลสุดท้าย ในแนวแกน x
y = (1:pp_cb_y); % กำหนดทำให้ภาพราบเรียบตั้งแต่พิกเซลแรกจนถึงพิกเซลสุดท้าย ในแนวแกน y
cs_value =  0.0025 % เป็นขนาดที่ใช้กำหนดว่าจะทำให้ภาพมีความราบเรียบขนาดไหน
palmprint_cubic_smooth = csaps({x,y},palmprint_crop_blocks,cs_value,{x,y});
% csaps คือคำสั่งที่จะทำให้ภาพมีความราบเรียบ โดยสามารถกำหนดได้ว่าจะทำให้ภาพมีความราบเรียบ
%       มากขนาดไหน (cs_value) และที่บริเวณใดของภาพ ({x,y})
% อ้างอิงการใช้งานคำสั่งจาก http://mathforum.org/kb/message.jspa?messageID=956102 [19]
```

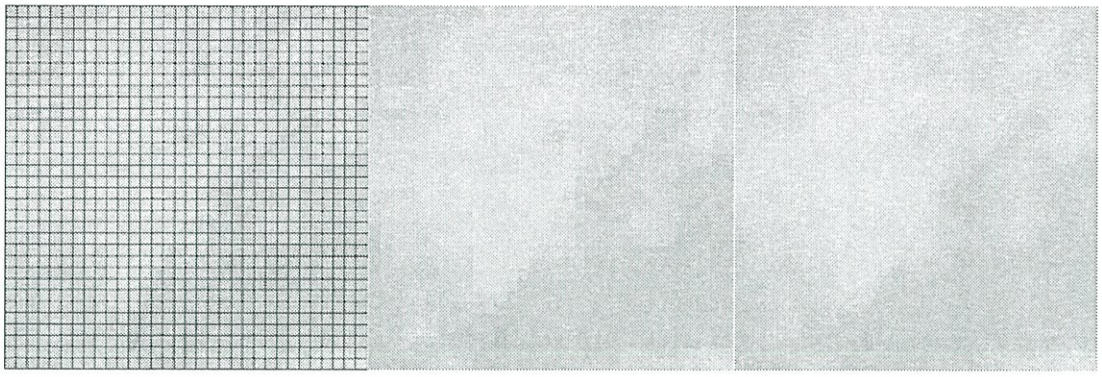
จากนั้นนำภาพฝ่ามือต้นฉบับคือภาพฝ่ามือในรูปที่ 3.28 (ก) มาลบออกด้วยภาพที่ผ่านการทำให้เรียบคือรูปที่ 3.28 (ค) จะได้ผลดังรูปที่ 3.28 (ง)

```
% นำภาพฝ่ามือต้นฉบับมาลบออกด้วยภาพที่ผ่านการทำให้เรียบ
palmprint_subtract = Palmprint_Crop - palmprint_cubic_smooth;
```

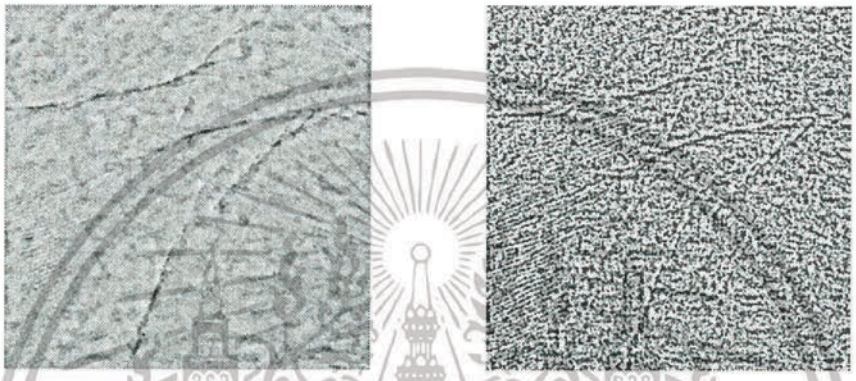
สุดท้ายคือการทำฮีสโตแกรมอีควอไลเซชั่น (Histogram Equalization) โดยการแบ่งรูปภาพออกเป็น 64 x 64 บล็อก แล้วทำฮีสโตแกรมอีควอไลเซชั่นแยกแต่ละบล็อกได้ผลดังรูปที่ 3.28 (จ)

```
% การทำ Histogram Equalization
% จะทำโดยใช้วิธีการเดียวกับการแบ่งภาพเป็นบล็อกจำนวน 32 x 32 บล็อก
% โดยเปลี่ยนค่า "subblock" จาก 32 เป็น 64 หรือเขียนค่าสั่งดังนี้
subblock = 64; % กำหนดให้แบ่งบล็อกจำนวน 64 x 64 บล็อก
% โดยนำภาพ 'palmprint_subtract' มาแบ่งบล็อกเป็นจำนวน 64 x 64 บล็อก
% จะได้ภาพที่มีชื่อตัวแปร 'palmprint_subblock' ออกมา
% และเปลี่ยนค่าสั่งจากการหาค่าเฉลี่ยในแต่ละบล็อก
palmprint_subblock_mean = mean(mean(palmprint_subblock));
% เป็นค่าสั่งทำ Histogram Equalization ในแต่ละบล็อกแทน คือ
palmprint_subblock_histeq = histeq(palmprint_subblock);
% จากนั้นนำ 'palmprint_subblock_histeq' ที่ไปสร้างบล็อกจำนวน 64 x 64 บล็อก
% ซึ่งใช้วิธีการเดียวกับการสร้างบล็อกจำนวน 32 x 32 บล็อก จะได้ตัวแปร
% 'palmprint_crop_blocks' ซึ่งเป็นภาพที่ผ่านการทำฮีสโตแกรมอีควอไลเซชั่นแยกแต่ละบล็อก
% จำนวน 64 x 64 บล็อกออกมา
```

จากผลการทดลองที่ในรูปที่ 3.28 (จ) พบว่าภาพไม่มีความชัดเจนมากเพียงที่จะแยกเส้นลายมือออกจากฝ่ามือได้ และเมื่อเปรียบเทียบกับภาพตัวอย่างจากงานวิจัยในรูปที่ 3.29 จะพบว่าภาพที่ได้จากการทดลองในปริญญานิพนธ์ มีความแตกต่างจากภาพที่ได้จากงานวิจัยอย่างมากทั้งในด้านความคมชัด และความชัดเจนของเส้นลายมือ

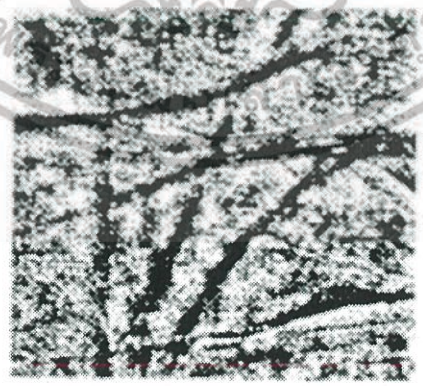


(ก) (ข) (ค)



(ง) (จ)

รูปที่ 3.28 การปรับปรุงภาพฝ่ามือ
(ก) ภาพฝ่ามือต้นฉบับ
(ข) ภาพฝ่ามือที่ผ่านการทำ 32x32 บล็อกและหาค่าเฉลี่ยในแต่ละบล็อกแล้ว
(ค) ภาพที่ผ่านการทำให้ราบเรียบด้วย Bi-cubic Interpolation
(ง) ผลลัพธ์จากการนำภาพฝ่ามือต้นฉบับลบออกด้วยภาพที่ผ่านการทำให้เรียบ
(จ) ภาพที่ผ่านการทำฮีสโตแกรมอีควอไลเซชั่นแยกแต่ละบล็อก



รูปที่ 3.29 ตัวอย่างภาพผลลัพธ์จากงานวิจัย [5]

สำหรับสาเหตุที่ผลลัพธ์ที่ได้จากการทดลองในรูปที่ 3.28 (จ) ไม่สามารถปรับปรุงภาพให้มีความชัดเจนได้เช่นเดียวกับผลที่ได้จากงานวิจัยในรูปที่ 3.29 คาดว่าสาเหตุเกิดจากความไม่เข้าใจในขั้นตอน

และวิธีการของงานวิจัย รวมถึงอาจมีข้อมูลหรือรายละเอียดส่วนอื่นที่งานวิจัยไม่ได้กล่าวถึงไว้ด้วย ดังนั้นจึงต้องศึกษางานวิจัยและวิธีการอื่นๆ เพื่อหาแนวทางที่เหมาะสม

3.5.2 ศึกษาข้อมูลจากงานวิจัยเรื่อง Palm Line Extraction and Matching for Personal Authentication

งานวิจัยนี้เขียนโดย Xiangqian Wu, David Zhang and Kuanquan Wang, IEEE Transactions on Systems, Man and Cybernetics, vol. 36, no. 5, September 2006. pp. 978-987. [6] จะกล่าวถึงการแยกเส้นลายมือออกจากภาพฝ่ามือและการระบุตัวบุคคลเช่นกัน โดยเนื้อหาที่ปริญญานิพนธ์นี้จะนำมาใช้ในการศึกษาคือขั้นตอนการแยกเส้นลายมือออกจากภาพฝ่ามือ สำหรับในหัวข้อนี้จะแยกนำเสนอในสองหัวข้อย่อยคือ หัวข้อแรกจะนำเสนอเกี่ยวกับขั้นตอนการปฏิบัติที่ได้จากงานวิจัย และหัวข้อที่สองนำเสนอเกี่ยวกับขั้นตอนในการปฏิบัติจริงที่นำไปใช้ในการเขียนโปรแกรม

3.5.2.1 ขั้นตอนการปฏิบัติที่ได้จากงานวิจัย

งานวิจัยนี้จะมีการใช้สมการเก๊าเซี้ยน (Gaussian) เป็นหนึ่งในขั้นตอนของการหาเส้นลายมือบนฝ่ามือ โดยสมการเก๊าเซี้ยนสามารถแสดงได้ดังสมการที่ (3.10)

$$G_{\sigma s} = exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3.10}$$

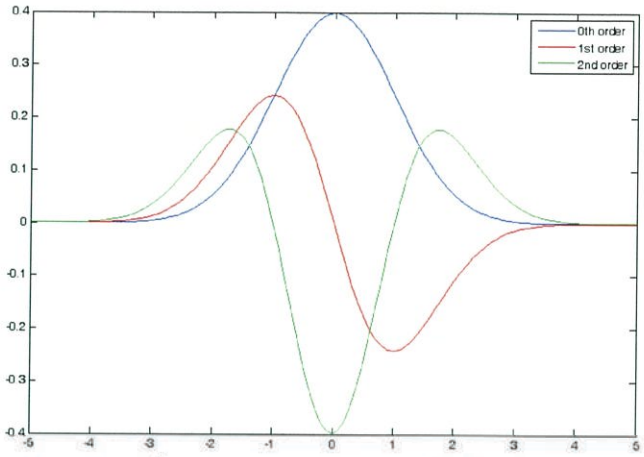
สมการอนุพันธ์อันดับที่หนึ่ง ของสมการเก๊าเซี้ยนสามารถแสดงได้ดังสมการที่ (3.11)

$$G'_{\sigma d} = -\frac{x}{\sigma^2} exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3.11}$$

และสมการอนุพันธ์อันดับที่สอง ของสมการเก๊าเซี้ยนสามารถแสดงได้ดังสมการที่ (3.12)

$$G''_{\sigma d} = \left(\frac{x^2-\sigma^2}{\sigma^4}\right) exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3.12}$$

ทั้ง 3 สมการเมื่อนำมาเขียนกราฟจะสามารถแสดงตัวอย่างลักษณะของกราฟได้ดังรูปที่ 3.30



รูปที่ 3.30 กราฟของสมการเก๊าเซี้ยน สมการอนุพันธุ์อันดับที่หนึ่งและสองของสมการเก๊าเซี้ยน [15]

การตรวจหาเส้นลายมือ (Palm Lines Detection) ในงานวิจัยนี้ จะใช้การตรวจจับเส้นขอบภาพในทิศทางที่แตกต่างกัน (Different Direction) โดยเราจะเรียกเส้นที่ถูกตรวจจับในทิศทาง θ ว่า เส้นทิศทาง θ องศา (θ-Directional Lines) และเรียกตัวตรวจจับที่ตรวจจับเส้นในทิศทาง θ องศา ว่าตัวตรวจจับเส้นในทิศทาง θ องศา (θ-Directional Lines Detectors)

การตรวจหาเส้นลายมือเพื่อแยกเส้นลายมือออกจากฝ่ามือเริ่มจาก สมมติให้ $I(x,y)$ แทนรูปภาพ จากนั้นใช้สมการเก๊าเซี้ยน (G_s) ในสมการที่ (3.10) คอนโวลูชั่นกับรูปภาพ I เพื่อกำจัดนอยซ์ (Noise) หรือสิ่งที่ไม่ใช่เส้นลายมือจากภาพ สามารถแสดงการคอนโวลูชั่นได้ดังสมการที่ (3.13)

$$I_s = I * G_{\sigma s} \tag{3.13}$$

เมื่อ I_s คือภาพที่ผ่านการคอนโวลูชั่นกับสมการเก๊าเซี้ยนแล้ว จากนั้นนำสมการอนุพันธุ์อันดับที่หนึ่ง (3.11) และสมการอนุพันธุ์อันดับที่สอง (3.12) ของสมการเก๊าเซี้ยนมาทำการทรานสโพส (Transpose) ซึ่งเป็นการจัดเรียงในทิศทางแนวตั้ง (Vertical) หรือสลับแถวและหลัก จากนั้นนำสมการที่สลับแถวและหลักแล้วมาคอนโวลูชั่นกับภาพ I_s แสดงได้ดังสมการที่ (3.14) และ (3.15)

$$\begin{aligned} I_0^1 &= I_s * (G'_{\sigma d})^T = (I * G_{\sigma s}) * (G'_{\sigma d})^T \\ I_0^1 &= I * (G_{\sigma s} * (G'_{\sigma d})^T) = I * H_0^1 \end{aligned} \tag{3.14}$$

$$\begin{aligned} I_0^2 &= I_s * (G''_{\sigma d})^T = (I * G_{\sigma s}) * (G''_{\sigma d})^T \\ I_0^2 &= I * (G_{\sigma s} * (G''_{\sigma d})^T) = I * H_0^2 \end{aligned} \tag{3.15}$$

เมื่อ I_0^1 และ I_0^2 คือภาพ I ที่ผ่านการคอนโวลูชั่นกับ H_0^1 และ H_0^2 ตามลำดับ โดยสามารถเขียน H_0^1 และ H_0^2 ในรูปสมการได้ดังสมการที่ (3.16) และ (3.17)

$$H_0^1 = G_s * {G'_d}^T \tag{3.16}$$

$$H_0^2 = G_s * G_d''^T \tag{3.17}$$

เมื่อ T คือทรานสโพส และ H_0^1 และ H_0^2 คือตัวตรวจจับเส้นในแนวนอน หรือทิศทาง 0 องศา (0-Directional Lines Detectors) เพื่อตรวจหาเส้นขอบภาพในแนว 0 องศา

เส้นในแนวนอนสามารถหาได้โดยการหาซีโร่ครอสพ้อยท์ (Zero Cross Point) คือการหาจุดที่ผ่านค่า 0 ของ I_0^1 ในทิศทางแนวตั้ง โดยค่าซีโร่ครอสพ้อยท์ของ I_0^1 นี้ยังมีความสัมพันธ์กับ I_0^2 ด้วย ดังนั้นจะนำภาพ I_0^1 และ I_0^2 มาเข้าเงื่อนไขในสมการที่ (3.18) และ (3.19) เพื่อหาเส้นขอบ

$$L_0^1(x,y) = \begin{cases} I_0^2(x,y), & \text{if } I_0^1(x,y) = 0 \text{ or } I_0^1(x,y) \times I_0^1(x,y+1) < 0 \\ 0 & , \text{ otherwise} \end{cases} \tag{3.18}$$

เมื่อ $L_0^1(x,y)$ คือ $I_0^2(x,y)$ ที่ผ่านเงื่อนไขในสมการที่ (3.18) โดยลักษณะของเส้นขอบจะมี 2 ลักษณะ คือลักษณะที่เป็นยอด (Peak) และมีลักษณะเป็นหุบ (Valley) จากภาพ $L_0^1(x,y)$ ค่าบวกจะแสดงลักษณะขอบภาพที่เป็นหุบ และค่าลบจะแสดงลักษณะขอบภาพที่เป็นยอด โดยเส้นลายมือที่ต้องการจะต้องมีลักษณะเป็นหุบ ดังนั้นจึงกำหนดเงื่อนไขดังสมการที่ (3.18) โดยนำภาพ $L_0^1(x,y)$ ไปเข้าเงื่อนไขที่กำหนดไว้ในสมการที่ (3.19) เพื่อเลือกค่า $L_0^1(x,y)$ ให้เหลือเพียงค่าบวก ซึ่งเป็นค่าที่แสดงถึงหุบหรือเส้นลายมือ

$$L_0^2(x,y) = \begin{cases} L_0^1(x,y), \text{ if } L_0^1(x,y) > 0 \\ 0 & , \text{ otherwise} \end{cases} \tag{3.19}$$

เมื่อ $L_0^2(x,y)$ คือ $L_0^1(x,y)$ ที่เหลือเพียงค่าบวก จากนั้นนำภาพ $L_0^2(x,y)$ ที่ได้มาสเกลค่าใหม่ให้มีค่าตั้งแต่ 0 ถึง 1 โดยใช้สมการที่ (3.20)

$$G_1(x,y) = \frac{L_0^2(x,y) - Min}{Max - Min} \times 255 \tag{3.20}$$

เมื่อ $G_1(x,y)$ คือภาพที่ผ่านการสเกลลิ่งของภาพ $L_0^2(x,y)$ มาแล้ว จากนั้นนำมาตัดระดับเทรชโฮลเพื่อทำให้เป็นภาพสองระดับ และเหลือเพียงเส้นลายมือเท่านั้น โดยภาพที่ได้ออกมาจะเป็นการหาเส้นขอบภาพในแนว 0 องศา

จากนั้นทำการหาเส้นขอบภาพในทิศทาง 45, 90 และ 135 องศา วิธีการหาเส้นขอบภาพในมุมต่างๆ นั้นทำได้โดยการหมุนตัวตรวจจับเส้นในแนวนอน (Horizontal) หรือทิศทาง 0 องศา คือหน้าต่าง H_0^1 และ H_0^2 ให้เรียงไปในทิศทางที่ต้องการ แล้วนำผลที่ได้จากการหาเส้นขอบภาพในทิศทางต่างๆ มารวมกัน สุดท้ายจะได้เส้นลายมือบนฝ่ามือออกมา

การหมุนตัวตรวจจับเส้นในแนวนอนหรือหน้าต่าง H_0^1 และ H_0^2 จะใช้วิธีการเดียวกัน ในปริญญานิพนธ์ฉบับนี้จะนำเสนอวิธีการหมุนหน้าต่างทั้งสองนี้ทั้งหมด 3 วิธีการ

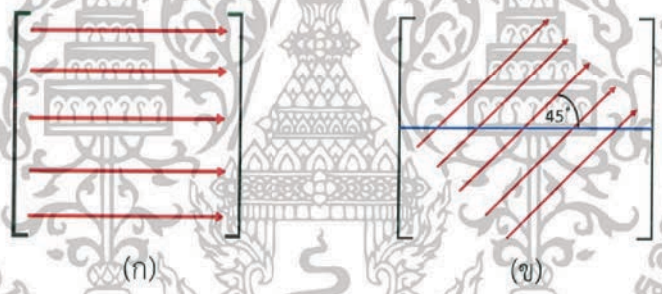
<u>วิธีการหมุนตัวตรวจจับเส้นในแนวนอนวิธีที่หนึ่ง</u>

ตัวอย่างหน้าต่างตัวตรวจจับเส้นในแนวนอนก่อนแสดงได้ในรูปที่ 3.31 เป็นตัวอย่างหน้าต่างขนาด 5 x 9 โดยสมมติให้ A1 ถึง I5 คือตัวเลขที่ได้จากการสร้างหน้าต่างตัวตรวจหาเส้นในแนวนอน

A1	B1	C1	D1	E1	F1	G1	H1	I1
A2	B2	C2	D2	E2	F2	G2	H2	I2
A3	B3	C3	D3	E3	F3	G3	H3	I3
A4	B4	C4	D4	E4	F4	G4	H4	I4
A5	B5	C5	D5	E5	F5	G5	H5	I5

รูปที่ 3.31 ตัวอย่างหน้าต่างตัวตรวจหาเส้นในแนวนอนขนาดหน้าต่าง 5 x 9

กำหนดให้ลักษณะการจัดเรียงหน้าต่างในรูปที่ 3.31 เป็นการจัดเรียงในแนวนอนหรือทิศทาง 0 องศา หรือแสดงทิศทางได้ในรูปที่ 3.32 (ก) จะจัดเรียงหน้าต่างนี้ใหม่โดยการจับวางให้หน้าต่างนี้เรียงในทิศทาง 45 องศา ซึ่งการจัดเรียงหน้าต่างให้มีทิศทาง 45 องศา จะอาศัยหลักการว่ามุม 45 องศาเป็นครึ่งหนึ่งของมุม 90 องศาซึ่งจะมีลักษณะตั้งฉาก ดังนั้นมุม 45 องศาจะสามารถแสดงได้ดังรูปที่ 3.32 (ข)



รูปที่ 3.32 ลักษณะการจับวางหน้าต่างให้เรียงในทิศทาง 45 องศา
(ก) หน้าต่างเรียงในทิศทาง 0 องศา
(ข) หน้าต่างเรียงในทิศทาง 45 องศา

ซึ่งลักษณะการจัดเรียงหน้าต่างในทิศทาง 45 องศา สามารถแสดงได้ในรูปที่ 3.33 โดยเมื่อจัดเรียงหน้าต่างให้อยู่ทิศทาง 45 องศาแล้ว หน้าต่างที่ถูกจัดเรียงใหม่นี้จะมีขนาดที่ใหญ่กว่าหน้าต่างที่อยู่ในทิศทาง 0 องศา เป็นหน้าต่างขนาด 9 x 9

0.0000	0.0000	0.0000	0.0000	I1	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	G1	H1	I2	0.0000	0.0000	0.0000
0.0000	0.0000	E1	F1	G2	H2	I3	0.0000	0.0000
0.0000	C1	D1	E2	F2	G3	H3	I4	0.0000
A1	B1	C2	D2	E3	F3	G4	H4	I5
0.0000	A2	B2	C3	D3	E4	F4	G5	H5
0.0000	0.0000	A3	B3	C4	D4	E5	F5	0.0000
0.0000	0.0000	0.0000	A4	B4	C5	D5	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	A5	B5	0.0000	0.0000	0.0000

รูปที่ 3.33 ตัวอย่างหน้าต่างตัวตรวจหาเส้นในทิศทาง 45 องศาขนาดหน้าต่าง 9 x 9

สำหรับตัวอย่างหน้าต่างที่ผ่านการจัดเรียงจริงในโปรแกรมสามารถแสดงได้ในรูปที่ 3.34 โดยเริ่มต้นสร้างหน้าต่างขนาด 5 x 9 และมีค่า σ_s = 3 และ σ_d = 0.8 แสดงได้ในรูปที่ 3.34 (ก) จากนั้นจัดเรียงค่าใหม่ในทิศทาง 45 องศา ซึ่งการจัดเรียงหน้าต่างโดยวิธีการจัดเรียงนี้ จะต้องสร้างหน้าต่างตัวตรวจหาเส้นในทิศทาง 45 องศาในโปรแกรม MATLAB ขึ้นมาเอง โดยการจับวางค่าที่อยู่ในหน้าต่าง 0 องศา ให้อยู่ในตำแหน่งต่างๆ ดังตัวอย่างในรูปที่ 3.33 จะได้ผลดังรูปที่ 3.34 (ข) และแสดงตัวอย่างผลของการหาเส้นขอบได้ในรูปที่ 3.34 (ค)

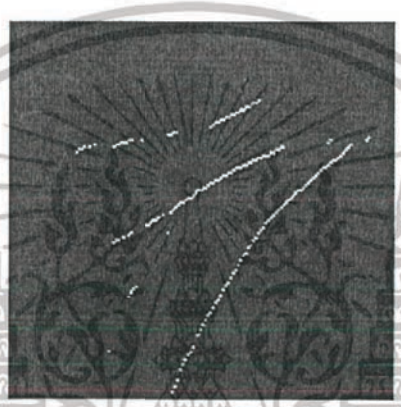
```
% การสร้างหน้าต่างตัวตรวจหาเส้นในทิศทาง 45 องศาในโปรแกรม MATLAB
% เป็นการสร้างเมตริกซ์ โดยการจับวางค่าที่อยู่ในหน้าต่าง 0 องศา ลงในตำแหน่งที่ต้องการ
% เมื่อกำหนดให้ H1_45deg คือ หน้าต่างตัวตรวจหาเส้นในทิศทาง 45 องศา
H1_45deg = [0.0000   0.0000   0.0000   0.0000   0.0564   0.0000   0.0000   0.0000   0.0000 ;
            0.0000   0.0000   0.0000   0.1099   0.0833   0.2941   0.0000   0.0000   0.0000 ;
            0.0000   0.0000   0.1373   0.1299   0.5728   0.4339   0.0000   0.0000   0.0000 ;
            0.0000   0.1099   0.1299   0.7154   0.6767   0.0000   0.0000  -0.2941   0.0000 ;
            0.0564   0.0833   0.5728   0.6767   0.0000   0.0000  -0.5728  -0.4339  -0.0564;
            0.0000   0.2941   0.4339   0.0000   0.0000  -0.7154  -0.6767  -0.1099  -0.0833;
            0.0000   0.0000   0.0000   0.0000  -0.5728  -0.6767  -0.1373  -0.1299   0.0000 ;
            0.0000   0.0000   0.0000  -0.2941  -0.4339  -0.1099  -0.1299   0.0000   0.0000 ;
            0.0000   0.0000   0.0000   0.0000  -0.0564  -0.0833   0.0000   0.0000   0.0000]
```

0.0564	0.0833	0.1099	0.1299	0.1373	0.1299	0.1099	0.0833	0.0564
0.2941	0.4339	0.5728	0.6767	0.7154	0.6767	0.5728	0.4339	0.2941
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-0.2941	-0.4339	-0.5728	-0.6767	-0.7154	-0.6767	-0.5728	-0.4339	-0.2941
-0.0564	-0.0833	-0.1099	-0.1299	-0.1373	-0.1299	-0.1099	-0.0833	-0.0564

(ก)

0.0000	0.0000	0.0000	0.0000	0.0564	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.1099	0.0833	0.2941	0.0000	0.0000	0.0000
0.0000	0.0000	0.1373	0.1299	0.5728	0.4339	0.0000	0.0000	0.0000
0.0000	0.1099	0.1299	0.7154	0.6767	0.0000	0.0000	-0.2941	0.0000
0.0564	0.0833	0.5728	0.6767	0.0000	0.0000	-0.5728	-0.4339	-0.0564
0.0000	0.2941	0.4339	0.0000	0.0000	-0.7154	-0.6767	-0.1099	-0.0833
0.0000	0.0000	0.0000	0.0000	-0.5728	-0.6767	-0.1373	-0.1299	0.0000
0.0000	0.0000	0.0000	-0.2941	-0.4339	-0.1099	-0.1299	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	-0.0564	-0.0833	0.0000	0.0000	0.0000

(ข)



(ค)

รูปที่ 3.34 ตัวอย่างหน้าต่างจริงที่จัดเรียงในโปรแกรม
(ก) หน้าต่างเรียงในทิศทาง 0 องศา ขนาด 5 x 9
(ข) หน้าต่างเรียงในทิศทาง 45 องศา ขนาด 9 x 9
(ค) ตัวอย่างเส้นขอบที่หาได้

ข้อดีของการใช้วิธีนี้ในการจัดเรียงหน้าต่างคือข้อมูลในหน้าต่างจะยังอยู่ครบโดยไม่สูญหายไป แต่ข้อเสียคือจะสามารถเรียงหน้าต่างได้เพียงในทิศทางที่ทราบองศาแน่นอนเท่านั้นคือ 0, 45, 90 และ 135 องศา หากต้องการจัดเรียงในทิศทางอื่นๆ เช่น 10, 30 หรือ 60 องศา จะไม่สามารถจัดเรียงได้ เนื่องจากมุมอื่นๆ จำเป็นต้องใช้ความละเอียดในการจัดวางมาก จึงได้นำเสนอวิธีการที่สอง ซึ่งสามารถหมุนหน้าต่างได้ทุกทิศทาง

<u>วิธีการหมุนตัวตรวจจับเส้นในแนวนอนวิธีที่สอง</u>

สำหรับวิธีการหมุนตัวตรวจจับเส้นในแนวทางที่สองนี้ จะใช้เครื่องมือในโปรแกรม MATLAB มาช่วยคือคำสั่ง imrotate เครื่องมือนี้สามารถที่จะหมุนวัตถุที่มีลักษณะเป็นเมตริกซ์ได้ทุกทิศทาง ในวิธีการที่สองนี้จะใช้ตัวอย่างหน้าต่างขนาด 5 x 9 และมีค่า σ_s = 3 และ σ_d = 0.8 เช่นเดิม สามารถแสดงได้ใน

รูปที่ 3.35 (ก) ใช้คำสั่ง imrotate หมุนหน้าต่างในรูปที่ 3.35 (ก) ไปในทิศทาง 45 องศา จะได้หน้าต่างขนาด 10 x 10 ดังแสดงในรูปที่ 3.35 (ข) และแสดงตัวอย่างผลของการหาเส้นขอบได้ในรูปที่ 3.35 (ค)

```
% การหมุนหน้าต่างตัวตรวจหาเส้นให้อยู่ทิศทาง 45 องศา
% กำหนดให้ H1_0deg คือ หน้าต่างตัวตรวจหาเส้นในทิศทาง 0 องศา
% กำหนดให้ H1_45deg คือ หน้าต่างตัวตรวจหาเส้นในทิศทาง 45 องศา
H1_45deg = imrotate(H1_0deg,45);
```

0.0564	0.0833	0.1099	0.1299	0.1373	0.1299	0.1099	0.0833	0.0564
0.2941	0.4339	0.5728	0.6767	0.7154	0.6767	0.5728	0.4339	0.2941
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-0.2941	-0.4339	-0.5728	-0.6767	-0.7154	-0.6767	-0.5728	-0.4339	-0.2941
-0.0564	-0.0833	-0.1099	-0.1299	-0.1373	-0.1299	-0.1099	-0.0833	-0.0564

(ก)

0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0564	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0833	0.2941	0.2941	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.1299	0.5728	0.4339	0.0000	-0.2941	0.0000
0.0000	0.0000	0.0000	0.1373	0.6767	0.6767	0.0000	-0.4339	-0.2941	-0.0564
0.0000	0.0000	0.1299	0.6767	0.7154	0.0000	-0.6767	-0.5728	-0.0833	0.0000
0.0000	0.0833	0.5728	0.6767	0.0000	-0.7154	-0.6767	-0.1299	0.0000	0.0000
0.0564	0.2941	0.4339	0.0000	-0.6767	-0.6767	-0.1373	0.0000	0.0000	0.0000
0.0000	0.2941	0.0000	-0.4339	-0.5728	-0.1299	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	-0.2941	-0.2941	-0.0833	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	-0.0564	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

(ข)

(ค)

รูปที่ 3.35 ตัวอย่างหน้าต่างที่หมุนโดยใช้คำสั่ง imrotate
(ก) หน้าต่างเรียงในทิศทาง 0 องศา ขนาด 5 x 9
(ข) หน้าต่างเรียงในทิศทาง 45 องศา ขนาด 10 x 10
(ค) ตัวอย่างเส้นขอบที่หาได้

แม้ข้อดีของการหมุนหน้าต่างด้วยวิธีการนี้จะทำให้สามารถหมุนหน้าต่างได้ในทุกทิศทาง แต่ข้อเสียคือค่าตัวเลขในบางตำแหน่งหลังจากที่หน้าต่างถูกหมุนจะสูญหายไป เช่น ในรูปที่ 3.35 (ก) ตัวเลขที่ถูกวงกลมไว้จะไม่ปรากฏในหน้าต่างในรูปที่ 3.35 (ข) เป็นต้น หากนำไปใช้ในการคอนโวลูชั่นกับรูปภาพอาจจะทำให้เกิดความผิดพลาดได้ ดังแสดงในรูปที่ 3.35 (ค) ที่เส้นขอบจะไม่สมบูรณ์เหมือนกับรูปที่ 3.34 (ค) ดังนั้นจึงจะนำเสนอวิธีการหมุนหน้าต่างวิธีการที่สาม

<u>วิธีการหมุนตัวตรวจจับเส้นในแนวนอนวิธีที่สาม</u>

สำหรับวิธีการหมุนตัวตรวจจับเส้นในแนวทางที่สามนี้ จะแตกต่างกับสองวิธีการที่ผ่านมา เนื่องจากสองวิธีการดังกล่าวจะใช้การหมุนหน้าต่างตัวตรวจจับเส้น แต่ในวิธีการที่สามนี้จะไม่ได้หมุนตัวตรวจจับเส้น แต่จะใช้วิธีการหมุนรูปภาพฝ่ามือไปในทิศทางต่างๆแทน โดยใช้คำสั่ง imrotate สำหรับตัวตรวจจับเส้นจะอยู่ในทิศทางเดิมตลอดคือทิศทาง 0 องศา ตัวอย่างการหมุนรูปภาพฝ่ามือไปในทิศทาง 45 องศา แสดงได้ในรูปที่ 3.36 (ก) และผลของการหาเส้นขอบที่หมุนภาพกลับมาในทิศทาง 0 องศาแล้ว แสดงได้ในรูปที่ 3.36 (ข)

```
% หมุนรูปภาพไปในทิศทาง 45 องศา
Palmprint_45deg = imrotate(Palmprint_Crop,45);
```

(ก) (ข)

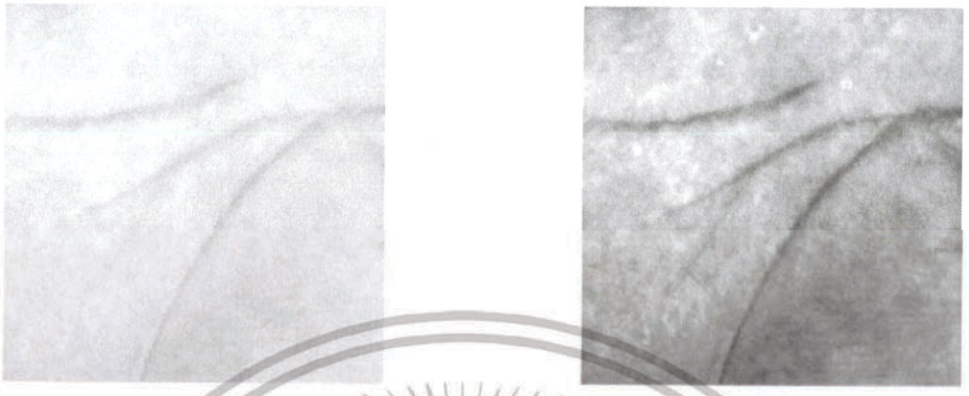
รูปที่ 3.36 ภาพฝ่ามือที่หมุนไปในทิศทาง 45 องศา และเส้นขอบที่หาได้
(ก) ตัวอย่างรูปภาพฝ่ามือที่หมุนไปในทิศทาง 45 องศา
(ข) ตัวอย่างเส้นขอบที่หาได้และหมุนกลับมาที่มุม 0 องศาแล้ว

โดยวิธีการนี้ข้อมูลทุกอย่างของหน้าต่างตัวตรวจจับเส้นจะอยู่ครบทั้งหมด ไม่สูญหายไปไหน ทำให้สามารถตรวจจับเส้นหรือขอบภาพได้ถูกต้องมากที่สุดจากทั้งสามวิธีการ จึงจะใช้วิธีการที่สามนี้ในการตรวจจับเส้นลายมือบนฝ่ามือ

3.5.2.2 ขั้นตอนการปฏิบัติจริงในการเขียนโปรแกรม

สำหรับขั้นตอนในการปฏิบัติจริง การดำเนินการจะเริ่มจากการนำรูปภาพฝ่ามือที่ตัดออกมาจากภาพมือ ไปปรับปรุงภาพเพื่อให้ภาพมีความคมชัดมากขึ้นและเห็นเส้นลายมือได้ชัดเจนมากขึ้น

โดยการใช้คำสั่ง imadjust ซึ่งเป็นเครื่องมือในโปรแกรม MATLAB เป็นการปรับความเข้มแสงของภาพ เพื่อเน้นให้ภาพมีความแตกต่างของความสว่างมากขึ้น สามารถแสดงภาพฝ่ามือก่อนและหลังปรับปรุงภาพได้ในรูปที่ 3.37



(ก) (ข)

รูปที่ 3.37 ขั้นตอนการปรับปรุงภาพฝ่ามือ เป็นภาพจากการทดลอง
(ก) ภาพฝ่ามือต้นฉบับ
(ข) ภาพที่ผ่านการปรับปรุงภาพโดยใช้คำสั่ง imadjust

จากนั้นสร้างหน้าต่าง H_0^1 และ H_0^2 ซึ่งเป็นตัวตรวจจับเส้นในแนวนอน หรือทิศทาง 0 องศา (0-Directional Lines Detectors) เพื่อตรวจจับเส้นขอบภาพในแนว 0 องศา ซึ่งงานวิจัยที่ได้ศึกษามีการกำหนดขนาดหน้าต่างตัวตรวจจับเส้นไว้ที่ 5 x 9 และจากการทดลองพบว่าขนาดหน้าต่างที่ใช้ในการตรวจจับเส้นที่เหมาะสมคือขนาด 5 x 9 เช่นกัน จึงจะใช้หน้าต่างตัวตรวจจับเส้นขนาด 5 x 9 และจากการทดลองจะกำหนดให้ค่า $\sigma_s = 3$ และ $\sigma_d = 0.8$ สามารถแสดงหน้าต่าง H_0^1 และ H_0^2 ได้ดังรูปที่ 3.38

```
% สร้างสมการ Gaussian ขนาดหน้าต่าง 1 x 9
Gs_row = 1; % แนวนอน
Gs_column = 9; % แนวตั้ง
Gs_x_number = (Gs_column-1)/2; % จำนวนของ x ที่มีค่าบวก
Gs_x = -Gs_x_number; % จุดเริ่มนับค่า x จะเริ่มนับจาก x ที่มีค่าลบตัวแรก
Gs_std = 3; % ค่าซิกม่าของ Gaussian
for Gs_column_count = 1: Gs_column
        % สร้าง Gaussian
        Gs(Gs_row,Gs_column_count) = exp(-(Gs_x^2)/(2*(Gs_std^2)));
        Gs_x = Gs_x+1; % ตัว x ตัวถัดไป
end
% สร้างสมการอนุพันธ์อันดับที่ 1 ของ Gaussian ขนาดหน้าต่าง 1 x 5
G1d_row = 1; % แนวนอน
G1d_column = 5; % แนวตั้ง
G1d_x_start = (G1d_column-1)/2; % จำนวนของ x ที่มีค่าบวก
G1d_x = -G1d_x_start; % จุดเริ่มนับค่า x จะเริ่มนับจาก x ที่มีค่าลบตัวแรก
G1d_std = 0.8; % ค่าซิกม่าของ Gaussian first derivative
```

```
for G1d_column_count = 1: G1d_column
    % สร้างสมการอนุพันธ์อันดับที่ 1 ของ Gaussian
    G1d(G1d_row,G1d_column_count) = -(G1d_x/(G1d_std^2)) *
                                    (exp(-(G1d_x^2)/(2*(G1d_std^2))));
    G1d_x = G1d_x+1; % ตัว x ตัวถัดไป
end
% สร้างสมการอนุพันธ์อันดับที่ 2 ของ Gaussian ขนาดหน้าต่าง 1 x 5
G2d_row = 1; % แนวนอน
G2d_column = 5; % แนวตั้ง
G2d_x_start = (G2d_column-1)/2; % จำนวนของ x ที่มีค่ามาก
G2d_x = -G2d_x_start; % จุดเริ่มแม่ค่า x จะเริ่มแม่จาก x ที่มีค่าลบตัวแรก
G2d_std = 0.8; % ค่าซิกม่าของ Gaussian second derivative
for G2d_column_count = 1: G2d_column
    % สร้างสมการอนุพันธ์อันดับที่ 2 ของ Gaussian
    G2d(G2d_row,G2d_column_count) = (((G2d_x^2)-(G2d_std^2))/(G2d_std^4)) *
                                    (exp(-(G2d_x^2)/(2*(G2d_std^2))));
    G2d_x = G2d_x+1; % ตัว x ตัวถัดไป
end
% สร้างหน้าต่าง H1_0deg
G1d_T = transpose(G1d); % Transpose สลับแถว-สลับหลัก
H1_0deg =   G1d_T * Gs % เอา Gs มาคูณกับ Gd 1st-derivative ได้ H10
% สร้างหน้าต่าง H2_0deg
G2d_T = transpose(G2d); % Transpose สลับแถว-สลับหลัก
H2_0deg =   G2d_T * Gs; % เอา Gs มาคูณกับ Gd 2nd-derivative ได้ H20
```

0.0564	0.0833	0.1099	0.1299	0.1373	0.1299	0.1099	0.0833	0.0564
0.2941	0.4339	0.5728	0.6767	0.7154	0.6767	0.5728	0.4339	0.2941
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-0.2941	-0.4339	-0.5728	-0.6767	-0.7154	-0.6767	-0.5728	-0.4339	-0.2941
-0.0564	-0.0833	-0.1099	-0.1299	-0.1373	-0.1299	-0.1099	-0.0833	-0.0564

(ก)

0.1482	0.2186	0.2886	0.3409	0.3604	0.3409	0.2886	0.2186	0.1482
0.1654	0.2441	0.3222	0.3806	0.4024	0.3806	0.3222	0.2441	0.1654
-0.6424	-0.9477	-1.2512	-1.4781	-1.5625	-1.4781	-1.2512	-0.9477	-0.6424
0.1654	0.2441	0.3222	0.3806	0.4024	0.3806	0.3222	0.2441	0.1654
0.1482	0.2186	0.2886	0.3409	0.3604	0.3409	0.2886	0.2186	0.1482

(ข)

รูปที่ 3.38 หน้าต่าง H_0^1 และ H_0^2
(ก) หน้าต่าง H_0^1
(ข) หน้าต่าง H_0^2

แต่จากการศึกษางานวิจัยของ John Canny, “A Computational Approach to Edge Detection”, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6, NOVEMBER 1986, pp. 679-698 [4] ได้มีการใช้เทคนิคเพิ่มเติมคือ ก่อนที่จะนำค่าที่ได้จากการกระจายของสมการเก๊าเซี้ยน $(G_{\sigma s})$ ไปคอนโวลูชั่นกับค่าที่ได้จากสมการอนุพันธุ์อันดับที่หนึ่งของเก๊าเซี้ยน $(G'_{\sigma d})$ และสมการอนุพันธุ์อันดับที่สองของเก๊าเซี้ยน $(G''_{\sigma d})$ จะมีการแทรกจุดที่มีค่าสูงสุดของชุดข้อมูลที่ได้จากสมการเก๊าเซี้ยน $(G_{\sigma s})$ ลงไปในชุดข้อมูลของสมการเก๊าเซี้ยน $(G_{\sigma s})$ เพื่อช่วยในเรื่องของการเชื่อมต่อของเส้นขอบภาพ

```
% ขั้นตอนการแทรกจุดที่มีค่ามากที่สุด
Gs_max = Gs(Gs_row,Gs_x_number+1); % ค่าที่มากที่สุดของ Gaussian
Gs_max_insert = 2 ;% 2 คือจำนวนจุดที่จะแทรก
Gs_column_count = 1 ;% กำหนดจุดที่จะเริ่มแม็พ-ข้อมูลเดิม
Gs_column_new = 1 ;% กำหนดจุดที่จะเริ่มแม็พ-ข้อมูลใหม่
% เนื่องจากจะแทรกจุดในหน้าต่าง Gaussian ซึ่งเดิมมีจำนวน 9 ตัว ดังนั้นการวนซ้ำ 20 ครั้งจึงเพียงพอ
for count = 1: 20
    % สร้าง Gaussian ตัวใหม่ที่ผ่านการแทรกค่าสูงสุดแล้ว
    Gs_new(Gs_row,Gs_column_new) = Gs(Gs_row,Gs_column_count);
    if Gs_column_count == Gs_x_number+1 % ถึงตำแหน่งที่มีค่าสูงสุด
        % เพิ่มจำนวนจุดที่จะแทรก เมื่อครบแล้วจะออกจากลูป
        for Gs_max_count = 1 : Gs_max_insert
            Gs_column_new = Gs_column_new + 1; % แทรกที่ตำแหน่งถัดจากค่าสูงสุด
            Gs_new(Gs_row,Gs_column_new) = Gs_max; % Gaussian ตัวใหม่
        end
    end
    if Gs_column_count == Gs_column % ออกจากลูปเมื่อสร้างหน้าต่าง Gs_new เสร็จ
        break                       % Gs_column คือจำนวนค่าที่มีในหน้าต่าง Gaussian เดิม
    end
    Gs_column_count = Gs_column_count + 1 ; % เพิ่มจุดถัดไป
    Gs_column_new = Gs_column_new + 1; % เพิ่มจุดถัดไป
end
% ---------------------------------------------------------------
% เมื่อแทรกค่าสูงสุดเรียบร้อยแล้ว ขั้นตอนต่อไปคือนำไปคูณกับสมการอนุพันธ์อันดับที่ 1 และ 2
% ของ Gaussian
% ---------------------------------------------------------------
```

สามารถแสดงหน้าต่าง H_0^1 และ H_0^2 ที่ผ่านการแทรกจุดแล้วในรูปที่ 3.39 โดยมีการแทรกจุดสูงสุดเข้าไปอีก 2 จุด คือจุดที่มีกรอบสีแดงล้อมรอบไว้ ทำให้หน้าต่างมีขนาด 5 × 11

0.0564	0.0833	0.1099	0.1299	0.1373	0.1373	0.1373	0.1299	0.1099	0.0833	0.0564
0.2941	0.4339	0.5728	0.6767	0.7154	0.7154	0.7154	0.6767	0.5728	0.4339	0.2941
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-0.2941	-0.4339	-0.5728	-0.6767	-0.7154	-0.7154	-0.7154	-0.6767	-0.5728	-0.4339	-0.2941
-0.0564	-0.0833	-0.1099	-0.1299	-0.1373	-0.1373	-0.1373	-0.1299	-0.1099	-0.0833	-0.0564

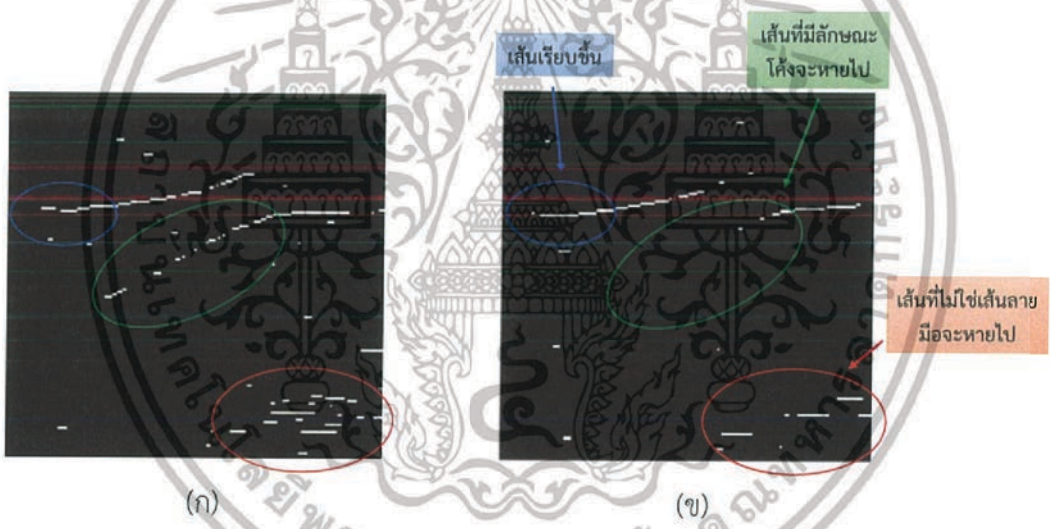
(ก)

0.1482	0.2186	0.2886	0.3409	0.3604	0.3604	0.3604	0.3409	0.2886	0.2186	0.1482
0.1654	0.2441	0.3222	0.3806	0.4024	0.4024	0.4024	0.3806	0.3222	0.2441	0.1654
-0.6424	-0.9477	-1.2512	-1.4781	-1.5625	-1.5625	-1.5625	-1.4781	-1.2512	-0.9477	-0.6424
0.1654	0.2441	0.3222	0.3806	0.4024	0.4024	0.4024	0.3806	0.3222	0.2441	0.1654
0.1482	0.2186	0.2886	0.3409	0.3604	0.3604	0.3604	0.3409	0.2886	0.2186	0.1482

(ข)

รูปที่ 3.39 หน้าต่าง H_0^1 และ H_0^2 ที่ผ่านการแทรกจุดแล้ว
(ก) หน้าต่าง H_0^1 ที่ผ่านการแทรกจุด
(ข) หน้าต่าง H_0^2 ที่ผ่านการแทรกจุด

โดยสามารถเปรียบเทียบผลการหาเส้นขอบก่อนการเพิ่มจุดสูงสุดและหลังเพิ่มจุดสูงสุดได้ในรูปที่ 3.40 ซึ่งจะเห็นได้ว่าหลังการเพิ่มจุดสูงสุดจะช่วยในเรื่องของการเชื่อมต่อเส้นให้มีความเรียบมากขึ้นดังที่วงกลมสีน้ำเงินไว้ และกำจัดเส้นที่ไม่ใช่เส้นลายมือได้ดังที่วงกลมสีแดงไว้ แต่วิธีการนี้ข้อเสียคือเส้นใดที่มีลักษณะโค้งก็จะถูกกำจัดออกไปด้วยดังที่วงกลมสีเขียวไว้



รูปที่ 3.40 ผลการหาเส้นขอบก่อนการเพิ่มจุดสูงสุดและหลังเพิ่มจุดสูงสุด
(ก) ผลการหาเส้นขอบก่อนการเพิ่มจุดสูงสุด
(ข) ผลการหาเส้นขอบหลังการเพิ่มจุดสูงสุด

การหาเส้นลายมือในทิศทาง 0 องศาจะเริ่มจากการนำหน้าต่าง H_0^1 และ H_0^2 ที่ทำการแทรกจุดแล้วไปคอนโวลูชั่นกับรูปฝ่ามือในรูปที่ 3.36 (ข) ตามสมการที่ (3.14) และ (3.15) จะได้ I_0' และ I_0'' ออกมา

```
% ขั้นตอนการ Convolution ภาพกับหน้าต่าง H10
PPC = Palmprint_Crop; % กำหนดตัวแปรใหม่
[PPC_x,PPC_y] = size(PPC); % หาขนาด Palm Print
[Gsnew_row,Gsnew_column]= size(Gs_new) % หาขนาดหน้าต่าง gaussian หลังจากแทรกจุด max
Gs_new_x = (Gsnew_column-1)/2; % จำนวนของ x ที่มีค่ามาก
% 'G1d_column' คือ จำนวนค่าของสมการอนุพันธ์อันดับที่ 1 ของ Gaussian (หน้าต่าง H10 ในแนวตั้ง)
% 'G1d_x_start' คือ จำนวนของ x ที่มีค่ามากของสมการอนุพันธ์อันดับที่ 1 ของ Gaussian
%
% การ Convolution ภาพกับหน้าต่าง H10
% เข้าพิกเซลรูปในแนวตั้ง โดยเว้นระยะเพื่อป้องกันการชนขอบ
for x = G1d_x_start+1: PPC_x-G1d_x_start
    % เข้าพิกเซลรูปในแนวนอน โดยเว้นระยะเพื่อป้องกันการชนขอบ
    for y = Gs_new_x+1: PPC_y-Gs_new_x
        G1d_count = G1d_x_start; % ใช้กำหนดจุดภาพที่จะเริ่ม Convolution ในแนวตั้ง
        for c_row = 1 : G1d_column % เข้าจุดหน้าต่าง H10 (แนวตั้ง)
            Gs_new_count = Gs_new_x; % ใช้กำหนดจุดภาพที่จะเริ่ม Convolution ในแนวนอน
            for c_column = 1 : Gsnew_column % เข้าจุดหน้าต่าง H10 (แนวนอน)
                % Convolution ภาพกับหน้าต่าง H10 ทีละจุด
                % โดยเรียงจากซ้ายไปขวา และจากบนลงล่าง
                H1_0deg_conv(c_row,c_column) = H1_0deg(c_row,c_column)*
                                               PPC(x-G1d_count,y-Gs_new_count);
                Gs_new_count = Gs_new_count - 1; % ใช้เลื่อนจุดภาพในแนวนอน
            end
        G1d_count = G1d_count - 1; % ใช้เลื่อนจุดภาพในแนวตั้ง
        end
        ppE_I1_0deg(x,y)  = sum(sum(H1_0deg_conv)); % ภาพ I1_0deg
    end
end
% ---------------------------------------------------------------------------
% การคอนโวลูชัน H20 กับภาพ Palm Print เพื่อหาภาพ I2_0deg หรือตัวแปร 'ppE_I2_0deg'
% ใช้วิธีการเดียวกันกับการคอนโวลูชัน H10 กับภาพ Palm Print
% ---------------------------------------------------------------------------
```

แล้วนำ I_0' และ I_0'' ไปคำนวณในสมการที่ (3.18) ได้ L_0^1 และนำค่าดังกล่าวไปคำนวณในสมการที่ (3.19) จะได้ L_0^2 ออกมา ภาพที่ได้ออกมาจะนำมาสเกลค่าใหม่ให้มีค่าตั้งแต่ 0 ถึง 1 โดยใช้สมการที่ (3.20) จะได้ผลลัพธ์ออกมาดังรูปที่ 3.41 (ก)

```
% หา L1_0deg โดยใช้ภาพ I1_0deg และ I2_0deg
[ppE_x,ppE_y] = size(I1_0deg); % หาขนาดภาพ I1_0deg
for x = 1: ppE_x-2
    for y = 1: ppE_y
        if I1_0deg(x,y) == 0 || (I1_0deg(x,y)*I1_0deg(x+1,y)) < 0
           L1_0deg(x,y) = I2_0deg(x,y);
        else
           L1_0deg(x,y) = 0;
        end
    end
end
```

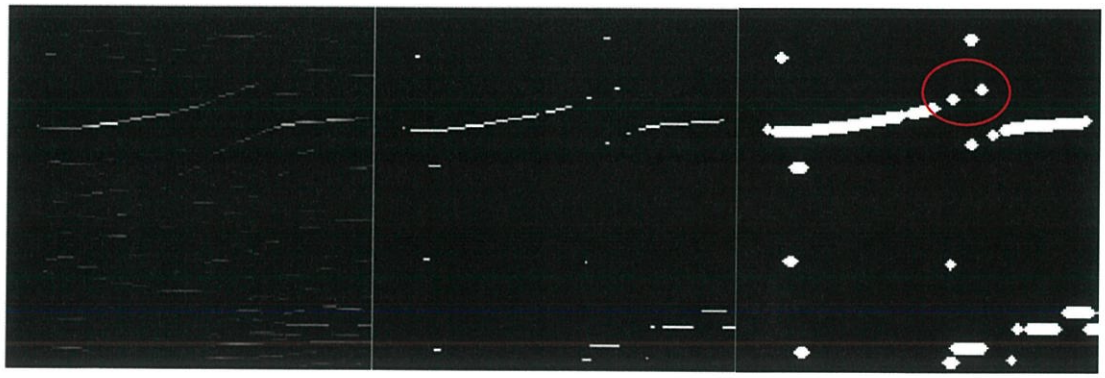
```
% หา L2_0deg โดยใช้ L1_0deg
[ppE_x,ppE_y] = size(L1_0deg); % หาขนาดภาพ L1_0deg
for x = 1: ppE_x
    for y = 1: ppE_y
        if L1_0deg(x,y) > 0
           L2_0deg(x,y) = L1_0deg(x,y);
        else
           L2_0deg(x,y) = 0;
        end
    end
end

% นำภาพ L2_0deg ไปสเกลให้มีค่าตั้งแต่ 0 ถึง 1
Max1 = max(max(L2_0deg)); % หาค่าสูงสุด
Min1 = min(min(L2_0deg)); % หาค่าต่ำสุด
% ภาพ L1_0deg กับภาพ L2_0deg มีขนาดเท่ากัน
for x = 2: ppE_x-1
    for y = 2: ppE_y-1
        % L2_0deg_scale คือ L2_0deg ที่ถูกสเกลค่าใหม่แล้ว
        L2_0deg_scale(x,y) = (((L2_0deg(x,y)-Min1)/(Max1-Min1)));
    end
end
```

จากนั้นนำรูปดังกล่าวไปตัดเทรชโฮลเพื่อให้เป็นภาพสองระดับดังรูปที่ 3.41 (ข) จะได้เส้นลายนิ้วมือในทิศทาง 0 องศาออกมา

```
% จากเส้นเอามาหาค่าเทรชโฮลและทำภาพระดับสีเทาให้กลายเป็นภาพสองระดับ
level_L2_0deg_scale = graythresh(L2_0deg_scale);
palmlines_0deg_bw = im2bw(L2_0deg_scale,level_L2_0deg_scale + 0.2);
```

และเพิ่มความหนาของเส้นโดยใช้คำสั่ง strel และ imdilate ร่วมกัน เพื่อให้จุดที่อยู่ใกล้เคียงกันเชื่อมต่อเป็นเส้นเดียวกัน โดยเส้นลายมือที่ผ่านการเพิ่มความหนาแล้วสามารถแสดงได้ในรูปที่ 3.41 (ค)

```
% การใช้คำสั่ง strel และ imdilate ร่วมกันเพื่อเพิ่มความหนาของเส้น
  se = strel('disk', 2); % Create a structuring element.
  Palm_Lines_0deg = imdilate(palmlines_0deg_bw,se);
% 'disk' คือ การเพิ่มความหนาเป็นวงกลมรอบจุดพิกเซล โดยเพิ่มความหนาขนาด 2 พิกเซล
% หากเพิ่มความหนาของเส้นมากกว่านี้ อาจทำให้เส้นลายมือเชื่อมกับเส้นที่ไม่ใช่เส้นลายมือได้
```



(ก) (ข) (ค)

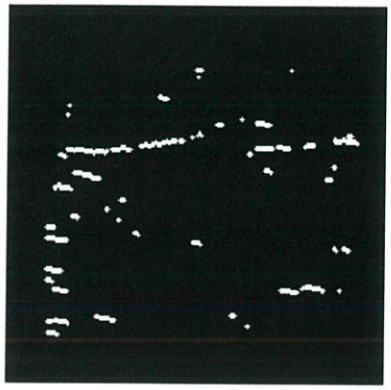
รูปที่ 3.41 เส้นลายมือในทิศทาง 0 องศาที่ตรวจจับได้
(ก) ภาพระดับสีเทา L_θ^2 ที่สเกลลิ่งแล้ว
(ข) เส้นลายนิ้วมือในทิศทาง 0 องศา เป็นภาพสองระดับ
(ค) เส้นลายนิ้วมือที่เพิ่มความหนาแล้ว

อย่างไรก็ตามจากรูปที่ 3.41 (ค) แม้จะเพิ่มความหนาของเส้นแล้ว แต่ในบางจุดเส้นลายมือก็ยังไม่สามารถที่จะเชื่อมกันได้ ดังที่วงกลมสีแดงไว้ ดังนั้นจะทำการหาเส้นลายนิ้วมือในทิศทางอื่นๆเพิ่มเติม โดยเป็นการหาในทิศทาง +10 องศา และ -10 องศา จากทิศทาง θ องศา ที่ต้องการตรวจจับ สาเหตุที่ต้องเพิ่มการตรวจในทิศทางดังกล่าวซ้ำแต่เป็นการหมุนไปในทิศทาง +10 องศา และ -10 องศา เนื่องจากเส้นที่ต้องการตรวจจับอาจไม่ได้อยู่ในทิศทางที่ตัวตรวจจับของทิศทางนั้นๆทำงานอยู่ ทำให้ไม่สามารถตรวจจับเส้นได้และทำให้เส้นลายมือที่ได้ไม่สมบูรณ์ ในที่นี้คือตรวจจับเส้นในแนวนอน หรือทิศทาง 0 องศา การเพิ่มทิศทางตรวจจับนี้จะทำให้เส้นลายมือที่ได้มีความสมบูรณ์มากขึ้น

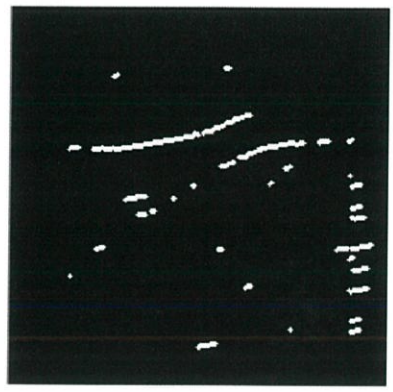
เริ่มจากการหมุนภาพฝ่ามือรูปที่ 3.37 (ข) ไป +10 และ -10 องศาด้วยการใช้คำสั่ง imrotate จากนั้นดำเนินการเช่นเดียวกับการหาเส้นลายมือในทิศทาง 0 องศา

```
Palm_Print_Rotate1 = imrotate(Palmprint_Crop,10); % หมุนรูปภาพไป +10 องศา
Palm_Print_Rotate2 = imrotate(Palmprint_Crop,-10); % หมุนรูปภาพไป -10 องศา
```

จากนั้นหมุนภาพเส้นลายมือที่ตรวจจับได้ให้กลับมาอยู่ในทิศทางเดิมคือ 0 องศา จะได้ผลลัพธ์ดังรูปที่ 3.42 (ก) และ (ข)



(ก)

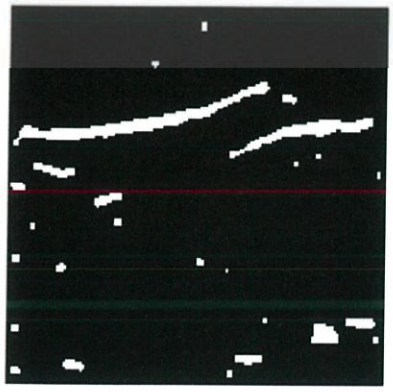


(ข)

รูปที่ 3.42 เส้นลายมือที่ตรวจจับได้จากทิศทางต่างๆ และหมุนกลับมาอยู่ในทิศทาง 0 องศาแล้ว
(ก) เส้นลายมือที่ตรวจจับได้จากทิศทาง +10 องศา
(ข) เส้นลายมือที่ตรวจจับได้จากทิศทาง -10 องศา

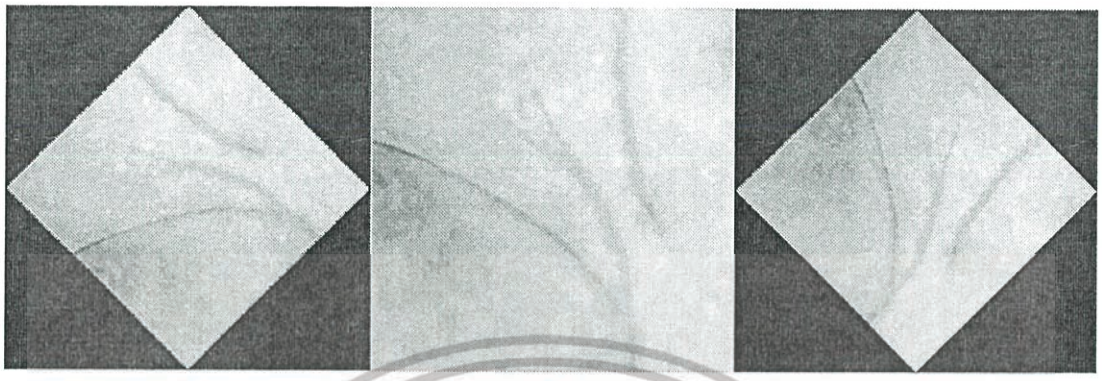
นำรูปภาพเส้นลายมือที่ได้จากทั้ง 3 ทิศทางมารวมกันโดยการดึงจุดพิกเซลที่มีค่าสูงสุดของแต่ละภาพมาใช้งาน จะได้ผลดังรูปที่ 3.43 เป็นรูปเส้นลายนิ้วมือในทิศทาง 0 องศา

```
% ขั้นตอนการดึงค่าสูงสุดจากทั้ง 3 ภาพมาใช้งาน
[Size_Row,Size_Column]=size(Palm_Lines_0deg); % หาขนาดภาพ
for row = 1 : Size_Row % ลำดับจุดพิกเซลในแนวแกน x
    for column = 1 : Size_Column % ลำดับจุดพิกเซลในแนวแกน y
        % ดึงค่า (Value) ในแต่ละจุดพิกเซลของภาพในทิศทางต่างๆขึ้นมาเพื่อเปรียบเทียบค่า
        Palm_Lines_0deg_Compare = [Palm_Lines_0deg(row,column)
                                   Palm_Lines_0deg_plus10deg(row,column)
                                   Palm_Lines_0deg_minus10deg(row,column)];
        % ดึงค่าสูงสุด (Max Value) มาใช้
        Palm_Lines_0deg_Combine(row,column) = max(Palm_Lines_0deg_Compare);
    end
end
% ---------------------------------------------------------------
%Palm_Lines_0deg_plus10deg คือเส้นลายมือที่หาในทิศทาง 0 องศาที่หมุนไปที่ +10 องศา
%Palm_Lines_0deg_minus10deg คือเส้นลายมือที่หาในทิศทาง 0 องศาที่หมุนไปที่ -10 องศา
% ---------------------------------------------------------------
```



รูปที่ 3.43 เส้นลายมือในทิศทาง 0 องศา ที่นำทิศทางต่างๆมารวมกัน

จากนั้นทำการหาเส้นลายนิ้วมือในทิศทาง 45, 90 และ 135 องศา โดยใช้การหมุนภาพไป 45, 90 และ 135 องศา โดยใช้คำสั่ง imrotate เช่นเดียวกัน สามารถแสดงผลการหมุนภาพฝ่ามือในทิศทาง 45, 90 และ 135 องศา ได้ในรูปที่ 3.44

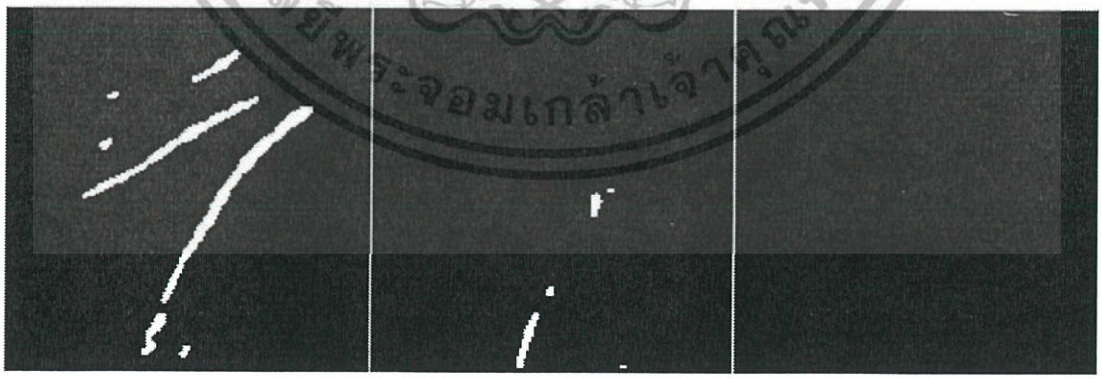


(ก) (ข) (ค)

รูปที่ 3.44 ภาพฝ่ามือในทิศทาง 45, 90 และ 135 องศา
(ก) ภาพฝ่ามือในทิศทาง 45 องศา
(ข) ภาพฝ่ามือในทิศทาง 90 องศา
(ค) ภาพฝ่ามือในทิศทาง 135 องศา

โดยหน้าต่างที่ใช้ตรวจจับเส้นขอบ จะใช้หน้าต่างตัวตรวจจับเส้นในแนวนอนหรือทิศทาง 0 องศา (0-Directional Lines Detectors) และในแต่ละทิศทางก็ยังใช้การหาเส้นลายมือเพิ่มเติมในทิศทาง +10 องศา และ -10 องศา จากทิศทาง θ องศา ที่ต้องการตรวจจับเช่นกัน

จากนั้นดำเนินการตรวจจับเส้นขอบด้วยวิธีการเดียวกับการตรวจจับเส้นขอบในทิศทาง 0 องศา จะได้ผลดังแสดงในรูปที่ 3.45 ซึ่งเป็นผลจากการตรวจจับเส้นในทิศทาง 45, 90 และ 135 องศา และได้หมุนภาพกลับมาในทิศทางเดิมคือ 0 องศาแล้ว



(ก) (ข) (ค)

รูปที่ 3.45 เส้นลายมือในทิศทาง 45, 90 และ 135 องศาที่ตรวจจับได้
(ก) เส้นลายมือในทิศทาง 45 องศา
(ข) เส้นลายมือในทิศทาง 90 องศา
(ค) เส้นลายมือในทิศทาง 135 องศา

สำหรับภาพ 3.45 (ค) ที่ไม่ปรากฏเส้นลายมือในทิศทาง 135 องศา เนื่องจากภาพมือที่เป็นภาพตัวอย่าง ไม่มีเส้นลายมือใดอยู่ในทิศทางดังกล่าว

จากนั้นนำภาพจากทั้งสี่ทิศทางคือ 0, 45, 90 และ 135 องศามารวมกันโดยการดึงจุดพิกเซลที่มีค่าสูงสุดของแต่ละภาพมาใช้งาน

```
% ขั้นตอนการดึงค่าสูงสุดจากแต่ละภาพมาใช้งาน
[Size_Row,Size_Column]=size(Palm_Lines_0deg_Compare); % หาขนาดภาพ
for row = 1 : Size_Row % ลำดับจุดพิกเซลในแนวแกน x
    for column = 1 : Size_Column % ลำดับจุดพิกเซลในแนวแกน y
        % ดึงค่า (Value) ในแต่ละจุดพิกเซลของภาพในทิศทางต่างๆขึ้นมาเพื่อเปรียบเทียบค่า
        Palm_Lines_Compare = [Palm_Lines_0deg_Compare(row,column)
                              Palm_Lines_45deg_Compare(row,column)
                              Palm_Lines_90deg_Compare(row,column)
                              Palm_Lines_135deg_Compare(row,column)];
        % ดึงค่าสูงสุด (Max Value) มาใช้จะได้ภาพเส้นลายมือ
        Palm_Lines_Combine(row,column) = max(Palm_Lines_Compare);
    end
end
% ------------------------------------------------------------------------
% Palm_Lines_0deg_Compare คือเส้นลายมือในทิศทาง 0 องศา
% Palm_Lines_45deg_Compare คือเส้นลายมือในทิศทาง 45 องศา
% Palm_Lines_90deg_Compare คือเส้นลายมือในทิศทาง 90 องศา
% Palm_Lines_135deg_Compare คือเส้นลายมือในทิศทาง 135 องศา
% ------------------------------------------------------------------------
```

จะได้เส้นลายมือบนฝ่ามือออกมาดังรูปที่ 3.46 (ก) จากนั้นทำเส้นลายมือให้บางลงเหลือความหนาหนึ่งพิกเซลด้วยวิธีที่นำเสนอในบทที่ 2 ดังรูปที่ 3.46 (ข)

(ก) (ข)

รูปที่ 3.46 เส้นลายมือที่รวมทุกทิศทางแล้วและผ่านการทำให้บาง
(ก) เส้นลายมือที่รวมทุกทิศทาง
(ข) เส้นลายมือที่ผ่านการทำให้บาง

อย่างไรก็ตามแม้ในรูปที่ 3.46 (ข) จะแสดงให้เห็นว่าภาพเส้นลายมือที่หาออกมามีความชัดเจนและต่อเนื่อง แต่เมื่อนำโปรแกรมและอัลกอริทึมนี้ไปทดลองกับภาพอื่นๆในฐานข้อมูล พบว่าภาพเส้นลายมือที่หาออกมาได้สมบูรณ์เช่นเดียวกับภาพที่ 3.46 (ข) นั้นมีจำนวนที่น้อย ภาพเส้นลายมือโดยส่วนมากที่ได้ออกมาจะได้เส้นลายมือที่ไม่มีความสมบูรณ์ คือไม่มีความชัดเจนของเส้นลายมือ ไม่สามารถระบุเส้นลายมือเส้นหลักได้ หรือเส้นลายมือที่หาออกมาได้ไม่มีความต่อเนื่อง โดยสามารถแสดงตัวอย่างภาพเส้นลายมือที่ไม่มีความสมบูรณ์ได้ในรูปที่ 3.47

รูปที่ 3.47 ตัวอย่างภาพเส้นลายมือที่ไม่มีความสมบูรณ์

สำหรับการแก้ไขปัญหาเส้นลายมือที่หาออกมาได้แต่ไม่มีความต่อเนื่อง หากเส้นลายมือที่ขาดออกจากกันมีระยะทางไม่มาก อาจแก้ไขโดยใช้วิธีการต่อเส้นลายมือทั้งสองด้านเข้าด้วยกัน แต่ถ้าหากเส้นลายมือที่ขาดออกจากกันมีระยะทางที่มาก อาจไม่สามารถใช้วิธีการต่อเส้นลายมือได้ ส่วนเส้นลายมือที่หาออกมาแล้วไม่มีความชัดเจนไม่สามารถระบุเส้นลายมือเส้นหลักได้ อาจต้องใช้การแยกเส้นลายมือด้วยวิธีการอื่น ดังนั้นจึงจะศึกษาแนวทางการแยกเส้นลายมือออกจากฝ่ามือด้วยวิธีการอื่น เพื่อหาแนวทางที่เหมาะสม

3.5.3 การแยกเส้นลายมือด้วยวิธี Canny

การแยกเส้นลายมือด้วยวิธี Canny จะใช้เครื่องมือที่มีอยู่แล้วในโปรแกรม MATLAB โดยใช้คำสั่ง edge ซึ่งมีรูปแบบการใช้งานดังนี้

```
Image_Output = edge(Image_Input, 'Canny',Threshold,Sigma)
```

มีคำอธิบายดังนี้

Image_Output คือ ผลลัพธ์จากการหาเส้นขอบด้วยวิธี Canny
Image_Input คือ รูปภาพที่ต้องการขอบภาพ
Canny คือ กำหนดวิธีหาขอบภาพด้วยวิธี Canny
Threshold คือ เลือกค่าตัวเลขเทรชโฮลที่ต้องการ โดยสามารถกำหนดค่าเทรชโฮลระดับเดียว หรือสองระดับก็ได้ หากต้องการใช้งานเทรชโฮลสองระดับมีรูปแบบการใช้ดังนี้ [Threshold_Low,Threshold_High] โดย Threshold_Low หมายถึง เทรชโฮลค่าต่ำ และ Threshold_High หมายถึงเทรชโฮลค่าสูง
Sigma คือ เลือกค่า σ ที่จะใช้ในสมการเก๊าเซี้ยน

สำหรับปริญญานิพนธ์ฉบับนี้จะใช้ค่า σ = 0.4 ซึ่งเป็นค่าที่ได้จากการทดลอง และหาค่าเทรชโฮลจากการใช้คำสั่ง graythresh และนำค่าเทรชโฮลนี้มาปรับแต่งให้เหมาะสม โดยเป็นการใช้ค่าเทรชโฮลระดับเดียว

```
% ค่าสั่งหาค่าเทรชโฮลโดยอัตโนมัติ
Threshold_Value = graythresh(Palmprint_Crop);
% การหาขอบภาพด้วยวิธี Canny
Palm_Lines = edge(Palmprint_Crop,'canny',Threshold_Value-0.5,0.4);
```

การหาเส้นขอบเริ่มจากการนำภาพฝ่ามือที่ตัดออกมาจากภาพมือมาปรับปรุงภาพโดยใช้คำสั่ง imadjust เพื่อทำให้ภาพมีความคมชัดและเห็นเส้นลายมือได้ชัดเจนมากขึ้น จากนั้นหาขอบภาพโดยใช้คำสั่ง edge วิธี Canny ในการหาเส้นขอบภาพแล้วจะได้ภาพออกมาดังรูปที่ 3.48

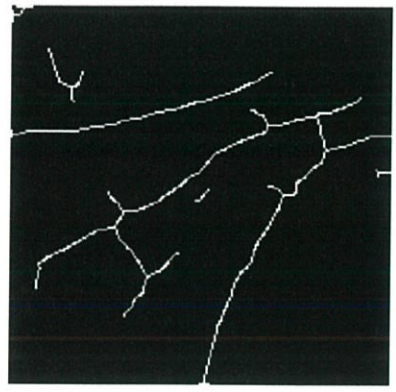
รูปที่ 3.48 ภาพฝ่ามือที่ปรับปรุงภาพแล้ว **รูปที่ 3.49** เส้นขอบที่หาออกมาได้จากคำสั่ง edge

จากภาพที่ 3.49 จะเห็นได้ว่าเส้นลายมือที่หาออกมาได้จะเป็นเส้นคู่ ซึ่งหลักการของการหาเส้นขอบภาพคือการตรวจหาการเปลี่ยนแปลงระดับความสว่างระหว่างจุดสองจุด ดังนั้นการที่เส้นลายมือบนฝ่ามือมีความหนา อาจจะทำให้หาเส้นขอบได้จากทั้งสองด้านของเส้นลายมือเพราะมีการเปลี่ยนแปลงระดับความสว่างสองจุด ต่อมาจึงเชื่อมเส้นขอบทั้งสองเข้าด้วยกัน โดยการใช้คำสั่ง strel และ imdilate ร่วมกัน

```
% การใช้คำสั่ง strel และ imdilate ร่วมกันเพื่อเพิ่มความหนาของเส้น
  se = strel('disk', 2); % Create a structuring element.
  Palm_Lines_Dilate = imdilate(Palm_Lines,se);
% 'disk' คือ การเพิ่มความหนาเป็นวงกลมรอบจุดพิกเซล โดยเพิ่มความหนาขนาด 2 พิกเซล
% ซึ่งเพียงพอแล้วในการเชื่อมเส้นลายมือให้เป็นเส้นเดียว
% ---------------------------------------------------------------------
% จากนั้นนำภาพเส้นลายมือที่เพิ่มความหนาแล้ว 'Palm_Lines_Dilate' ไปทำให้บางเหลือ
% เพียง 1 พิกเซล ได้ภาพ 'Palm_Lines' ออกมา
% ---------------------------------------------------------------------
```

การใช้คำสั่งดังกล่าวเพื่อทำให้เส้นขอบมีความหนามากขึ้นและเชื่อมเส้นขอบทั้งสองด้านเข้ากันเพื่อให้เป็นเส้นเดียวกัน สามารถแสดงผลของการเพิ่มความหนาเส้นลายมือได้ดังรูปที่ 3.50 (ก) จากนั้นจะทำเส้นขอบภาพให้บางลงเหลือเพียง 1 พิกเซลแสดงได้ดังรูปที่ 3.50 (ข) ได้เส้นลายมือจำนวน 3 เส้นออกมา

(ก)



(ข)

รูปที่ 3.50 เส้นลายมือที่ถูกเพิ่มความหนาและผ่านการทำให้บาง
(ก) เส้นลายมือที่ถูกเพิ่มความหนา
(ข) เส้นลายมือที่ผ่านการทำให้บาง

จากรูปที่ 3.50 (ข) เส้นลายมือที่หาออกมาได้ จะมีเส้นที่แยกออกมาจากเส้นลายมือเส้นหลักและมีเส้นที่ไม่ใช่เส้นลายมืออยู่ ซึ่งจุดประสงค์ของปริญญานิพนธ์ฉบับนี้คือการหาและจัดเก็บจุดพิกัดของเส้นลายมือเส้นหลักจำนวน 3 เส้นเท่านั้น อย่างไรก็ตามการจัดเก็บจุดพิกัดของเส้นลายมือ 3 เส้นหลัก สามารถกระทำได้โดยการใช้เงื่อนไขในขั้นตอนของการหาและจัดเก็บจุดพิกัดเส้นลายมือซึ่งจะกล่าวถึงในหัวข้อถัดไป เพื่อให้สามารถเก็บจุดพิกัดเฉพาะเส้นลายมือเส้นหลักเท่านั้น สามารถที่จะแสดงตัวอย่างเส้นลายมือ 3 เส้นหลักที่ถูกจัดเก็บได้ในรูปที่ 3.51 โดยเส้นสีแดงคือเส้นลายมือที่ถูกจัดเก็บเส้นที่หนึ่ง เส้นสีเขียวคือเส้นลายมือที่ถูกจัดเก็บเส้นที่สอง และเส้นสีน้ำเงินคือเส้นลายมือที่ถูกจัดเก็บเส้นที่สาม

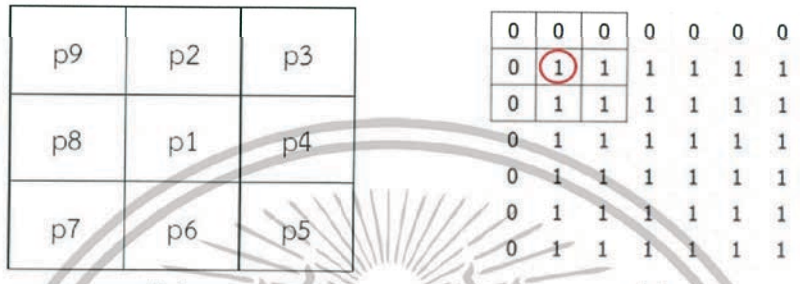
รูปที่ 3.51 เส้นลายมือ 3 เส้นหลักที่ถูกจัดเก็บ

จากรูปที่ 3.51 จะเห็นได้ว่าการจัดเก็บจุดพิกัดเส้นลายมือเส้นหลักทั้ง 3 เส้นสามารถจัดเก็บได้โดยไม่มีการขาดช่วง และไม่มีการจัดเก็บเส้นลายมือที่แยกออกมาจากเส้นลายมือเส้นหลักที่อยู่ในรูปที่ 3.50 (ข) ดังนั้นจึงจะใช้วิธีการแยกเส้นลายมือด้วยวิธี Canny ในการแยกเส้นลายมือออกจากฝ่ามือ ร่วมกับการใช้เงื่อนไขในขั้นตอนการหาจุดพิกัดของเส้นลายมือเพื่อจัดเก็บส้นลายมือเส้นหลักทั้ง 3 เส้น

3.6 การหาจุดพิกัดของเส้นลายมือ (Palm Lines Coordinate Detection)

สำหรับการจุดพิกัดของเส้นลายมือนั้น จากรูปที่ 3.51 จะเห็นได้ว่าเส้นลายมือจะมีเส้นหลักอยู่ 3 เส้นและในเส้นลายมือหลักแต่ละเส้นจะมีเส้นลายมือย่อยแตกแขนงออกมา อย่างไรก็ตามจุดประสงค์ของการเก็บจุดพิกัดในปริญญานิพนธ์ฉบับนี้จะเก็บเพียงเส้นลายมือ 3 เส้นหลักเท่านั้น

การหาจุดพิกัดเส้นลายมือจะเป็นการหาจุดพิกัดในแนวแกน x และ y เริ่มจากการหาจุดพิกัดที่มีค่าเป็น 1 เมื่อพบแล้วนำหน้าต่างขนาด 3 x 3 ในรูปที่ 3.52 (ก) มาวางทาบโดยให้จุด p1 อยู่บนจุดพิกเซลที่มีค่า 1 นั้น ดังรูปที่ 3.52 (ข)



รูปที่ 3.52 การวางหน้าขนาด 3 x 3 ลงบนภาพ
(ก) หน้าต่างขนาด 3 x 3
(ข) วางหน้าต่างขนาด 3 x 3 ลงบนจุดภาพแรกที่ตรวจพบ

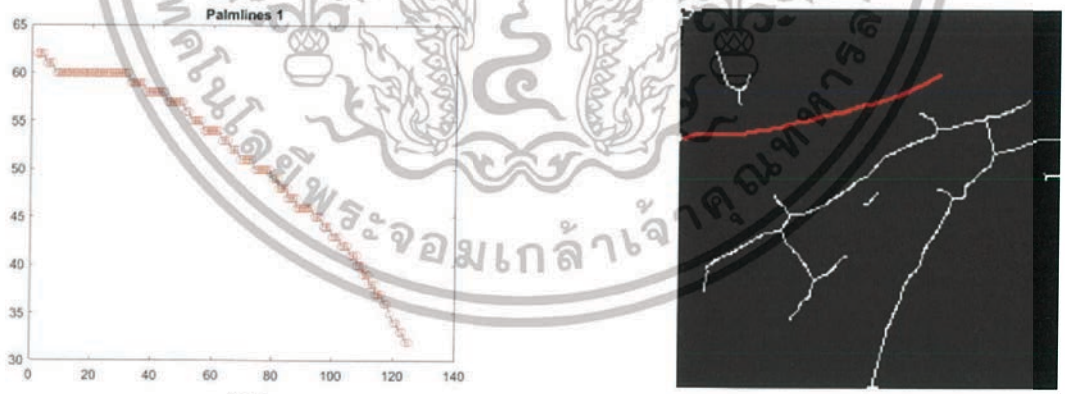
การหาเส้นลายมือเส้นที่หนึ่ง จะเริ่มจากการหาจุดพิกัดที่มีค่า 1 โดยหาในทิศทางจากบนลงล่าง และจากซ้ายไปขวา จุดพิกเซลที่มีค่า 1 ที่พบจุดแรกจะเป็นจุดเริ่มต้นของเส้นลายมือเส้นที่หนึ่ง เมื่อนำหน้าต่างขนาด 3 x 3 มาวางแล้วจะทำการเก็บค่าจุดพิกัดนั้นแยกกันในแนวแกน x และ y จากนั้นตรวจสอบจุดพิกัด p4, p2, p3 และ p5 ตามลำดับ

```
% โปรแกรมจัดเก็บเส้นลายมือเส้นที่หนึ่ง
% การเขียนโปรแกรมจะเหมือนกับโปรแกรมการจัดเก็บเส้นของลายมือ
% โดยเปลี่ยนแปลงตัวแปรและเงื่อนไขที่ใช้จัดเก็บจุดพิกัดเท่านั้น
% หาเส้นลายมือเส้นที่หนึ่ง
[m,n] = size(Palm_Lines); % หาขนาดของภาพเส้นลายมือ
false_detect_1st_pixel = 20; % กรณีเจอจุดพิกัดที่ไม่ใช่เส้นลายมือเส้นที่หนึ่งให้เริ่มหาค่า 1
                             % ใหม่ กำหนดให้วนซ้ำ 20 ครั้ง เนื่องจากภาพเส้นลายมือ
                             % มีจุดพิกัดที่ไม่ใช่เส้นลายมือในจำนวนที่ไม่มาก
false_detect_coordinate = 100; % จำนวนจุดพิกัดขั้นต่ำของแต่ละเส้นลายมือ หากน้อยกว่า
                               % ที่กำหนดแสดงว่าจุดพิกัดนั้นไม่ใช่เส้นลายมือ
% กรณีเจอจุดพิกัดที่ไม่ใช่เส้นลายมือให้เริ่มหาค่า 1 ใหม่
for find = 1 : false_detect_1st_pixel
    END = 1;% กำหนดค่าเริ่มต้นเพื่อใช้ในการตรวจสอบว่าได้จัดเก็บจุดพิกัดเส้นลายมือได้ถูกต้อง
%   หาจุดพิกเซลที่มีค่า 1 จุดแรกโดยการสแกนภาพจากบนลงล่างและจากซ้ายไปขวา
    for y = 2: n-1 % สแกนภาพในทิศทางซ้ายไปขวา
        for x = 2: m-1 % สแกนภาพในทิศทางบนลงล่าง
            if Palm_Lines(x,y) == 1  % ถ้าเจอพิกเซลที่มีค่า 1
               break % พบจุดพิกัดที่มีค่า 1 แล้วออกจากลูป x = 2: m-1
            end
```

```
            end
            if Palm_Lines(x,y) == 1  % ถ้าเจอพิกเซลที่มีค่า 1
               break % พบจุดพิกัดที่มีค่า 1 แล้วออกจากลูป y = 2: n-1
            end       % ซึ่งคือการออกจากลูปค้นหาจุดพิกัดที่มีค่า 1 จุดแรก
        end
    % การจัดเก็บเส้นลายมือเส้นที่หนึ่ง
    PL1_start(x,y) = Palm_Lines(x,y); % จุดเริ่มต้นของเส้นลายมือเส้นที่หนึ่ง
                                      % จัดเก็บในรูปของเมตริกซ์ที่มีขนาด x คูณ y
    a = 1; % เริ่มนับจุดพิกัดจุดแรก
    % จำนวนจุดพิกัดของเส้นลายมือจะมีจำนวนสูงสุดไม่เกิน 1000 จุด จึงให้วนซ้ำจำนวน 1000 ครั้ง
    for c = 1 : 1000
        PL1(x,y) = Palm_Lines(x,y); %ใช้ตัวแปร PL1 ในการจัดเก็บจุดพิกัดเส้นลายมือเส้นที่หนึ่ง
        Palm_Lines(x,y) = 10; % ลบจุดพิกัดเส้นลายมือเส้นที่หนึ่งที่จัดเก็บไปแล้ว
                              % โดยเปลี่ยนค่าจาก 1 เป็น 10
        xPL1(a) = x; % จัดเก็บตำแหน่งจุดพิกัด x ของเส้นลายมือเส้นที่หนึ่ง
        yPL1(a) = y; % จัดเก็บตำแหน่งจุดพิกัด y ของเส้นลายมือเส้นที่หนึ่ง
        a = a + 1; % เพิ่ม a ทีละ 1 เมื่อจัดเก็บจุดพิกัดแล้ว
        if (x == 1) || (x == m) || (y == n) || (y == 1) % ป้องกันการชนขอบของภาพ
            % ถ้าจำนวนจุดพิกัดน้อยกว่าที่กำหนด แสดงว่าไม่ใช่จุดพิกัดเส้นลายมือเส้นที่หนึ่ง
            if a < false_detect_coordinate
               clear PL1 PL1_start x y xPL1 yPL1 % ลบค่าตัวแปรที่ใช้ในการจัดเก็บจุดพิกัด
               break % ออกจากลูปจัดเก็บจุดพิกัดแล้วเริ่มหาจุดที่มีค่า 1 ใหม่
            end
               a = a-1; % ลดค่า a ลง 1 ในกรณีสิ้นสุดการจัดเก็บ เพราะเพิ่มค่าเกินจำเป็น
               PL1_end(x,y) = PL1(x,y); % จัดเก็บจุดสิ้นสุดเส้นลายมือเส้นที่หนึ่งในรูปเมตริกซ์
               disp(' END collected Palmlines 1 ')
               END = 1000;  % ใช้ในการตรวจสอบว่าได้จัดเก็บจุดพิกัดเส้นลายมือได้ถูกต้อง
            break % ออกจากลูปจัดเก็บจุดพิกัด
        end
        if Palm_Lines(x,y+1) == 1 % ตรวจสอบตำแหน่ง p4
            x = x;
            y = y+1;
        elseif Palm_Lines(x-1,y) == 1 % ตรวจสอบตำแหน่ง p2
            x = x-1;
            y = y;
        elseif Palm_Lines(x-1,y+1) == 1 % ตรวจสอบตำแหน่ง p3
            x = x-1;
            y = y+1;
        elseif Palm_Lines(x+1,y+1) == 1 % ตรวจสอบตำแหน่ง p5
            x = x+1;
            y = y+1;
    %  สิ้นสุดการจัดเก็บเส้นลายเส้นมือเส้นที่หนึ่ง เมื่อไม่พบพิกเซลที่มีค่า 1 ที่จุด p4 p2 p3 p5
        elseif (Palm_Lines(x,y+1) ~= 1) || (Palm_Lines(x-1,y) ~= 1) ||
               (Palm_Lines(x-1,y+1) ~= 1) || (Palm_Lines(x+1,y+1) ~= 1)
            % ถ้าจำนวนจุดพิกัดน้อยกว่าที่กำหนด แสดงว่าไม่ใช่จุดพิกัดเส้นลายมือเส้นที่หนึ่ง
            if a < false_detect_coordinate
               clear PL1 PL1_start x y xPL1 yPL1 % ลบค่าตัวแปรที่ใช้ในการจัดเก็บจุดพิกัด
               break % ออกจากลูปจัดเก็บจุดพิกัดแล้วเริ่มหาจุดที่มีค่า 1 ใหม่
            end
            a = a-1; % ลดค่า a ลง 1 ในกรณีสิ้นสุดการจัดเก็บ เพราะเพิ่มค่าเกินจำเป็น
            PL1_end(x,y) = PL1(x,y); % จัดเก็บจุดสิ้นสุดเส้นลายมือเส้นที่หนึ่งในรูปเมตริกซ์
            disp(' END collected Palmlines 1 ')
            END = 1000; % ใช้ในการตรวจสอบว่าได้จัดเก็บจุดพิกัดเส้นลายมือได้ถูกต้อง
            break % ออกจากลูปจัดเก็บจุดพิกัด
```

```
   end
end
if END == 1000 % ถ้า END คือ 1000 แสดงว่าจัดเก็บจุดพิกัดเส้นลายมือเส้นที่หนึ่งถูกต้อง
               % ถ้า END คือ 1 แสดงว่าไม่ใช่จุดพิกัดเส้นลายมือเส้นที่หนึ่ง หาจุดที่มีค่า 1 ใหม่
   break       % ออกจากโปรแกรมจัดเก็บจุดพิกัดเส้นลายมือเส้นที่หนึ่ง
end
end
% ---------------------------------------------------------------------
% การหาเส้นลายมือเส้นที่สองใช้หลักการและโปรแกรมเดียวกับการหาเส้นลายมือเส้นที่หนึ่ง
% ---------------------------------------------------------------------
% การหาเส้นลายมือเส้นที่สามใช้หลักการเดียวกับการหาเส้นลายมือเส้นที่หนึ่ง
% แต่เปลี่ยนเงื่อนไขจากการตรวจสอบภาพในตำแหน่ง p4, p2, p3 และ p5 ตามลำดับ
% เป็นการตรวจสอบภาพในตำแหน่ง p2, p4 และ p3 ตามลำดับ
% ---------------------------------------------------------------------
```

สาเหตุที่ต้องเก็บจุดพิกัดตามลำดับดังกล่าว เพราะเส้นลายมือเส้นแรกที่ต้องการเก็บจะมีลักษณะที่ขนานไปกับแนวนอนจึงจะตรวจสอบจุดบริเวณด้านขวาของหน้าต่างขนาด 3 x 3 ก่อน ส่วนที่ต้องตรวจสอบจุด p5 ด้วยแม้ว่าจุดดังกล่าวจะอยู่บริเวณขวาล่างของ p1 เนื่องจากเส้นลายมือเส้นที่หนึ่งนี้มีลักษณะที่ขนานไปทางแนวนอน ในบางครั้งอาจจะมีเส้นลายมือที่มีลักษณะเอียงในทิศทางลงได้ หากพบจุดใดมีค่าเป็น 1 ก็จะเคลื่อนหน้าต่าง 3 x 3 ไปที่จุดพิกเซลที่มีค่า 1 นั้น โดยให้จุด p1 อยู่บนจุดพิกเซลที่มีค่า 1 นั้น การเก็บค่าพิกัดจะเก็บอยู่ในรูป X_n และ Y_n เมื่อ X และ Y คือพิกัดในแนวแกน x และ y ตามลำดับ และ n คือจำนวนจุดพิกัดที่จัดเก็บ และดำเนินการซ้ำแบบเดิมจนกว่าจะเก็บจุดพิกัดเส้นลายมือเส้นที่หนึ่งได้หมด โดยเราจะทราบว่าเก็บจุดพิกัดหมดแล้วก็ต่อเมื่อตรวจสอบจุดพิกัดโดยรอบ p1 ตามเงื่อนไขแล้วไม่พบจุดพิกเซลที่มีค่า 1 อีก รูปที่ 3.53 (ข) จะแสดงถึงเส้นลายมือเส้นที่หนึ่ง (เส้นสีแดง) ที่ถูกจัดเก็บ



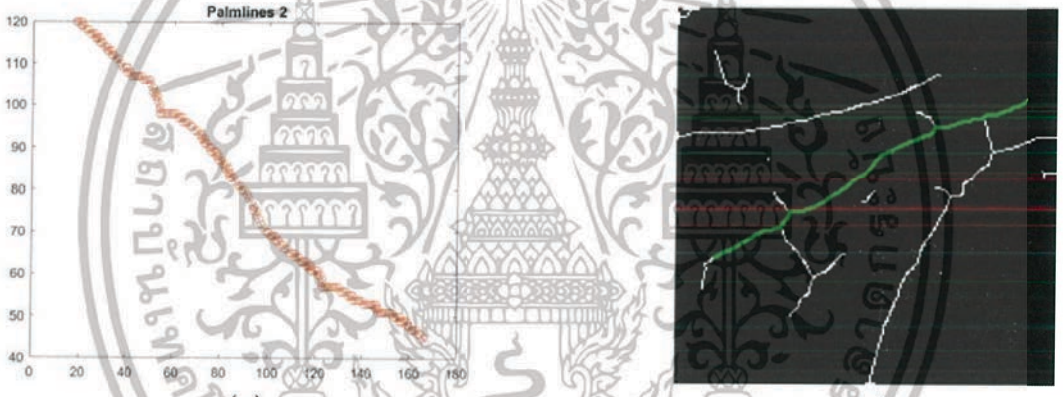
(ก) (ข)

รูปที่ 3.53 จุดพิกัดที่จัดเก็บของเส้นลายมือเส้นที่หนึ่ง (เส้นสีแดง)
(ก) พล็อทจุดพิกัดเส้นลายมือเส้นที่หนึ่ง
(ข) เส้นลายมือเส้นที่หนึ่งที่ถูกจัดเก็บ (เส้นสีแดง)

เมื่อหาพิกัดเส้นลายมือเส้นที่หนึ่งได้ครบแล้ว ก็จะทำการลบจุดพิกัดดังกล่าวในรูปที่ 3.50 (ข) เช่นเดียวกับขั้นตอนการเก็บพิกัดเส้นขอบนิ้วมือ เพื่อป้องกันการจัดเก็บข้อมูลที่ซ้ำซ้อนกัน เนื่องจาก

การเก็บเส้นลายมือเส้นที่สองจะเริ่มจากการหาจุดพิกัดที่มีค่า 1 ด้วยการหาจุดพิกัดจากบนลงล่างและซ้ายไปขวา หากไม่ลบจุดพิกัดเส้นลายมือเส้นที่หนึ่ง อาจทำให้เกิดการเก็บค่าซ้ำซ้อนได้

จากนั้นเมื่อลบเส้นลายมือเส้นที่หนึ่งแล้ว ก็จะหาเส้นลายนิ้วมือเส้นที่สอง ด้วยวิธีการเดิมคือหาจุดพิกัดที่มีค่าเป็น 1 ในทิศทางจากบนลงล่าง และจากซ้ายไปขวา เมื่อพบแล้วก็จะนำหน้าต่างขนาด 3 x 3 มาวางแล้วจะทำการเก็บค่าจุดพิกัดนั้นแยกกันในแนวแกน x และ y จากนั้นตรวจสอบจุดพิกัด p4, p2, p3 และ p5 ตามลำดับเช่นเดิม เนื่องจากเส้นลายมือเส้นที่สองนี้จะมีลักษณะที่คล้ายกับเส้นลายมือเส้นที่หนึ่งคือมีลักษณะที่ขนานไปกับแนวนอน แต่จากรูปที่ 3.54 (ข) จะเห็นได้ว่าเส้นลายมือเส้นที่สอง (เส้นสีเขียว) จะมีเส้นลายมือที่แยกออกมาจากเส้นลายมือเส้นหลักด้วย อย่างไรก็ตาม เนื่องจากเส้นลายมือเส้นแยก มักจะแยกออกจากเส้นลายมือหลักในทิศทางขึ้นข้างบนหรือบริเวณด้านบนของหน้าต่างขนาด 3 x 3 และแยกในทิศทางลงด้านล่างหรือบริเวณด้านล่างของหน้าต่างขนาด 3 x 3 ดังนั้นจากเงื่อนไขที่กำหนดให้ตรวจสอบจุด p4 เป็นจุดแรก คือการตรวจสอบบริเวณด้านขวาของหน้าต่างขนาด 3 x 3 จึงทำให้สามารถที่จะหลีกเลี่ยงการเก็บเส้นลายมือเส้นแยกได้ จากนั้นดำเนินการเช่นเดียวกับการเก็บพิกัดเส้นลายมือเส้นที่หนึ่ง เมื่อเก็บพิกัดครบแล้วก็จะลบจุดพิกัดเช่นเดียวกัน



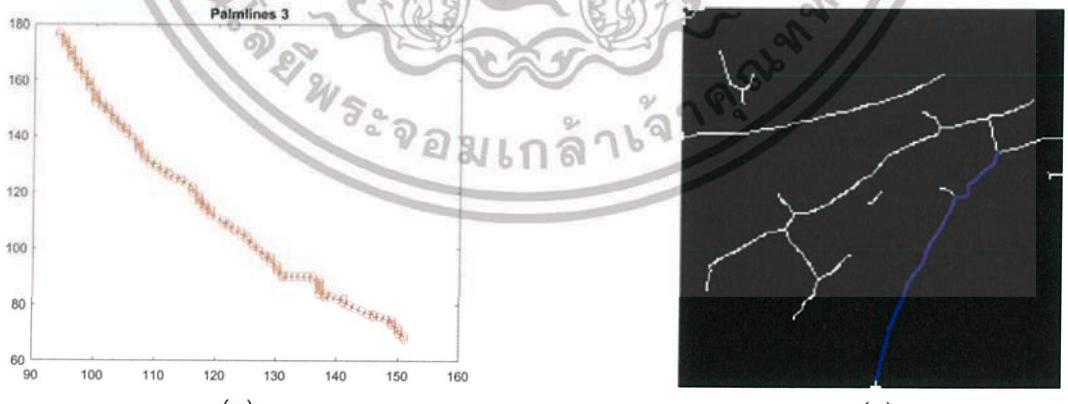
รูปที่ 3.54 จุดพิกัดที่จัดเก็บของเส้นลายมือเส้นที่สอง (เส้นสีเขียว)
(ก) พล็อทจุดพิกัดเส้นลายมือเส้นที่สอง
(ข) เส้นลายมือเส้นที่สองที่ถูกจัดเก็บ (เส้นสีเขียว)

สุดท้ายคือการจัดเก็บเส้นลายมือเส้นที่สาม สำหรับเส้นลายมือเส้นที่สามนี้จะมีวิธีการหาจุดพิกัดที่มีค่าเป็น 1 จะแตกต่างกับวิธีการหาเส้นลายมือทั้ง 2 เส้นที่ผ่านมา โดยจะค้นหาจุดพิกเซลที่มีค่าเป็น 1 ในทิศทางจากซ้ายไปขวา และจากล่างขึ้นบน เนื่องจากเส้นลายมือเส้นที่สามจะมีลักษณะเฉียงขึ้นไปทางขวา

```
% โปรแกรมค้นหาเส้นลายมือเส้นที่สาม
% หาจุดพิกเซลที่มีค่า 1 จุดแรกโดยการสแกนภาพจากซ้ายไปขวาและจากล่างขึ้นบน
  for cx = 2: m-1
      x = m - cx - 1; % สแกนภาพในทิศทางจากด้านล่างขึ้นด้านบน
      for y = 2: n-1 % สแกนภาพในทิศทางจากซ้ายไปขวา
```

```
            if Palm_Lines(x,y) == 1 % ถ้าเจอพิกเซลที่มีค่า 1
               break % พบจุดพิกัดที่มีค่า 1 แล้วออกจากลูป y = 2: n-1
            end
        end
        if Palm_Lines(x,y) == 1 % ถ้าเจอพิกเซลที่มีค่า 1
           break % พบจุดพิกัดที่มีค่า 1 แล้วออกจากลูป cx = 2: m-1
                 % ซึ่งคือการออกจากลูปค้นหาจุดพิกัดที่มีค่า 1 จุดแรก
        end
    end
% ------------------------------------------------------------------
% การเก็บจุดพิกัดเส้นลายมือเส้นที่สามใช้หลักการเดียวกับการหาเส้นลายมือเส้นที่หนึ่ง
% แต่เปลี่ยนเงื่อนไขจากการตรวจสอบภาพในตำแหน่ง p4, p2, p3 และ p5 ตามลำดับ
% เป็นการตรวจสอบภาพในตำแหน่ง p2, p4 และ p3 ตามลำดับ
% ------------------------------------------------------------------
```

เมื่อพบจุดพิกเซลที่มีค่า 1 แล้วก็จะดำเนินการเช่นเดียวกับการเก็บพิกัดเส้นลายมือเส้นที่หนึ่ง แต่จะตรวจสอบจุดพิกัด p2, p4 และ p3 ตามลำดับ สาเหตุที่ต้องเก็บจุดพิกัดตามลำดับดังกล่าว เนื่องจากเส้นลายมือเส้นที่สามที่ต้องการเก็บจะมีลักษณะที่เฉียงในทิศทาง 45 องศา หรือเฉียงขึ้นไปทางขวาจนถึงเกือบขนานกับแนวตั้ง จึงจะตรวจสอบจุดบริเวณด้านบน ด้านขวา และบริเวณมุมขวาบนของจุด p1 การตรวจสอบเพียง 3 จุดดังกล่าวจึงเพียงพอแล้วในการจัดเก็บเส้นลายมือ แต่จากรูปที่ 3.55 (ข) จะเห็นได้ว่าเส้นลายมือเส้นที่สาม (เส้นสีน้ำเงิน) จะมีเส้นลายมือที่แยกออกมาจากเส้นลายมือเส้นหลักด้วย อย่างไรก็ตามเนื่องจากเส้นลายมือเส้นแยก มักจะแยกออกจากเส้นลายมือเส้นหลักในทิศทางด้านข้างทั้ง 2 ด้าน หรือบริเวณด้านซ้ายและด้านขวาของหน้าต่างขนาด 3 x 3 ดังนั้นจากเงื่อนไขที่กำหนดให้ตรวจสอบจุด p2 เป็นจุดแรก ซึ่งคือการตรวจสอบบริเวณด้านบนของหน้าต่างขนาด 3 x 3 จึงทำให้สามารถที่จะหลีกเลี่ยงการจัดเก็บเส้นลายมือเส้นแยกได้ จากนั้นดำเนินการเช่นเดียวกับการเก็บพิกัดเส้นลายมือเส้นที่หนึ่ง เมื่อเก็บพิกัดครบแล้วก็จะลบจุดพิกัดเช่นเดียวกัน



(ก) (ข)

รูปที่ 3.55 จุดพิกัดที่จัดเก็บของเส้นลายมือเส้นที่สาม (เส้นสีน้ำเงิน)
(ก) พล็อตจุดพิกัดเส้นลายมือเส้นที่สาม
(ข) เส้นลายมือเส้นที่สามที่ถูกจัดเก็บ (เส้นสีน้ำเงิน)

อย่างไรก็ตามเงื่อนไขในการจัดเก็บจุดพิกัดเส้นลายมือยังไม่สมบูรณ์และมีความเหมาะสมเพียงพอ ดังจะเห็นได้จากรูปที่ 3.55 (ข) ที่เส้นลายมือที่ถูกจัดเก็บ (เส้นสีน้ำเงิน) ยังไม่สามารถที่จะจัดเก็บเส้นลายมือเส้นที่สามได้ทั้งหมด เนื่องจากมีเส้นลายมือเส้นแยกที่แยกออกจากเส้นลายมือเส้นหลักในทิศทางขึ้นด้านบนทำให้การจัดเก็บจุดพิกัดผิดพลาด ดังนั้นเงื่อนไขการจัดเก็บเส้นลายมือนี้ยังจำเป็นที่จะต้องปรับปรุงโดยอาจใช้แนวคิดว่า ขณะจัดเก็บเส้นลายมือหากพบจุดแยก ให้กำหนดเป็นจุดแยกที่หนึ่งและกำหนดว่าให้ทำการจัดเก็บเส้นลายมือในทิศทางใดก่อน จากนั้นพิจารณาว่าเส้นลายมือเส้นที่ถูกจัดเก็บไว้เป็นเส้นลายมือเส้นหลักหรือไม่ หากไม่ใช่ให้ทำการลบจุดพิกัดที่จัดเก็บไว้ แล้วเริ่มจัดเก็บเส้นลายมือใหม่จากจุดแยกเดิมที่พบคือจุดแยกที่หนึ่งในทางแยกที่เหลือ

เมื่อเก็บจุดพิกัดเส้นลายมือเส้นหลักครบทั้ง 3 เส้นแล้ว ก็จะนำจุดพิกัดของเส้นขอบนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นไปนอร์มอไลเซชันและอินเตอร์โพเลชันต่อไป

3.7 การนอร์มอลไลเซชันและอินเตอร์โพเลชันของนิ้วมือและลายเส้นมือบนฝ่ามือ

หลังจากที่ได้พิกัดของเส้นขอบนิ้วมือทั้ง 4 นิ้วได้แก่ นิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อย และเส้นลายมือทั้ง 3 เส้นแล้ว ก็จะนำจุดพิกัดที่ได้จัดเก็บไว้ไปเข้าสู่กระบวนการนอร์มอลไลเซชันคือการหาจุดศูนย์กลางของนิ้วและจุดศูนย์กลางของเส้นลายมือ และหมุนจุดพิกัดให้อยู่ในแนวแกนเดียวกัน เนื่องจากการวางภาพมือบนแผ่นรับอาจมีการหมุนมือไปในทิศทางที่แตกต่างกัน ทำให้ภาพมือที่ถ่ายออกมามีองศาของการวางมือที่ต่างกันไปด้วย เป็นผลให้จุดพิกัดของแต่ละนิ้วและแต่ละเส้นมีความแตกต่างกันและตำแหน่งพิกัดมีลักษณะที่แตกต่างกันไปด้วย ทำให้การหาเอกลักษณ์มีความผิดพลาดได้ จึงต้องหมุนจุดพิกัดให้มาอยู่ในแนวแกนเดียวกันเพื่อให้อยู่ในรูปแบบเดียวกัน

การนอร์มอลไลเซชันเริ่มจากการนำค่าพิกัดแนวแกน x และ y ของเส้นขอบนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นมาหาค่าเฉลี่ย หาค่าเบี่ยงเบน หาค่าเมตริกซ์ความแปรปรวนร่วม (Covariant Matrix) หาค่าไอเกน (Eigen value) และค่าไอเกนเวคเตอร์ (Eigen Vector) โดยการหาค่าเฉลี่ยของตำแหน่งพิกัด หาได้ดังสมการที่ (3.21) และ (3.22)

$$\bar{x}^{(k)} = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} x_i^{(k)} \tag{3.21}$$

$$\bar{y}^{(k)} = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} y_i^{(k)} \tag{3.22}$$

เมื่อ

$\bar{x}^{(k)}$ และ $\bar{y}^{(k)}$ คือ ค่าเฉลี่ยหรือจุดกึ่งกลางของจุดพิกัดของนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นในแนวแกน x และ y ตามลำดับ

$x_i^{(k)}$ และ $y_i^{(k)}$ คือ จุดพิกัดของนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นในแนวแกน x และ y ตามลำดับ

i คือ ตำแหน่งจุดพิกัดมีค่าตั้งแต่ 1, 2, 3 จนถึง N

k คือ นิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อย เมื่อ $k = 1, 2, 3, 4$ และเส้นลายมือเส้นที่หนึ่ง สอง และสาม เมื่อ $k = 5, 6, 7$

N คือ จำนวนจุดพิกัดทั้งหมดของนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นที่ถูกคำนวณ

จากนั้นหาค่าเบี่ยงเบน $(\mathbb{x}_i^{(k)}, \mathbb{y}_i^{(k)})$ จากจุดศูนย์กลางหรือค่าเฉลี่ยของจุดพิกัดของนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นได้จากสมการที่ (3.23)

$$\mathbb{x}_i^{(k)} = x_i^{(k)} - \bar{x}^{(k)} \quad \text{และ} \quad \mathbb{y}_i^{(k)} = y_i^{(k)} - \bar{y}^{(k)} \tag{3.23}$$

และหาค่าเมทริกซ์ความแปรปรวนร่วม (Covariant Matrix : $C^{(k)}$) ดังสมการที่ (3.24)

$$C^{(k)} = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} \begin{bmatrix} (\mathbb{x}_i^{(k)}, \mathbb{x}_i^{(k)}) & (\mathbb{x}_i^{(k)}, \mathbb{y}_i^{(k)}) \\ (\mathbb{y}_i^{(k)}, \mathbb{x}_i^{(k)}) & (\mathbb{y}_i^{(k)}, \mathbb{y}_i^{(k)}) \end{bmatrix} \tag{3.24}$$

หาค่าไอเกน (Eigen Value) และค่าไอเกนเวคเตอร์ (Eigen Vector) เพื่อนำมาใช้ในการปรับตำแหน่งของจุดพิกัดของนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นให้หมุนมาอยู่ในแนวแกนเดียวกัน ค่าไอเกนและค่าไอเกนเวคเตอร์สามารถหาได้จากสมการที่ (3.25)

$$C^{(k)} v^{(k)} = \lambda^{(k)} v^{(k)} \tag{3.25}$$

เมื่อ

$\lambda^{(k)}$ คือ ค่าไอเกน

$v^{(k)}$ คือ ค่าไอเกนเวคเตอร์

ในการหาค่าไอเกนและค่าไอเกนเวคเตอร์จะใช้คำสั่ง eig ซึ่งเป็นเครื่องมือในโปรแกรม MATLAB มีรูปแบบการใช้คำสั่งดังนี้

```
[Eigen_Vecter,Eigen_Value] = eig(Covariant_Matrix)
```

มีคำอธิบายดังนี้

Eigen_Vecter คือ ค่าไอเกนเวคเตอร์ที่หาออกมาได้

Eigen_Value คือ ค่าไอเกนที่หาออกมาได้

Covariant_Matrix คือ เมตริกซ์ความแปรปรวนร่วม (Covariant Matrix) จากสมการที่ (3.23)

เมื่อได้ค่าไอเกนและไอเกนเวคเตอร์มาแล้ว จะนำค่าเบี่ยงเบนของจุดพิกัดของเส้นขอบนิ้วมือและเส้นลายมือทั้งในแนวแกน x และ y มาคูณกับค่าไอเกนเวคเตอร์ (Eigen vector) เพื่อคำนวณหาจุดพิกัดอันใหม่ของเส้นขอบนิ้วมือและเส้นลายมือ ดังสมการที่ (3.26)

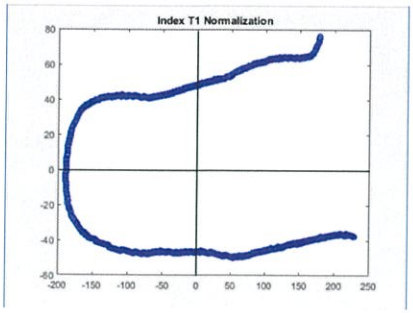
$$\mathbb{X}_i^{(k)} = \mathbb{x}_i^{(k)} v^{(k)} \quad \text{และ} \quad \mathbb{Y}_i^{(k)} = \mathbb{y}_i^{(k)} v^{(k)} \tag{3.26}$$

เมื่อ

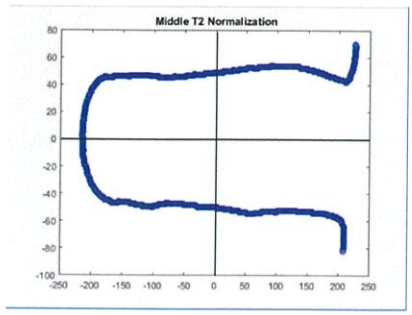
$\mathbb{X}_i^{(k)}$และ $\mathbb{Y}_i^{(k)}$ คือ จุดพิกัดอันใหม่ของเส้นขอบนิ้วมือและเส้นลายมือหรือจุดพิกัดที่ถูกนอร์มอลไลซ์แล้ว

```
% ตัวอย่าง : ขั้นตอนการนอร์มอลไลเซชันของน้ำชี้
% 'RxT1' คือจุดพิกัดเส้นขอบน้ำชี้ในแนวแกน x
% 'RyT1' คือจุดพิกัดเส้นขอบน้ำชี้ในแนวแกน y
% หาค่าเฉลี่ยของน้ำชี้
xT1_ave = round(sum(RxT1)/size(RxT1,2)); % ในแนวแกน x
yT1_ave = round(sum(RyT1)/size(RyT1,2)); % ในแนวแกน y
% หาค่าเบี่ยงเบนจากจุดศูนย์กลางหรือค่าเฉลี่ย และหาเมทริกซ์ความแปรปรวนร่วมของน้ำชี้
xT1_size = size(RxT1,2); % จำนวนจุดพิกัดของน้ำชี้ในแนวแกน x
yT1_size = size(RyT1,2); % จำนวนจุดพิกัดของน้ำชี้ในแนวแกน y
a = 1; % เริ่มเข้าจุดพิกัดจุดแรก
for c = 1 : xT1_size % xT1_size คือจำนวนจุดพิกัดของน้ำชี้
    % หาค่าเบี่ยงเบนจากค่าเฉลี่ยของน้ำชี้
    xT1_devia(a) = RxT1(a) - xT1_ave; % ในแนวแกน x
    yT1_devia(a) = RyT1(a) - yT1_ave; % ในแนวแกน y
    % ตัวแปรใน Covarient matrix
    xxT1_CoM(a) = xT1_devia(a) * xT1_devia(a); % ตำแหน่ง (x, x)
    xyT1_CoM(a) = xT1_devia(a) * yT1_devia(a); % ตำแหน่ง (x, y)
    yxT1_CoM(a) = yT1_devia(a) * xT1_devia(a); % ตำแหน่ง (y, x)
    yyT1_CoM(a) = yT1_devia(a) * yT1_devia(a); % ตำแหน่ง (y, y)
    a = a + 1; % จุดพิกัดจุดถัดไป
end
% หาเมทริกซ์ความแปรปรวนร่วมของน้ำชี้
Co_Matrix_T1 = (1/xT1_size) * [sum(xxT1_CoM) sum(xyT1_CoM);
                               sum(yxT1_CoM) sum(yyT1_CoM)];
% หาค่าไอเกนและไอเกนเวคเตอร์ของน้ำชี้
% 'EVec_T1' คือค่าไอเกนเวคเตอร์
% 'EVal_T1' คือค่าไอเกน
[EVec_T1,EVal_T1] = eig(Co_Matrix_T1);
% ค่าเฉลี่ยจุดพิกัดอันใหม่ของน้ำชี้
a = 1; % เริ่มเข้าจุดพิกัดจุดแรก
for c = 1 : xT1_size % xT1_size คือจำนวนจุดพิกัดของน้ำชี้
    % นำค่าเบี่ยงเบนของจุดพิกัดน้ำชี้ในแนวแกน x และ y มาคูณกับค่าไอเกนเวคเตอร์
    A = EVec_T1 * [xT1_devia(a);yT1_devia(a)];
    yT1_Norm(a) = A(1,1); % จุดพิกัดของน้ำชี้ในแนวแกน y ที่ถูกนอร์มอลไลซ์แล้ว
    xT1_Norm(a) = A(2,1); % จุดพิกัดของน้ำชี้ในแนวแกน x ที่ถูกนอร์มอลไลซ์แล้ว
    a = a + 1; % จุดพิกัดจุดถัดไป
end
%-------------------------------------------------------
% น้ำมือน้ำอื่นๆ และเส้นลายมือใช้วิธีการเดียวกันในการทำนอร์มอลไลเซชัน
%-------------------------------------------------------
```

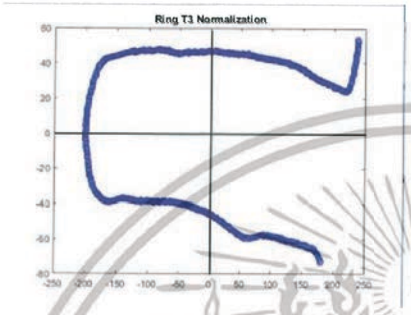
จะได้จุดพิกัดของนิ้วมือและเส้นลายมือที่ผ่านการนอร์มอลไลซ์ออกมาดังรูปที่ 3.56 และ 3.57



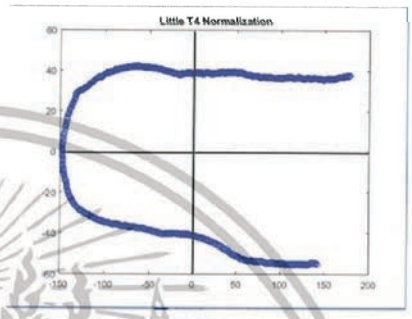
(ก)



(ข)

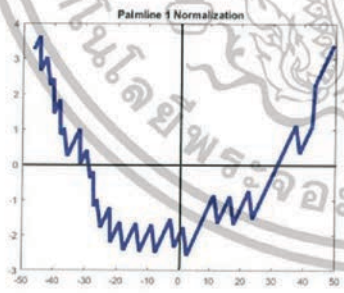


(ค)

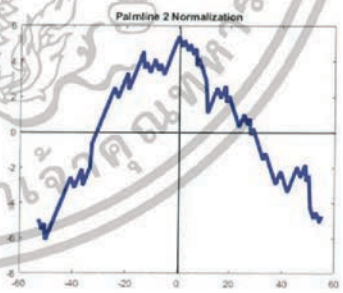


(ง)

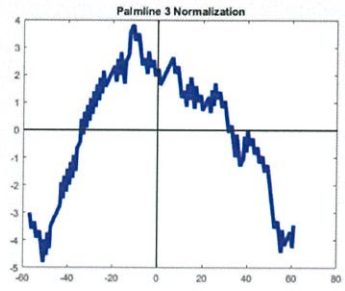
รูปที่ 3.56 นิ้วมือที่ผ่านการนอร์มอลไลซ์
(ก) นิ้วชี้ที่ผ่านการนอร์มอลไลซ์
(ข) นิ้วกลางที่ผ่านการนอร์มอลไลซ์
(ค) นิ้วนางที่ผ่านการนอร์มอลไลซ์
(ง) นิ้วก้อยที่ผ่านการนอร์มอลไลซ์



(ก)



(ข)



(ค)

รูปที่ 3.57 เส้นลายมือที่ผ่านการนอร์มอลไลซ์
(ก) เส้นลายมือเส้นที่หนึ่งที่ผ่านการนอร์มอลไลซ์
(ข) เส้นลายมือเส้นท่าสองที่ผ่านการนอร์มอลไลซ์
(ค) เส้นลายมือเส้นที่สามที่ผ่านการนอร์มอลไลซ์

เนื่องจากภาพมือและฝ่ามือแต่ละภาพจะมีจำนวนจุดพิกัดที่ไม่เท่ากัน มาจากการที่การถ่ายภาพแต่ละครั้งอาจจะมีระยะการถ่ายที่ไม่เท่ากัน เช่น ถ่ายภาพใกล้หรือไกล ทำให้ได้ขนาดภาพมือและฝ่ามือที่ไม่เท่ากัน อาจจะได้ภาพที่เล็กหรือใหญ่ที่มีขนาดแตกต่างกัน รวมถึงความสว่างและแสงเงาของบริเวณที่ถ่ายภาพและลักษณะการวางมือบนแผ่นราบ ก็มีผลทำให้จำนวนจุดพิกัดของภาพมือและฝ่ามือมีจำนวนไม่เท่ากัน อีกทั้งความยาวของนิ้วมือแต่ละนิ้วก็มีความยาวที่ไม่เท่ากัน เช่น นิ้วกลางที่จะมีความยาวมากกว่านิ้วก้อย หากนำภาพที่มีจำนวนจุดพิกัดไม่เท่ากันมาประมวลผล อาจทำให้เกิดความยุ่งยากในขั้นตอนการระบุตัวบุคคลและอาจจะเกิดความผิดพลาดขึ้นได้ ดังนั้นเพื่อความสะดวกในการประมวลผลจึงต้องมีการแทรกจุดพิกัดเพื่อให้นิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นมีจำนวนจุดพิกัดที่เท่ากัน การแทรกจุดจะใช้วิธีการอินเตอร์โพเลชัน คือการแทรกค่าจุดพิกัดระหว่างจุดพิกัดสองจุด เพื่อให้มีจำนวนจุดพิกัดตามที่เราต้องการ สามารถทำโดยใช้คำสั่ง interp1 ซึ่งเป็นเครื่องมือในโปรแกรม MATLAB

```
% ตัวอย่าง : ขั้นตอนการ Interpolation ของน้ำชี้
Tz_Interp = 300; % จำนวนจุดพิกัดที่ต้องการ
% 'xT1_size' คือจำนวนจุดพิกัดของน้ำชี้ในแนวแกน x
sam = 1:1:xT1_size; % จำนวนจุดที่ต้องการ sampling ตั้งแต่จุดที่ 1 ถึงจุดที่ต้องการ
% 'xT1_Norm' จุดพิกัดของน้ำชี้ในแนวแกน x ที่ถูกนอร์มอลไลซ์แล้ว
var = xT1_Norm; % ตัวแปรที่ต้องการให้ sampling
a = xT1_size/Tz_Interp; % กำหนดขนาดที่จะแทรกจุดว่าให้แทรกทุกๆ a จุด
samq = 1:a:xT1_size; % จะแทรกจุดทุกๆกี่จุด ตั้งแต่จุดที่ 1 ถึงจุดที่ต้องการ
% จุดพิกัดของน้ำชี้ในแนวแกน x ที่ทำ Interpolation แล้ว
xT1_Interp = interp1(sam,var,samq,'Nearest');
%
% 'yT1_size' คือจำนวนจุดพิกัดของน้ำชี้ในแนวแกน y
sam = 1:1:yT1_size; % จำนวนจุดที่ต้องการ sampling ตั้งแต่จุดที่ 1 ถึงจุดที่ต้องการ
% 'yT1_Norm' จุดพิกัดของน้ำชี้ในแนวแกน y ที่ถูกนอร์มอลไลซ์แล้ว
var = yT1_Norm; % ตัวแปรที่ต้องการให้ sampling
a = yT1_size/Tz_Interp; % กำหนดขนาดที่จะแทรกจุดว่าให้แทรกทุกๆ a จุด
```

```
samq = 1:a:yT1_size; % จะแทรกจุดทุกๆกี่จุด ตั้งแต่จุดที่ 1 ถึงจุดที่ต้องการ
% จุดพิกัดของเข้าชี้ในแนวแกน y ที่ทำ Interpolation แล้ว
yT1_Interp = interp1(sam,var,samq,'Nearest');
%-----------------------------------------------------
% เข้ามือเข้าอื่นๆ และเส้นลายมือใช้วิธีการเดียวกันในการทำ Interpolation
%-----------------------------------------------------
```

จากวิทยานิพนธ์ของ ณัฐภัทร ธีรเบญจกุล “การระบุตัวบุคคลด้วย FIR SYSTEM” [1] ได้มีการแทรกจุดพิกัดให้เส้นขอบนิ้วมือแต่ละนิ้วมีจำนวนจุดพิกัดจำนวน 600 จุด และทำการดาวน์แซมปลิ้ง (Down Sampling) เพื่อลดจำนวนจุดพิกัดให้เหลือ 300 จุด ซึ่งจากการทดลองพบว่าเส้นขอบนิ้วมือของแต่ละนิ้วจะมีจำนวนจุดพิกัด 400 ถึง 600 จุด และเส้นลายมือแต่ละเส้นจะมีจำนวนจุดพิกัด 150 ถึง 200 จุด แตกต่างกันไปตามแต่ละบุคคล ดังนั้นเพื่อความสะดวกในการแทรกจุดพิกัด ในปริญญานิพนธ์ฉบับนี้จะแทรกจุดพิกัดให้เส้นขอบนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นมีจำนวนจุดพิกัดจำนวน 300 จุด และทำการดาวน์แซมปลิ้งเพื่อลดจำนวนจุดพิกัดให้เหลือ 150 จุด โดยใช้คำสั่ง downsample ซึ่งเป็นเครื่องมือในโปรแกรม MATLAB จากนั้นจะนำจุดพิกัดของเส้นขอบนิ้วมือและเส้นลายมือมาหาค่าสัมประสิทธิโคไซน์ต่อไป

```
% ตัวอย่าง : ขั้นตอนการ Downsample เข้าชี้
xT1_Dsam = downsample(xT1_Interp,2); % จุดพิกัดเข้าชี้ในแนวแกน x ที่ถูก Downsample
yT1_Dsam = downsample(yT1_Interp,2); % จุดพิกัดเข้าชี้ในแนวแกน y ที่ถูก Downsample
%-----------------------------------------------------
% เข้ามือเข้าอื่นๆ และเส้นลายมือใช้วิธีการเดียวกันในการทำ Downsample
%-----------------------------------------------------
```

3.8 การหาค่าสัมประสิทธิ์โคไซน์ของนิ้วมือและเส้นลายมือ

ต่อมานำค่าพิกัดที่ผ่านการอินเตอร์โพเลชันและดาวน์แซมปลิ้งของนิ้วมือและเส้นลายมือมาแทนค่าในสมการดีสครีตโคไซน์ทรานส์ฟอร์ม (Discrete Cosine Transform : DCT) เพื่อแปลงค่าจุดพิกัดเป็นค่าสัมประสิทธิ์ DCT ทั้งในแนวแกน x และ y โดยแทนค่าลงในสมการที่ (3.27) และ (3.28)

$$c_1^{(k)}(u) = w(u)\sum_{n=0}^{D-1}\tilde{x}_n^{(k)}\cos\frac{(2n+1)u\pi}{2D} \tag{3.27}$$

$$c_2^{(k)}(u) = w(u)\sum_{n=0}^{D-1}\tilde{y}_n^{(k)}\cos\frac{(2n+1)u\pi}{2D} \tag{3.28}$$

เมื่อ

$$w(u) = \begin{cases} \sqrt{\frac{1}{D}}, & u = 0 \\ \sqrt{\frac{2}{D}}, & 1 \le u \le D-1 \end{cases}$$

เมื่อ

u คือ จำนวนค่าสัมประสิทธิ์โคไซน์ กำหนดให้ u = 0, 1, 2, 3, ..., D-1

$\tilde{x}_n^{(k)}$ คือ ตำแหน่งจุดพิกัดในแนวแกน x ที่ผ่านการดาวน์แซมปลิ้งของเส้นขอบนิ้วมือและเส้นลายมือ

$\tilde{y}_n^{(k)}$ คือ ตำแหน่งจุดพิกัดในแนวแกน y ที่ผ่านการดาวน์แซมปลิ้งของเส้นขอบนิ้วมือและเส้นลายมือ

$c_1^{(k)}(u)$ และ $c_2^{(k)}(u)$ คือ ค่าสัมประสิทธิ์โคไซน์ของแนวแกน x และ y

k คือ นิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อย เมื่อ $k = 1, 2, 3, 4$ และเส้นลายมือเส้นที่หนึ่ง สอง และสาม เมื่อ $k = 5, 6, 7$

$w(u)$ คือ ค่าคงที่

การเขียนโปรแกรมจะยกตัวอย่างการเขียนโปรแกรมของนิ้วชี้ โดยนิ้วมือนิ้วอื่นๆและเส้นลายมือเส้นอื่นๆ ใช้หลักการเดียวกันในการหาค่าสัมประสิทธิ์โคไซน์

```
% ตัวอย่าง : การหาค่าสัมประสิทธิ์โคไซน์ (Discrete Cosine Transform : DCT) ของนิ้วชี้
% 150 คือจำนวนจุดพิกัดของนิ้วชี้
% 'xT1_Dsam' คือ % จุดพิกัดนิ้วชี้ในแนวแกน x ที่ถูก Downsample
% 'yT1_Dsam' คือ % จุดพิกัดนิ้วชี้ในแนวแกน y ที่ถูก Downsample
u1 = 1; % ใช้ในการเรียกข้อมูลและเก็บข้อมูลของตัวแปร u
u0 = 0; % ใช้ในการคำนวณของตัวแปร u
for c = 1 : 150 % จำนวนจุดพิกัดของเส้นขอบนิ้วชี้
    if u0 == 0
        xw(u1) = 1/sqrt(150);
        yw(u1) = 1/sqrt(150);
    elseif u0 >= 1
        xw(u1) = sqrt(2/150);
        yw(u1) = sqrt(2/150);
    end
    n1 = 1; % ใช้ในการเรียกข้อมูลและเก็บข้อมูลของตัวแปร n
    n0 = 0; % ใช้ในการคำนวณของตัวแปร n
    for c1 = 1 : 150 % จำนวนจุดพิกัดของเส้นขอบนิ้วชี้
        xT1_ans1(n1) = xT1_Dsam(n1) * cos(((2*n0)+1)*u0*pi)/(2*150));
        yT1_ans1(n1) = yT1_Dsam(n1) * cos(((2*n0)+1)*u0*pi)/(2*150));
        n1 = n1 + 1; % ตัวถัดไป
        n0 = n0 + 1; % ตัวถัดไป
    end
    xT1_DCT(u1) = xw(u1) * sum(xT1_ans1); % ค่าสัมประสิทธิ์โคไซน์นิ้วชี้ในแนวแกน x
    yT1_DCT(u1) = yw(u1) * sum(yT1_ans1); % ค่าสัมประสิทธิ์โคไซน์นิ้วชี้ในแนวแกน y
    u1 = u1 + 1; % ตัวถัดไป
    u0 = u0 + 1; % ตัวถัดไป
end
%-----------------------------------------------------------------
% นิ้วมือนิ้วอื่นๆ และเส้นลายมือใช้วิธีการเดียวกันในการหาค่าสัมประสิทธิ์โคไซน์
% โดยเปลี่ยนจากตัวแปร T1 ซึ่งหมายถึงนิ้วชี้ เป็นตัวแปร T2, T3 และ T4 คือนิ้วกลาง นิ้วนาง
% และนิ้วก้อย
% สำหรับกรณีของเส้นลายมือให้เปลี่ยนเป็น PL1, PL2 และ PL3 คือเส้นลายมือเส้นที่หนึ่ง สอง
% และสาม ตามลำดับ
%-----------------------------------------------------------------
```

ค่าที่ได้ออกมาคือ $c_1^{(k)}(u)$ และ $c_2^{(k)}(u)$ เป็นค่าสัมประสิทธิ์โคไซน์ในแนวแกน x และ y ตามลำดับ เมื่อนำค่าสัมประสิทธิ์ดังกล่าวมาพล็อทกราฟจะแสดงตัวอย่างได้ดังรูปที่ 3.58 เป็นตัวอย่างค่าสัมประสิทธิ์โคไซน์ของนิ้วชี้ในแนวแกน x และ y ที่ได้จากการคำนวณในโปรแกรม

รูปที่ 3.58 กราฟค่าสัมประสิทธิ์ของ DCT
(ก) ค่าสัมประสิทธิ์โคไซน์ในแนวแกน x, $c_1^{(k)}(u)$
(ข) ค่าสัมประสิทธิ์โคไซน์ในแนวแกน y, $c_2^{(k)}(u)$

3.9 การหาค่าตอบสนองอิมพัลส์ในระบบ FIR (Finite Impulse Response)

จากนั้นนำค่าสัมประสิทธิ์ของ DCT ที่คำนวณได้ของเส้นขอบนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นทั้งในแนวแกน x และ y ไปเข้าระบบ FIR เพื่อหาค่าความเป็นเอกลักษณ์ของนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้น โดยการทำงานของระบบ FIR จะใช้ค่าสัมประสิทธิ์ที่ได้จากการคำนวณในสมการ DCT มาคำนวณในระบบ FIR จะได้ผลตอบสนองอิมพัลส์ (Impulse Response) ออกมาซึ่งจะแสดงความเป็นเอกลักษณ์ของนิ้วแต่ละนิ้วและเส้นลายมือแต่ละเส้น ความสัมพันธ์ของระบบ FIR แสดงได้ดังสมการที่ (3.29)

$$\tilde{Y}_n^{(k)} = \sum_{m=0}^{S} \delta_m^{(k)} \tilde{X}_{n-m}^{(k)} \tag{3.29}$$

เมื่อ

$\tilde{X}_{n-m}^{(k)}$ คือ ค่าอินพุทระบบ FIR

$\delta_m^{(k)}$ คือ ค่าสัมประสิทธิ์ผลตอบสนองของอิมพัลส์ของระบบ FIR

$\tilde{Y}_n^{(k)}$ คือ ค่าเอาท์พุตระบบ FIR

S คือ จำนวนลำดับของระบบ FIR

k คือ นิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อย เมื่อ $k = 1, 2, 3, 4$ และเส้นลายมือเส้นที่หนึ่ง สอง และสาม เมื่อ $k = 5, 6, 7$

การคำนวณผลตอบสนองอิมพัลส์เริ่มจากการจัดเรียงค่าอินพุต $\tilde{X}_{n-m}^{(k)}$ ใหม่ให้อยู่ในรูปเมตริกซ์สามเหลี่ยมล่าง (Lower Triangular Matrix : $T^{(k)}$) ดังสมการที่ (3.30)

$$T^{(k)} = \begin{bmatrix} \tilde{X}_0^{(k)} & 0 & \cdots & 0 \\ \tilde{X}_1^{(k)} & \tilde{X}_0^{(k)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{X}_D^{(k)} & \tilde{X}_{D-1}^{(k)} & \cdots & \tilde{X}_0 \end{bmatrix} \tag{3.30}$$

เมื่อ

D คือ จำนวนสัมประสิทธิ์ที่ผ่านการแปลง DCT

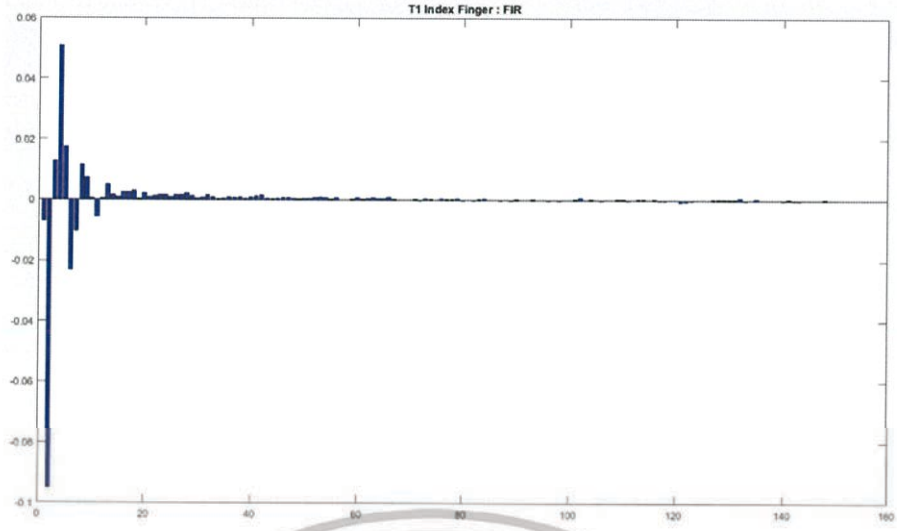
นำค่า $T^{(k)}$ ที่ได้ไปหาค่าผลตอบสนองอิมพัลส์ $\delta_m^{(k)}$ โดยใช้สมการที่ (3.31)

$$\delta_m^{(k)} = \left(T^{(k)T} * T^{(k)}\right)^{-1} * T^{(k)T} * \tilde{Y}_n^{(k)T} \tag{3.31}$$

จะใช้สมการที่ (3.31) นี้ในการหาค่าผลตอบสนองอิมพัลส์ซึ่งเป็นเอกลักษณ์ของนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้นออกมา

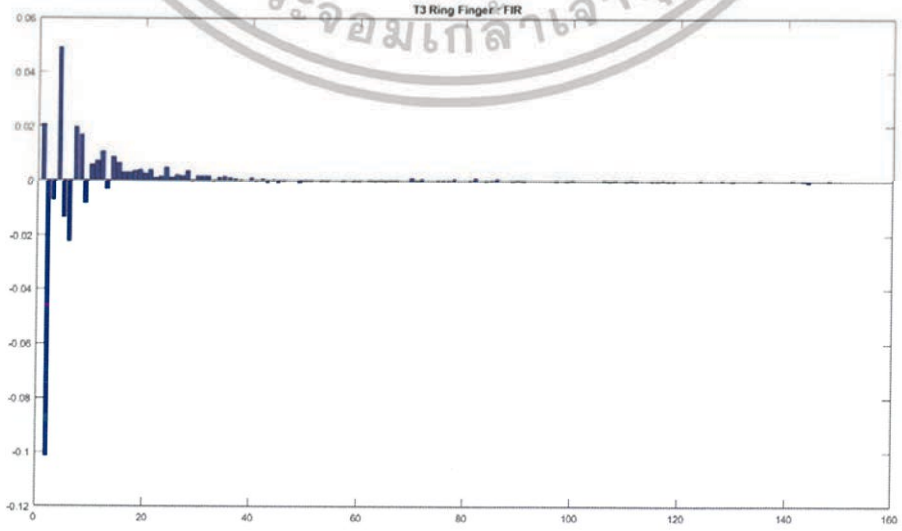
```
% ตัวอย่าง : การหาค่าตอบสนองอิมพัลส์ในระบบ FIR (Finite Impulse Response) ของน้ำชี้
% สร้าง Lower Triangular Matrix
% วิธีการสร้างเมตริกซ์สามเหลี่ยมล่างคือจะสร้างเมตริกซ์ทีละหลักไปเรื่อยๆจนครบทุกหลัก
% 150 คือจำนวนผลตอบสนองอิมพัลส์ในระบบ FIR
% 'xT1_DCT' คือ ค่าสัมประสิทธิ์โคไซน์ของน้ำชี้ในแนวแกน x
% 'yT1_DCT' คือ ค่าสัมประสิทธิ์โคไซน์ของน้ำชี้ในแนวแกน y
m = 1; % กำหนด Row เริ่มต้น
n = 1; % กำหนด Column เริ่มต้น
for c = 1 : 150 % จำนวนผลตอบสนองอิมพัลส์ในระบบ FIR
    a = 1; % เริ่มนับจุดพิกัด
    m = n; % กำหนด Row เริ่มต้นใน Lower Triangular Matrix
    for c1 = 1 : 150 % จำนวนผลตอบสนองอิมพัลส์ในระบบ FIR
        LTM_T1(m,n) = xT1_DCT(1,a); % สร้าง Lower Triangular Matrix
        m = m + 1; % เลื่อนไป Row ถัดไป
        a = a + 1; % จุดพิกัดถัดไป
    end
    n = n + 1; % เลื่อนไป Column ถัดไป
end
% หาค่า Impulse Response ของน้ำชี้
T1_Identity = pinv(LTM_T1) * (transpose(yT1_DCT));
%-------------------------------------------------------------------------
% น้ำมือน้ำอีเฒ่า และเส้นลายมือใช้วิธีการเดียวกันในการหาค่า Impulse Response ของระบบ FIR
%-------------------------------------------------------------------------
```

เมื่อนำค่าผลตอบสนองอิมพัลส์ $\delta_m^{(k)}$ มาพล็อตกราฟจะได้ออกมาดังรูปที่ 3.59 แสดงตัวอย่างค่าผลตอบสนองอิมพัลส์ซึ่งเป็นเอกลักษณ์ของนิ้วมือแต่ละนิ้ว และรูปที่ 3.60 แสดงตัวอย่างค่าผลตอบสนองอิมพัลส์ซึ่งเป็นเอกลักษณ์ของเส้นลายมือแต่ละเส้นที่ได้จากการคำนวณในโปรแกรม โดยได้จำนวนผลตอบสนองอิมพัลส์ทั้งหมด 150 ค่า

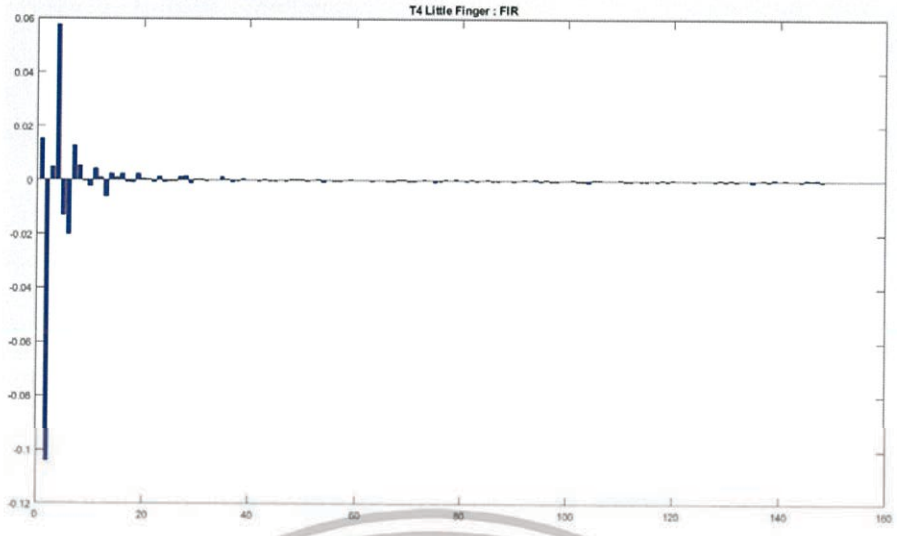


(ก)

(ข)

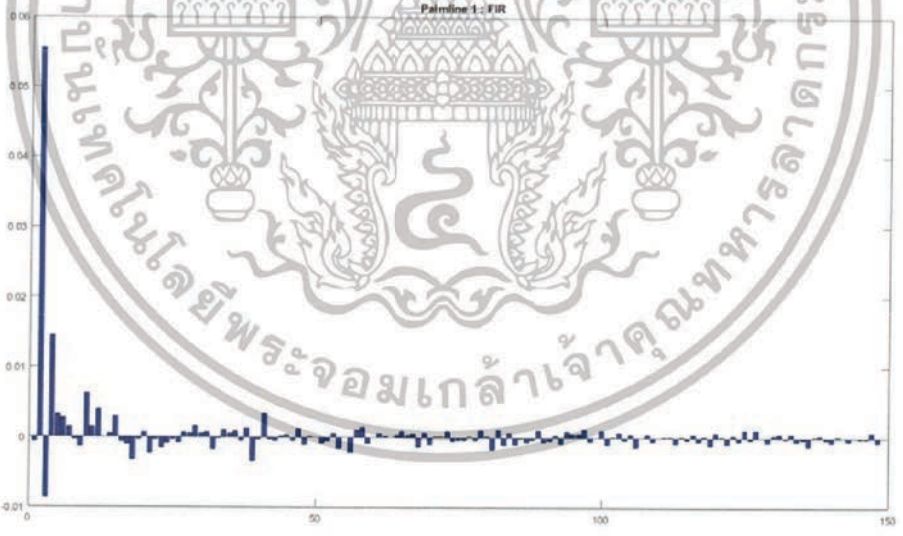


(ค)

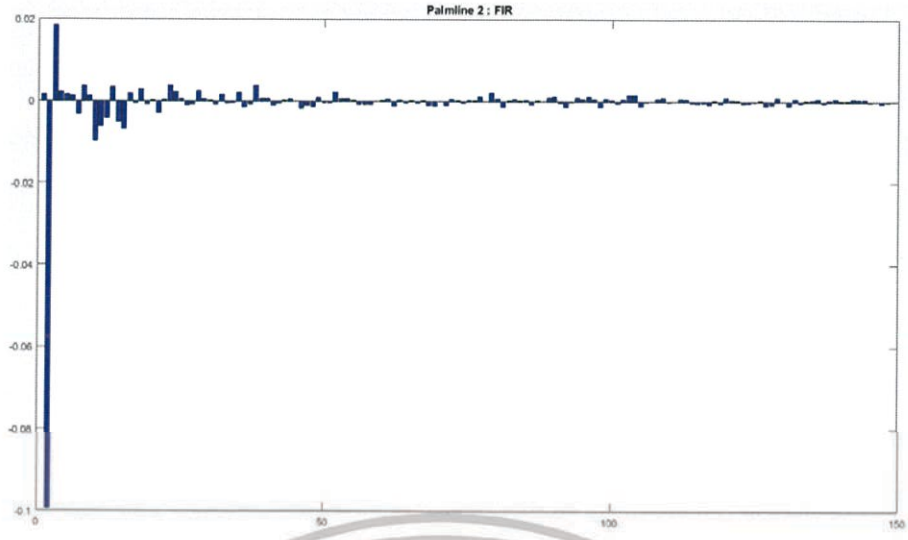


(ง)

รูปที่ 3.59 กราฟค่าผลตอบสนองอิมพัลส์ของนิ้วมือแต่ละนิ้วที่ได้จากระบบ FIR จำนวน 150 ค่า
(ก) กราฟค่าผลตอบสนองอิมพัลส์ของนิ้วชี้
(ข) กราฟค่าผลตอบสนองอิมพัลส์ของนิ้วกลาง
(ค) กราฟค่าผลตอบสนองอิมพัลส์ของนิ้วนาง
(ง) กราฟค่าผลตอบสนองอิมพัลส์ของนิ้วก้อย



(ก)



(ข)

(ค)

รูปที่ 3.60 กราฟค่าผลตอบสนองอิมพัลส์ของเส้นลายมือแต่ละเส้นที่ได้จากระบบ FIR จำนวน 150 ค่า
(ก) กราฟค่าผลตอบสนองอิมพัลส์ของเส้นลายมือเส้นที่หนึ่ง
(ข) กราฟค่าผลตอบสนองอิมพัลส์ของเส้นลายมือเส้นที่สอง
(ค) กราฟค่าผลตอบสนองอิมพัลส์ของเส้นลายมือเส้นที่สาม

3.10 การเก็บค่าผลตอบสนองอิมพัลส์ของนิ้วมือและเส้นลายมือเพื่อใช้เป็นฐานข้อมูลแม่แบบ

การเก็บค่าผลตอบสนองอิมพัลส์เพื่อใช้เป็นฐานข้อมูลแม่แบบนั้น ทำเพื่อใช้ในการเปรียบเทียบกับข้อมูลภาพที่ต้องการนำมาประมวลผลว่าสามารถแยกแยะและระบุตัวบุคคลได้

การจัดเก็บค่าผลตอบสนองอิมพัลส์ซึ่งจะแสดงให้เห็นถึงเอกลักษณ์ของนิ้วมือและเส้นลายมือนั้น ในวิทยานิพนธ์ของ ณัฐภัทร ธีรเบญจกุล “การระบุตัวบุคคลด้วย FIR SYSTEM” [1] จะเก็บค่า

ผลตอบสนองอิมพัลส์ของแต่ละนิ้วเพื่อจัดทำเป็นฐานข้อมูลแม่แบบจำนวน 50 ค่า แต่เนื่องจากในปริญญานิพนธ์ฉบับนี้จะมีการจัดเก็บผลตอบสนองอิมพัลส์ของเส้นลายมือด้วย ซึ่งจำนวนจุดพิกัดของเส้นลายมือที่จัดเก็บไว้ในตอนต้นก่อนการแทรกจุดพิกัดจะมีจำนวน 150 ถึง 200 จุด และเมื่อหาผลตอบสนองอิมพัลส์ออกมาจะได้ 150 ค่า จะเห็นได้ว่าจำนวนของค่าผลตอบสนองอิมพัลส์ที่แสดงให้เห็นถึงเอกลักษณ์ของเส้นลายมือจะมีจำนวนที่ใกล้เคียงกับจำนวนจุดพิกัดของเส้นลายมือ หากลดจำนวนผลตอบสนองอิมพัลส์เพื่อจัดทำเป็นฐานข้อมูลลงเหลือ 50 จุด อาจทำให้การเปรียบเทียบตัวบุคคลเกิดความผิดพลาดได้ ดังนั้นเพื่อความสะดวกในการจัดเก็บค่าผลตอบสนองอิมพัลส์ ในปริญญานิพนธ์ฉบับนี้จะกำหนดให้เก็บค่าผลตอบสนองอิมพัลส์ของนิ้วมือทั้ง 4 นิ้ว และเส้นลายมือทั้ง 3 เส้น อยู่ที่ 150 ค่า

ในปริญญานิพนธ์ฉบับนี้จะยังไม่มีการจัดทำฐานข้อมูลเป็นข้อมูลชุดเดียว แต่จะใช้การจัดเก็บชุดข้อมูลแยกแต่ละบุคคลแทน โดยจะใช้คำสั่ง save ซึ่งเป็นเครื่องมือในโปรแกรม MATLAB หมายถึงคำสั่งในการบันทึกเพื่อเก็บค่าผลตอบสนองอิมพัลส์ลงในไฟล์ข้อมูลของแต่ละบุคคล โดยจะเก็บค่าผลตอบสนองอิมพัลส์ของนิ้วมือทุกนิ้วและเส้นลายมือทุกเส้น มีรูปแบบการใช้คำสั่งดังนี้

```
save Person1 T1 T2 T3 T4 PL1 PL2 PL3
```

คำอธิบาย

Person1 คือชื่อไฟล์ที่ต้องการบันทึกของแต่ละบุคคล โดยจะเป็นไฟล์นามสกุล .mat

T1, T2, T3, T4 คือ ตัวแปรของค่าผลตอบสนองอิมพัลส์จำนวน 150 ค่า ของนิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อย ตามลำดับ

PL1, PL2, PL3 คือ ตัวแปรของค่าผลตอบสนองอิมพัลส์จำนวน 150 ค่า ของเส้นลายมือเส้นที่หนึ่ง สอง และสาม ตามลำดับ

```
% ตัวอย่าง : การจัดเก็บค่าผลตอบสนองอิมพัลส์ของนิ้วมือและเส้นลายมือของบุคคลที่ 1
  save Person1 T1_Identity T2_Identity T3_Identity T4_Identity
               PL1_Identity PL2_Identity PL3_Identity
%-------------------------------------------------------------------
% บุคคลอื่นๆ ใช้วิธีการเดียวกันในการจัดเก็บค่าผลตอบสนองอิมพัลส์ของนิ้วมือและเส้นลายมือ
%-------------------------------------------------------------------
```

3.11 การระบุตัวบุคคลโดยการเปรียบเทียบกับฐานข้อมูล

การระบุตัวบุคคลโดยเปรียบเทียบกับฐานข้อมูล จะใช้นิ้วมือ 4 นิ้ว และเส้นลายมือ 3 เส้น ของรูปภาพที่รับเข้ามาใหม่ โดยขั้นตอนการแยกส่วนของนิ้วมือและแยกเส้นลายมือจะมีขั้นตอนที่เหมือนกับขั้นตอนที่ได้อธิบายไปในหัวข้อที่ 3.1 ถึง 3.9 โดยผลที่ได้ออกมาก่อนเข้าสู่ขั้นตอนการเปรียบเทียบกับฐานข้อมูลคือค่าผลตอบสนองอิมพัลส์ซึ่งแสดงถึงเอกลักษณ์ของนิ้วมือแต่ละนิ้วและเส้นลายมือแต่ละเส้น แสดงตัวอย่างค่าผลตอบสนองอิมพัลส์ได้ในรูปที่ 3.59 และ 3.60

นำค่าผลตอบสนองอิมพัลส์ของรูปภาพที่รับเข้ามาใหม่นี้ มาทำการเปรียบเทียบกับค่าผลตอบสนองอิมพัลส์ที่มีอยู่แล้วในฐานข้อมูลทีละตัว และคำนวณหาค่าความผิดพลาดของการเปรียบเทียบโดยใช้สมการที่ (3.31) และ (3.32)

$$e^k = \sum_{m=0}^{S} \left[(\delta_{ref})_m^{(k)} - \delta_m^{(k)} \right]^2 \tag{3.32}$$

$$e^h = \sum_{m=0}^{S} \left[(\delta_{ref})_m^{(h)} - \delta_m^{(h)} \right]^2 \tag{3.33}$$

เมื่อ

δ_{ref} คือ ค่าผลตอบสนองอิมพัลส์ของนิ้วมือและเส้นลายมือที่อยู่ในฐานข้อมูล
δ คือ ค่าผลตอบสนองอิมพัลส์ของนิ้วมือและเส้นลายมือของภาพมือที่รับเข้ามา
k คือ นิ้วชี้ นิ้วกลาง นิ้วนาง และนิ้วก้อย เมื่อ $k = 1, 2, 3, 4$
h คือ เส้นลายมือเส้นที่หนึ่ง สอง และสาม เมื่อ $h = 1, 2, 3$
S คือ จำนวนลำดับในระบบ FIR
m คือ ลำดับข้อมูลของค่าผลตอบสนองอิมพัลส์ของนิ้วมือและเส้นลายมือ

โดยค่าความผิดพลาดของนิ้วมือแต่ละนิ้วจะถูกนำมารวมกันโดยใช้สมการที่ (3.34) และค่าความผิดพลาดรวมของเส้นลายมือแต่ละเส้นจะถูกนำมารวมกันโดยใช้สมการที่ (3.35)

$$E_1 = \sum_{k=1}^{4} e^k \tag{3.34}$$

$$E_2 = \sum_{h=1}^{3} e^h \tag{3.35}$$

และเมื่อเขียนในรูปค่าความผิดพลาดรวมที่จะรวมค่าความผิดพลาดจากนิ้วมือทั้ง 4 นิ้วและเส้นลายมือทั้ง 3 เส้น จะเขียนได้ดังสมการที่ (3.36)

$$E = E_1 + E_2 \tag{3.36}$$

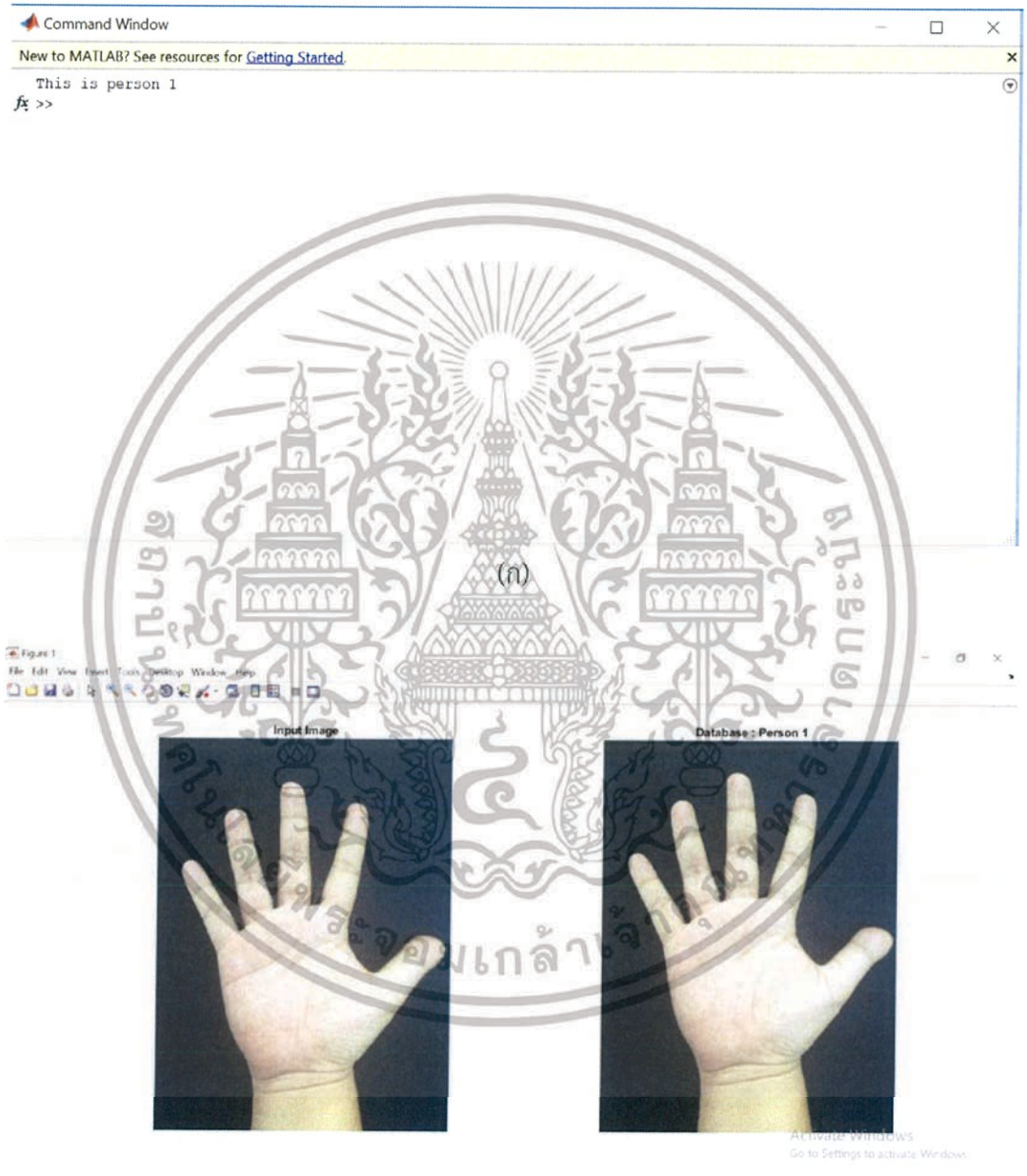
```
% ตัวอย่างการคำนวณค่าความผิดพลาดเพื่อใช้ในการระบุตัวบุคคลโดยเปรียบเทียบกับบุคคลที่ 1
load Person1 % โหลดไฟล์ที่จัดเก็บไว้เข้ามาในโปรแกรม MATLAB ในที่นี้คือบุคคลที่ 1
% ข้อมูลที่ได้จะมีผลตอบสนองอิมพัลส์ของนิ้วมือทั้ง 4 นิ้ว (T1_Identity ถึง T4_Identity)
% และผลตอบสนองอิมพัลส์ของเส้นลายมือทั้ง 3 เส้น (PL1_Identity ถึง PL3_Identity)
person = 1; % เปรียบเทียบผลตอบสนองอิมพัลส์กับบุคคลที่ 1
% หาความผิดพลาดของนิ้วมือทั้ง 4 นิ้ว
T_error(1) = sum((T1_Identity - T1_Test).^2); % ความผิดพลาดของนิ้วชี้
T_error(2) = sum((T2_Identity - T2_Test).^2); % ความผิดพลาดของนิ้วกลาง
T_error(3) = sum((T3_Identity - T3_Test).^2); % ความผิดพลาดของนิ้วนาง
T_error(4) = sum((T4_Identity - T4_Test).^2); % ความผิดพลาดของนิ้วก้อย
```

```
T_sum_error(person) = sum(T_error); % ความผิดพลาดรวมของน้ำมือทั้ง 4 นิ้ว
% หาความผิดพลาดของเส้นลายมือทั้ง 3 เส้น
PL_error(1) = sum((PL1_Identity - PL1_Test).^2);%ความผิดพลาดของเส้นลายมือเส้นที่หนึ่ง
PL_error(2) = sum((PL2_Identity - PL2_Test).^2);%ความผิดพลาดของเส้นลายมือเส้นที่สอง
PL_error(3) = sum((PL3_Identity - PL3_Test).^2);%ความผิดพลาดของเส้นลายมือเส้นที่สาม
PL_sum_error(person) = sum(PL_error); % ความผิดพลาดรวมของเส้นลายมือทั้ง 3 เส้น
% หาความผิดพลาดรวมทั้งของน้ำมือและเส้นลายมือ
All_error(person) = T_sum_error(person) + PL_sum_error(person);
%-------------------------------------------------------------------------
% เมื่อ T1_Test ถึง T4_Test และ PL1_Test ถึง PL3_Test คือ ผลตอบสนองอิมพัลส์ของน้ำมือ
% และเส้นลายมือที่รับเข้ามาทดสอบ
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% จะใช้วิธีการเดียวกันนี้ในการเปรียบเทียบค่าความผิดพลาดกับบุคคลในฐานข้อมูลบุคคลอื่นๆ
%-------------------------------------------------------------------------
```

ในการระบุตัวบุคคลจะใช้ค่าความผิดพลาดที่น้อยที่สุด (Minimum Error) ในการระบุตัวบุคคล

```
% ตัวอย่างการยืนยันตัวบุคคล
% จะยกตัวอย่างการยืนยันตัวบุคคลโดยใช้เส้นขอบน้ำมือ
T_sum_error_min = min(T_sum_error); % หาความผิดพลาดที่น้อยที่สุด
% เปรียบเทียบกับบุคคลที่ 1 ว่าตรงกับค่าผิดพลาดน้อยที่สุดหรือไม่
if T_sum_error(1) == T_sum_error_min
    % ถ้าเปรียบเทียบแล้วมีค่าตรงกับค่าความผิดพลาดที่น้อยที่สุด แสดงว่าเป็นบุคคลที่ 1
    disp ('This is person 1');
    figure(1) % แสดงภาพเพื่อเปรียบเทียบภาพที่รับเข้ามาทดสอบกับฐานข้อมูล
    subplot(1,2,1);
    imshow(Image_Input); % แสดงภาพที่รับเข้ามาทดสอบ
    title('Input Image')
    image_Database = imread('D:\ที่อยู่ของไฟล์ภาพ\บุคคลที่1.jpg'); % ภาพที่จัดเก็บในฐานข้อมูล
    subplot(1,2,2);
    imshow(image_Database); % แสดงภาพในฐานข้อมูลของบุคคลที่ถูกระบุ
    title('Database : Person 1')
% เปรียบเทียบกับบุคคลที่ 2 ว่าตรงกับค่าผิดพลาดน้อยที่สุดหรือไม่
elseif T_sum_error(2) == T_sum_error_min
        % ถ้าเปรียบเทียบแล้วมีค่าตรงกับค่าความผิดพลาดที่น้อยที่สุด แสดงว่าเป็นบุคคลที่ 2
        disp ('This is person 2');
        figure(1) % แสดงภาพเพื่อเปรียบเทียบภาพที่รับเข้ามาทดสอบกับฐานข้อมูล
        subplot(1,2,1);
        imshow(Image_Input); % แสดงภาพที่รับเข้ามาทดสอบ
        title('Input Image')
        image_Database = imread('D:\ที่อยู่ของไฟล์ภาพ\บุคคลที่2.jpg'); % ภาพที่จัดเก็บในฐานข้อมูล
        subplot(1,2,2);
        imshow(image_Database); % แสดงภาพในฐานข้อมูลของบุคคลที่ถูกระบุ
        title('Database : Person 2')
%-------------------------------------------------------------------------
% จะใช้วิธีการเดียวกันนี้ในการตรวจสอบตัวบุคคล กับบุคคลในฐานข้อมูลบุคคลอื่นๆ
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% จะใช้วิธีการเดียวกันนี้ในการยืนยันตัวบุคคลโดยใช้เส้นลายมือ
% และใช้วิธีการเดียวกันนี้ในการยืนยันตัวบุคคลโดยใช้เส้นขอบน้ำมือร่วมกับเส้นลายมือ
%-------------------------------------------------------------------------
```

สามารถแสดงตัวอย่างผลการระบุตัวบุคคลได้ในรูปที่ 3.61 โดยเป็นการรับภาพมือของบุคคลที่ต้องการทดสอบเข้ามาทดสอบในโปรแกรม แล้วระบบระบุว่าภาพมือของบุคคลที่รับเข้ามาทดสอบนั้นตรงกับภาพมือของบุคคลที่ 1 ที่ใช้เป็นฐานข้อมูล โดยแสดงคำว่า “This is person 1” ในคอมมานด์วินโดว์ (Command Window) แสดงได้ในรูปที่ 3.61 (ก) และนำภาพมือของบุคคลที่รับเข้ามาทดสอบมาเปรียบเทียบกับภาพมือของบุคคลที่ถูกใช้เป็นฐานข้อมูลที่ถูกระบุ แสดงได้ในรูปที่ 3.61 (ข)



(ก)

(ข)

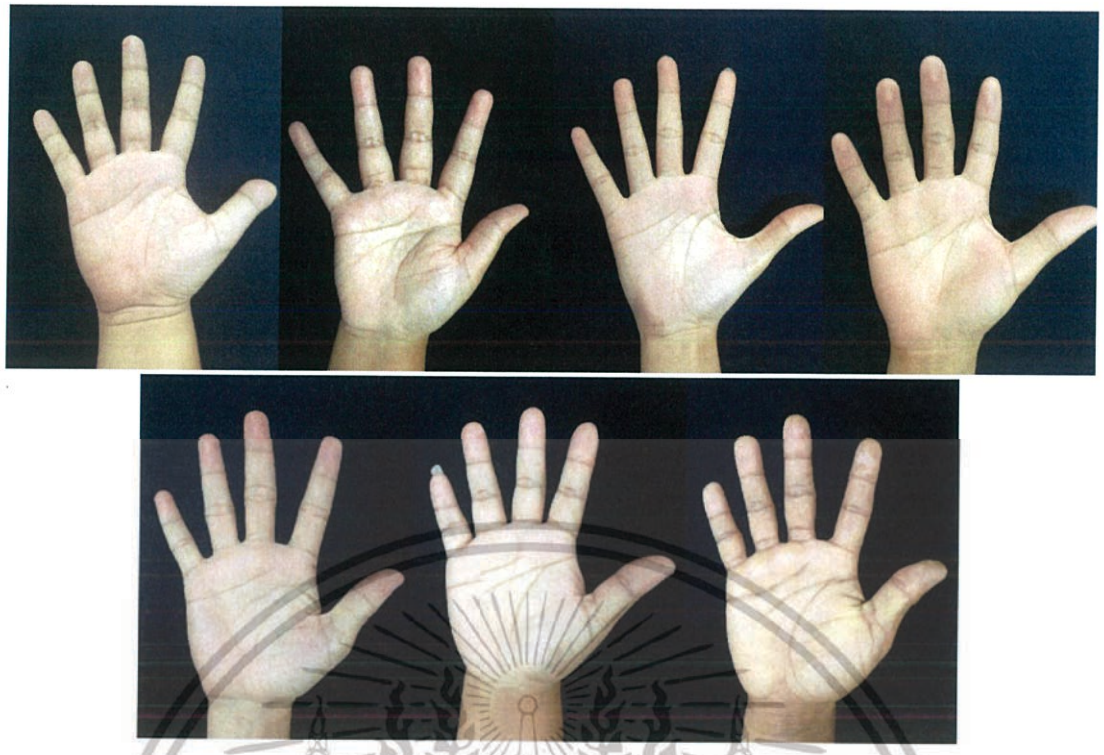
รูปที่ 3.61 ตัวอย่างผลการระบุตัวบุคคลเมื่อเปรียบเทียบกับฐานข้อมูลที่จัดเก็บไว้
(ก) การระบุตัวบุคคลที่ระบบระบุว่าเป็นบุคคลที่ 1
(ข) เปรียบเทียบภาพที่รับเข้ามา (ภาพซ้าย) กับภาพบุคคลที่ถูกระบุ (ภาพขวา)

บทที่ 4
ผลการทดลองการระบุตัวบุคคลโดยใช้ FIR System

ในส่วนของผลการทดลองนี้จะแบ่งออกเป็น 3 ส่วน คือ 1. ผลการทดลองการระบุตัวบุคคลโดยใช้เส้นขอบนิ้วมือ 2. ผลการทดลองการระบุตัวบุคคลโดยใช้ลายเส้นบนฝ่ามือ และ 3. ผลการทดลองการระบุตัวบุคคลโดยใช้เส้นขอบนิ้วมือและลายเส้นบนฝ่ามือ การทดลองนี้จะใช้ฐานข้อมูลของ KMITL ที่มีอยู่แล้วเป็นแม่แบบ โดยภาพถ่ายมือของแต่ละบุคคลจะเป็นภาพถ่ายที่ได้จากกล้องดิจิตอล Nikon S2600 ภาพสี 24 บิต ขนาดภาพ 1600×1200 ที่ความละเอียด 72 พิกเซล/นิ้ว ภาพถ่ายมีระยะห่างระหว่างมือกับกล้อง 20 เซนติเมตร [1]

ในปริญญานิพนธ์ฉบับนี้ได้คัดเลือกบุคคลจำนวน 7 บุคคล บุคคลละ 5 ภาพจากฐานข้อมูลเพื่อใช้ในการทดลอง โดยองค์ประกอบของแต่ละภาพจะมีองค์ประกอบที่แตกต่างกันคือ ภาพแรกที่ใช้เป็นฐานข้อมูลจะเป็นภาพที่ถ่ายในระยะ 20 เซนติเมตร จากนั้นอีก 4 ภาพที่เหลือจะนำมาใช้ในการทดลองยืนยันตัวบุคคล โดยภาพลำดับที่ 1 เป็นภาพถ่ายที่ถ่ายซ้ำในตำแหน่งเดิม ภาพลำดับที่ 2 เป็นภาพถ่ายที่กางนิ้วมือให้ห่างกว่าเดิม ภาพลำดับที่ 3 เป็นภาพถ่ายที่ถ่ายในระยะใกล้กว่าเดิม และภาพลำดับที่ 4 เป็นภาพถ่ายที่ถ่ายในระยะที่ไกลกว่าเดิม โดยสาเหตุที่ใช้บุคคลเพียง 7 คนในการทดลองเนื่องจากยังคงมีปัญหาในเรื่องของการแยกเส้นลายมือออกจากฝ่ามือและการจัดเก็บค่าจุดพิกัดของเส้นลายมือที่ยังไม่สามารถจัดเก็บจุดพิกัดได้อย่างสมบูรณ์ ซึ่งในการทดลองสามารถที่จะแยกเส้นลายมือออกจากฝ่ามือได้เพียง 7 บุคคล

ในการทดลองการยืนยันตัวบุคคล การที่จะตัดสินว่าภาพที่รับเข้ามาทดสอบเป็นบุคคลใดจะใช้ความผิดพลาดรวมที่น้อยที่สุด โดยรูปมือของแต่ละบุคคลที่นำมาใช้ในการทดลองทั้ง 7 บุคคลสามารถแสดงได้ในรูปที่ 4.1



รูปที่ 4.1 ภาพมือที่ใช้ในการทดลองจำนวน 7 บุคคล

ผลการทดลองเปรียบเทียบค่าความผิดพลาดของแต่ละภาพของแต่ละบุคคลที่รับเข้ามากับบุคคลทั้ง 7 คนในฐานข้อมูลสามารถแสดงได้ในหัวข้อที่ 4.1, 4.2 และ 4.3

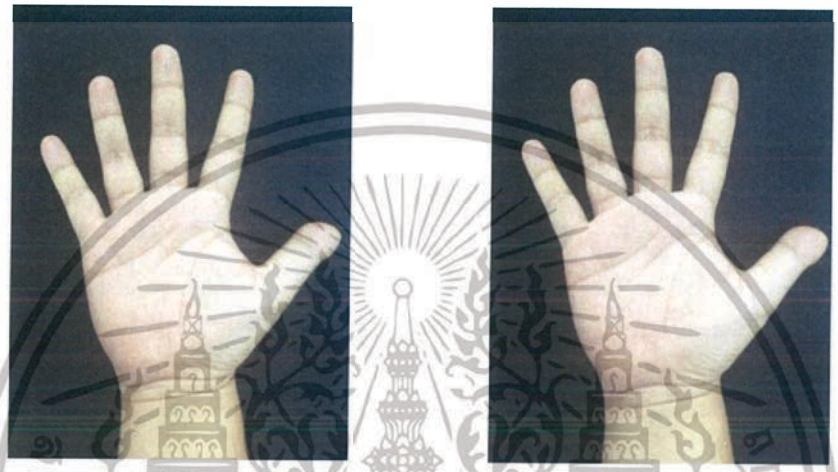
4.1 ผลการทดลองการยืนยันตัวบุคคลโดยใช้เส้นขอบนิ้วมือ

ในหัวข้อนี้จะแสดงผลการทดลองของการยืนยันตัวบุคคลโดยใช้เส้นขอบนิ้วมือ โดยใช้บุคคลในการทดสอบจำนวน 7 คน บุคคลละ 4 ภาพ รวมทั้งหมด 28 ภาพ การทดสอบจะรับภาพเข้ามาทดสอบทีละภาพโดยเริ่มตั้งแต่ บุคคลที่ 1 ภาพที่ 1, บุคคลที่ 1 ภาพที่ 2, บุคคลที่ 1 ภาพที่ 3 ไปเรื่อยๆ จนถึง บุคคลที่ 7 ภาพที่ 4 ซึ่งเป็นภาพสุดท้าย เมื่อรับภาพเข้ามาแล้วจะทำการหาผลตอบสนองอิมพัลส์ จากนั้นนำผลตอบสนองนี้ไปเปรียบเทียบกับผลตอบสนองอิมพัลส์ที่จัดเก็บไว้ในฐานข้อมูลของบุคคลทั้ง 7 คน โดยจะแสดงผลการเปรียบเทียบค่าความผิดพลาดได้ในตารางที่ 4.1 หากพบว่าผลการเปรียบเทียบกับฐานข้อมูลของบุคคลใดมีความผิดพลาดน้อยที่สุด ระบบจะระบุว่าเป็นบุคคลนั้น

ตารางที่ 4.1 ผลการเปรียบเทียบค่าความผิดพลาดของแต่ละภาพของแต่ละบุคคลกับฐานข้อมูลโดยใช้เส้นขอบนิ้วมือ

บุคคล	ภาพที่	ค่าความผิดพลาดเมื่อเปรียบเทียบกับบุคคลในฐานข้อมูล บุคคลที่ 1 ถึง 7							บุคคลที่ผิดพลาดน้อยสุด
		1	2	3	4	5	6	7	
1	1	1.2E-4	0.0052	0.0040	0.0027	0.0039	0.0043	0.0058	1
	2	1.3E-4	0.0055	0.0043	0.0027	0.0040	0.0044	0.0058	1
	3	1.0E-4	0.0053	0.0042	0.0027	0.0039	0.0044	0.0057	1
	4	1.1E-4	0.0055	0.0043	0.0027	0.0040	0.0044	0.0058	1
2	1	0.0050	2.1E-5	0.0032	0.0077	0.0060	0.0062	0.0108	2
	2	0.0051	1.0E-4	0.0031	0.0076	0.0061	0.0061	0.0109	2
	3	0.0052	6.2E-5	0.0033	0.0078	0.0061	0.0063	0.0109	2
	4	0.0051	4.0E-5	0.0032	0.0077	0.0060	0.0061	0.0108	2
3	1	0.0042	0.0032	7.6E-5	0.0056	0.0041	0.0050	0.0087	3
	2	0.0038	0.0031	1.1E-4	0.0053	0.0037	0.0047	0.0081	3
	3	0.0041	0.0033	9.4E-5	0.0057	0.0042	0.0051	0.0089	3
	4	0.0041	0.0032	7.5E-5	0.0055	0.0038	0.0049	0.0083	3
4	1	0.0032	0.0084	0.0061	1.2E-4	0.0027	0.0032	0.0036	4
	2	0.0028	0.0076	0.0053	1.1E-4	0.0023	0.0031	0.0036	4
	3	0.0031	0.0085	0.0062	1.9E-4	0.0026	0.0032	0.0035	4
	4	0.0026	0.0075	0.0052	1.7E-4	0.0023	0.0031	0.0037	4
5	1	0.0042	0.0061	0.0041	0.0025	7.9E-5	0.0046	0.0031	5
	2	0.0041	0.0060	0.0040	0.0024	6.8E-5	0.0045	0.0032	5
	3	0.0041	0.0059	0.0040	0.0025	7.0E-5	0.0045	0.0033	5
	4	0.0041	0.0060	0.0041	0.0025	6.7E-5	0.0045	0.0032	5
6	1	0.0047	0.0064	0.0050	0.0032	0.0046	6.0E-5	0.0060	6
	2	0.0046	0.0064	0.0050	0.0031	0.0046	8.9E-5	0.0060	6
	3	0.0047	0.0065	0.0050	0.0031	0.0047	1.0E-4	0.0062	6
	4	0.0047	0.0064	0.0050	0.0031	0.0046	7.7E-5	0.0060	6
7	1	0.0062	0.0109	0.0084	0.0039	0.0033	0.0060	1.4E-5	7
	2	0.0058	0.0102	0.0079	0.0035	0.0030	0.0056	1.2E-4	7
	3	0.0058	0.0107	0.0083	0.0035	0.0032	0.0057	9.5E-5	7
	4	0.0060	0.0105	0.0078	0.0035	0.0033	0.0054	1.3E-4	7
รวม	28					จำนวนภาพที่ระบุได้ถูกต้อง			28
						ความถูกต้องคิดเป็นร้อยละ			100

จากตารางที่ 4.1 จะยกตัวอย่างการระบุบุคคลโดยใช้ค่าความผิดพลาดที่น้อยที่สุด เช่น บุคคลที่ 1 ภาพที่ 1 เมื่อเปรียบเทียบกับฐานข้อมูลแล้วพบว่าค่าความผิดพลาดที่น้อยที่สุดคือ เมื่อเปรียบเทียบกับบุคคลที่ 1 มีค่าความผิดพลาด 1.2E-4 ซึ่งน้อยที่สุดเมื่อเปรียบเทียบกับบุคคลที่ 2 ถึง 7 ดังนั้นระบบจะระบุว่าภาพที่รับเข้ามาทดสอบเป็นบุคคลที่ 1 โดยช่องที่ระบายแถบสีไว้คือช่องที่มีค่าความผิดพลาดน้อยที่สุดเมื่อนำบุคคลที่ต้องการเปรียบเทียบมาเปรียบเทียบกับบุคคลในฐานข้อมูล โดยสามารถแสดงภาพมือของบุคคลที่รับเข้ามาทดสอบคือบุคคลที่ 1 ภาพที่ 1 ได้ดังรูปที่ 4.2 (ก) และภาพมือของบุคคลที่ใช้เป็นฐานข้อมูลที่ถูกระบุว่าเป็นบุคคลที่ 1 ได้ดังรูปที่ 4.2 (ข)



(ก) (ข)

รูปที่ 4.2 ตัวอย่างภาพมือที่ใช้เป็นฐานข้อมูลและภาพมือที่รับเข้ามาทดสอบการระบุบุคคลโดยใช้เส้นขอบนิ้วมือ
(ก) บุคคลที่ 1 ภาพที่ 1 ที่รับเข้ามาทดสอบ
(ข) ภาพมือบุคคลที่ 1 ที่ใช้เป็นฐานข้อมูลที่ถูกระบุ

จากรูปที่ 4.2 จะเห็นได้ว่าภาพมือของบุคคลที่รับเข้ามาทดสอบ เป็นบุคคลเดียวกับฐานข้อมูล ซึ่งตรงกับข้อมูลจากตารางที่ระบุว่าเมื่อเปรียบเทียบความผิดพลาดแล้วมีค่าความผิดพลาดน้อยที่สุด

ดังนั้นผลการทดลองจากตารางที่ 4.1 แสดงให้เห็นว่าการระบุตัวบุคคลโดยใช้เส้นขอบนิ้วมือของทั้ง 7 บุคคล บุคคลละ 4 ภาพ รวม 28 ภาพ สามารถที่จะระบุบุคคลได้ถูกต้องจำนวน 28 ภาพ

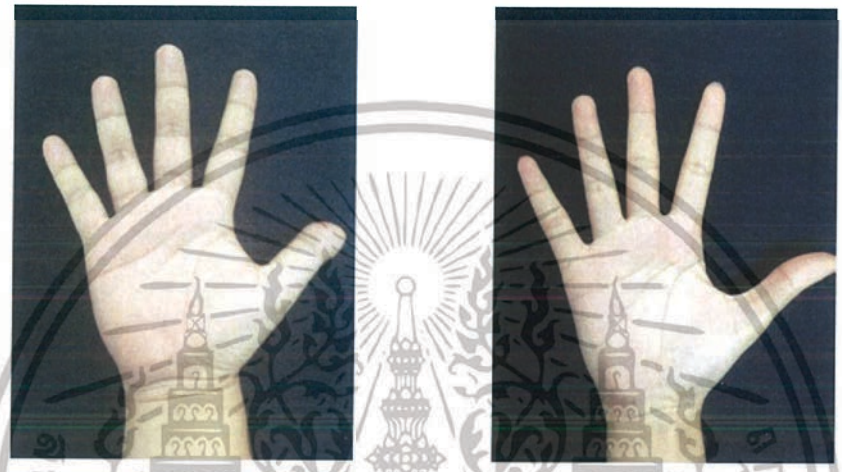
4.2 ผลการทดลองการยืนยันตัวบุคคลโดยใช้ลายเส้นบนฝ่ามือ

ในหัวข้อนี้จะแสดงผลการทดลองของการยืนยันตัวบุคคลโดยใช้ลายเส้นบนฝ่ามือ โดยใช้บุคคลในการทดสอบจำนวน 7 คน บุคคลละ 4 ภาพ รวมทั้งหมด 28 ภาพ เช่นเดียวกัน เมื่อรับภาพเข้ามาแล้วจะทำการหาผลตอบสนองอิมพัลส์จากนั้นนำผลตอบสนองนี้ไปเปรียบเทียบกับผลตอบสนองที่จัดเก็บไว้ในฐานข้อมูลของบุคคลทั้ง 7 คน โดยจะแสดงผลการเปรียบเทียบค่าความผิดพลาดได้ในตารางที่ 4.2 หากพบว่าผลการเปรียบเทียบกับฐานข้อมูลของบุคคลใดมีความผิดพลาดน้อยที่สุด ระบบจะระบุว่าเป็นบุคคลนั้น

ตารางที่ 4.2 ผลการเปรียบเทียบค่าความผิดพลาดของแต่ละภาพของแต่ละบุคคลกับฐานข้อมูลโดยใช้ลายเส้นบนฝ่ามือ

บุคคล	ภาพที่	ค่าความผิดพลาดเมื่อเปรียบเทียบกับบุคคลในฐานข้อมูลบุคคลที่ 1 ถึง 7							บุคคลที่ผิดพลาดน้อยสุด
		1	2	3	4	5	6	7	
1	1	0.0118	0.0193	0.0059	0.0070	0.0095	0.0125	0.0094	3
	2	0.0029	0.0176	0.0049	0.0101	0.0151	0.0142	0.0094	1
	3	0.0076	0.0118	0.0067	0.0064	0.0113	0.0113	0.0052	4
	4	0.0029	0.0176	0.0049	0.0101	0.0151	0.0142	0.0094	1
2	1	0.0166	0.0026	0.0177	0.0196	0.0237	0.0165	0.0119	2
	2	0.0179	0.0020	0.0187	0.0195	0.0240	0.0165	0.0116	2
	3	0.0184	0.0035	0.0194	0.0218	0.0261	0.0185	0.0144	2
	4	0.0170	0.0031	0.0185	0.0203	0.0250	0.0174	0.0122	2
3	1	0.0026	0.0246	0.0051	0.0120	0.0140	0.0159	0.0121	1
	2	0.0022	0.0233	0.0038	0.0110	0.0135	0.0151	0.0107	1
	3	0.0066	0.0192	0.0028	0.0070	0.0141	0.0124	0.0083	3
	4	0.0062	0.0196	0.0026	0.0077	0.0148	0.0134	0.0089	3
4	1	0.0106	0.0264	0.0079	0.0039	0.0064	0.0074	0.0083	4
	2	0.0052	0.0256	0.0075	0.0054	0.0065	0.0078	0.0092	4
	3	0.0109	0.0268	0.0094	0.0040	0.0053	0.0072	0.0088	4
	4	0.0095	0.0232	0.0079	0.0027	0.0064	0.0079	0.0070	4
5	1	0.0135	0.0351	0.0194	0.0132	0.0101	0.0123	0.0156	5
	2	0.0082	0.0268	0.0164	0.0117	0.0124	0.0109	0.0117	4
	3	0.0132	0.0284	0.0157	0.0093	0.0067	0.0087	0.0108	5
	4	0.0134	0.0353	0.0191	0.0129	0.0097	0.0120	0.0152	5
6	1	0.0205	0.0372	0.0222	0.0153	0.0159	0.0139	0.0194	6
	2	0.0121	0.0169	0.0131	0.0070	0.0083	1.0E-4	0.0078	6
	3	0.0304	0.0428	0.0258	0.0262	0.0310	0.0284	0.0299	3
	4	0.0204	0.0375	0.0225	0.0152	0.0161	0.0138	0.0191	6
7	1	0.0082	0.0147	0.0088	0.0060	0.0073	0.0090	0.0023	7
	2	0.0115	0.0125	0.0097	0.0058	0.0083	0.0091	0.0025	7
	3	0.0098	0.0136	0.0092	0.0060	0.0093	0.0105	0.0030	7
	4	0.0113	0.0131	0.0094	0.0058	0.0087	0.0094	0.0027	7
รวม	28					จำนวนภาพที่ระบุได้ถูกต้อง			22
						ความถูกต้องคิดเป็นร้อยละ			78.57

จากตารางที่ 4.2 จะยกตัวอย่างของการระบุตัวบุคคลที่ผิดพลาด คือ บุคคลที่ 1 รูปภาพที่ 1 เมื่อเปรียบเทียบกับฐานข้อมูลแล้วพบว่าค่าความผิดพลาดที่น้อยที่สุดคือ เมื่อเปรียบเทียบกับบุคคลที่ 3 มีค่าความผิดพลาดอยู่ที่ 0.0059 ซึ่งน้อยที่สุดเมื่อเปรียบเทียบกับบุคคลอื่นๆ ในฐานข้อมูล ดังนั้นระบบจะระบุว่าภาพที่รับเข้ามาทดสอบเป็นบุคคลที่ 3 โดยช่องที่ระบายแถบสีไว้คือช่องที่มีค่าความผิดพลาดน้อยที่สุดเมื่อนำบุคคลที่ต้องการเปรียบเทียบมาเปรียบเทียบกับบุคคลในฐานข้อมูล โดยสามารถแสดงภาพมือของบุคคลที่รับเข้ามาทดสอบคือบุคคลที่ 1 ภาพที่ 1 ได้ดังรูปที่ 4.3 (ก) และภาพมือของบุคคลที่ใช้เป็นฐานข้อมูลที่ถูกระบุว่าเป็นบุคคลที่ 3 ได้ดังรูปที่ 4.3 (ข)



(ก) (ข)

รูปที่ 4.3 ตัวอย่างภาพมือที่ใช้เป็นฐานข้อมูลและภาพมือที่รับเข้ามาทดสอบการระบุบุคคลโดยใช้ลายเส้นบนฝ่ามือ
(ก) บุคคลที่ 1 ภาพที่ 1 ที่รับเข้ามาทดสอบ
(ข) ภาพมือบุคคลที่ 3 ที่ใช้เป็นฐานข้อมูลที่ถูกระบุ

จากรูปที่ 4.3 จะเห็นได้ว่าภาพมือของบุคคลที่รับเข้ามาทดสอบ ไม่ใช่บุคคลเดียวกันกับฐานข้อมูล ดังนั้นการระบุตัวบุคคลของบุคคลที่ 1 ภาพที่ 1 นี้จึงเป็นการระบุตัวบุคคลที่ผิดพลาด

สำหรับสาเหตุของความผิดพลาดนั้นเกิดจากขั้นตอนการแยกเส้นลายมือออกจากฝ่ามือ ที่ยังไม่สามารถแยกเส้นลายมือออกมาได้สมบูรณ์เท่าที่ควร และในขั้นตอนการเก็บจุดพิกัดเส้นลายมือที่เงื่อนไขในการจัดเก็บยังไม่สมบูรณ์ทำให้ยังไม่สามารถจัดเก็บเส้นลายมือได้ทั้งหมด หรือเก็บเส้นลายมือผิดพลาด

ดังนั้นผลการทดลองจากตารางที่ 4.2 แสดงให้เห็นว่าการระบุตัวบุคคลโดยใช้เส้นลายมือของทั้ง 7 บุคคล บุคคลละ 4 ภาพ รวม 28 ภาพ สามารถที่จะระบุบุคคลได้ถูกต้องจำนวน 22 ภาพ

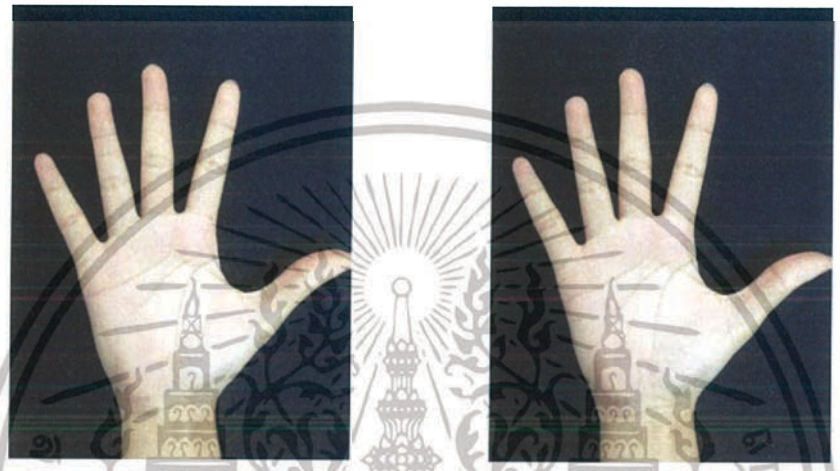
4.3 ผลการทดลองการยืนยันตัวบุคคลโดยใช้เส้นขอบนิ้วมือและลายเส้นบนฝ่ามือ

ในหัวข้อนี้จะแสดงผลการทดลองของการยืนยันตัวบุคคลโดยใช้เส้นขอบนิ้วมือและลายเส้นบนฝ่ามือ โดยใช้บุคคลในการทดสอบจำนวน 7 คน บุคคลละ 4 ภาพ รวมทั้งหมด 28 ภาพ เช่นเดียวกัน โดยนำภาพไปเปรียบเทียบกับข้อมูลที่จัดเก็บไว้ในฐานข้อมูลของบุคคลทั้ง 7 คน โดยจะแสดงผลการเปรียบเทียบค่าความผิดพลาดได้ในตารางที่ 4.3 หากพบว่าผลการเปรียบเทียบกับฐานข้อมูลของบุคคลใดมีความผิดพลาดน้อยที่สุด ระบบจะระบุว่าเป็นบุคคลนั้น

ตารางที่ 4.3 ผลการเปรียบเทียบค่าความผิดพลาดของแต่ละภาพของแต่ละบุคคลกับฐานข้อมูลโดยใช้เส้นขอบนิ้วมือและลายเส้นบนฝ่ามือ

บุคคล	ภาพที่	ค่าความผิดพลาดเมื่อเปรียบเทียบกับบุคคลในฐานข้อมูลบุคคลที่ 1 ถึง 7							บุคคลที่ผิดพลาดน้อยสุด
		1	2	3	4	5	6	7	
1	1	0.0119	0.0246	0.0099	0.0097	0.0134	0.0168	0.0152	4
	2	0.0030	0.0230	0.0093	0.0128	0.0192	0.0186	0.0151	1
	3	0.0077	0.0172	0.0109	0.0091	0.0152	0.0158	0.0109	1
	4	0.0030	0.0230	0.0093	0.0128	0.0192	0.0186	0.0151	1
2	1	0.0216	0.0026	0.0208	0.0273	0.0296	0.0227	0.0227	2
	2	0.0230	0.0021	0.0219	0.0271	0.0300	0.0226	0.0225	2
	3	0.0235	0.0036	00227	0.0296	0.0322	0.0248	0.0254	2
	4	0.0221	0.0031	0.0217	0.0280	0.0310	0.0235	0.0230	2
3	1	0.0067	0.0277	0.0051	0.0176	0.0181	0.0209	0.0208	3
	2	0.0060	0.0264	0.0039	0.0163	0.0172	0.0198	0.0189	3
	3	0.0108	0.0225	0.0029	0.0127	0.0183	0.0185	0.0172	3
	4	0.0103	0.0228	0.0027	0.0132	0.0186	0.0183	0.0172	3
4	1	0.0138	0.0348	0.0140	0.0040	0.0091	0.0106	0.0119	4
	2	0.0080	0.0333	0.0128	0.0055	0.0088	0.0109	0.0128	4
	3	0.0140	0.0353	0.0156	0.0041	0.0079	0.0105	0.0123	4
	4	0.0121	0.0307	0.0132	0.0029	0.0086	0.0111	0.0107	4
5	1	0.0177	0.0412	0.0235	0.0157	0.0102	0.0169	0.0187	5
	2	0.0123	0.0327	0.0204	0.0141	0.0124	0.0154	0.0149	1
	3	0.0173	0.0343	0.0196	0.0118	0.0068	0.0132	0.0141	5
	4	0.0175	0.0413	0.0232	0.0154	0.0098	0.0165	0.0184	5
6	1	0.0251	0.0436	0.0272	0.0184	0.0205	0.0140	0.0253	6
	2	0.0166	0.0233	0.0181	0.0101	0.0129	0.0011	0.0138	6
	3	0.0351	0.0493	0.0309	0.0293	0.0357	0.0285	0.0361	6
	4	0.0241	0.0439	0.0275	0.0183	0.0207	0.0139	0.0251	6
7	1	0.0144	0.0256	0.0172	0.0099	0.0106	0.0150	0.0024	7
	2	0.0173	0.0227	0.0176	0.0093	0.0113	0.0147	0.0026	7
	3	0.0156	0.0242	0.0175	0.0096	0.0124	0.0162	0.0031	7
	4	0.0173	0.0236	0.0172	0.0093	0.0120	0.0148	0.0028	7
รวม	28					จำนวนภาพที่ระบุได้ถูกต้อง			26
						ความถูกต้องคิดเป็นร้อยละ			92.86

จากตารางที่ 4.3 จะยกตัวอย่างการระบุบุคคลโดยใช้ค่าความผิดพลาดที่น้อยที่สุด คือ บุคคลที่ 3 ภาพที่ 1 เมื่อเปรียบเทียบกับฐานข้อมูลแล้วพบว่าค่าความผิดพลาดที่น้อยที่สุดคือ เมื่อเปรียบเทียบกับบุคคลที่ 3 มีค่าความผิดพลาด 0.0051 ซึ่งน้อยที่สุดเมื่อเปรียบเทียบกับบุคคลอื่นๆ ดังนั้นระบบจะระบุว่าภาพที่รับเข้ามาทดสอบเป็นบุคคลที่ 3 โดยช่องที่ระบายแถบสีไว้คือช่องที่มีค่าความผิดพลาดน้อยที่สุดเมื่อนำบุคคลที่ต้องการเปรียบเทียบมาเปรียบเทียบกับบุคคลในฐานข้อมูล โดยสามารถแสดงภาพมือของบุคคลที่รับเข้ามาทดสอบคือบุคคลที่ 3 ภาพที่ 1 ได้ดังรูปที่ 4.4 (ก) และภาพมือของบุคคลที่ใช้เป็นฐานข้อมูลที่ถูกระบุว่าเป็นบุคคลที่ 3 ได้ดังรูปที่ 4.4 (ข)



(ก) (ข)

รูปที่ 4.4 ตัวอย่างภาพมือที่ใช้เป็นฐานข้อมูลและภาพมือที่รับเข้ามาทดสอบการระบุบุคคลโดยใช้เส้นขอบนิ้วมือและลายเส้นบนฝ่ามือ
(ก) บุคคลที่ 3 ภาพที่ 1 ที่รับเข้ามาทดสอบ
(ข) ภาพมือบุคคลที่ 3 ที่ใช้เป็นฐานข้อมูลที่ถูกระบุ

จากรูปที่ 4.4 จะเห็นได้ว่าภาพมือของบุคคลที่รับเข้ามาทดสอบ เป็นบุคคลเดียวกับฐานข้อมูล ซึ่งตรงกับข้อมูลจากตารางที่ระบุว่าเมื่อเปรียบเทียบความผิดพลาดแล้วมีค่าความผิดพลาดน้อยที่สุด

อย่างไรก็ตามบุคคลที่ 3 ภาพที่ 1 นี้เมื่อใช้เส้นขอบนิ้วมือแล้วระบุตัวบุคคลได้ถูกต้องตามตารางที่ 4.1 แต่เมื่อใช้ลายเส้นบนฝ่ามือในการระบุบุคคลแล้วจะผิดพลาดตามตารางที่ 4.2 และเมื่อใช้เส้นขอบนิ้วมือและลายเส้นบนฝ่ามือร่วมกันในการระบุบุคคลจะระบุตัวบุคคลได้ถูกต้องตามตารางที่ 4.3 สาเหตุที่การระบุตัวบุคคลในตารางที่ 4.3 สามารถระบุได้ถูกต้องเนื่องมาจาก ตารางที่ 4.1 การระบุตัวบุคคลโดยใช้เส้นขอบนิ้วมือมีค่าความผิดพลาดเพียง 7.6E-5 ซึ่งเป็นจำนวนที่น้อยมาก เมื่อนำมารวมกับค่าความผิดพลาดจากการระบุตัวบุคคลโดยใช้ลายเส้นบนฝ่ามือในตารางที่ 4.2 ซึ่งมีค่าความผิดพลาด 0.0051 จะได้ค่าความผิดพลาดรวมตามตารางที่ 4.3 อยู่ที่ 0.0051 เมื่อเปรียบเทียบค่าความผิดพลาดนี้กับบุคคลอื่นๆในฐานข้อมูลของตารางที่ 4.3 จึงมีค่าความผิดพลาดที่น้อยที่สุด

ดังนั้นผลการทดลองจากตารางที่ 4.3 แสดงให้เห็นว่าการระบุตัวบุคคลโดยใช้ลายเส้นบนฝ่ามือของทั้ง 7 บุคคล บุคคลละ 4 ภาพ รวม 28 ภาพ สามารถที่จะระบุบุคคลได้ถูกต้องจำนวน 26 ภาพ

อย่างไรก็ตามผลการทดลองดังกล่าวนี้ยังเป็นเพียงผลการทดลองในเบื้องต้นที่แสดงถึงวิธีการระบุตัวบุคคลโดยนำภาพที่รับเข้ามาเปรียบเทียบกับบุคคลในฐานข้อมูลแล้วใช้ค่าความผิดพลาดที่น้อยที่สุดในการระบุตัวบุคคล ยังไม่สามารถนำไปใช้อ้างอิงใดๆได้ เนื่องจากจำนวนบุคคลที่ใช้ในการทดลองมีจำนวนน้อยคือใช้เพียง 7 บุคคลในการทดลอง อีกทั้งจากผลการทดลองจะเห็นได้ว่าในตารางที่ 4.2 การเปรียบเทียบค่าความผิดพลาดของแต่ละภาพของแต่ละบุคคลกับฐานข้อมูลโดยใช้ลายเส้นบนฝ่ามือยังพบความผิดพลาดของการระบุตัวบุคคลจำนวน 6 ภาพ จากทั้งหมด 28 ภาพ และในตารางที่ 4.3 การเปรียบเทียบค่าความผิดพลาดของแต่ละภาพของแต่ละบุคคลกับฐานข้อมูลโดยใช้เส้นขอบนิ้วมือและลายเส้นบนฝ่ามือ ยังพบความผิดพลาดของการระบุตัวบุคคลจำนวน 2 ภาพจากจำนวนทั้งหมด 28 ภาพ ทั้งนี้สาเหตุที่จำนวนบุคคลที่ใช้ในการทดลองมีจำนวนน้อย เนื่องมาจากปัญหาในขั้นตอนการแยกเส้นลายมือออกจากฝ่ามือที่ยังไม่สามารถแยกเส้นลายมือออกมาได้สมบูรณ์ และขั้นตอนในการจัดเก็บเส้นลายมือที่ยังไม่สามารถระบุเงื่อนไขในการจัดเก็บเส้นลายมือให้สามารถเก็บเส้นลายมือได้อย่างถูกต้องและครบถ้วน

บทที่ 5
สรุปผลการทดลอง ปัญหาและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

ปริญญานิพนธ์ฉบับนี้ได้นำเสนอวิธีการในการแยกนิ้วมือออกจากภาพมือ และเส้นลายมือออกจากภาพฝ่ามือเพื่อการระบุตัวบุคคลโดยใช้ผลตอบสนองอิมพัลส์ของระบบ FIR จากการทดลองและเขียนโปรแกรมในโปรแกรม MATLAB สามารถแยกเส้นขอบนิ้วมือออกจากภาพมือ และแยกภาพฝ่ามือออกจากมือได้ แต่ในส่วนของการแยกเส้นลายมือออกจากภาพฝ่ามือนั้นยังไม่สามารถที่จะปรับปรุงภาพและแยกเส้นลายมือออกมาได้สำเร็จเท่าที่ควร อีกทั้งในขั้นตอนของการเก็บเส้นลายมือ เงื่อนไขที่กำหนดไว้ยังไม่สามารถที่จะจัดเก็บเส้นลายมือได้อย่างถูกต้องและครบถ้วน อย่างไรก็ตามจากการทดลองโปรแกรมการยืนยันตัวบุคคลซึ่งแบ่งออกเป็น 3 ส่วน คือการยืนยันตัวบุคคลโดยใช้เส้นขอบนิ้วมือ การยืนยันตัวบุคคลโดยใช้เส้นลายมือ และการยืนยันตัวบุคคลโดยใช้เส้นขอบนิ้วมือและเส้นลายมือร่วมกัน โดยการคัดเลือกรูปภาพจากฐานข้อมูล KMITL ที่สามารถแยกในส่วนของเส้นขอบนิ้วมือและเส้นลายมือออกมาได้ทั้งสองส่วน ได้บุคคลทั้งหมดจำนวน 7 บุคคล บุคคลละ 5 ภาพ จากนั้นนำภาพ 1 ภาพของทุกบุคคลจัดทำเป็นฐานข้อมูล และใช้ภาพที่เหลืออีก 4 ภาพของทุกบุคคลเป็นภาพในการทดลองการยืนยันตัวบุคคลรวมทั้งหมด 28 ภาพ ในส่วนของการยืนยันตัวบุคคลโดยใช้เส้นขอบนิ้วมือสามารถยืนยันตัวบุคคลได้ถูกต้อง 28 ภาพจากทั้งหมด 28 ภาพ ความถูกต้องคิดเป็นร้อยละ 100 การยืนยันตัวบุคคลโดยใช้เส้นลายมือสามารถยืนยันตัวบุคคลได้ถูกต้องจำนวน 26 ภาพจากทั้งหมด 28 ภาพ ความถูกต้องคิดเป็นร้อยละ 78.57 และการยืนยันตัวบุคคลโดยใช้เส้นขอบนิ้วมือและเส้นลายมือร่วมกันสามารถยืนยันตัวบุคคลได้ถูกต้องจำนวน 26 ภาพ จากทั้งหมด 28 ภาพ ความถูกต้องคิดเป็นร้อยละ 92.86

5.2 ปัญหาและข้อเสนอแนะ

สำหรับผลการทดลองที่ได้มานั้น จะมีการใช้รูปภาพในการระบุตัวบุคคลเพียง 7 บุคคลเท่านั้น ซึ่งจะแยกการวิเคราะห์ออกเป็นสองส่วนคือการหาเส้นขอบนิ้วมือและการหาเส้นลายมือ ในส่วนแรกทำการหาเส้นขอบนิ้วมือเพียงอย่างเดียว ในขั้นตอนของการปรับปรุงภาพและแยกเส้นขอบนิ้วมือออกมานั้นสามารถที่จะปรับปรุงภาพและแยกเส้นลายมือออกมาได้เป็นส่วนใหญ่ เนื่องจากภาพมือจะมีระดับสีที่แตกต่างกับพื้นหลังอย่างชัดเจน จึงทำให้สามารถที่หาเส้นขอบนิ้วมือได้ ปัญหาที่พบส่วนมากจะเกิดจากขั้นตอนการปรับปรุงภาพในส่วนของฝ่ามือและขั้นตอนการจัดเก็บเส้นลายมือ ซึ่งสามารถแยกปัญหาที่พบได้เป็น 2 ส่วน ในส่วนแรกเนื่องจากเส้นลายมือที่อยู่บนฝ่ามือไม่ได้มีความแตกต่างอย่างชัดเจนเช่นเดียวกับการหาเส้นขอบนิ้วมือที่ภาพมือจะมีระดับสีที่แตกต่างกับพื้นหลังอย่างชัดเจน ทำให้การปรับปรุงภาพฝ่ามือและแยกเส้นลายมือออกจากฝ่ามือยังไม่สำเร็จเท่าที่ควร เช่น เส้นลายมือที่แยกออกมาได้ไม่มีความต่อเนื่อง เป็นต้น ภาพที่สามารถนำมาหาเส้นลายมือได้จึงมีจำนวนน้อย

ในส่วนที่สองด้วยลักษณะของเส้นลายมือที่จะมีเส้นลายมือเส้นหลักที่เห็นได้ชัดประมาณ 3 ถึง 4 เส้น แต่ในเส้นลายมือเส้นหลักนี้ก็จะมีเส้นลายมือที่แยกออกจากเส้นลายมือเส้นหลักด้วย ซึ่งจุดประสงค์ของปริญญานิพนธ์ฉบับนี้คือการหาเส้นลายมือเส้นหลักเท่านั้น อีกทั้งขั้นตอนการเก็บจุด

พิกัดเส้นลายมือของโปรแกรมในปริญญานิพนธ์ฉบับนี้ ยังไม่ครอบคลุมเงื่อนไขที่จะจัดเก็บเพียงจุดพิกัดเส้นลายมือเส้นหลัก ทำให้ยังไม่สามารถที่จะจัดเก็บจุดพิกัดของเส้นลายมือให้สมบูรณ์ได้

จากปัญหาที่พบใน 2 ส่วนดังกล่าว จึงมีข้อเสนอแนะดังนี้

1. ในขั้นตอนการปรับปรุงภาพฝ่ามือนั้นต้องมีการศึกษาข้อมูลเพิ่มเติมเพื่อหาวิธีการที่สามารถปรับปรุงภาพให้มีความเหมาะสมในการแยกเส้นลายมือออกจากภาพฝ่ามือ
2. ขั้นตอนการเก็บจุดพิกัดเส้นลายมือของโปรแกรมต้องมีการกำหนดเงื่อนไขเพิ่มเติม ในกรณีที่เจอจุดแยกหรือเจอเส้นลายมือที่แยกออกมาจากเส้นลายมือเส้นหลัก เพื่อให้สามารถจัดเก็บเส้นลายมือ 3 เส้นหลักได้อย่างถูกต้อง
3. ศึกษาวิธีการที่จะแยกเส้นลายมือเส้นหลักเพียงอย่างเดียว โดยไม่ให้มีเส้นลายมือที่แยกออกจากเส้นลายมือเส้นหลักถูกแยกออกมาด้วย เพื่อความสะดวกในการจัดเก็บจุดพิกัด
4. สำหรับกรณีที่เส้นลายมือเส้นหลักถูกแยกออกมาแล้วเส้นลายมือไม่ต่อเนื่อง อาจใช้วิธีการต่อเส้นลายมือทั้งสองเส้นเข้าด้วยกัน

บรรณานุกรม

[1] ณัฐภัทร ธีรเบญจกุล, "การระบุตัวบุคคลด้วย FIR SYSTEM", วิทยานิพนธ์วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมการวัดคุม, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2557

[2] Rafael C. Gonzalez, and Richard E. Woods, "Digital Image Processing", New Jersey : Prentice-Hall, Inc. 1993.

[3] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, "Digital image processing using MATLAB", NJ : Pearson/Prentice Hall, Inc. 2004.

[4] John Canny, "A Computational Approach to Edge Detection", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6, NOVEMBER 1986, pp. 679-698.

[5] Badrinath G. S., Naresh Kumar Kachhi and Phalguni Gupta, "Palmprint based Verification System Robust to Occlusion using Low-order Zernike Moments of Sub-images", BMVC, 2009, pp. 1-11.

[6] Xiangqian Wu, David Zhang and Kuanquan Wang, "Palm Line Extraction and Matching for Personal Authentication", IEEE Transactions on Systems, Man and Cybernetics, vol. 36, no. 5, September 2006. pp. 978-987.

[7] Slobodan Ribaric and Ivan Fratric, "An Online Biometric Authentication System Based on Eigenfingers and Finger-Geometry", EUSIPCO, 2005

[8] Digital Image Processing, "Chapter 6: Color Image Processing", May, 2016. [Online] Available : https://slideplayer.com/slide/7019748/ (Feb 13, 2019)

[9] ผศ.ดร. อุรฉัตร โคแก้ว, "Color Model (Part Model (Part Model (Part I)", [ระบบออนไลน์] แหล่งที่มา : https://home.kku.ac.th/urachart/graphic/load/color2 (สืบค้นเมื่อ 21 พฤษภาคม 2562)

[10] Trucco and Jain et al., "Edge detection", [Online] Available : https://www.cse.unr.edu/~bebis/CS791E/Notes/EdgeDetection.pdf (May 20, 2019)

[11] "EDGE DETECTION", [Online] Available : https://candokamera.wordpress.com/edge-detection/ (May 21, 2019)

[12] MACHINE VISION, "Chapter 5 Edge Detection", [Online] Available : http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter5.pdf (Dec 19, 2018)

[13] "การวิเคราะห์ภาพดิจิทัล", [ระบบออนไลน์] แหล่งที่มา : http://www.cpe.eng.cmu.ac.th/wp-content/uploads/CPE752_07.pdf (สืบค้นเมื่อ 21 พฤษภาคม 2562)

บรรณานุกรม (ต่อ)

[14] ณัฏฐ์ อรุณ, "การบีบอัดข้อมูลภาพดิจิทัลโดยใช้ฟังก์ชันการแปลโคซายน์แบบไม่ต่อเนื่อง",2553. [ระบบออนไลน์] แหล่งที่มา : https://www.bu.ac.th/knowledgecenter/executive_journal/30_2/pdf/aw21.pdf (สืบค้นเมื่อ 21 พฤษภาคม 2562)

[15] CAMP, "1D and 2D Gaussian Derivatives", May 20, 2010. [Online] Available : http://campar.in.tum.de/Chair/HaukeHeibelGaussianDerivatives (Feb 13, 2019)

[16] "How to crop and save plot square on an image using 4 points coordinates", Sep 26, 2017. [Online] Available : https://www.mathworks.com/matlabcentral/answers/358442-how-to-crop-and-save-plot-square-on-an-image-using-4-points-coordinates (Sep 12, 2018)

[17] "แนะนำสู่การประมวลผลภาพดิจิตอล" , [ระบบออนไลน์] แหล่งที่มา : http://fivedots.coe.psu.ac.th/~montri/Teaching/image/chap1.htm (สืบค้นเมื่อ 21 พฤษภาคม 2562)

[18] ผศ.ดร.พนมขวัญ ริยะมงคล, Digital Image Processing "Chapter6: Color Image Processing", [ระบบออนไลน์] แหล่งที่มา : http://www.ecpe.nu.ac.th/panomkhawn/imagepro/pdf/ch06.pdf (สืบค้นเมื่อ 21 พฤษภาคม 2562)

[19] "smoothing splines in matlab (csaps)", Dec 10, 2003. [Online] Available : http://mathforum.org/kb/message.jspa?messageID=956102 (Oct 10, 2018)

ภาคผนวก

เข้าร่วมงาน ICA SP-CON 2019 ครั้งที่ 10 ณ สถาบันเทคโนโลยีนานาชาติสิรินธร

เป็นงานที่นำโครงงานของนักศึกษาชั้นปีที่ 4 คณะวิศวกรรมศาสตร์ของสาขาวิชาที่เกี่ยวข้องกันอันได้แก่ สาขาวัดคุม ควบคุม และอัตโนมัติ มานำเสนอในงานประชุม

IEEE THAILAND SECTION — ICA SP-CON 2019 — ECTI Association

This is to certify that

Sawakarn Hiriwatanawong

has successfully completed the presentation of the project entitled

Palm Lines and Finger Contours Extraction for Personal Identification using FIR System

at the Tenth Instrumentation, Control, and Automation Senior Project Conference (ICA SP-CON 2019) on April 30, 2019 at Sirindhorn International Institute of Technology, Thammasat University, Thailand.

Assoc. Prof. Dr. Waree Kongprawechnon
SIIT, Thammasat University
ICA SP-CON 2019 General Chair

Asst. Prof. Dr. Itthisek Nilkhamhang
SIIT, Thammasat University
ICA SP-CON 2019 Technical Program Chair

และจากการเข้าร่วมงานในครั้งนี้ยังได้รางวัลการนำเสนอโปสเตอร์ดีเด่นด้วย

IEEE THAILAND SECTION — ICA SP-CON 2019 — ECTI Association

This is to certify that

Sawakarn Hiriwatanawong

has received the **BEST POSTER PRESENTATION AWARD** for the project entitled

Palm Lines and Finger Contours Extraction for Personal Identification using FIR System

at the Tenth Instrumentation, Control, and Automation Senior Project Conference (ICA SP-CON 2019) on April 30, 2019 at Sirindhorn International Institute of Technology, Thammasat University, Thailand.

Assoc. Prof. Dr. Waree Kongprawechnon
SIIT, Thammasat University
ICA SP-CON 2019 General Chair

Asst. Prof. Dr. Itthisek Nilkhamhang
SIIT, Thammasat University
ICA SP-CON 2019 Technical Program Chair