บทที่ 1
บทนำ

1.1. ที่มาของโครงการ

ในการตรวจเช็คค่าตัวเลขการใช้น้ำหรือไฟฟ้าจากมิเตอร์ตามบ้านเรือนนั้น ซึ่งโดยปกติแล้วจะมีเจ้าหน้าที่คอยจดบันทึกค่าตัวเลขการใช้น้ำหรือไฟฟ้าจากมิเตอร์ในแต่ละบ้าน ลงในสมุดบันทึก และนำตัวเลขที่บันทึกนั้นไปป้อน ตรวจสอบ และคำนวณ การใช้น้ำหรือไฟฟ้าที่ใช้ไปในแต่ละเดือนเป็นกี่หน่วย และในแต่ละเดือนจะต้องจ่ายค่าน้ำหรือไฟฟ้าเท่าไหร่ ซึ่งพนักงานมีโอกาสที่จะเกิดข้อผิดพลาดขึ้นได้ อาจเนื่องมาจาก การทำงานเดิม ๆ ซ้ำ ๆ เป็นระยะเวลานาน ซึ่งปัญหาที่เกิดขึ้นเหล่านี้เป็นการทำให้เกิดผลเสียต่อสำนักงาน

โปรแกรมการอ่านค่าตัวเลขการใช้น้ำหรือไฟฟ้าจากมิเตอร์ เป็นโปรแกรมที่พัฒนาขึ้นมาเพื่อที่จะช่วยในการอ่านค่าตัวเลขการใช้น้ำหรือไฟฟ้าจากมิเตอร์ โดยใช้แนวคิดที่จะใช้การถ่ายภาพแทนการจดบันทึก โดยใช้วิธีการตรวจจับภาพและใช้โปรแกรมช่วยในการวิเคราะห์ภาพเพื่อให้คอมพิวเตอร์สามารถอ่านและเข้าใจได้ว่าภาพที่ถ่ายจากมิเตอร์น้ำหรือไฟฟ้านั้นมีตัวเลขใด ซึ่งการใช้โปรแกรมนั้นเป็นการช่วยแก้ปัญหาการอ่านตัวเลขที่ผิดพลาด เนื่องจากข้อผิดพลาดที่เกิดจากมนุษย์นั่นเอง

นอกจากนี้ตัวโปรแกรมยังสามารถที่จะพัฒนาเพื่อใช้ประยุกต์ใช้กับโปรแกรมอื่น ๆ ได้อีกด้วย เช่น โปรแกรมที่ใช้ตรวจสอบมิเตอร์น้ำหรือไฟฟ้าออนไลน์ ซึ่งสามารถที่จะตรวจสอบการใช้น้ำหรือไฟฟ้าจากอินเตอร์เน็ตได้ด้วยตัวเอง และนอกจากนี้การใช้โปรแกรมนั้นเป็นการช่วยอำนวยความสะดวกต่าง ๆ แก่เจ้าหน้าที่พนักงาน ได้หลากหลายสถานการณ์

1.2. วัตถุประสงค์ของโครงการ

พัฒนาโปรแกรมที่สามารถวิเคราะห์ค่าตัวเลขการใช้น้ำหรือไฟฟ้าบนมิเตอร์ จากภาพถ่ายได้อย่างถูกต้อง เพื่อนำไปประยุกต์ใช้กับโปรแกรมประยุกต์อื่น ๆ ตามวัตถุประสงค์ของผู้ใช้

1.3. ขอบเขตของโครงการ

พัฒนาโปรแกรมที่สามารถอ่านค่าตัวเลขการใช้น้ำหรือไฟฟ้าจากมิเตอร์ที่เป็นแบบมาตรฐานได้อย่างถูกต้อง โดยได้จากการวิเคราะห์ภาพถ่ายเพื่อให้ได้ข้อมูลตัวเลขนั้นมา

1.4. แผนการดำเนินงาน

1. ศึกษา ค้นคว้า เรื่อง Image processing และการเขียน C++
2. นำข้อมูลที่ได้มาใช้ในการออกแบบโปรแกรม
3. พัฒนาโปรแกรมที่ได้ออกแบบไว้
4. ทดสอบโปรแกรมเพื่อหาข้อผิดพลาดและแก้ไขโปรแกรม
5. เขียนคู่มือการใช้งานและสรุปโครงการ

1.5. ขั้นตอนการดำเนินงาน

โครงการนี้มีระยะเวลาในการจัดทำ รวมทั้งสิ้น 9 เดือน โดยเริ่มตั้งแต่เดือนมิถุนายน ถึงเดือนกุมภาพันธ์ พ.ศ. 2555 ซึ่งมีขั้นตอนการดำเนินงานแสดงดังตารางที่ 1.1

ตารางที่ 1.1 ตารางแสดงขั้นตอนการดำเนินงาน

ลำดับ	แผนการดำเนินงาน	ระยะเวลา	2554							2555	
			มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.
1	ศึกษาค้นคว้าเรื่อง การประมวลผลภาพ และการเขียน C++	4 สัปดาห์									
2	นำข้อมูลที่ได้มาใช้ในการออกแบบโปรแกรม	5 สัปดาห์									
3	พัฒนาโปรแกรมที่ได้ออกแบบไว้	12 สัปดาห์									
4	ทดสอบโปรแกรมเพื่อหาข้อผิดพลาดและแก้ไขโปรแกรม	8 สัปดาห์									
5	เขียนคู่มือการใช้งานและสรุปโครงการ	6 สัปดาห์									

1.6. ประโยชน์ที่คาดว่าจะได้รับ

1. ได้โปรแกรมที่สามารถวิเคราะห์ตัวเลขการใช้น้ำหรือไฟฟ้าจากภาพถ่ายได้อย่างถูกต้อง เพื่อนำไปใช้ประโยชน์ต่อไปได้
2. ได้รับความรู้เรื่อง Image processing และการเขียน C++

บทที่ 2
ความรู้พื้นฐาน และทฤษฎีที่เกี่ยวข้อง

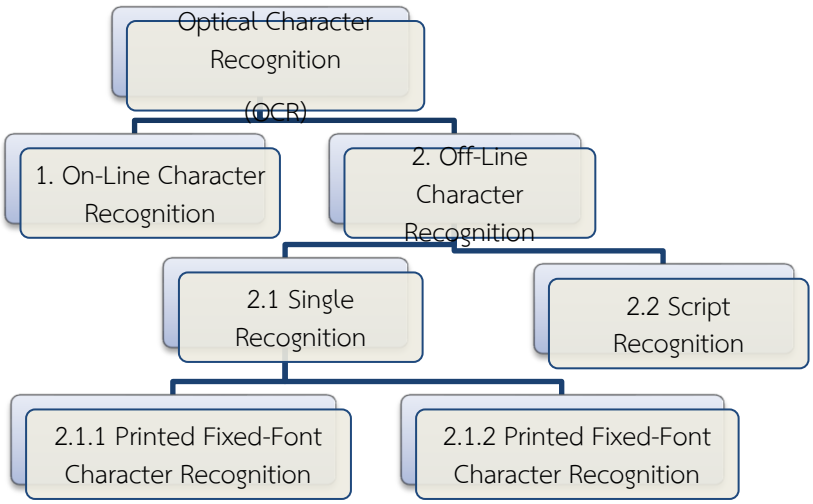
ในบทนี้จะกล่าวถึงความรู้พื้นฐาน และทฤษฎีที่เกี่ยวข้อง ที่ใช้ในการทำโครงการ

2.1. ความหมายของ OCR (Optical Character Recognition)

OCR (Optical Character Recognition) คือ การแปลงข้อมูลที่อยู่ในรูปแบบของเอกสารรูปภาพ (Image File) ให้อยู่ในรูปแบบของเอกสารข้อความ(Text File) ซึ่งประโยชน์ของ OCR คือ สะดวกในการปรับแต่ง แก้ไขเอกสาร และประหยัดพื้นที่ในการจัดเก็บข้อมูล เป็นต้น

OCR (Optical Character Recognition) สามารถแบ่งตามคุณลักษณะตามแหล่งที่มาของตัวอักษร[3] ดังแสดงในรูปที่ 1.1 ได้ดังนี้

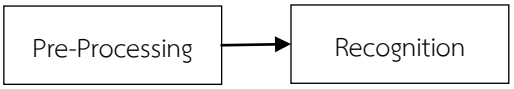
1. การรู้จำตัวอักษรแบบออนไลน์ (On-line Character Recognition)
2. การรู้จำตัวอักษรแบบออฟไลน์ (Off-line Character Recognition)
    2.1. ตัวอักษรโดด (Single Character)
        2.1.1. การรู้จำตัวพิมพ์แบบฟอนต์เฉพาะ (Printed Fixed-Font Character Recognition)
        2.1.2. การรู้จำลายมือเขียนแบบตัวโดด (Isolated Handprint Character Recognition)
    2.2. การรู้จำลายมือแบบเขียนต่อเนื่อง (Script recognition)



รูปที่ 2.1 แผนผังประเภทของ OCR[3]

2.2. ขั้นตอนการทำงานของ OCR (Optical Character Recognition)[3]

ขั้นตอนการทำงานของ OCR แบ่งออกเป็น 2 ขั้นตอนหลัก ๆ ดังแสดงในรูปที่ 2.2 ดังนี้



รูปที่ 2.2 รูปขั้นตอนการทำงานของ OCR[3]

2.2.1. ขบวนการประมวลผลขั้นต้น (Pre-Processing)[3]

ขบวนการประมวลผลขั้นต้น (Pre-Processing) เป็นขั้นตอนแรกของกระบวนการที่เรียกว่า OCR เพราะก่อนที่โปรแกรมจะสามารถวิเคราะห์ลักษณะของตัวอักษรว่าเป็นตัวอักษรใดได้นั้น จะต้องวิเคราะห์จากภาพที่มีความเหมาะสมในการตรวจสอบและวิเคราะห์ ซึ่งในบางครั้งภาพตัวอย่างที่ได้มาอาจมีการรบกวนจากสภาพแวดล้อมต่าง ๆ เช่น แสงสะท้อน เปื้อนโคลน ตัวอักษรเอียง เป็นต้น จึงต้องอาศัยกระบวนการนี้มาช่วย ซึ่งแบ่งออกเป็นขั้นตอนดังต่อไปนี้

2.2.1.1. การกรองข้อมูลแทรกซ้อน (Noise Filtering)[3]

การกรองข้อมูลแทรกซ้อน (Noise Filtering) เป็นวิธีการที่ทำหน้าที่กำจัดสิ่งที่ไม่พึงประสงค์ที่ปรากฏอยู่บนภาพที่ต้องการจะนำมาประมวลผล ซึ่งเป็นสาเหตุสำคัญที่ทำให้ผลลัพธ์ที่ได้มีความผิดพลาด จึงจำเป็นอย่างมากที่ต้องจัดการกับปัจจัยรบกวนเหล่านี้ วิธีในการกำจัดสิ่งรบกวนของภาพนั้นมีหลายวิธี แต่วิธีหนึ่งที่น่าสนใจ คือ วิธีการที่นำ Mask มาช่วยในการประมวลผล ซึ่งความสามารถของวิธี Mark คือ สามารถเปลี่ยนขนาดได้ ซึ่งทำให้มีประสิทธิภาพในการกำจัดสิ่งรบกวนมากยิ่งขึ้น แต่การทำ Mark จะทำให้ภาพเสียความคมชัด วิธีการใช้งานคือการนำ Average Filter มาใช้ โดยค่าดังกล่าวหาได้จากค่าเฉลี่ยของ Mask ซึ่งหาได้จากสมการที่ (2.1) และ Mask มีลักษณะดังต่อไปนี้

ตารางที่ 2.1 ตารางแสดง Mask 3x3 [3]

$f(i-1, j-1)$	$f(i, j-1)$	$f(i+1, j-1)$
$f(i-1, j)$	$f(i, j)$	$f(i+1, j)$
$f(i-1, j-1)$	$f(i, j+1)$	$f(i+1, j+1)$

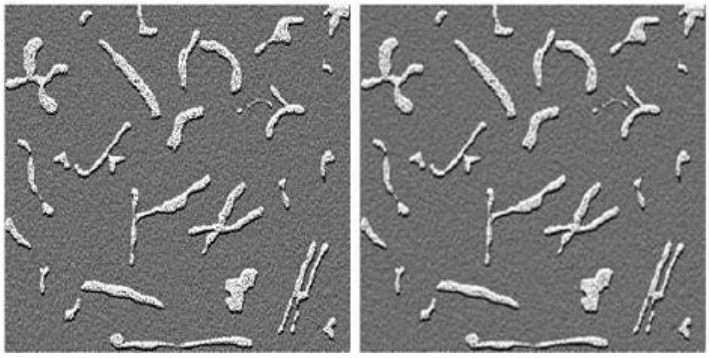
$$g(i,j) = \frac{1}{n} \sum_{i=1,j=1}^{m,n} f(i,j) \qquad (2.1)$$

โดยที่

$g(i,j)$ คือ ค่าเฉลี่ยของตำแหน่ง (i,j) ที่นำ Mask ไปครอบ

$f(i,j)$ คือ ตำแหน่งที่นำ Mask ไปครอบ

m,n คือ ขนาดของ Mask มีขนาดเป็น Square Matrix



(a) (b)

รูปที่ 2.3 (a) ภาพตัวอย่างจากการทำ Mask ภาพต้นฉบับ (b) ภาพที่ถูกแก้ไขแล้ว[3]

จากรูปที่ 2.3 จะพบว่าภาพที่ผ่านการกำจัดสิ่งรบกวนแล้ว อาจมีความคมชัดของภาพลดลง ซึ่งถ้ามองผ่าน ๆ แล้ว อาจคิดว่าทำให้คุณภาพของภาพแย่ลง แต่ในความเป็นจริงแล้วไม่เป็นเช่นนั้น เพราะการลดความคมชัดของภาพสามารถที่จะทดแทนส่วนที่หายไปของภาพได้ นอกจากวิธีข้างต้นแล้วยังมีฟิลเตอร์(Filter) ตัวอื่น ๆ ที่ใช้ในการจำกัดสิ่งรบกวนของภาพ ตัวอย่างเช่น Gaussian Filter ซึ่งมีสมการ ดังนี้

สมการที่ (2.2) สมการ Gaussian Filter [3]

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2/2\sigma^2} \qquad (2.2)$$

สมการที่ (2.3) สมการ Gaussian Filter 2 Dimensions [3]

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \qquad (2.3)$$

โดยที่ σ คือ ค่าความแปรปรวน

x คือ ค่าของตำแหน่งจุดภาพ(Pixel)

y คือ ค่าของตำแหน่งจุดภาพ(Pixel) ในกรณีสองมิติ

$G(x), G(x,y)$ คือ ค่าของตำแหน่งจุดภาพที่ได้จากการคำนวณ

2.2.1.2. การปรับแต่งข้อมูล (Normalization)[3]

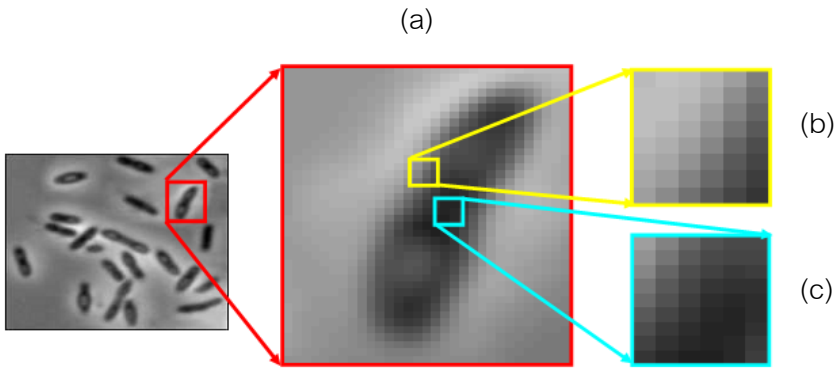
การปรับแต่งข้อมูลเป็นวิธีการปรับแต่งข้อมูลเพื่อให้ภาพที่จะนำไปประมวลผลนั้น มีความเหมาะสมที่ระบบการทำงานต่อไปต้องการ เช่น การปรับขนาดตัวอักษร การทำเกรย์สเกล (Grey Scale) การทำภาพขาวดำ(Binarization) เป็นต้น

2.2.1.3. การตัดแบ่งพื้นที่ใช้งาน (Cropping)[3]

การตัดแบ่งพื้นที่ใช้งานเป็นการแยกเอาเฉพาะส่วนตัวอักษรที่ต้องการนำไปวิเคราะห์ เพื่อความถูกต้องของข้อมูล แต่ในการตัดแบ่งพื้นที่ใช้งานนั้นอาจมีอุปสรรคของตัวอักษร เช่น ตัวอักษรที่เขียนติดกัน ซึ่งจำเป็นจะต้องอาศัยอัลกอริทึมที่ดีในการแยกตัวอักษรออกจากกัน ซึ่งข้อดีของการตัดแบ่งพื้นที่ใช้งานคือ การลดจำนวนข้อมูลรูปภาพที่ไม่จำเป็นในการวิเคราะห์ลง จัดระเบียบข้อมูลในรูปภาพให้เป็นกลุ่มได้ดีขึ้น เป็นต้น หลักการในการแยกองค์ประกอบของภาพ ดังแสดงในรูปที่ 2.4 [3] แบ่งได้ดังนี้

2.2.1.3.1. การแยกองค์ประกอบตามความเหมือนกัน(Similarity) ของคุณสมบัติจุดภาพของรูปภาพที่อยู่ภายในพื้นที่เดียวกัน

2.2.1.3.2. การแยกองค์ประกอบโดยดูจากความไม่ต่อเนื่อง(Discontinuity) ของคุณสมบัติจุดภาพของรูปภาพบริเวณรอยต่อระหว่างวัตถุในภาพกับฉากหลัง



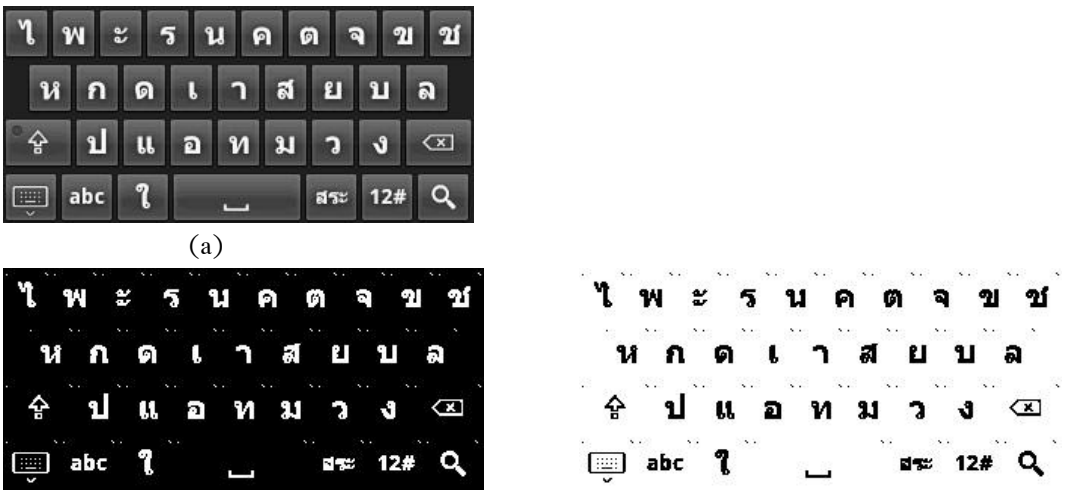
รูปที่ 2.4 การแยกองค์ประกอบของภาพ (a) ภาพที่ขยายจากต้นฉบับ
(b) แยกโดยใช้ความแตกต่างบริเวณรอยต่อ
(c) แยกโดยความเหมือนของจุดภาพ[3]

การตัดแบ่งพื้นที่ใช้งาน ซึ่งกระบวนการหลักๆ ของการตัดแบ่งพื้นที่คือ การนำภาพเข้าไปประมวลผลเพื่อทำให้ให้ภาพที่ได้นั้นสมบูรณ์ที่สุด เพื่อให้ง่ายแก่การวิเคราะห์ จากนั้นเป็นการตรวจสอบลักษณะของจุดภาพและทิศทางของสี และทำการแปลงภาพให้เป็นภาพขาวดำเพื่อง่ายต่อการตรวจสอบขอบของภาพ และสุดท้ายคือการกำหนดลักษณะของกรอบภาพที่ได้ การทำงานโดยรวมอธิบายดังรูปที่ 2.5

รูปที่ 2.5 ภาพกระบวนการทำงานของการตัดแบ่งพื้นที่ใช้งาน(Segmentation Process)[2]

2.2.1.4. Thresholding[2]

Thresholding เป็นวิธีการเปลี่ยนค่าของจุดภาพให้มีเพียงสองค่า(Binary) คือ ขาวและดำ(0,1) ดังแสดงในรูปที่ 2.6



(a) (b) (c)

รูปที่ 2.6 ภาพที่ผ่านการ Thresholding

(a) ภาพต้นฉบับ grey-scale

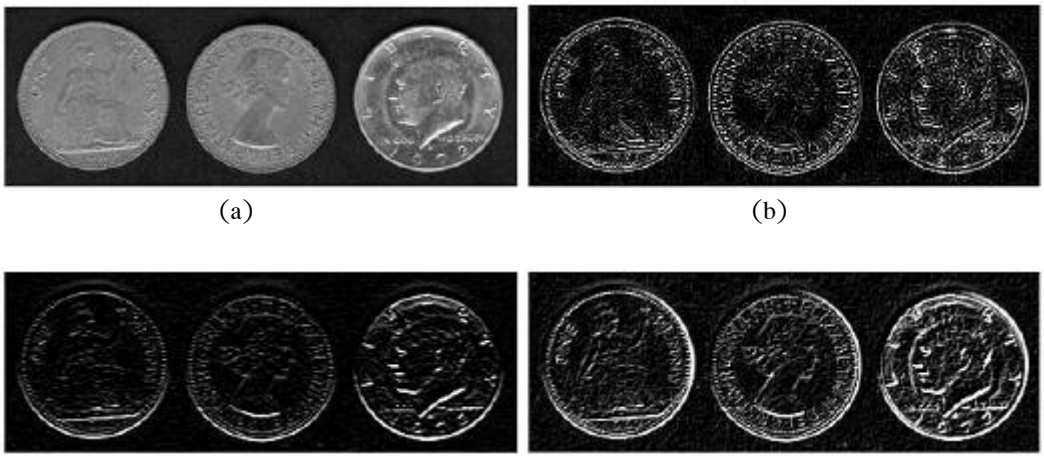
(b) ภาพที่ผ่านการ Thresholding

(c) ภาพที่ผ่านการ Inverted จาก (b)

2.2.1.5. Edge Detection [2]

การตรวจหาเส้นขอบของภาพ(Edge Detection) เป็นวิธีการที่สำคัญในการประมวลผลภาพ เพราะหากมีอัลกอริทึมที่ดี ผลลัพธ์ของภาพที่ตัดแบ่งออกมาจะมีประสิทธิภาพ และส่งผลให้ง่ายต่อการทำงานในขั้นตอนต่อไป ซึ่งขั้นต่อไปคือ กระบวนการการรู้จำ(Recognition) ซึ่งการตรวจหาเส้นขอบของภาพ จะอาศัยค่าความแตกต่างของค่าสีในพื้นที่ของภาพ ซึ่งเปรียบเทียบโดยค่าเกรย์สเกล
(grey scale)

การหาขอบภาพด้วย Mask เป็นการหาวัตถุในภาพ หรือลายเส้นที่มีระดับความแตกต่างของความเข้มแสงมาก ๆ ซึ่งมีวิธีที่ใช้ในการหาหลายวิธี เช่น Sobel Operator, Prewit Operator และ Laplacian Operator เป็นต้น ซึ่งผลลัพธ์ที่ออกมานั้นมีความคล้ายกันอย่างมาก ต่างกันเพียงแค่ Mask ของแต่ละวิธีเท่านั้นเอง ดูผลลัพธ์ของแต่ละวิธีดังรูปที่ 2.7



(a) (b)

(c) (d)

รูปที่ 2.7 ภาพผลลัพธ์ที่ได้จากการหาขอบภาพ[2]

(a) ภาพต้นฉบับ

(b) ภาพการหาขอบด้วยวิธี Laplacian Operator

(c) ภาพการหาขอบด้วยวิธี Prewit Operator

(d) ภาพการหาขอบด้วยวิธี Sobel Operator

2.2.1.6. การสกัดลักษณะสำคัญ (Feature Extraction) [2]

การสกัดลักษณะสำคัญเป็นกระบวนการ ซึ่งจะอยู่ระหว่างขั้นตอนสองขั้นตอนแรกคือ ระหว่างขั้นตอนประมวลผลขั้นต้นและขั้นตอนการรู้จำ การทำงานของการสกัดลักษณะสำคัญมีการทำงานคือ มีการนำเอาโครงสร้างพื้นฐานสำคัญของตัวอักษร(Character) แต่ละตัวอักษรออกมา แล้วนำมาเปรียบเทียบเทียบกับโครงสร้างพื้นฐานที่กำหนดไว้ เช่น ตัวเลขซึ่งประกอบไปด้วยโครงสร้างพื้นฐานต่าง ๆ คือ เส้นตรงทั้งแนวตั้งและแนวนอน เส้นเอียง เส้นโค้ง วงกลม วงรี จุดตัด เป็นต้น และเมื่อได้ผลลัพธ์จากการเปรียบเทียบแล้ว จะถูกส่งต่อไปยังกระบวนการต่อไป

ในกระบวนการประมวลผลภาพ สามารถที่จะใช้ตัวดำเนินการทางคณิตศาสตร์เข้ามาช่วย เช่น Union Intersection Transposition และ XOR เป็นต้น

2.2.1.7. การขยายภาพ(Dilation) และการกร่อนภาพ(Erosion) [2]

การขยายภาพและการกร่อนภาพ ทั้งสองวิธีการเป็นขั้นตอนการดำเนินการทางตรรกะ ดังแสดงในรูปที่ 2.8 การขยายภาพ(Dilation) นั้นเป็นการขยายโครงสร้างของภาพให้ใหญ่ขึ้น ดังสมการที่ (2.4)

สมการ Dilation [2]

$$A \oplus B^t = \{p | B_p \cap A \neq \emptyset\} \quad ; B \text{ is denoted } A \oplus B^t \qquad (2.4)$$

โดยที่ A คือ ภาพที่ต้องการขยายภาพ(Dilation)

B^t คือ ส่วนกลับ(Transposed) ของ B

B_p คือ ค่ากลาง ณ จุด p

การกร่อนภาพ(Erosion) เป็นการตัดภาพ ขยายภาพให้เล็กลง ดังสมการที่ (2.5)

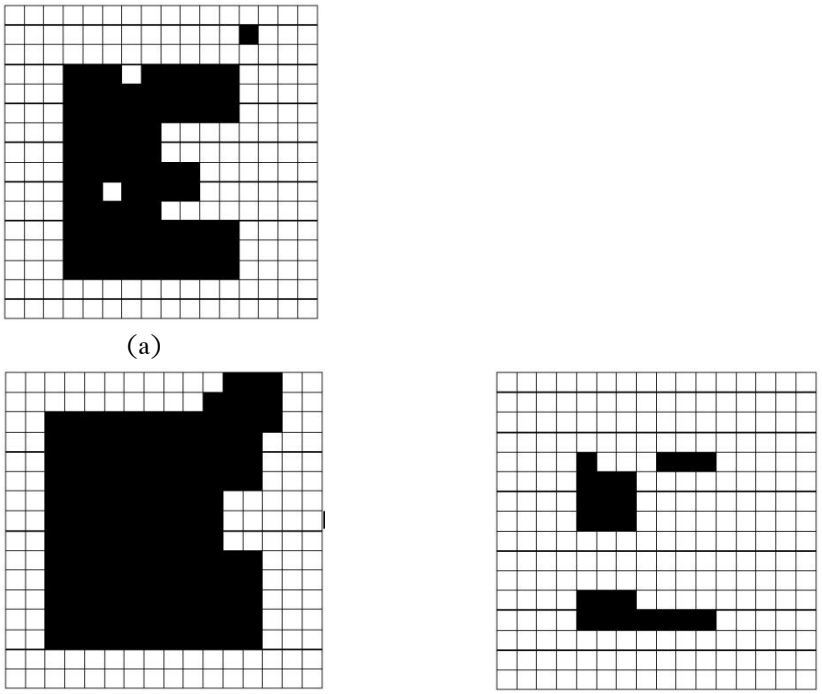
สมการ Erosion [2]

$$A\theta B^t = \{p | B_p \subset A\} \quad ; \; B \; is \; denoted \; A\theta B^t \qquad (2.5)$$

โดยที่ A คือ ภาพที่ต้องการขยายภาพ(Dilation)

B^t คือ ส่วนกลับ(Transposed) ของ B

B_p คือ ค่ากลาง ณ จุด p



(a) (b) (c)

รูปที่ 2.8 Dilation and Erosin [2] (a) ภาพต้นฉบับ(Original Image)
(b) ภาพที่ถูก dilation จากภาพต้นฉบับ
(c) ภาพที่ถูก erosion จากภาพต้นฉบับ

2.2.1.8. การเปิดช่อง(Opening) และการปิดช่อง(Closing) [2]

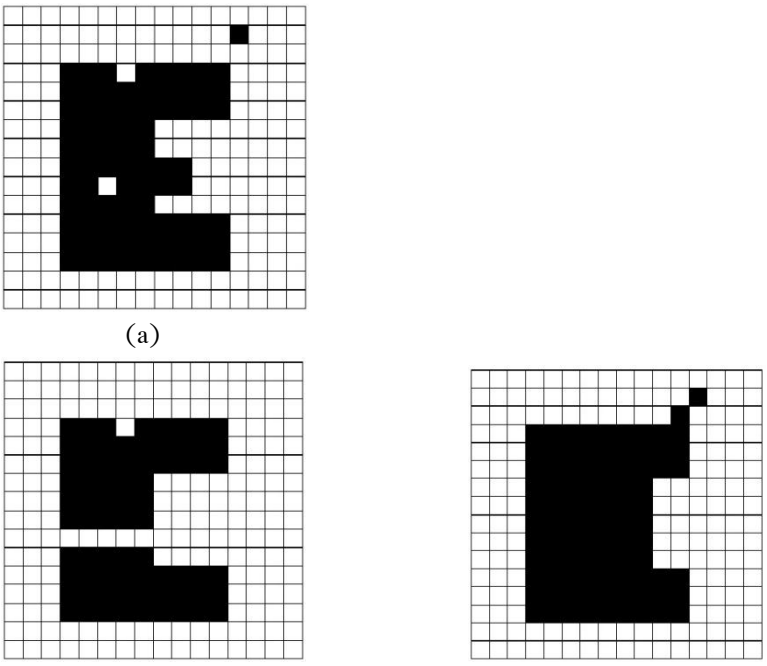
การเปิดช่องและการปิดช่องวิธีทั้งสองเป็นการทำให้วัตถุในภาพมีผิวขอบที่ราบเรียบดังรูปที่ 2.9 คือเป็นการกำจัดส่วนที่ยื่นหรือโผล่ออกมา ซึ่งทั้งสองวิธีจะทำงานแตกต่างกัน โดยการเปิดช่อง(Opening) นั้นเป็นการตัดหรือทำลายส่วนที่เกินมาของวัตถุในภาพ ซึ่งทำให้วัตถุในภาพมีขนาดเล็กลง แต่การปิดช่อง(Closing) เป็นการเติมเต็มส่วนที่โผล่หรือยื่นออกมาของวัตถุในภาพ โดยทั้งสองวิธีการมีสมการ ดังนี้

สมการ Opening ดังสมการที่ (2.6)

$$Opening: \quad (\mathrm{A}\theta\mathrm{B}^{\mathrm{t}}) \oplus B; \quad i.e. erode\ then\ dilate \qquad (2.6)$$

สมการ Closing[5] ดังสมการที่ (2.7)

$$Closing: \quad (A \oplus B^{t})\theta B; \quad i.e. dilate\ then\ erode \qquad (2.7)$$



(a)

(b) (c)

รูปที่ 2.9 ภาพที่ได้จากการทำ Opening และ Closing [2]
(a) เป็นภาพต้นฉบับ(Original Image)
(b) เป็นภาพที่ถูก Opened แล้ว
(c) เป็นภาพที่ถูก Closed จากต้นฉบับ

2.2.2. การรู้จำ (Recognition) [1][2]

การรู้จำเป็นขั้นตอนที่สำคัญของ OCR เพราะเป็นขั้นตอนที่ทำหน้าที่ตัดสินใจว่าภาพที่ส่งเข้าไปวิเคราะห์นั้นเป็นตัวอักษรอะไร ซึ่งมีวิธีการหลากหลายวิธีในการทำงาน ซึ่งสามารถแบ่งออกเป็นหลายวิธี โดยเน้นกฎทางทฤษฎีเป็นหลักในการแบ่งวิธีต่าง ๆ ซึ่งวิธีการต่าง ๆ เหล่านี้มีข้อดีและข้อเสียที่แตกต่างกัน ซึ่งถ้าเราต้องการข้อมูลที่มีความแม่นยำ โปรแกรมจำเป็นที่จะต้องอาศัยวิธีการต่าง ๆ เหล่านี้มาช่วยในการตัดสินใจ ซึ่งวิธีการต่าง ๆ ได้แบ่งออกได้ดังนี้

2.2.2.1. วิธีทางการเข้าคู่รูปแบบ (Template Matching)

เป็นวิธีที่ทางผู้ทำโครงการนี้ใช้ในการรู้จำ ซึ่งการทำงานของวิธีการเข้าคู่รูปแบบ คือ การนำรูปภาพที่ต้องการหาค่า ไปเปรียบเทียบกับรูปแบบตัวอย่าง (Template) ซึ่งรูปแบบตัวอย่างนี้ จะเก็บค่า และกำหนด

ลักษณะสำคัญต่าง ๆ ที่สามารถแยกความแตกต่างของตัวอักษรต่าง ๆ ได้ และเมื่อนำภาพตั้งต้นที่มีมาเปรียบเทียบกับรูปแบบที่มี เพื่อตรวจสอบความคล้ายคลึงกันของภาพทั้งสอง ซึ่งอาศัยค่าระดับที่ตั้งขึ้นมาเพื่อตรวจสอบหรือตัดสินใจ แต่ข้อเสียของวิธีการนี้มีค่อนข้างมาก ซึ่งข้อเสียของวิธีการนี้ คือ มีความอ่อนไหวต่อข้อมูลที่มีการแทรกซ้อน ขนาด และความเอียง ซึ่งปัญหาเหล่านี้สามารถแก้ได้ แต่นั่นต้องอาศัยการทำงานในขั้นตอนประมวลผลขั้นต้นที่ดี

Template Matching เป็นเทคนิคที่นิยมใช้กันอย่างกว้างขวาง เพราะให้ผลค่อนข้างมีประสิทธิภาพ แต่ข้อเสียคือช้า เทคนิคนี้เหมาะสำหรับการหาลักษณะเฉพาะที่มีขนาดไม่ใหญ่บนภาพ เทคนิค Template Matching จะมีความคล้ายคลึงกับเทคนิคที่เรียกว่า Convolution สำหรับเทคนิค Template Matching เราจะทำการเคลื่อน Template ไปยังจุดต่าง ๆ จากซ้ายไปขวาบนลงล่าง เพื่อคำนวณหาค่าความเหมือนของ template กับบริเวณต่าง ๆ บนภาพ ซึ่งขบวนการที่ทำเช่นนี้เรียกว่าการ Correlation และค่าความเหมือนนี้มีชื่ออีกอย่างว่า Correlation Coefficient สามารถคำนวณได้จากสมการที่ (2.8)

$$c = \frac{\sum_{i=1}^{W\times H}(f_i - < f >_i) \times \sum_{j=1}^{N\times M}(w_i - < w >)}{\sqrt{\sum(f_i - < f >_i)^2 \times \sum(w_i - < w >)^2}} \qquad (2.8)$$

c คือค่า Correlation Coefficient

f คือ ค่าความเข้มสีแต่ละจุดสีบนภาพ

w คือ ค่าความเข้มสีแต่ละจุดบน Template

<f> และ <w> คือ ค่าเฉลี่ยสีของภาพ (ขนาดเท่า template) และ template ตามลำดับ

ดังนั้นถ้าบริเวณใดของภาพเหมือนกับ template พอดี ค่า c ที่ได้ก็จะมีค่าเป็น 1 แต่หากไม่เหมือนกัน ค่า c ก็จะมีค่าลดลง

2.2.2.2. วิธีทางสถิติ (Statistical Approach)

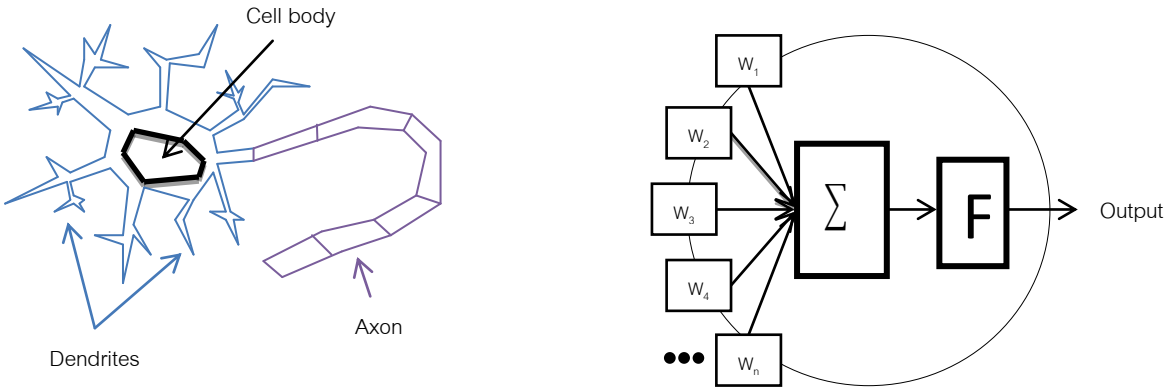
วิธีการนี้อาศัยภาพที่ได้จากขั้นตอนการสกัดลักษณะสำคัญ (Feature Extraction) เมื่อนำภาพเข้าสู่ขั้นตอนการรู้จำเพื่อตรวจสอบลักษณะเฉพาะของตัวอักษรแต่ละตัว ซึ่งอาศัยการตัดสินใจจากหลักการทางสถิติ โดยอาศัยค่าความน่าจะเป็นหรือฟังก์ชั่นการแจกแจงความน่าจะเป็น ซึ่งเมื่อผ่านขั้นตอนแล้ว จะได้ผลลัพธ์ออกมาว่าเป็นตัวอักษรอะไร หรือถ้าต้องการความแม่นยำที่มากขึ้น สามารถนำผลลัพธ์ที่ได้เข้าตรวจสอบกับกระบวนการรู้จำอื่น ๆ ได้

2.2.2.3. วิธีการวิเคราะห์ทางโครงสร้าง (Structural Analysis)

เป็นวิธีที่อาศัยผลลัพธ์ที่ได้จากการสกัดลักษณะสำคัญ(Feature Extraction) โดยอาศัยหลักการทางโครงสร้างของของตัวอักษร เพราะตัวอักษร หรือตัวเลขแต่ละตัวนั้น ล้วนมีองค์ประกอบเฉพาะของแต่ละตัว เพื่อที่จะประกอบเป็นตัวอักษร หรือตัวเลขขึ้นมา โดยในขั้นตอนการรู้จำลักษณะของตัวอักษรต่าง ๆ นั้น จะถูกตรวจสอบหรือวิเคราะห์กฎการเขียนอักษร เช่น โครงสร้างกราฟ หรือ โครงสร้างต้นไม้เป็นต้น เพื่อที่จะสามารถระบุได้ว่าเป็นตัวอักษรใด ซึ่งอาศัยการตัดสินใจโดยพิจารณาที่ลักษณะโครงสร้างเฉพาะของตัวอักษร ซึ่งข้อดีของวิธีการนี้คือ มีความยืดหยุ่นต่อความหลากหลายของตัวอักษร ซึ่งเหมาะแก่การทำโปรแกรม OCR ที่เป็นการแปลงตัวอักษรในหลาย ๆ รูปแบบ

2.2.2.4. วิธีทางโครงข่ายประสาทเทียม (Neural Network)

เป็นวิธีการที่เป็นที่นิยมใช้ เนื่องจากเป็นวิธีการที่อาศัยการทำงานที่จำลองมาจากการทำงานของสมองมนุษย์ โดยอาศัยการรู้จำของต้นแบบตัวอักษรหลาย ๆ รูปแบบในการจดจำ เพื่อที่เมื่อนำข้อมูลอินพุตเข้ามาโปรแกรมสามารถที่จะวิเคราะห์และตัดสินใจได้ว่าเป็นตัวอักษรตัวใด ซึ่งต้องอาศัยวิธีการสกัดลักษณะสำคัญเพื่อนำค่าผลลัพธ์ที่ได้มาวิเคราะห์ในขั้นตอนนี้ เครือข่ายใยประสาท (neural network) ซึ่งเป็นโมเดลทางคณิตศาสตร์ สำหรับการประมวลผลสารสนเทศด้วยการคำนวณแบบคอนเนคชันนิสต์(Connectionist) เพื่อจำลองการทำงานของสมองมนุษย์ดังแสดงในรูปที่ 2.10 โดยอาศัยการรู้จำของต้นแบบตัวอักษรหลาย ๆ รูปแบบในการจดจำ



รูปที่ 2.10 (a) Model ของ Neural ในสมองมนุษย์ (b) Model ของ Neural ในคอมพิวเตอร์[1]

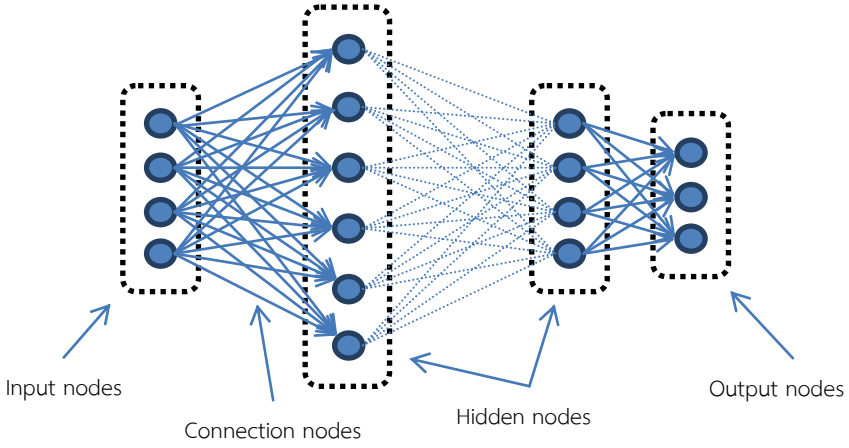
2.2.2.4.1. หลักการทำงานของเครือข่ายใยประสาท (neural network)

ใยประสาท(neural) ในคอมพิวเตอร์ ประกอบด้วยอินพุต และเอาต์พุต โดยจำลองให้อินพุตแต่ละอันมี weight เป็นตัวกำหนดน้ำหนักของอินพุต ใยประสาทแต่ละหน่วยจะมีค่า threshold เป็นตัวกำหนดว่าน้ำหนักรวมของอินพุต ต้องมากขนาดไหนจึงจะสามารถส่งเอาต์พุตไปยังใยประสาทตัวอื่นได้ เมื่อนำใยประสาทแต่ละหน่วยมาต่อกันให้ทำงานร่วมกัน การทำงานนี้ในทางตรรกะแล้วก็จะเหมือนกับปฏิกิริยาเคมีที่เกิดในสมอง เพียงแต่ในคอมพิวเตอร์ทุกอย่างเป็นตัวเลข[1]

การทำงานของเครือข่ายใยประสาท เมื่อมีอินพุต เข้ามายังเครือข่าย เราจะนำอินพุต มาคูณกับ weight ในแต่ละขา ผลที่ได้จากอินพุตทุก ๆ ขา ของใยประสาทจะนำมารวมกันแล้วนำมาเทียบกับ threshold ที่กำหนดไว้ ถ้าผลรวมมีค่ามากกว่า threshold แล้วใยประสาท จะส่งเอาต์พุตออกไป เอาต์พุตนี้ก็จะถูกส่งไปยังอินพุตของใยประสาทอื่น ๆ ที่เชื่อมกันในเครือข่าย ถ้าค่าน้อยกว่า threshold ก็จะไม่เกิดเอาต์พุต แสดงได้ในรูปที่ 2.11

เขียนอัลกอริทึมออกมาได้ดังนี้

if (sum(input * weight) > threshold) then output



รูปที่ 2.11 แสดงโครงสร้างวงจร Neural Network[1]

สิ่งสำคัญคือ เราต้องทราบค่า weight และ threshold สำหรับสิ่งที่เราต้องการ เพื่อให้คอมพิวเตอร์รู้จำ ซึ่งเป็นค่าที่ไม่แน่นอน แต่สามารถกำหนดให้คอมพิวเตอร์ปรับค่าเหล่านั้นได้โดยสอนให้รู้จักแบบแผน(pattern) ของสิ่งที่เราต้องการให้รู้จำ เรียกว่า "back propagation" ซึ่งเป็นกระบวนการย้อนกลับของการรู้จำ ในการฝึก feed-forward neural networks จะมีการใช้อัลกอริทึมแบบ back-propagation[1] เพื่อใช้ในการปรับปรุงน้ำหนักคะแนนของเครือข่าย (network weight) หลังจากเราใส่รูปแบบข้อมูลสำหรับการฝึกให้แก่เครือข่ายในแต่ละครั้งแล้ว ค่าเอาต์พุตที่ได้จากเครือข่ายจะถูกนำไปเปรียบเทียบกับผลที่คาดหวัง แล้วทำการคำนวณหาค่าความผิดพลาด ตามสมการที่ (2.9) ซึ่งค่าความผิดพลาดนี้จะถูกส่งกลับเข้าสู่เครือข่ายเพื่อใช้แก้ไขค่าน้ำหนักคะแนนต่อไป

Output ของแต่ละ Node

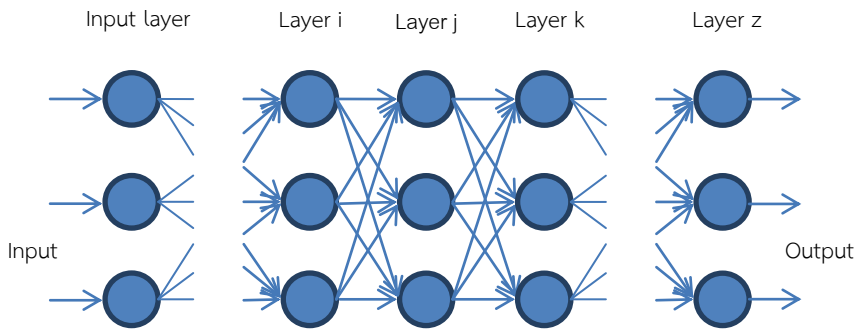
$$y_i = f(w_i^1 x_1 + w_i^2 x_2 + w_i^3 x_3 + \cdots + w_i^m x_m) = f(\sum_j w_i^j x_j) \qquad (2.9)$$

เมื่อ x_i = input จากโหนดอื่น ๆ

w_i^j = น้ำหนัก (weight) ของแต่ละแขน (connection)

2.2.2.4.2. Back propagation Algorithm

Back-propagation Algorithm เป็นอัลกอริทึมที่ใช้ในการเรียนรู้ของเครือข่ายใยประสาทวิธีหนึ่งที่ใช้ใน multilayer perceptron เพื่อปรับค่าน้ำหนักในเส้นเชื่อมต่อระหว่างโหนดให้เหมาะสม โดยการปรับค่านี้จะขึ้นกับความแตกต่างของค่าเอาต์พุตที่คำนวณได้กับค่าเอาต์พุตที่ต้องการ พิจารณารูปที่ 2.12 ประกอบ



รูปที่ 2.12 แสดงรูปแบบ Back-propagation neural network[1]

ขั้นตอนของ Back-propagation Algorithm มีดังนี้ [1]

1. กำหนดค่าอัตราเร็วในการเรียนรู้ (rate parameter)

2. สำหรับแต่ละตัวอย่างอินพุตให้ทำตามขั้นตอนต่อไปนี้จนกว่าได้ระดับ performance ที่ต้องการ

- คำนวณหาค่าเอาต์พุตโดยใช้ค่าน้ำหนักเริ่มต้นซึ่งอาจได้จากการสุ่ม
- คำนวณหาค่า β : แทนประโยชน์ที่จะได้รับสำหรับการเปลี่ยนค่าเอาต์พุตของแต่ละโหนด
- ในชั้นเอาต์พุต (Output Layer) คำนวณได้ตามสมการที่ (2.10)

$$\beta_z = d_z - o_z \qquad (2.10)$$

เมื่อ d_z = ค่าเอาต์พุตที่ต้องการ

o_z = ค่าเอาต์พุตที่คำนวณได้

- ในชั้นซ่อน (Hidden Layer) คำนวณได้ตามสมการที่ (2.11)

$$\beta_j = \sum w_{jk} o_k (1 - o_k) \beta_k \qquad (2.11)$$

เมื่อ w_{jk} = น้ำหนักของเส้นเชื่อมระหว่างชั้นที่ j กับ k

- คำนวณค่าน้ำหนักที่เปลี่ยนแปลงไปสำหรับในทุกน้ำหนัก ด้วยสมการที่ (2.12)

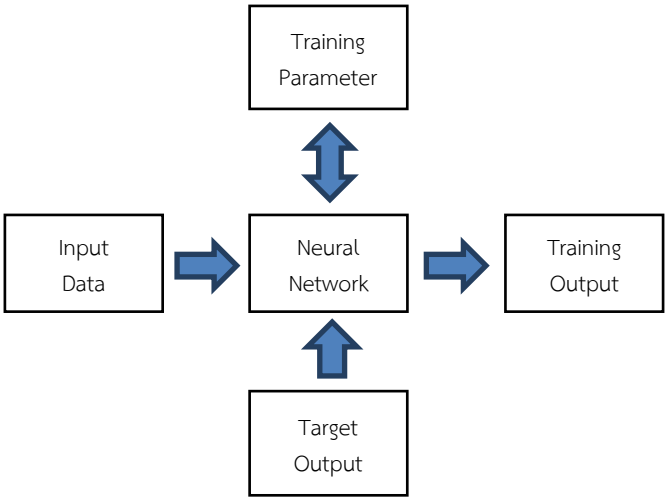
$$\Delta w_{ij} = r o_i o_j (1 - o_j) \beta_j \qquad (2.12)$$

- เพิ่มค่าน้ำหนักที่เปลี่ยนแปลง สำหรับตัวอย่างอินพุตทั้งหมด และเปลี่ยนค่าน้ำหนัก

2.2.2.4.3. การเรียนรู้ของ Neural Network[1]

2.2.2.4.3.1. การเรียนแบบมีการสอน (Supervised Learning)

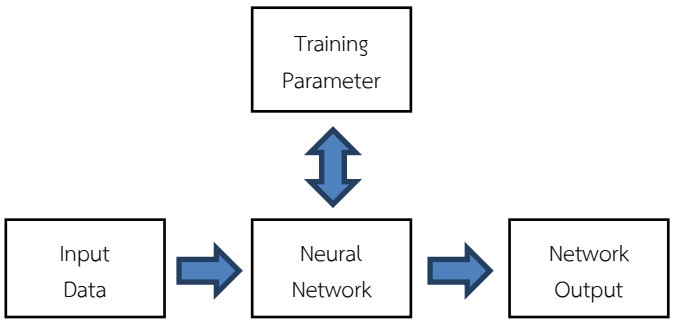
เป็นการที่เรียนรู้แบบที่มีการตรวจคำตอบเพื่อให้ วงจรข่ายปรับตัว ซึ่งชุดข้อมูลที่ใช้สอนวงจรข่าย จะมีคำตอบไว้ตรวจดูว่าวงจรข่ายให้คำตอบถูกต้องหรือไม่ ถ้าคำตอบผิด วงจรข่ายจะปรับตัวเองเพื่อให้ได้คำตอบที่ดีขึ้น ดังรูปที่ 2.13



รูปที่ 2.13 แสดงการเรียนรู้แบบมีการสอน (Supervised Leaning)[1]

2.2.2.4.3.2. การเรียนแบบไม่มีการสอน (Unsupervised Learning)

เป็นการที่เรียนรู้แบบที่ไม่มีการตรวจคำตอบว่าถูกหรือผิด วงจรข่ายจะจัดเรียงโครงสร้างด้วยตัวเองตามลักษณะของข้อมูล ดังรูปที่ 2.14

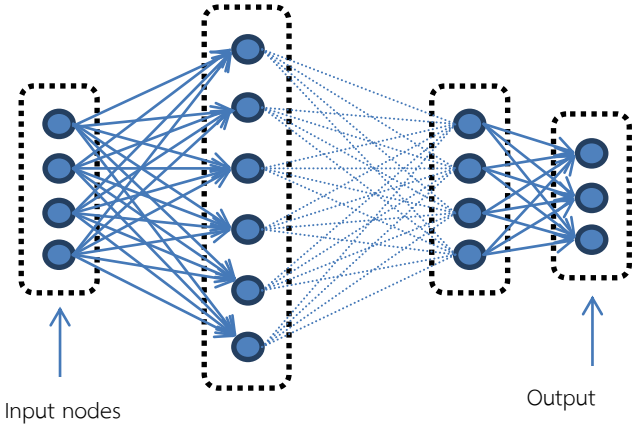


รูปที่ 2.14 แสดงการเรียนรู้แบบไม่มีการสอน (Unsupervised Learning)[1]

2.2.2.4.4. Network Architecture[1]

2.2.2.4.4.1. Feedforward network

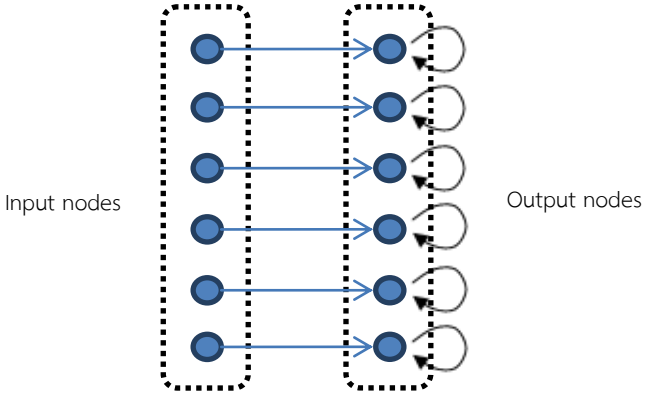
ข้อมูลที่ประมวลผลในวงจรข่ายจะถูกส่งไปทิศทางเดียวจาก Input nodes และจะถูกส่งประมวลผลเรื่อย ๆ จนถึง Output nodes โดยไม่มีการย้อนกลับของข้อมูล และแต่ละ Nodes ใน Layer จะไม่มีการเชื่อมต่อกัน ดังรูปที่ 2.15



รูปที่ 2.15 แสดงสถาปัตยกรรมของ Feedforward network[1]

2.2.2.4.4.2. Feedback network

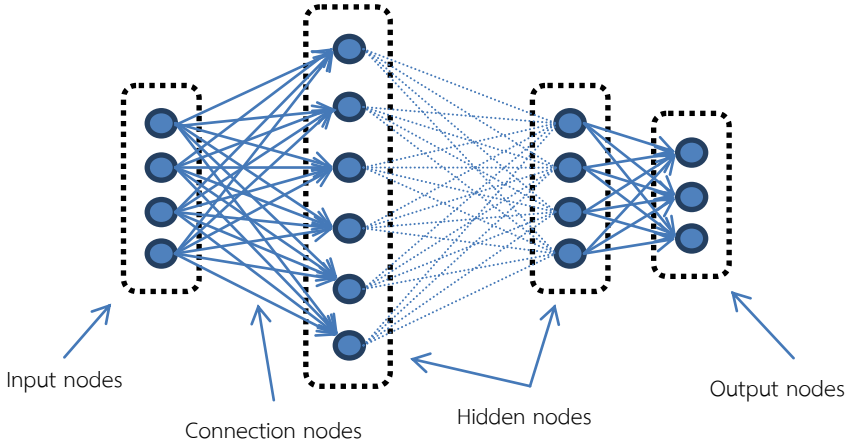
ข้อมูลที่ประมวลผลในวงจรข่าย จะมีการป้อนกลับเข้าไปยังวงจรข่ายหลาย ๆ ครั้ง จนกระทั่งได้คำตอบออกมา ดังรูปที่ 2.16



รูปที่ 2.16 แสดงสถาปัตยกรรม Feedback network[1]

2.2.2.4.4.3. Network Layer

Network Layer จะประกอบไปด้วย 3 Layer ได้แก่ Input Units Hidden Units และ Output Units ดังรูปที่ 2.17



รูปที่ 2.17 แสดงสถาปัตยกรรม Network Layer[1]

2.2.2.4.4.4. Architecture of Layer

Architecture of Layer สามารถจำแนกสถาปัตยกรรมของชั้น (Layer) ออกเป็น 2 ประเภท คือ Single-layer และ Multi-layer

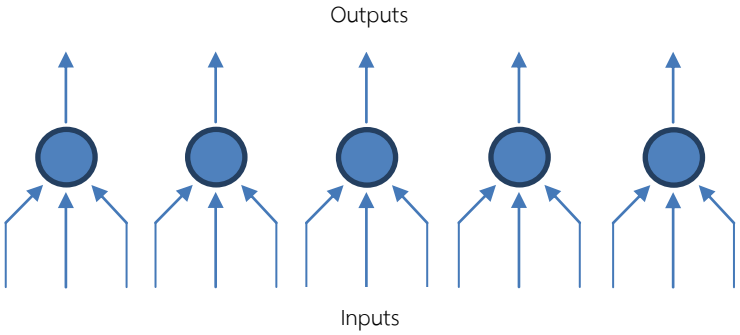
2.2.2.4.4.4.1. Single-layer perceptron เครือข่ายใยประสาท (Neural Network) ที่ประกอบด้วยชั้นเพียงชั้นเดียวตามรูปที่ 2.18 จำนวน Input nodes ขึ้นอยู่กับจำนวน components ของ Input data และ Activation function ขึ้นอยู่กับลักษณะข้อมูลของเอาต์พุต เช่น ถ้าเอาต์พุตที่ต้องการเป็น “ใช่” หรือ “ไม่ใช่” เราจะต้องใช้ Threshold function ตามสมการที่ (2.13) ดังนี้[1]

$$f(x) = \begin{cases} 1, & if\ x \geq T \\ 0, & if\ x < T \end{cases} \qquad (2.13)$$

T = Threshold level

หรือถ้า Output เป็นค่าตัวเลขที่ต่อเนื่อง เราต้องใช้ Continuous function เช่น Sigmoid function ตามสมการที่ (2.14)[1]

$$f(x) = \frac{1}{1 + e^{-\alpha x}} \qquad (2.14)$$



รูปที่ 2.18 แสดงสถาปัตยกรรม Single-layer perceptron[1]

2.2.2.4.4.4.2. Multi-layer perceptron เครือข่ายใยประสาท(Neural Network) ที่ประกอบด้วยหลายชั้นโดยในแต่ละชั้นจะประกอบด้วยโหนด(nodes) หรือเปรียบได้กับเซลล์ประสาท(Neurals) ค่าน้ำหนักของเส้นที่เชื่อมต่อระหว่างโหนดของแต่ละชั้น(เมทริก W), ค่า Bias vector(b) และค่า Output vector(a) โดย m เป็นตัวเลขบอกลำดับชั้นกำกับไว้ด้านบน เมื่อ p เป็น Input vecter การคำนวณค่า

เอาต์พุทสำหรับเครือข่ายใยประสาทที่มี M ชั้น ได้สมการที่ (2.14) ดังนี้

$$a^{m+1} = f^{m+1}(W^{m+1}a^m + b^{m+1}) \qquad (2.14)$$

เมื่อ $m = 0,2,\ldots, M-1$

$$a^0 = p$$
$$a = a^m$$

และ f เป็น Transfer function

บทที่ 3
การออกแบบโปรแกรม

3.1. หลักการออกแบบ

ออกแบบและพัฒนาโปรแกรมบนภาษา C++ โดยใช้ Visual C++ 2008 Express Edition เนื่องจากเป็นโปรแกรมกลุ่ม IDE(integrated development environment)เพื่อพัฒนาโปรแกรมบนภาษา C++ โดยเฉพาะเนื่องจากง่ายและสะดวกต่อการใช้งาน นอกจากนั้นยังอยู่ในโครงการซอฟแวร์เพื่อการศึกษาจากไมโครซอฟต์ ทางผู้จัดทำโครงการจึงสามารถใช้งานได้อย่างถูกต้องตามลิขสิทธิ์

โดยทางผู้จัดทำโครงการได้เลือกใช้ Library ที่มีชื่อว่า OpenCV (Open Source Computer Vision) ซึ่งเป็น library ที่ใช้เขียนโปรแกรมเกี่ยวกับ Computer Vision เป็นหลัก เหตุผลที่ทางผู้จัดทำโครงการเลือกใช้ OpenCV เนื่องจากเป็น libraryที่สามารถนำไปใช้ได้เลยโดยมี algorithm พื้นฐานสามารถเรียกใช้งานและสามารถนำ algorithm เหล่านี้ไปประยุกต์ใช้งานในรูปแบบที่ต้องการได้ ซึ่งเหมาะสมกับโครงการเป็นอย่างมาก

จากแผนงานที่วางไว้ ทางผู้จัดทำได้วางขั้นตอนการทำงานหลัก ๆ ของโปรแกรม คือ Pre-Processing และ Recognition ซึ่งทางผู้จัดทำโครงการได้เลือกทำที่ส่วนของขั้นตอน Recognition เนื่องจากเป็นขั้นตอนการทำงานที่เป็นส่วนสำคัญที่สุดของโปรแกรม ซึ่งนั่นทำให้เห็นภาพรวมความคืบหน้าของโครงการดีกว่าขั้นตอน Pre-Processing โดยการดำเนินงานในขั้นตอน Recognition นั้น ทางผู้จัดทำโครงการได้เลือกใช้วิธีการ Matching by correlation ซึ่งเป็นฟังก์ชั่นที่มีอยู่ใน OpenCV library

3.2. ขั้นตอนการพัฒนา

แบ่งออกเป็น 2 ขั้นตอนหลัก ๆ คือ

3.2.1. ขบวนการประมวลผลขั้นต้น (Pre-Processing) สามารถแบ่งการทำงานย่อย ๆ ได้ดังนี้

3.2.1.1. การกรองข้อมูลแทรกซ้อน (Noise Filtering) เป็นวิธีการที่ทำหน้าที่กำจัดสิ่งที่ไม่พึงประสงค์ที่ปรากฏอยู่บนภาพที่ต้องการจะนำมาประมวลผล

3.2.1.2. การตัดแบ่งพื้นที่ใช้งาน (Cropping) การตัดแบ่งพื้นที่ใช้งานเป็นการแยกเอาเฉพาะส่วนตัวอักษรที่ต้องการนำไปวิเคราะห์ เพื่อความถูกต้องของข้อมูล

3.2.1.3. Thesholding เป็นวิธีการเปลี่ยนค่าของจุดภาพให้มีเพียงสองค่า (Binary) คือ ขาวและดำ หรือ (0,1)

3.2.1.4. การปรับขนาด (Resize) คือ การปรับขนาดรูปภาพที่เข้ามาในโปรแกรมให้มีความเหมาะสมในการวิเคราะห์

3.2.2. ขบวนการการรู้จำ (Recognition)

การรู้จำเป็นขั้นตอนที่สำคัญของ OCR เพราะเป็นขั้นตอนที่ทำหน้าที่ตัดสินใจว่าภาพที่ส่งเข้าไปวิเคราะห์นั้นเป็นตัวอักษรอะไร ซึ่งมีวิธีการหลากหลายวิธีในการทำงาน ซึ่งสามารถแบ่งออกเป็นหลายวิธี โดยเน้นกฎทางทฤษฎีเป็นหลักในการแบ่งวิธีต่าง ๆ ซึ่งวิธีการต่าง ๆ เหล่านี้มีข้อดีและข้อเสียที่แตกต่างกัน ซึ่งถ้าเราต้องการข้อมูลที่มีความแม่นยำ โปรแกรมจำเป็นที่จะต้องอาศัยวิธีการต่าง ๆ เหล่านี้มาช่วยในการตัดสินใจ ซึ่งวิธีการที่ทางผู้พัฒนาเลือกใช้ก็คือ

3.2.2.1. วิธีทางการเข้าคู่รูปแบบ (Template Matching) คือ การนำรูปภาพที่ต้องการหาค่าไปเปรียบเทียบกับรูปแบบตัวอย่าง (Template) ซึ่งรูปแบบตัวอย่างนี้ จะเก็บค่า และกำหนดลักษณะสำคัญต่าง ๆ ที่สามารถแยกความแตกต่างของตัวอักษรต่าง ๆ ได้ และเมื่อนำภาพตั้งต้นที่มีมาเปรียบเทียบกับรูปแบบที่มี เพื่อตรวจสอบความคล้ายคลึงกันของภาพทั้งสอง ซึ่งอาศัยค่าระดับที่ตั้งขึ้นมาเพื่อตรวจสอบหรือตัดสินใจ แต่ข้อเสียของวิธีการนี้มีค่อนข้างมาก ซึ่งข้อเสียของวิธีการนี้ คือ มีความอ่อนไหวต่อข้อมูลที่มีการแทรกซ้อนขนาดและความเอียง ซึ่งปัญหาเหล่านี้สามารถแก้ได้ แต่นั่นต้องอาศัยการทำงานในขั้นตอนประมวลผลขั้นต้นที่ดี

ในการทำงานของ Matching by Correlation นั้นจำเป็นที่จะต้องมีภาพสองส่วนหลัก ๆ คือ Template Image และ Source Image โดยส่วนที่เรียกว่า Template Image นั้นเป็นส่วนของรูปแบบ(Pattern)ของภาพ ซึ่งจะเก็บรูปแบบภาพทั้ง 10 ตัว คือเลข 0 ถึง 9 ซึ่งอาจมีชุดของรูปแบบตัวเลขที่เพิ่มเข้ามาอีก ส่วนที่สองคือ Source Image คือส่วนของชุดรูปภาพตัวเลขบนมิเตอร์ที่นำเข้าโปรแกรมเพื่อวิเคราะห์ผล (Recognition) โดยหลักการทำงานของการ Matching ตัวเลข ทางผู้จัดทำโครงการได้นำภาพ Template Image ไปเปรียบเทียบกับชุดตัวเลข 4 หลักของ Source Image ดังรูปที่ 1 กล่าวคือ เริ่มต้นที่นำภาพ Template Image ตัวแรก คือ เลข 0 ไปเปรียบเทียบกับชุดตัวเลขของ Source Image เมื่อเปรียบเทียบเสร็จก็นำภาพตัวต่อไปมาเปรียบเทียบเรื่อย ๆ จนถึงตัวเลขสุดท้ายนั่นก็คือ เลข 9

โดยฟังก์ชั่นที่ทางผู้จัดทำโครงการเรียกใช้นั่นคือ cvMatchTemplate ซึ่งมีรูปแบบของฟังก์ชั่น ดังตารางที่ 3.1

ตารางที่ 3.1 รูปแบบโครงสร้างของฟังก์ชั่น cvMatchTemplate

```
Void  cvMatchTemplate (
            const CvArr*      image,
            const CvArr*      templ,
                  CvArr*      result,
            int   method        )  ;
```

จากรูปแบบโครงสร้างของฟังก์ชั่นอธิบายได้โดย

image คือ ภาพที่ต้องการนำมาทำการรันในโปรแกรม หรือภาพที่จะถูกทำการค้นหาและเปรียบเทียบ

templ คือ ภาพที่จะถูกนำไปเปรียบเทียบและค้นหาใน image ซึ่งต้องมีชนิดของข้อมูลตรงกันกับ image

result คือ ภาพผลลัพธ์จากการเปรียบเทียบและค้นหา

method คือ หลักการที่ใช้ของ templ ที่จะไปทำการเปรียบเทียบและค้นหา

การทำงานของฟังก์ชั่นคือทำการเปรียบเทียบภาพ templ กับภาพ image โดยการเปรียบเทียบโดยใช้ขนาด w x h โดยเลือกใช้ method ในการเลือกใช้วิธีการค้นหาและเปรียบเทียบและทำการเก็บผลลัพธ์ไปที่ result

ต่อไปกล่าวถึงรูปแบบของ method ที่ได้ทำการเลือกใช้ซึ่งแทนตัวแปรต่าง ๆ ดังนี้

I denotes image, T templ, R result

x' = 0... w − 1, y' = 0... h − 1

ซึ่งรูปแบบโครงสร้าง Method: CV_TM_CCORR_NORMED แสดงในสมการที่ (15)

$$R(x,y) = \frac{\sum_{x',y'}\left(T(x',y').I'(x+x',y+y')\right)}{\sqrt{\sum_{x',y'}T(x',y')^2.\sum_{x',y'}I(x+x',y+y')^2}} \qquad (15)$$

3.3. อุปกรณ์ เครื่องมือ ในการออกแบบและพัฒนา

3.3.1. C++ 2008 Express Edition

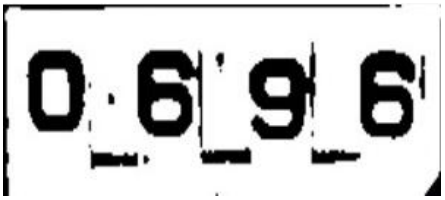
3.3.2. OpenCV Library

3.3.3. กล้องถ่ายภาพ

3.4. ขั้นตอนการทดลอง

3.4.1. เตรียมอุปกรณ์ และภาพถ่ายที่ใช้ในการนำมาวิเคราะห์ถ่ายภาพจากมิเตอร์ และนำภาพเหล่านั้นมาทำการตกแต่งเพื่อให้มีความสมบูรณ์ของภาพมากที่สุด เพื่อที่จะนำภาพเหล่านั้นมาเป็นภาพ Template Image ดังรูปที่ 3.1 และเป็นภาพ Source Image ดังรูปที่ 3.2

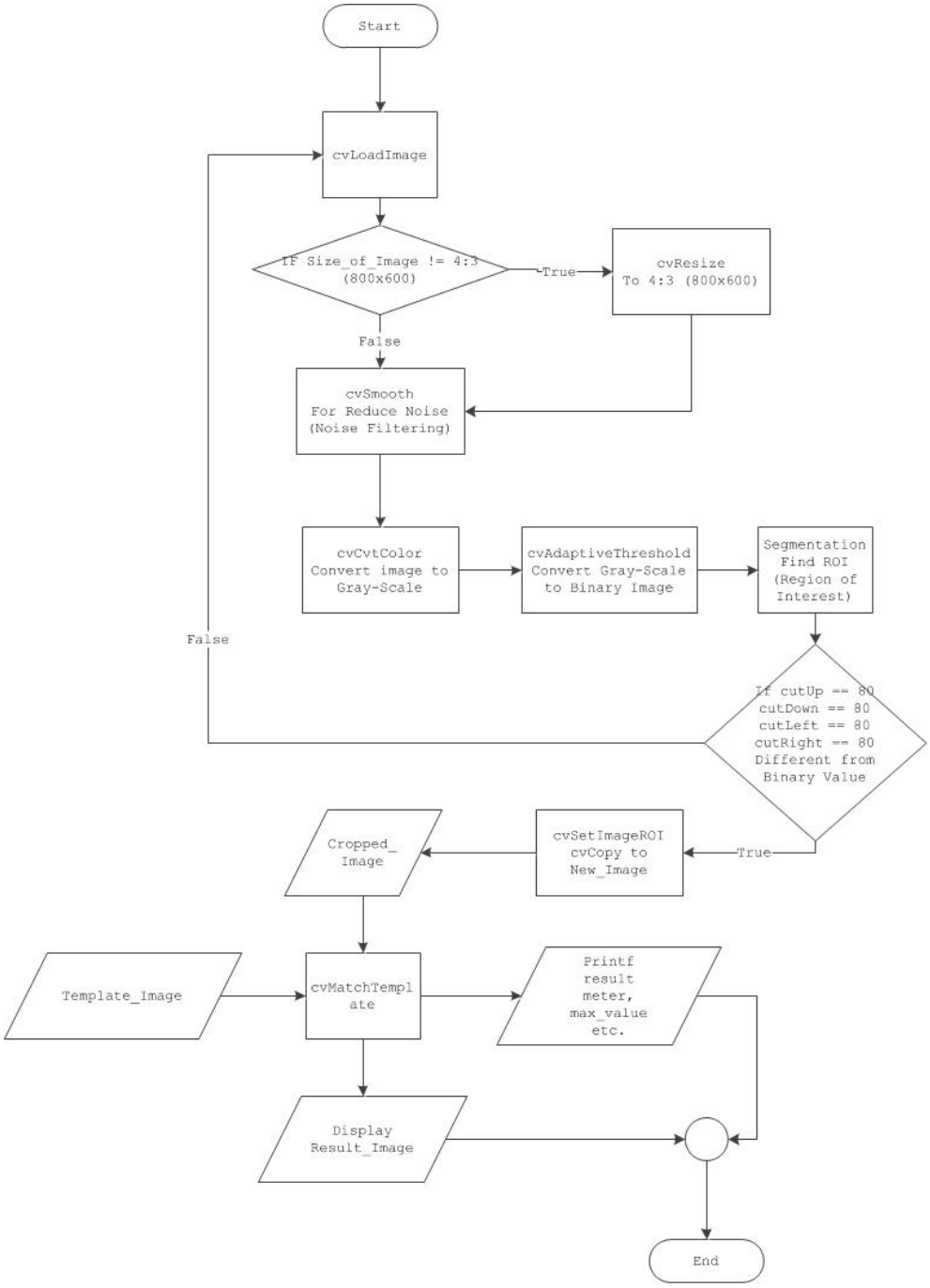
รูปที่ 3.1 ภาพของ Template Image จากมิเตอร์จริง



รูปที่ 3.2 ตัวอย่างภาพ Source Image

3.4.2. เขียนอัลกอริทึ่ม และออกแบบโปรแกรม

ในส่วนนี้ทางผู้จัดทำได้ศึกษาถึงโครงสร้างการทำงานของ OpenCV Library เพื่อหาในส่วนของฟังก์ชั่นการทำงานที่เหมาะสมกับโปรแกรมที่พัฒนา โดยอัลกอริทึ่มที่ได้มาเห็นดั้งแสดงในหัวข้อ 3.1 หลักการออกแบบ โดยสามารถแสดงการทำงานได้ตาม Flowchart ดังรูปที่ 3.4



รูปที่ 3.3 Flowchart แสดงการทำงานของโปรแกรม

บทที่ 4
การพัฒนาโปรแกรมและผลการทดลอง

ในส่วนของการพัฒนาโปรแกรมนั้น จะถูกแบ่งออกเป็นทั้งหมด 2 ส่วนใหญ่ ๆ คือ ส่วนของการประมวลผลขั้นต้น(Preprocessing) และส่วนของการรู้จำ(Recognition)

4.1. การประมวลผลขั้นต้น(Preprocessing)

การดำเนินงานเริ่มต้นจากการศึกษาหาความรู้เกี่ยวกับเรื่อง Image Processing และการรู้จำ (Recognition) เพื่อช่วยในการสร้างโค้ดที่ใช้ในการวิเคราะห์ภาพและเปลี่ยนแปลงภาพให้เป็นภาพที่ง่ายต่อการอ่านหมายเลขมิเตอร์ไฟฟ้าโดยมีขั้นตอนดังนี้

1. ถ่ายภาพมิเตอร์ โดยให้กรอบสีดำของมิเตอร์อยู่ในขอบเขตที่กำหนด และจุดสีแดงต้องอยู่ภายในกรอบสีดำ ดังในรูปที่ 4.1 และ 4.2

รูปที่ 4.1 ตัวอย่างการถ่ายภาพที่ถูกวิธี

รูปที่ 4.2 ตัวอย่างการถ่ายภาพที่ผิดวิธี

2. โหลดภาพมิเตอร์ไฟฟ้าที่ถ่ายมาได้เข้าโปรแกรม visual c++ โดยใช้คำสั่งในตารางที่ 3.1 ตารางที่

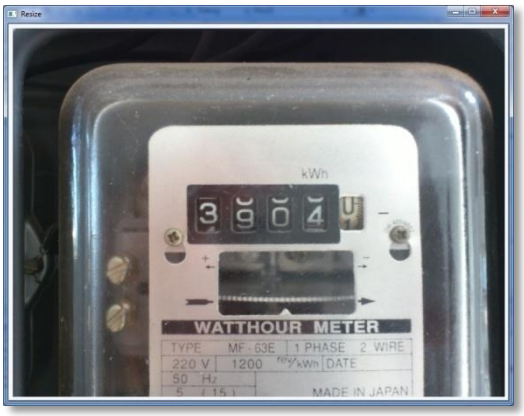
ตารางที่ 4.1 ตัวอย่างโค้ดในการโหลดรูปเข้ามาในโปรแกรม

```
IplImage *Imgsrc = cvLoadImage("NewPic44.jpg", 1);
```

3. นำภาพของมิเตอร์ไฟฟ้าที่ถ่ายได้ มาทำการลดขนาดของภาพ(Resize) ให้เหลือขนาด 800×600 pixel ตามตัวอย่างในตารางที่ 4.2 และจะได้ผลลัพธ์ดังในรูปที่ 4.3

ตารางที่ 4.2 ตัวอย่างโค้ดในการลดขนาดภาพ (Resize)

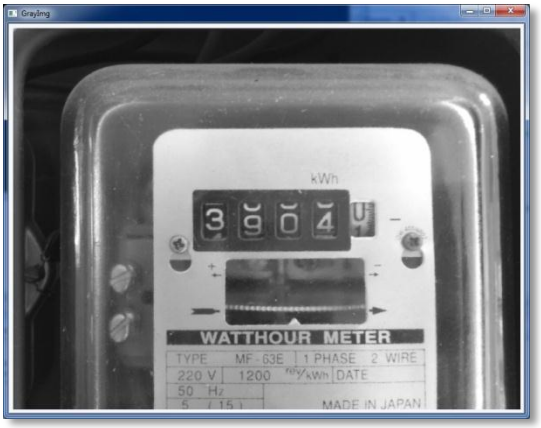
```
// ลดขนาดของภาพ(resize) ขนาด 800x600 pixel
IplImage * Resize = cvCreateImage(cvSize(800,600),IPL_DEPTH_8U,3);
cvResize(Imgsrc,Resize,1);
```



รูปที่ 4.3 ภาพที่ลดขนาดแล้วขนาด 800×600 pixel

4. นำภาพที่ได้มาแปลงภาพเป็นแบบ Gray scale ตามตัวอย่างในตารางที่ 4.3 เพื่อที่จะทำให้เหลือเพียง 1 channel เพราะปกติภาพสีจะมี 3 channel (RGB) เมื่อแปลงแล้วจะได้ผลตามรูปที่ 4.4ตารางที่ 4.3 โค้ดตัวอย่างการแปลงภาพสีเป็น Grayscale

```
IplImage* GrayImg = cvCreateImage(cvGetSize(Resize),IPL_DEPTH_8U,1);
cvCvtColor(Resize, GrayImg, CV_BGR2GRAY );
```



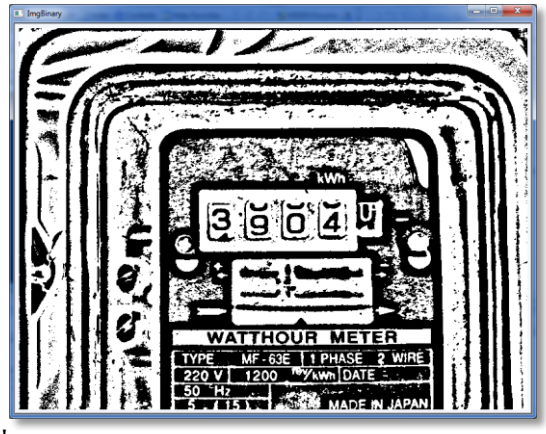
รูปที่ 4.4 ภาพมิเตอร์ไฟฟ้าแบบ Grayscale

5. แปลงภาพให้เป็นภาพขาว-ดำ โดยใช้วิธีการ AdaptiveThreshold เพื่อที่จะสามารถเข้าถึง pixel แต่ละ pixel ได้ โดยใช้คำสั่งในตารางที่ 4.4 เมื่อใช้คำสั่งนี้แล้วจะได้ภาพขาวดำ ซึ่งการที่ภาพจะ

กลายเป็นภาพขาวดำได้นั้น จะต้องแปลงภาพให้เป็น Grayscale ก่อนเสมอ ซึ่งจะได้ภาพดังในรูปที่ 4.4

ตารางที่ 4.4 ตัวอย่างโค้ดแปลงภาพให้เป็นขาวดำ

```
IplImage* ImgBinary = cvCloneImage(GrayImg);
cvAdaptiveThreshold(GrayImg,ImgBinary,255,
CV_ADAPTIVE_THRESH_MEAN_C,C    V_THRESH_BINARY_INV,29,0);
```

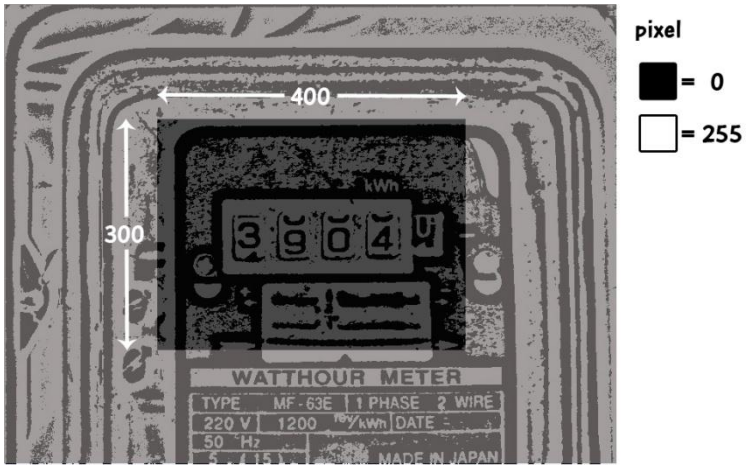


รูปที่ 4.5 ภาพขาวดำของมิเตอร์ไฟฟ้าแบบ AdaptiveThreshold

6. เมื่อภาพเป็นภาพขาวดำแล้วค่าของ pixel จะเหลือเพียงค่าเดียวคือ 0(สีดำ) หรือ 255(สีขาว) แล้วเก็บค่าของ pixel ลงใน Array ขนาด 300×400 ดังแสดงในรูปที่ 4.5 และโค้ดตัวอย่างการดึงค่าของ pixel เก็บไว้ใน array แสดงในตารางที่ 4.5

ตารางที่ 4.5 ตัวอย่างโค้ดการดึงค่าของ pixel จากภาพถ่าย เก็บไว้ใน array

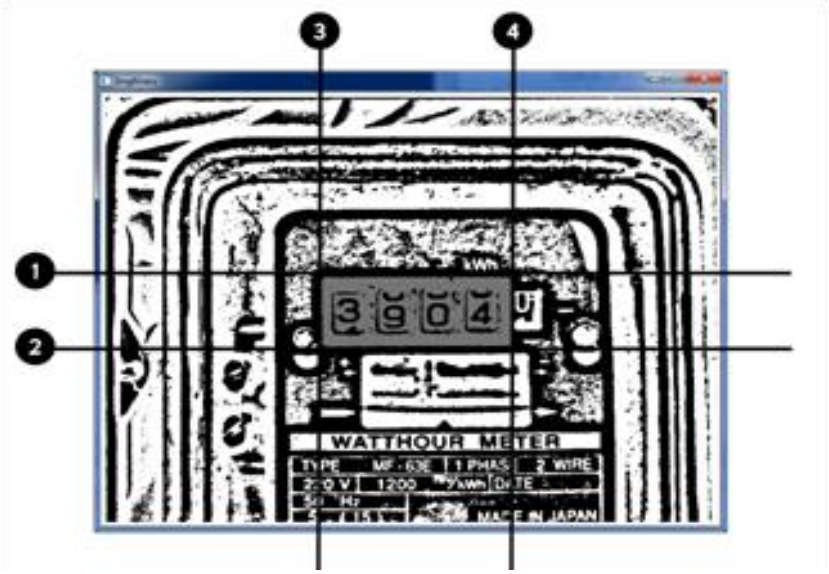
```
int width     = ImgBinary->width; //ดึงค่าขนาดความกว้างของ ImgBinary
int height    = ImgBinary->height; //ดึงค่าขนาดความสูงของ ImgBinary
int nchannels = ImgBinary->nChannels; //ดึงค่า channels ของ  ImgBinary
int step      = ImgBinary->widthStep; //ดึงค่า step ของ  ImgBinary

        uchar *data = ( uchar* )ImgBinary->imageData;


        int keep[300][400]; //เก็บค่าสีของ pixel ลงใน array ขนาด 300×400 pixel


        for( int y = 150 ; y < 450 ; y++ ) {
                for( int x = 200 ; x < 600 ; x++ ) {

                  keep[y-150][x-200] = data[y*step + x*nchannels + 0 ];
         }
}
```



รูปที่ 4.6 แสดงขอบเขตการเก็บค่าของ pixel ลงใน array

7. ทำการตัดกรอบสีดำโดยอ้างอิงจากจุดตรงกลางของภาพมิเตอร์ โดยใช้วิธีการหาเส้นสีดำที่ต่อเนื่องกันของกรอบ และโปรแกรมจะทำการหาตำแหน่งขอบบนโดยใช้คำสั่งในตารางที่ 4.6 หาตำแหน่งขอบล่างโดยใช้คำสั่งในตารางที่ 4.7 หาตำแหน่งขอบซ้ายโดยใช้คำสั่งในตารางที่ 4.8 และหาตำแหน่งขอบขวาโดยใช้คำสั่งในตารางที่ 4.9 เมื่อเราได้ตำแหน่งทั้ง 4 ตำแหน่งแล้ว เราก็จะ

สามารถตัดกรอบสีดำออกมาได้ดังแสดงในรูปที่ 4.7



รูปที่ 4.7 แสดงลำดับการตัดกรอบสีดำ

รูปที่ 4.8 ภาพที่ตัดแล้ว

หมายเลข 1 ตำแหน่งตัดขอบบน

หมายเลข 2 ตำแหน่งตัดขอบล่าง

หมายเลข 3 ตำแหน่งตัดขอบซ้าย

หมายเลข 4 ตำแหน่งตัดขอบขวา

ตารางที่ 4.6 ตัวอย่างโค้ดหาตำแหน่งที่ใช้ตัดขอบบนของกรอบสีดำของมิเตอร์
ดังแสดงตำแหน่งการตัดที่หมายเลข ❶ ในรูปที่ 4.7

```
//ตัดขอบบน
int cutUp=0 ; //เก็บค่าของตำแหน่ง y ที่จะตัด (มีค่าเดียว)
int count1=0; //นับจำนวน pixel ที่จะตัด

for(int y = 150 ; y>0 ; y--){

                        if(count1==80)break; //ออกจากการทำงานในการตัดขอบบน
                        count1 = 0; //เริ่มนับใหม่เมื่อเจอ pixelสีขาว(สีดำไม่ต่อเนื่องกัน)

        for(int x = 200 ; x<300 ; x++){

                if(keep[y][x]==0) count1++; //ถ้าเจอ pixelสีดำให้เพิ่มค่าcount1
                if(keep[y][x]==255) break; //ถ้าเจอ pixelสีขาวให้หยุดนับcount1แล้วเริ่มนับใหม่

if(count1==80) { //ถ้านับครบ 80 pixel เราจะตัดแกน y ที่ตำแหน่งนี้
                        cutUp = y; //เก็บค่าตำแหน่ง y ที่ได้ลงใน cutUp เพื่อรอการตัด
                        break;
                                }
                }
}
```

ตารางที่ 4.7 ตัวอย่างโค้ดหาตำแหน่งที่ใช้ตัดขอบล่างของกรอบสีดำของมิเตอร์
ดังแสดงตำแหน่งการตัดที่หมายเลข ❷ ในรูปที่ 4.7

```
//ตัดขอบล่าง
int cutDown=0 ; //เก็บค่าของตำแหน่ง y ที่จะตัด (มีค่าเดียว)
int count2=0; //นับจำนวน pixel ที่จะตัด

for(int y = 150 ; y<300 ; y++){

                        if(count2==80)break; //ออกจากการทำงานในการตัดขอบล่าง
                        count2 = 0; //เริ่มนับใหม่เมื่อเจอ pixelสีขาว(สีดำไม่ต่อเนื่องกัน)

        for(int x = 200 ; x<300 ; x++){

                if(keep[y][x]==0) count2++; //ถ้าเจอ pixelสีดำให้เพิ่มค่าcount2
                if(keep[y][x]==255) break; //ถ้าเจอ pixelสีขาวให้หยุดนับcount2แล้วเริ่มนับใหม่

                if(count2==80) { //ถ้านับครบ 80 pixel เราจะตัดแกน y ที่ตำแหน่งนี้
                        cutDown = y; //เก็บค่าตำแหน่ง y ที่ได้ไว้ใน cutDown เพื่อรอการตัด
                        break;
                                }
                }
}
```

ตารางที่ 4.8 ตัวอย่างโค้ดหาตำแหน่งที่ใช้ตัดขอบซ้ายของกรอบสีดำของมิเตอร์
ดังแสดงตำแหน่งการตัดที่หมายเลข ❸ ในรูปที่ 4.7

```
//ตัดขอบซ้าย
int cutLeft=0 ; //เก็บค่าของตำแหน่ง x ที่จะตัด (มีค่าเดียว)
int count3=0; //นับจำนวน pixel ที่จะตัด

for(int x = 200 ; x>0 ; x--){

                        if(count3==10)break; //ออกจากการทำงานในการตัดขอบซ้าย
                        count3 = 0; //เริ่มนับใหม่เมื่อเจอ pixelสีขาว(สีดำไม่ต่อเนื่องกัน)

        for(int y = cutUp ; y < cutUp+50 ; y++){

                if(keep[y][x]==0) count3++; //ถ้าเจอ pixelสีดำให้เพิ่มค่าcount3
                if(keep[y][x]==255) break; //ถ้าเจอ pixelสีขาวให้หยุดนับcount3แล้วเริ่มนับใหม่

                if(count3==10) { //ถ้านับครบ 10 pixel เราจะตัดแกน x ที่ตำแหน่งนี้
                        cutLeft = x; //เก็บค่าตำแหน่ง x ที่ได้ไว้ใน cutLeft เพื่อรอการตัด
                        break;
                                }
                }
}
```

ตารางที่ 4.9 ตัวอย่างโค้ดหาตำแหน่งที่ใช้ตัดขอบขวาของกรอบสีดำของมิเตอร์
ดังแสดงตำแหน่งการตัดที่หมายเลข ❹ ในรูปที่ 4.7

```
//ตัดขอบขวา
int cutRight=0 ; //เก็บค่าของตำแหน่ง x ที่จะตัด (มีค่าเดียว)
int count4=0; //นับจำนวน pixel ที่จะตัด

for(int x = 200 ; x<400 ; x++){

                        if(count4==10)break; //ออกจากการทำงานในการตัดขอบขวา
                        count4 = 0; //เริ่มนับใหม่เมื่อเจอ pixelสีขาว(สีดำไม่ต่อเนื่องกัน)

        for(int y = cutUp ; y < cutUp+50 ; y++){

                if(keep[y][x]==0) count4++; //ถ้าเจอ pixelสีดำให้เพิ่มค่าcount4
                if(keep[y][x]==255) break; //ถ้าเจอ pixelสีขาวให้หยุดนับcount4แล้วเริ่มนับใหม่

                if(count4==10) { //ถ้านับครบ 10pixel เราจะตัดแกน x ที่ตำแหน่งนี้
                        cutRight = x; //เก็บค่าตำแหน่ง x ที่ได้ไว้ใน cutRight เพื่อรอการตัด
                        break;
                                }
                }
}
```

ตารางที่ 4.10 ตัวอย่างโค้ดแจ้งเตือนให้ถ่ายรูปใหม่ เนื่องจากไม่สามารถหาตำแหน่งที่จะตัดได้

```
if( cutUp==0||cutDown==0||cutLeft==0||cutRight==0 ){ //ไม่สามารถตัดได้
                printf("Please take new picture !!! \n"); //กรุณาถ่ายรูปใหม่
                return 0 ;
                                }
```

ตารางที่ 4.11 ตัวอย่างโค้ดการ crop รูป

```
cvSetImageROI( ImgBinary, cvRect(cutLeft+200,cutUp+150,
(cutRight+200)-(cutLeft+200),(cutDown+150)-(cutUp+150))); //กรอกข้อมูลตำแหน่ง xและyที่ได้
ทั้ง 4จุด ซึ่งเป็นตำแหน่งทั้ง 4 จุด ในรูป ImgBinary

IplImage* CropImg = cvCreateImage(cvGetSize(ImgBinary),8, 1); //สร้างพื้นที่รอการตัด
cvCopy( ImgBinary, CropImg, 0); //ทำการตัดรูปออกมา
```

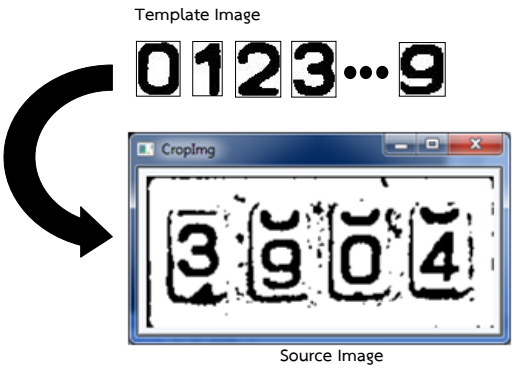
8. ลดขนาดของรูปที่ crop มาอีกครั้ง เพื่อให้ขนาดใกล้เคียงกับ Template ที่เตรียมไว้ แสดงตัวอย่างโค้ดตามตารางที่ 4.10 ซึ่งเมื่อเสร็จขั้นตอนนี้แล้วจะได้รูปที่ crop ออกมาแล้ว ดังแสดงในรูปที่ 4.8 ซึ่งขั้นตอนนี้จะเป็นขั้นตอนสุดท้ายของการ pre-processing แล้ว

ตารางที่ 4.12 ตัวอย่างโค้ดการลดขนาดของภาพ (Resize)

```
IplImage* ResizeImage2 = cvCreateImage( cvSize(300,130),IPL_DEPTH_8U,1 ); //ลดขนาดของภาพ
(resize) ขนาด300x130 pixel
cvResize(CropImg,ResizeImage2,1);
//แสดงผลภาพ
cvNamedWindow("CropImg", 1); //โชว์หน้าต่างชื่อ CropImg
cvShowImage("CropImg", ResizeImage2); //โชว์ภาพ ResizeImage2 ในหน้าต่าง CropImg
```

4.2. ส่วนรู้จำเลขบนมิเตอร์(Recognition)

หลังจากผ่านขั้นตอนการตัดกรอบสีดำของมิเตอร์แล้วก็จะมาถึงในส่วนนี้คือ ซึ่งการรู้จำของเรานั้นได้ใช้วิธีเข้าคู่รูปแบบ(Template Matching) ใน opencv เราสามารถเรียกใช้งานได้ และเรานำมาพัฒนาและประยุกต์ใช้ ซึ่งเป็นการนำตัวเลข (Template) ไปเทียบกับภาพที่ต้องการ ดังแสดงในรูปที่ 4.9 โดยใช้เทคนิคคือ จะทำการเคลื่อนตัวเลข (Template) ไปยังจุดต่าง ๆ จากซ้ายไปขวา บนลงล่าง เพื่อคำนวณหาค่าความเหมือนของตัวเลข(Template) กับบริเวณต่าง ๆ บนภาพ โดยมีขั้นตอนการทำงานดังนี้ แสดงในตารางที่ 4.11 นำภาพที่ทำการตัดแล้วเข้ามา ในรูปที่ 4.8 เตรียม template ตัวเลข 0-9 ดังแสดงในรูปที่ 4.9 ทำการ match ตัวเลขที่เตรียมไว้ 0-9 กับรูปภาพที่นำเข้ามา จัดเรียงตัวเลขให้อยู่ในตำแหน่งที่ถูกต้อง และได้ผลลัพธ์ออกมาเป็นตัวเลขจำนวนเต็มเมื่อทำทั้ง 5 ขั้นตอนเสร็จแล้วจะได้ผลลัพธ์ออกมาดังแสดงในรูปที่ 4.10



รูปที่ 4.9 การนำ Template ที่เตรียมไว้ นำไปเทียบกับภาพที่เสร็จขั้นตอนที่ 8 แล้ว

ตารางที่ 4.13 ตัวอย่างโค้ดในการรู้จำตัวเลขบนมิเตอร์

```
1. นำภาพที่ทำการตัดแล้วเข้ามา
IplImage* imgOriginal = cvCloneImage(ResizeImage2); //เอารูปที่ได้ในขั้นตอนที่ 8 มาใช้งาน
IplImage* imgTemplate;    //Template

int xDimension[5]; //กำหนดขนาดของอะเรย์ ใช้เก็บตัวเลขมิเตอร์
int dataNumber[5]; //เก็บตัวเลขที่พบในภาพ
int count= 0 ; // เอาไว้นับตำแหน่งตัวเลขในอะเรย์

for(double a= 0.95 ; a >= 0.75; )
{
        for(int j=0;j<=19;j++)     {

2. เตรียม template ตัวเลข 0-9
        //template ชุดที่ 1
        if(j==0){imgTemplate = cvLoadImage("number0.jpg", 0);}
        if(j==1){imgTemplate = cvLoadImage("number1.jpg", 0);}
        if(j==2){imgTemplate = cvLoadImage("number2.jpg", 0);}
        if(j==3){imgTemplate = cvLoadImage("number3.jpg", 0);}
        if(j==4){imgTemplate = cvLoadImage("number4.jpg", 0);}
        if(j==5){imgTemplate = cvLoadImage("number5.jpg", 0);}
        if(j==6){imgTemplate = cvLoadImage("number6.jpg", 0);}
        if(j==7){imgTemplate = cvLoadImage("number7.jpg", 0);}
        if(j==8){imgTemplate = cvLoadImage("number8.jpg", 0);}
        if(j==9){imgTemplate = cvLoadImage("number9.jpg", 0);}
```

```
        //template ชุดที่ 2
        if(j==10){imgTemplate = cvLoadImage("pack2_number0.jpg", 0);}
        if(j==11){imgTemplate = cvLoadImage("pack2_number1.jpg", 0);}
        if(j==12){imgTemplate = cvLoadImage("pack2_number2.jpg", 0);}
        if(j==13){imgTemplate = cvLoadImage("pack2_number3.jpg", 0)
        if(j==14){imgTemplate = cvLoadImage("pack2_number4.jpg", 0);}
        if(j==15){imgTemplate = cvLoadImage("pack2_number5.jpg", 0)
        if(j==16){imgTemplate = cvLoadImage("pack2_number6.jpg", 0);}
        if(j==17){imgTemplate = cvLoadImage("pack2_number7.jpg", 0);}
        if(j==18){imgTemplate = cvLoadImage("pack2_number8.jpg", 0)
        if(j==19){imgTemplate = cvLoadImage("pack2_number9.jpg", 0);}
```

3. ทำการ match ตัวเลขที่เตรียมไว้ 0-9 กับรูปภาพที่นำเข้ามา

```
for(int i=0 ;i<=3; i++) { //ใช้ตรวจสอบตัวเลขซ้ำของมิเตอร์

        if(count==5){ //ตัวเลขเกิน 4 ตัว  ภาพมิเตอร์ที่ตัดมาไม่สามารถอ่านได้
        printf("Meter = ERROR \n");
        return 0 ;
                                }

int cv1=imgOriginal->width-imgTemplate->width, cv2=imgOriginal
   ->height-imgTemplate->height;

        double min_val=0, max_val=0;
IplImage* imgResult = cvCreateImage(cvSize(cv1+1, cv2+1),
IPL_DEPTH_32F, 1);

        cvZero(imgResult);

        //เป็นส่วนสำคัญที่สุด คือ การนำ Template ที่เตรียมไว้ มา Match กับรูปภาพที่ตัดไว้ในขั้นตอนที่ 8
        cvMatchTemplate(imgOriginal, imgTemplate, imgResult,
        CV_TM_CCORR_NORMED);

        //เป็นฟังก์ชันที่ใช้ต่อเนื่องกับ ฟังก์ชัน cvMatchTemplate ซึ่งฟังก์ชั่นนี้จะดึงค่า
          min_val,max_val,min_loc และ max_loc
        CvPoint min_loc, max_loc;
        cvMinMaxLoc(imgResult, &min_val, &max_val, &min_loc, &max_loc);
```

```
//บวกตำแหน่งเลขที่ไม่ใช่เลข 1 ไป 15 pixel เพื่อง่ายต่อการตรวจสอบตำแหน่งของตัวเลขให้ถูกต้อง
if(j == 0 || j == 2 ||j == 3 || j == 4 || j == 5 || j == 6 || j == 7    || j == 8 || j == 9 ||j == 10 || j == 12
||j == 13 || j == 14
|| j == 15 || j == 16 || j == 17 || j == 18 || j == 19)
max_loc.x =  max_loc.x + 15;


//กำหนดตำแหน่งตัวเลขในรูปภาพที่เราสนใจ
if( (groupTemplate==1 && j>=0 && j<=9 && max_val> a && max_loc.x > 25      && max_loc.x <
60)
|| (groupTemplate==1 && j>=0 && j<=9 && max_val> a && max_loc.x > 95 && max_loc.x < 130)
|| (groupTemplate==1 && j>=0 && j<=9 && max_val> a && max_loc.x > 165 && max_loc.x < 200)
|| (groupTemplate==1 && j>=0 && j<=9 && max_val> a && max_loc.x > 235 && max_loc.x < 270)
)        {

//เมื่อเจอตัวเลขลบตัวเลขนั้นทิ้งได้เลย
        cvRectangle(imgOriginal, cvPoint(max_loc.x,max_loc.y -
                max_loc.y), cvPoint(max_loc.x+imgTemplate->width,
                max_loc.y+130), cvScalar(255), 40 , 0);
        dataNumber[count]= j;//เก็บตัวเลขที่ได้ไว้
        xDimension[count]= max_loc.x;//เก็บตำแหน่งที่พบตัวเลขนี้

    count++; //เพิ่มค่า countขึ้นไปอีก1
                        }

//กำหนดตำแหน่งตัวเลขในรูปภาพที่เราสนใจ
if( ( groupTemplate==2 && j>=10 && j<=19 && max_val> a && max_loc.x >  15 && max_loc.x <
50)
|| (groupTemplate==2 && j>=10 && j<=19 && max_val> a && max_loc.x > 90 && max_loc.x <
130)
|| (groupTemplate==2 && j>=10 && j<=19 && max_val> a && max_loc.x > 170 && max_loc.x <
205)
|| (groupTemplate==2 && j>=10 && j<=19 && max_val> a && max_loc.x > 245 && max_loc.x <
280) ) {

                j= j-10;
//เมื่อเจอตัวเลขลบตัวเลขนั้นทิ้งได้เลย
        cvRectangle(imgOriginal, cvPoint(max_loc.x,max_loc.y -
```

```
                        max_loc.y), cvPoint(max_loc.x+imgTemplate->width,
                        max_loc.y+130), cvScalar(255), 40 , 0);
        dataNumber[count]= j; //เก็บตัวเลขที่ได้ไว้
        xDimension[count]= max_loc.x; //เก็บตำแหน่งที่พบตัวเลขนี้

        count++; //เพิ่มค่า countขึ้นไปอีก1

                        }
         }
}

if(count==4 && a>0.78) //ครบ 4 ตัวแล้วออกจากการทำงานในส่วนนี้
                break ;
if((count==1 && a <0.78)||(count==2 && a <0.78)||(count==3 && a
  <0.78)||(count==4 && a <0.78))
                {
                count = 5 ;
                break ;
                }
        if(count!=4) //ยังไม่ครบ4ตัวกลับไปทำอีกครั้งจนกว่าจะครบ
                a=a-0.01;
                continue ;

}
```

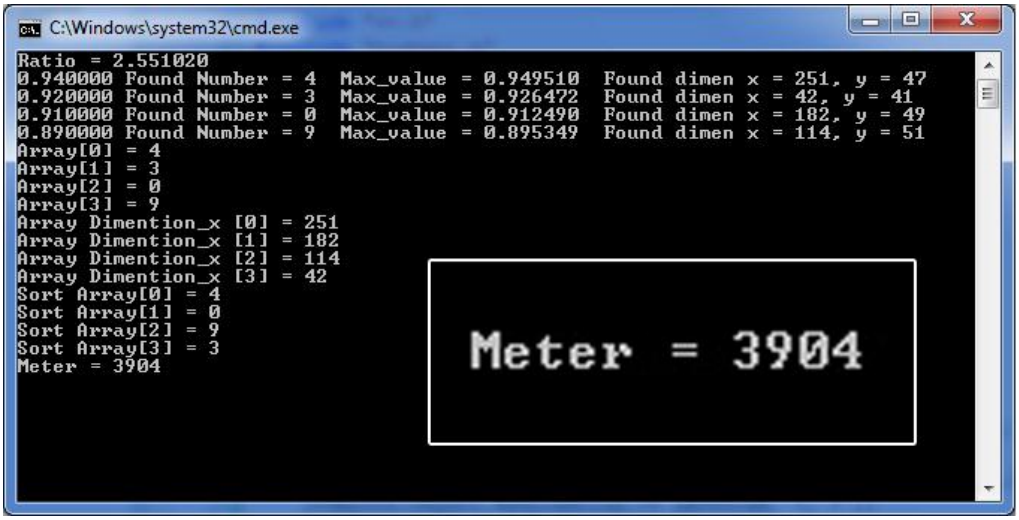
4. จัดเรียงตัวเลขให้อยู่ในตำแหน่งที่ถูกต้อง

```
        //จัดเรียงตัวเลขใหม่
        for (int c=0;c<=2;c++){
        int tempNumber=0,tempX=0;
        for (int d=0;d<=2;d++){

                if(xDimension[d+1]>xDimension[d]){
                        tempNumber = dataNumber[d];
                        tempX = xDimension[d];
                        dataNumber[d] = dataNumber[d+1];
                        xDimension[d] = xDimension[d+1];
                        dataNumber[d+1] = tempNumber;
```

```
                        xDimension[d+1] = tempX;
                        }
                }
        }
```

5. ได้ผลลัพธ์ออกมาเป็นตัวเลขจำนวนเต็ม

```
        //calculate meter คำนวนค่ามิเตอร์ เอาค่าใน array ทั้ง 4 มาบวกกัน
        int meter = 0;
        for (int d=0;d<=3;d++){
                printf("Sort Array[%d] = %d \n",d , dataNumber[d]);
                meter = meter + dataNumber[d]*pow(10.0,d);
        }

return 0;

}
```



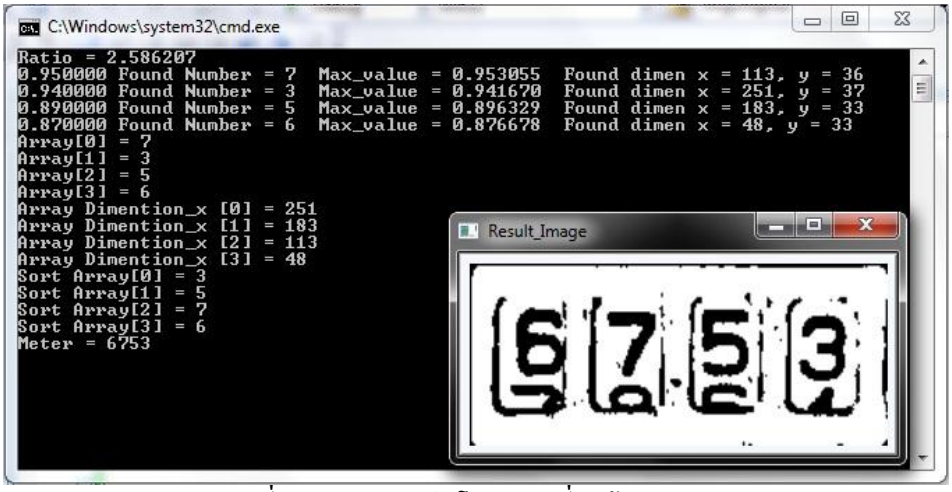
รูปที่ 4.10 ผลการรันโปรแกรม

4.3. ผลการทดลอง

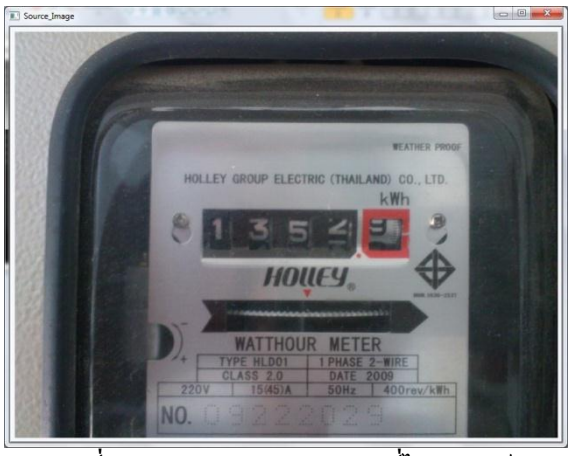
จากการทดลองรันโปรแกรมจากภาพถ่ายที่ถ่ายได้อย่างถูกต้องสมบูรณ์ ผลลัพธ์ที่ออกมามีความถูกต้องทุกรูป แต่หากภาพที่เข้ามาถูกรบกวนจากสภาพแวดล้อมขณะถ่ายภาพ ผลลัพธ์ที่ออกมาจะมีความผิดพลาดในหัวข้อ 5.1 สรุปผลการทดลอง

ภาพตัวอย่างการทำงานของโปรแกรมจากภาพที่มีความสมบูรณ์ซึ่งจะได้ผลลัพธ์ที่ถูกต้อง ดังรูปที่ 4.11 และ 4.12 และภาพที่ไม่มีความสมบูรณ์ซึ่งจะทำให้ไม่มีความสมบูรณ์ถูกต้อง ดังรูปที่ 4.13 และ 4.14 ซึ่งจากการทดลองจาก 100 ภาพได้ผลลัพธ์ที่ถูกต้อง 90 ภาพ หรือคิดเป็นร้อยละ 90 ซึ่งภาพที่ผลลัพธ์ผิดพลาดนั้นเกิดจากภาพถ่ายที่ถูกรบกวนจากสภาพแวดล้อมในขณะที่ถ่ายภาพ

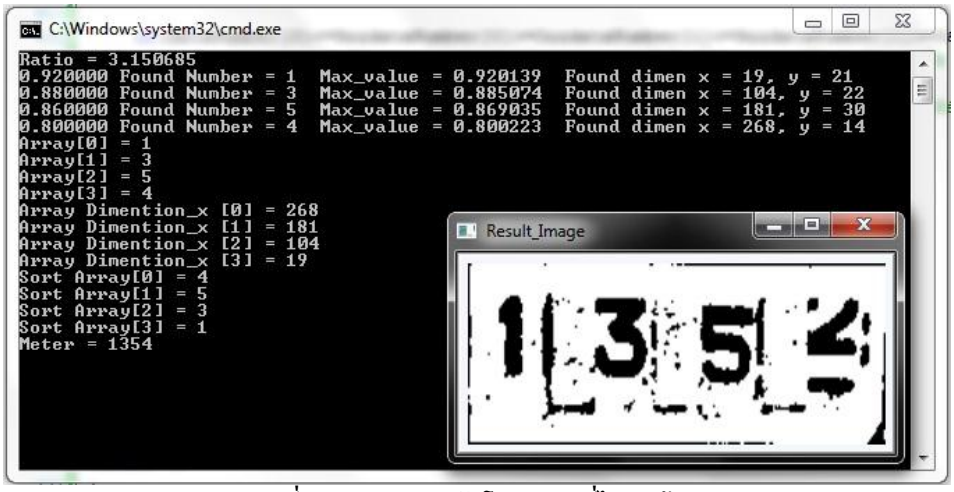
รูปที่ 4.11 ภาพ Source Image ที่สมบูรณ์



รูปที่ 4.12 ภาพผลรันโปรแกรมที่ถูกต้องแม่นยำ



รูปที่ 4.13 ภาพ Source Image ที่ไม่สมบูรณ์



รูปที่ 4.14 ภาพผลรันโปรแกรมที่ไม่ถูกต้อง

บทที่ 5
สรุปผลการดำเนินงาน

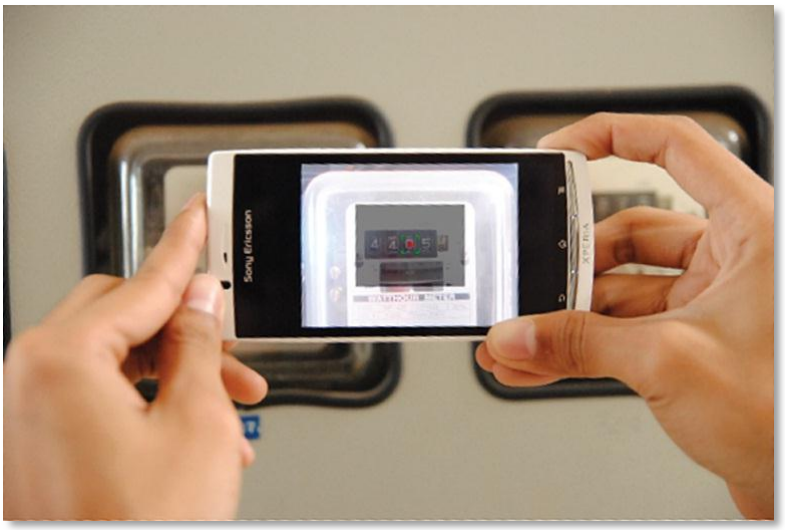
5.1. สรุป

โปรแกรมอ่านเลขมิเตอร์เป็นโปรแกรมที่พัฒนาขึ้นโดยใช้ Microsoft Visual C++ 2008 และ OpenCV Library ที่สามารถวิเคราะห์ค่าตัวเลขการไฟฟ้าบนมิเตอร์ จากภาพถ่ายได้อย่างถูกต้องและแม่นยำ (เฉพาะตัวเลข) เพื่อที่จะนำผลที่ได้นี้ไปประยุกต์ใช้ต่อ

ซึ่งผลลัพธ์ของโปรแกรมอ่านเลขมิเตอร์ยังข้อจำกัดอยู่คือ สามารถตรวจสอบหาได้เฉพาะแต่ตัวเลขเท่านั้น และในขณะที่ทำการถ่ายภาพนั้น จะต้องมีสภาพแวดล้อมของการถ่ายภาพที่เหมาะสมเพื่อผลลัพธ์ที่ถูกต้องและแม่นยำ ยกตัวอย่างเช่น ในกรณีแสงสะท้อนกระทบกับกรอบพลาสติกที่ครอบตัวมิเตอร์ไว้ หรือภาพที่ถ่ายออกมามีสิ่งรบกวนติดอยู่ รวมไปถึงกรณีการถ่ายภาพเอียง และกรณีของการที่ตัวเลขกำลังหมุนไปอยู่ระหว่างกึ่งกลาง ซึ่งกรณีเหล่านี้ผู้จัดทำจะได้นำไปพัฒนาต่อไป

5.2. ขอบเขตการทำงาน

กรณีที่ผู้ใช้ถ่ายภาพบนอุปกรณ์จะต้องถ่ายให้ตรงตามกรอบที่โปรแกรมตั้งไว้ ดังรูปที่ 5.1 เพื่อให้ได้ผลลัพธ์การทำงานที่ถูกต้องแม่นยำ และรวดเร็ว



รูปที่ 5.1 ภาพตัวอย่างจำลองขอบเขตของการถ่ายภาพมิเตอร์จากผู้ใช้งาน

5.3. ข้อเสนอแนะ

เพื่อให้เป็นประโยชน์ต่อไปควรมีการทำงานของโปรแกรมบนเซิร์ฟเวอร์ ที่ติดต่อไปยังผู้ให้บริการต่อไป เพื่อให้ผู้ใช้สามารถตรวจสอบค่าไฟฟ้าในขณะที่ต้องการได้ เป็นต้น รวมไปถึงการพัฒนาบนอุปกรณ์สื่อสารที่สามารถทำงานได้บนอุปกรณ์สื่อสารได้ทันที

เอกสารอ้างอิง

[1] “Artificial Neural Network”,Retrieved 15 Jul,2011,from http://202.28.94.55/web/320417/2548/work1/.../Report_Neural%20Network.doc

[2] Awcock, G.J. and Thomas R., “Applied Image Processing”, Macmillan Press Ltd, 1995.

[3] Thai OCR, http://thaiocr.phaisarn.com, July 10, 2011.

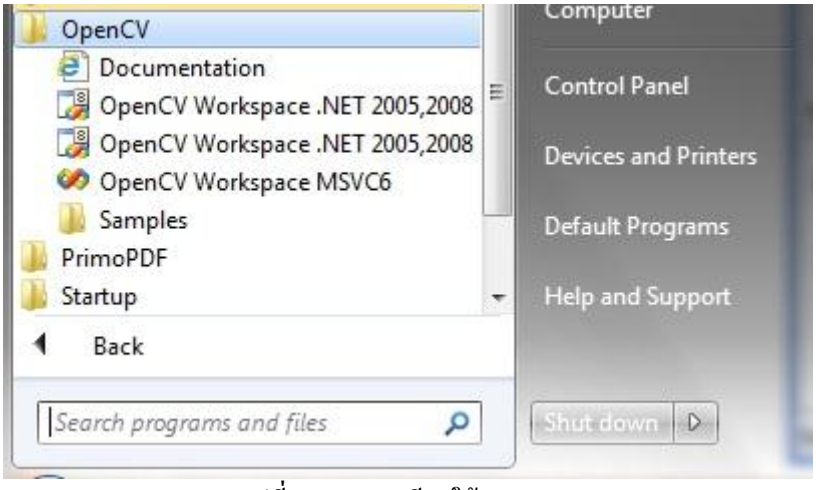
ภาคผนวก

วิธีติดตั้ง และเรียกใช้งาน OpenCV Library

1. เริ่มต้นการติดตั้ง OpenCV Library โดยดาวน์โหลดตัวติดตั้ง ได้ที่ http://sourceforge.net/projects/opencvlibrary ดังรูปที่ 1

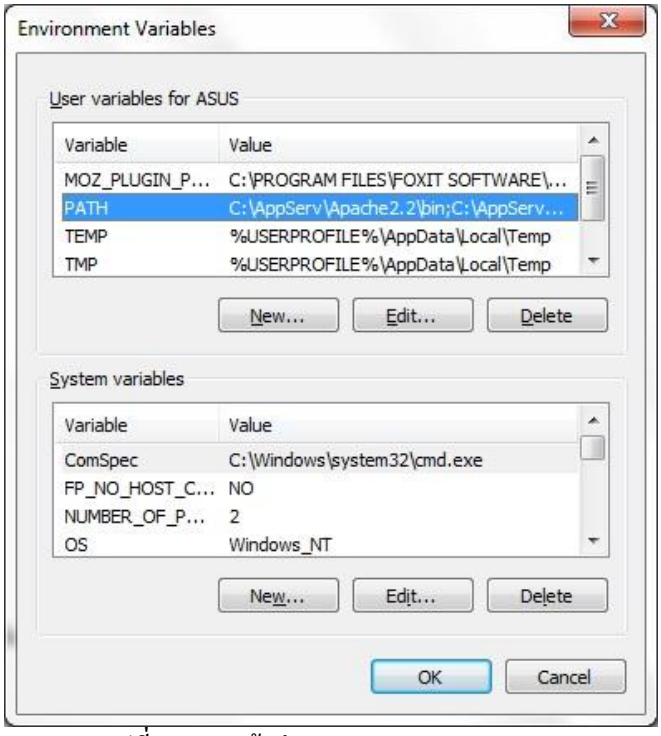
รูปที่ 1 ภาพหน้าต่างดาวน์โหลดตัวติดตั้ง

2. เมื่อติดตั้งตามขั้นตอนเสร็จเรียบร้อยแล้ว การเรียกใช้โปรแกรมสามารถเรียกใช้ที่ Start Taskbar ดังรูปที่ 2

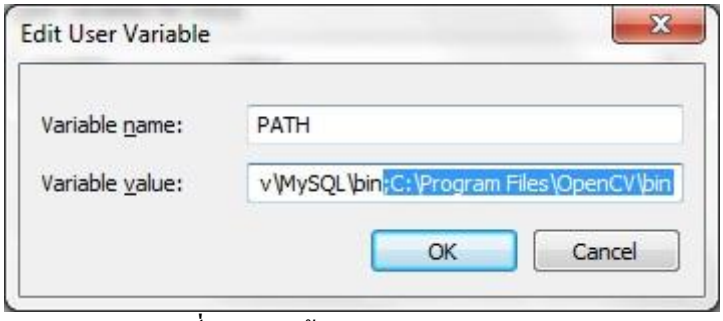


รูปที่ 2 ภาพการเรียกใช้ OpenCV

3. การตั้งค่า Environment Variables เพื่อให้ระบบสามารถใช้งานร่วมกับ Visual C++ 2008 Express Edition ได้ โดยเริ่มต้นที่การตั้งค่าระบบเพื่อค้นหา DLL Files โดยสามารถทำได้โดยการเพิ่มตำแหน่ง Path ที่เก็บ DLL Files โดยเพิ่ม ";C:\Program Files\OpenCV\bin" ต่อท้าย โดยเรียกหน้าต่าง Environment Variables ขึ้นมาทำได้ ดังรูปที่ 3-4

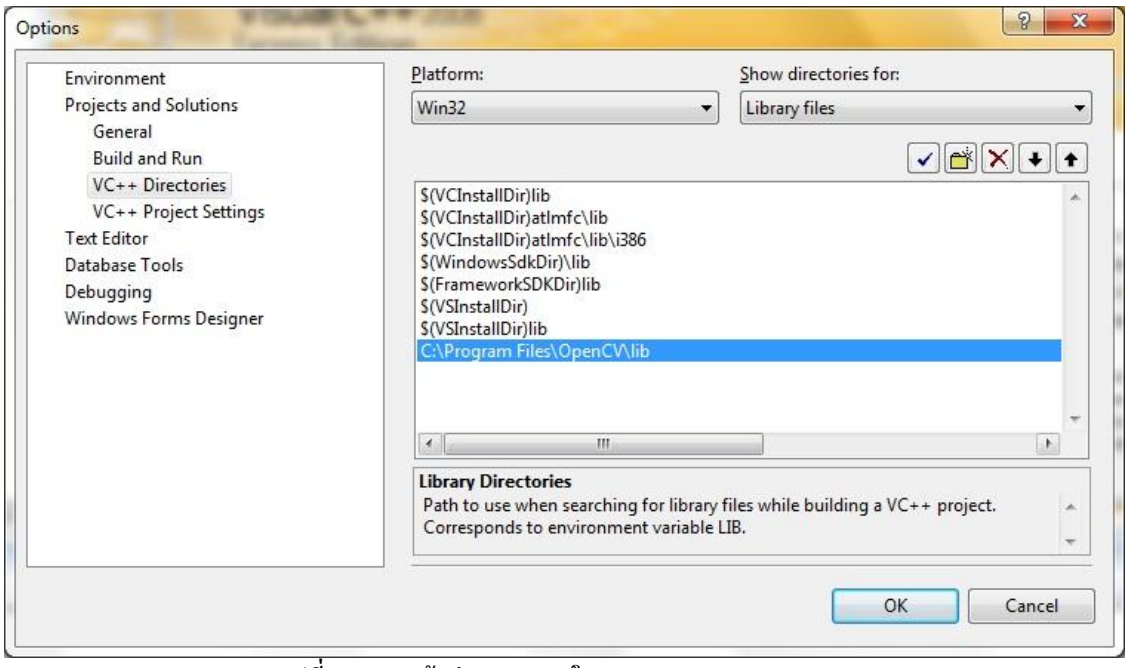


รูปที่ 3 ภาพหน้าต่าง Environment Variables



รูปที่ 4 ภาพหน้าต่าง Edit User Variable

4. ตั้งค่าเพื่อให้โปรแกรมสามารถ Compile ซึ่งในการ Compile แต่ละครั้งนั้นโปรแกรมจำเป็นที่จะต้องเรียกใช้ Function ของ OpenCV ดังนั้นจึงจำเป็นอย่างยิ่งในการทำให้ Visual C++ 2008 Express Edition รองรับ Function ใน Library ดังกล่าว ซึ่งการตั้งค่าเริ่มต้นนั้น ทำได้โดยเปิดโปรแกรม Visual C++ 2008 Express Edition ขึ้นมาแล้วเลือกที่ Tools จากนั้นเลือก Options... เมื่อกดเข้าไปจะพบหน้าต่าง ดังรูปที่ 5



รูปที่ 5 ภาพหน้าต่าง Option ใน Visual C++ 2008 EE

5. เลือกแถบรายการในหน้าต่างด้านซ้ายมือที่ Projects and Solutions จากนั้นเลือก VC++ Directories แล้วไปที่ไปที่ Show directories for: ทางขวาบนของหน้าต่างเลือก Library files จากนั้นเพิ่ม Path ต่อไปนี้ลงไปในรายการสุดท้าย

C:\Program Files\OpenCV\lib

หลังจากเพิ่ม Path ของ Library files เรียบร้อยแล้ว ต่อไปเป็นการเพิ่ม Path ของ Include files และ Source files โดยเปลี่ยนหัวข้อที่ Show directories for: ดังรูปที่ 6-7

โดยเพิ่ม Path ต่อไปนี้ข้างท้ายของรายการ

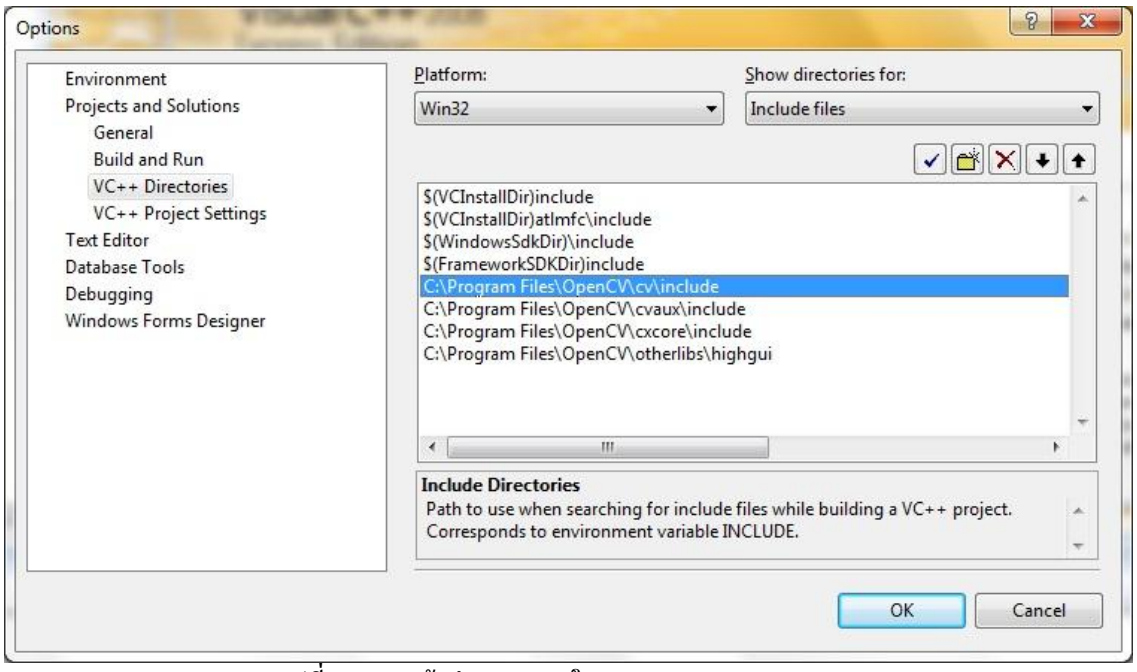
Include files

C:\Program Files\OpenCV\cv\include

C:\Program Files\OpenCV\cvaux\include

C:\Program Files\OpenCV\cxcore\include

C:\Program Files\OpenCV\otherlibs\highgui



รูปที่ 6 ภาพหน้าต่าง Option ใน Visual C++ 2008 EE

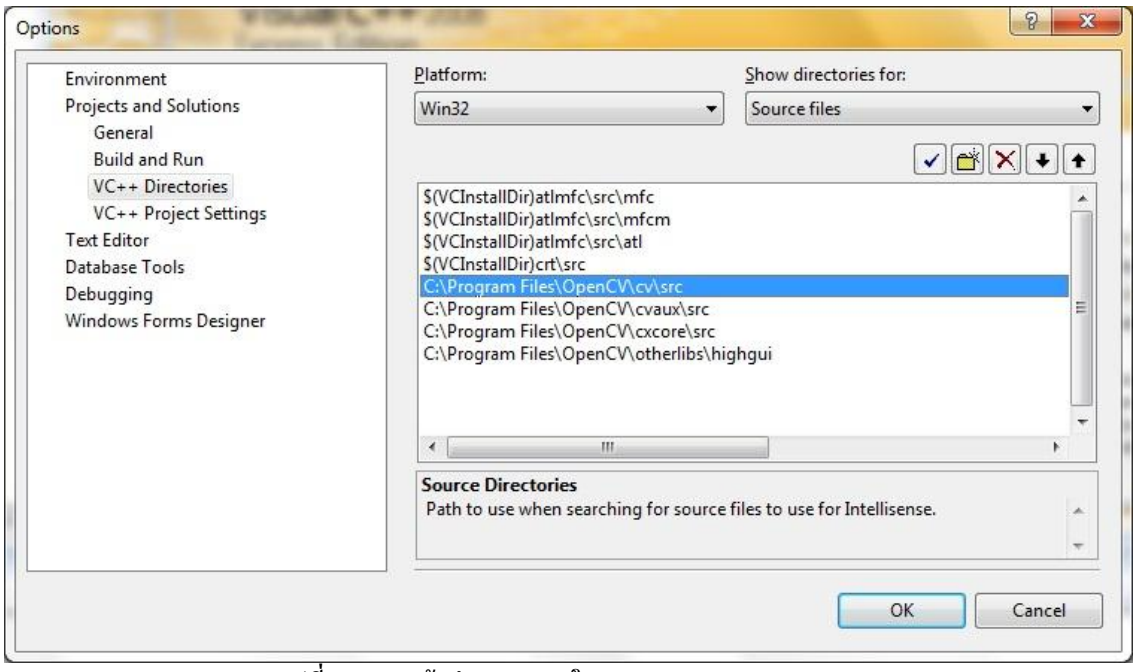
Source files

C:\Program Files\OpenCV\cv\src

C:\Program Files\OpenCV\cvaux\src

C:\Program Files\OpenCV\cxcore\src

C:\Program Files\OpenCV\otherlibs\highgui



รูปที่ 7 ภาพหน้าต่าง Option ใน Visual C++ 2008 EE

6. การตั้งค่าระบบเพื่อสามารถเรียกใช้งาน Library files เนื่องจาก Library files ต่าง ๆ ที่มาพร้อมกับตัวติดตั้งที่กล่าวมาข้างต้น เป็น Library ที่ถูกสร้างขึ้นเพื่อใช้กับ Visual C++ 2005 จึงจำเป็นที่จะต้องแก้ไข ซึ่งวิธีที่แนะนำ คือ ทำการ Download และติดตั้ง Package เสริมจาก Microsoft ซึ่งสามารถ Download ได้ที่ http://www.microsoft.com/downloads/details.aspx?familyid=200B2FD9-AE1A-4A14-984D-389C36F85647&displaylang=en เมื่อกดเข้าไปจะปรากฏ ดังรูปที่ 8

รูปที่ 8 ภาพหน้าต่างดาวน์โหลด Package เสริมจาก Microsoft

7. ขั้นตอนสุดท้ายคือการตั้งค่าสำหรับการ Build Executable File จากการทำตามขั้นตอนข้างต้นทั้งหมดก็พร้อมที่จะสร้าง project ใน Visual C++ ได้แล้ว แต่ในการ run project นั้นโปรแกรมจำเป็นที่จะต้องทำขั้นตอนในส่วนของ linking เพื่อเพิ่มรายการ Library ของ OpenCV โดยทำได้โดยเข้าโปรแกรม Visual C++ 2008 Express Edition แล้วคลิ๊กขวาที่ชื่อ Project แล้วเลือก Properties จากนั้นเลือกเลือก Configuration Properties >>> Linker >>> Input แล้วทำการเพิ่มรายชื่อ Library ดังต่อไปนี้

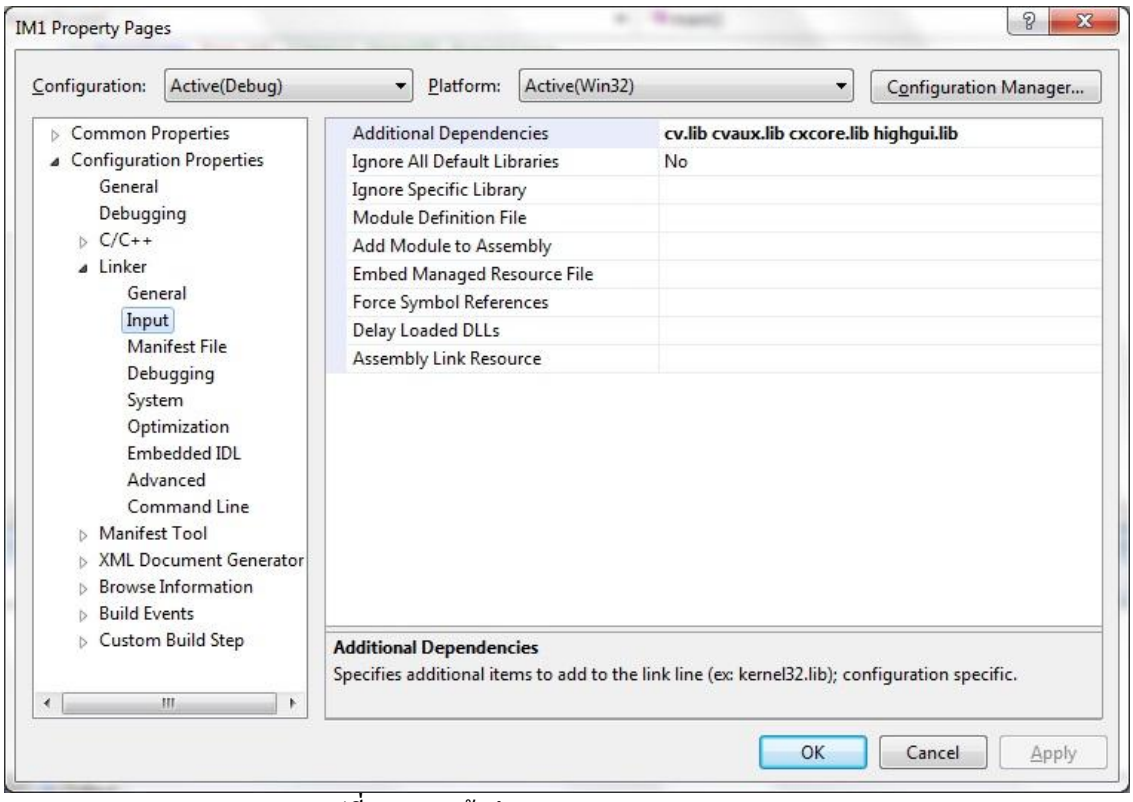
cv.lib

cvaux.lib

cxcore.lib

highgui.lib

ในช่อง Additional Dependecies จะได้ ดังรูปที่ 9 แล้วกดปุ่ม OK เป็นอันเสร็จสิ้นขั้นตอนการติดตั้ง Library OpenCV



รูปที่ 9 ภาพหน้าต่างการ Build Executable File