ติดตามผู้เขียน
ผ่านช่องทางยูทูป

 https://www.youtube.com/c/KongRuksiamOfficial/  https://www.facebook.com/KongRuksiamTutorial/

หรือสแกน QRCODE

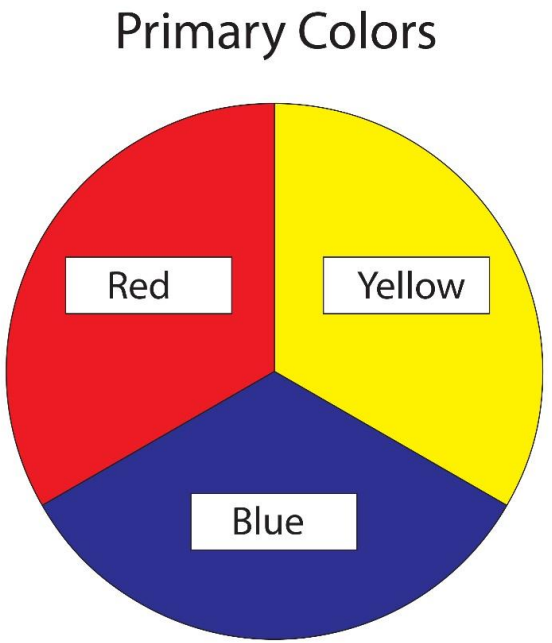
เรียนเนื้อหา Python & OpenCV ได้ที่

https://bit.ly/3bzfVRo

ทฤษฎีสี - (Color Theory)

แม่สี - (Primary Color)

แม่สี คือกลุ่มของสีพื้นฐานที่มนุษย์มองเห็น มี 3 สี ได้แก่ สีแดง สีเขียว และสีน้ำเงิน ที่เรียกว่า ทฤษฎีตัวกระตุ้นสีทั้งสาม และสามารถนำแม่สีมาผสมรวมกันเพื่อทำให้เกิดสีต่างๆ สำหรับการใช้ในชีวิตประจำวันของมนุษย์



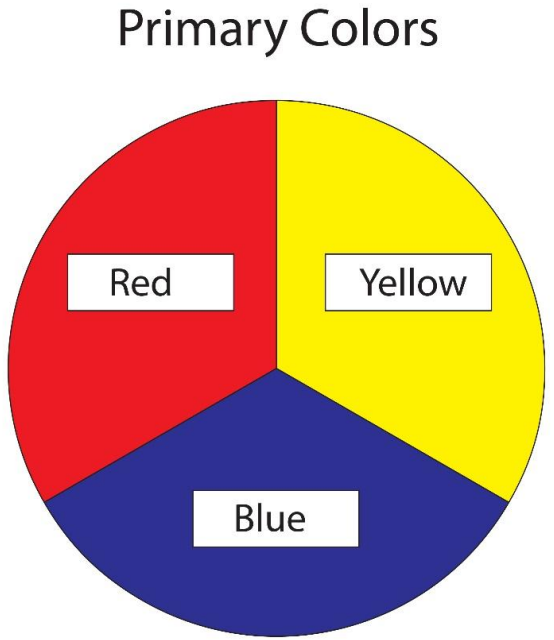
ทฤษฎีสี - (Color Theory)

การผสมสีแบบบวก (Additive Color)

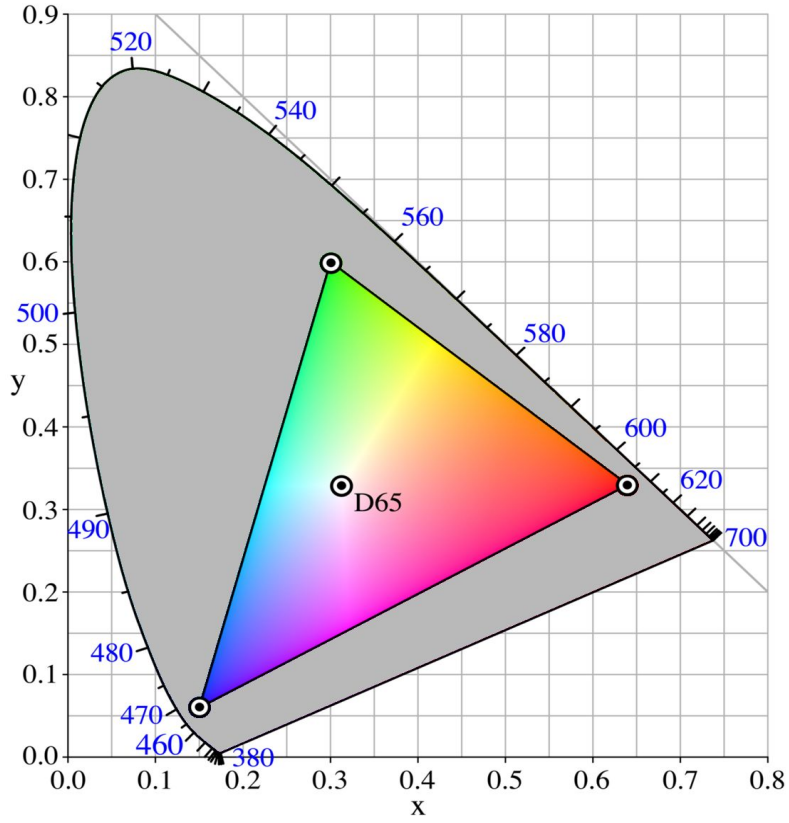
- สีแดง (R) สีเขียว (G) สีน้ำเงิน (B)

การผสมสีแบบลบ (Subtractive Color)

- สีน้ำเงินอมเขียว (C : Cyan) สีแดงอมม่วง (M : Magenta) และสีเหลือง (Y: Yellow)

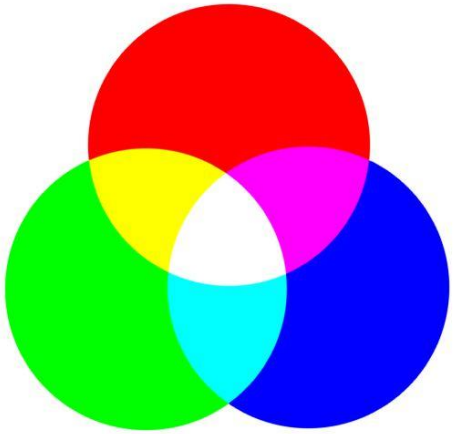


รูปแบบสี (Color Model)



รูปแบบสีหรือระบบสี คือแบบจำลองทางคณิตศาสตร์ซึ่งโดยปกติจะมีองค์ประกอบของสี 3 หรือ 4 ค่า โดยเราจะเรียกกลุ่มของสีนี้ว่า **พื้นที่สี**

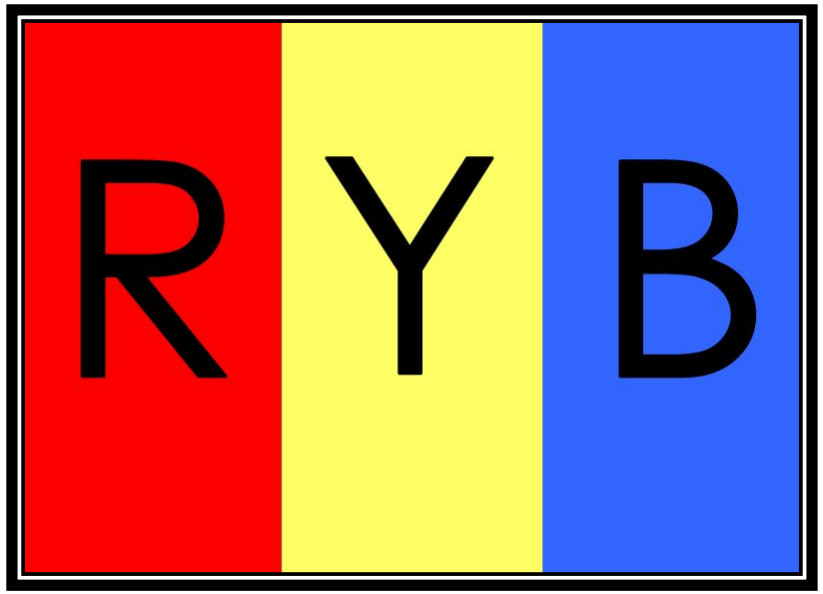
รูปแบบสี RGB



รูปแบบนี้เป็นการรวมเข้าด้วยกันของแม่สีแบบบวก

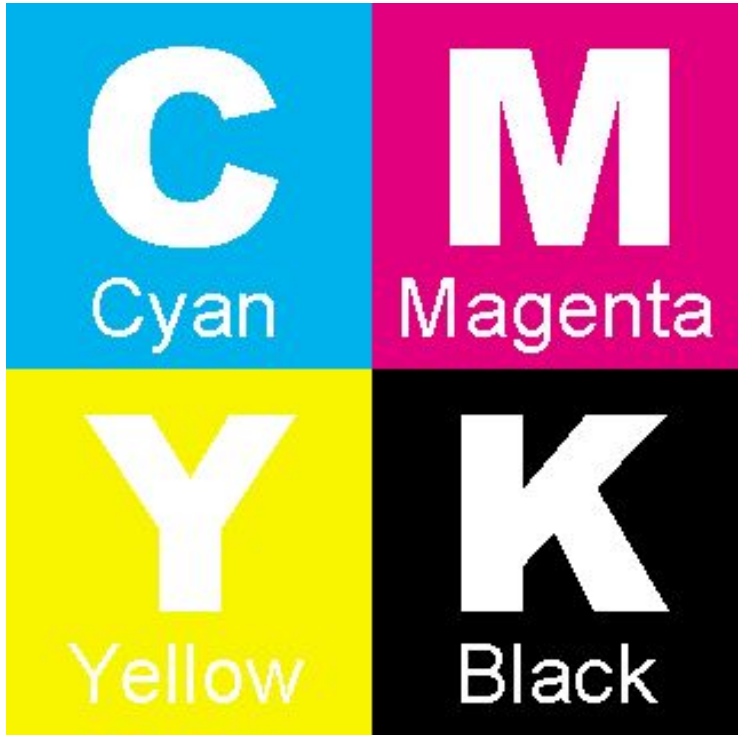
ได้แก่ สีแดง สีเขียว และสีน้ำเงิน วัตถุประสงค์หลักของรูปแบบสี RGB เพื่อการตรวจวัดและการแสดงผลของภาพในระบบอิเล็กทรอนิกส์ เช่น โทรทัศน์ และคอมพิวเตอร์ และเป็นพื้นฐานในการรับรู้สีของมนุษย์

ระบบสี RYB (Red Yellow and Blue)



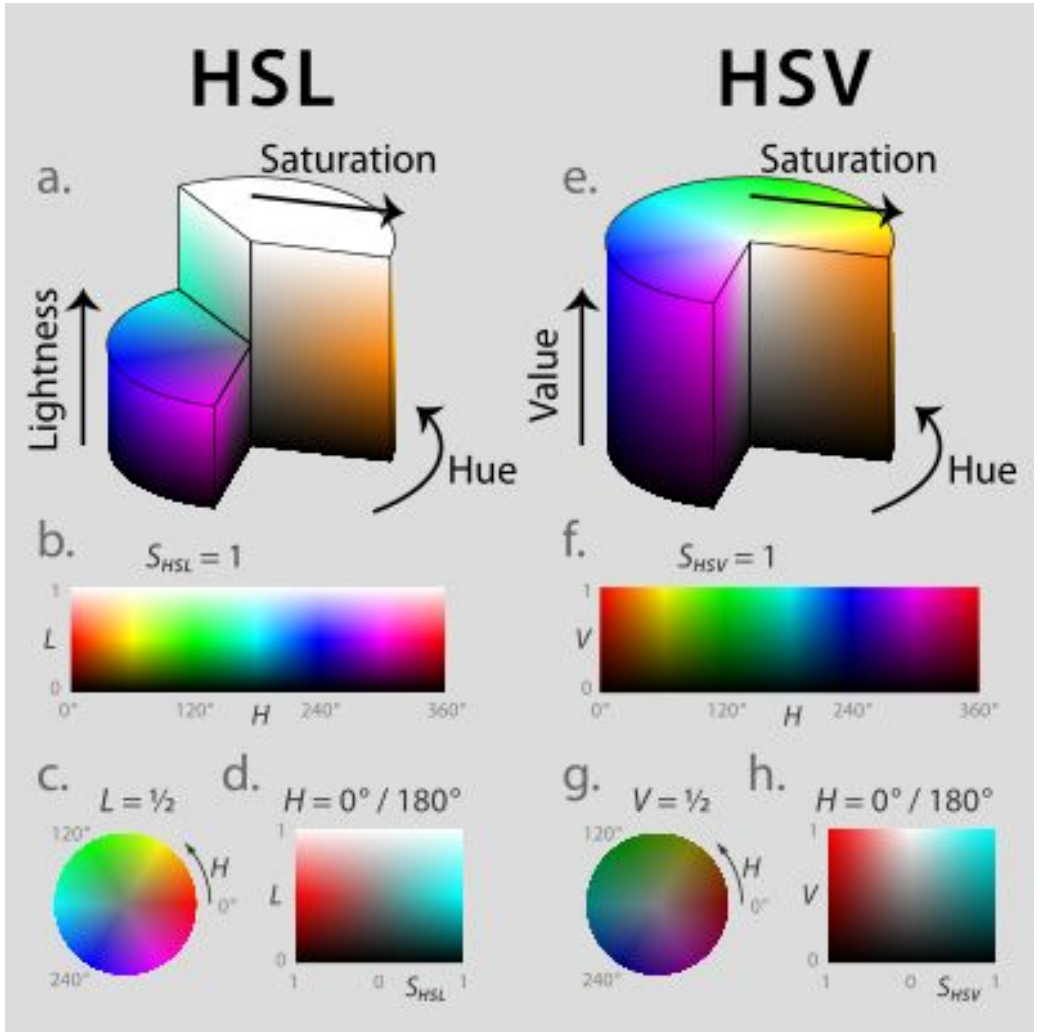
ระบบสี RYB **(สีแดง สีเหลือง สีน้ำเงิน)**
คือ รูปแบบมาตรฐานของชุดแม่สีแบบลบที่ใช้สำหรับการผสมสารสีซึ่งใช้ในงานศิลปะ

ระบบสี CMYK



ระบบสี CMYK เป็นระบบผสมสีแบบลบ ใช้ในงานพิมพ์สีเป็นส่วนใหญ่ ได้แก่ สีน้ำเงินอมเขียว (C) สีแดงอมม่วง (M) และสีเหลือง (Y) และสุดท้าย คือ สีดำ (K) ซึ่งเป็นอักษรย่อสุดท้ายของคำว่า Black เพื่อจะไม่ได้ซ้ำกับคำว่า Blue ที่แปลว่า สีน้ำเงิน

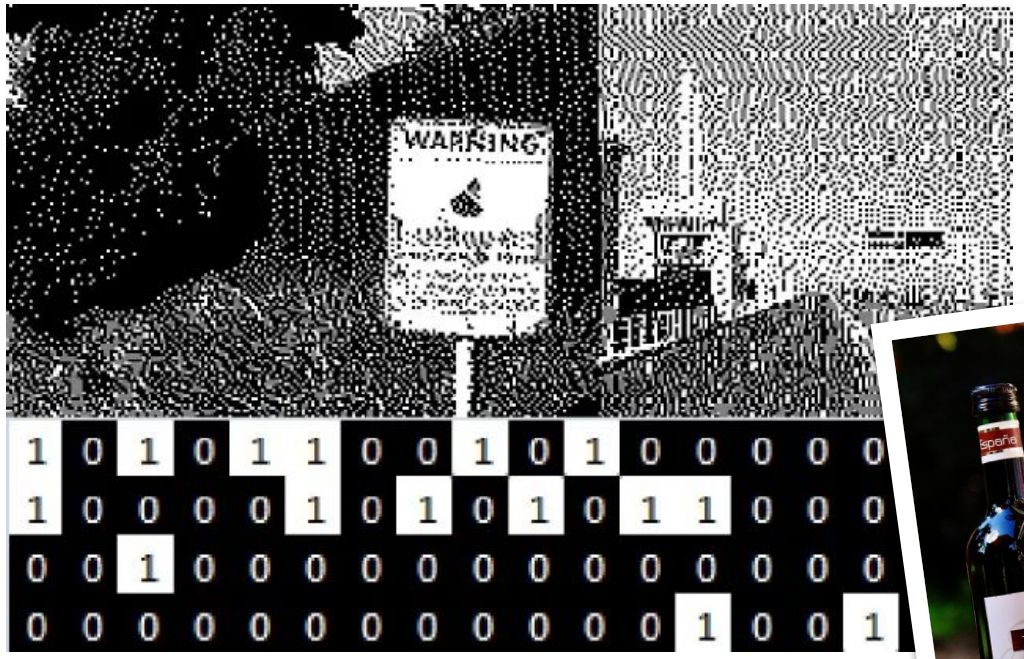
ระบบสี - HSV และ HSL



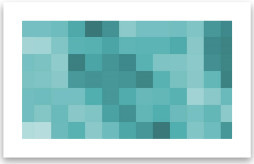
ระบบสี HSV (Hue , Saturation , Value) และ HSL (Hue,Saturation,Lightness) และมีการใช้อย่างแพร่หลายในคอมพิวเตอร์กราฟิก โดยเฉพาะอย่างยิ่งเป็นซอฟต์แวร์การแก้ไขภาพ

HSV และ HSL จะปรับปรุงระบบที่เป็นรูปลูกบาศก์ของระบบสี RGB ด้วยการจัดเรียงแต่ละสีในแนวรัศมีรอบแกนกลางของสีที่เป็นกลาง ซึ่งจะมีสีดำอยู่ที่ด้านล่างและมีสีขาวอยู่ที่ด้านบน สีที่สดใสเต็มที่ของแต่ละสีจะอยู่ในวงกลมเป็นวงล้อสี

ชนิดของภาพ


WARNING
1 0 1 0 1 1 0 0 1 0 1 0 0 0 0 0
1 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1

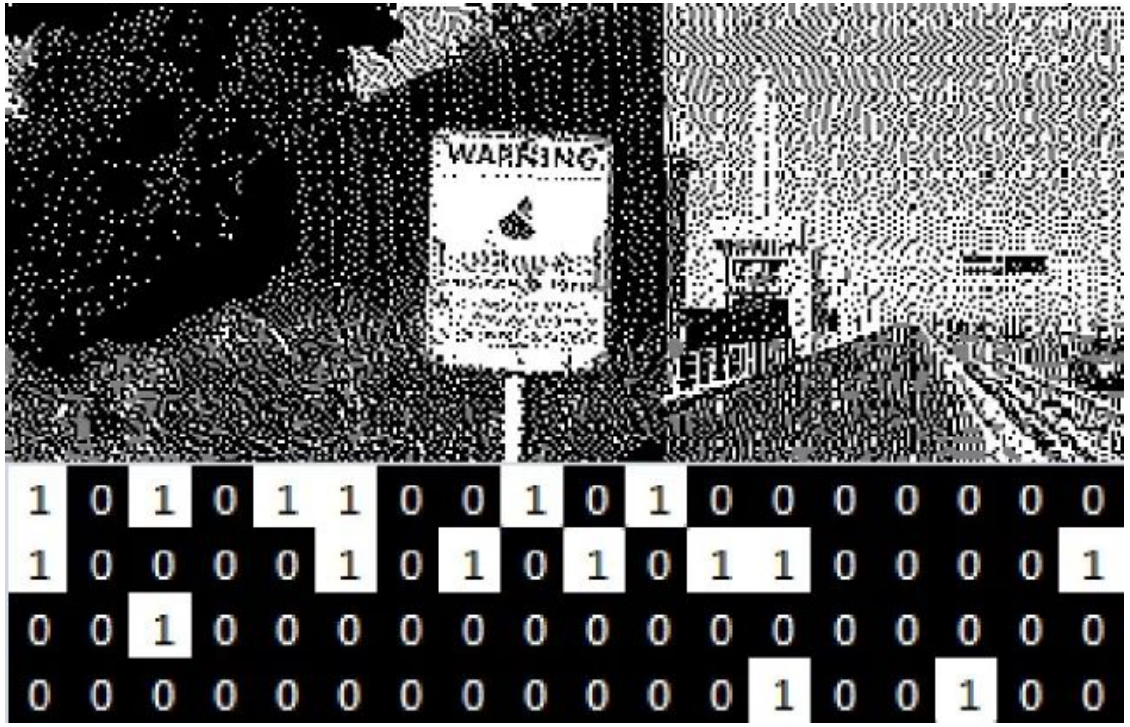
Pixel คืออะไร



Pixel มาจากคำว่า Picture (ภาพ) กับคำว่า Element (พื้นฐาน)

คือ หน่วยพื้นฐานซึ่งเล็กที่สุดของภาพดิจิตอล เทียบได้กับจุดสีของภาพ 1 จุด หลากหลายสี หลายๆจุดที่เรียงชิดติดกันถูกรวมกันทำให้เกิดเป็นภาพนั่นเอง ***แต่ 1 Pixel จะเป็นสีหนึ่งสีใดเพียงสีเดียวเท่านั้นจะมีสีอื่นไม่ได้*** เนื่องจากว่าเป็นส่วนที่เล็กที่สุดของการแสดงผลพิกเซลนั้นมีความสำคัญต่อการสร้างกราฟฟิกของคอมพิวเตอร์มาก เพราะทุกๆส่วนของกราฟฟิก เช่น จุด เส้น แบบลายและสีของภาพล้วนเกิดจากพิกเซลทั้งสิ้น

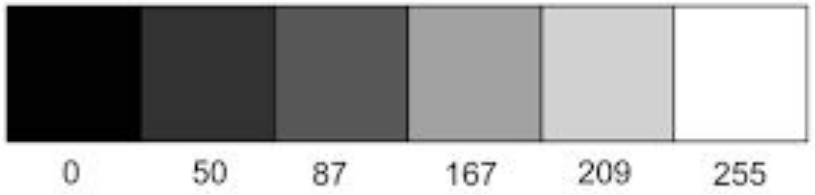
Binary Image



ไบนารีในทางดิจิตอล หมายถึงมีเพียง 2 สถานะ คือ 0 และ 1

ภาพไบนารี่ก็จะมีแค่ความเข้ม 2 ค่าเท่านั้นคือ 0 และ 1 หมายความว่า พิกเซลใดที่มีค่าเป็น 0 ก็จะหมายถึงว่าพิกเซลนั้นจะแสดงสีดำ และพิกเซลใดที่มีค่าเป็น 1 ก็จะหมายถึงว่าพิกเซลนั้นจะแสดงสีขาว

Grayscale Image



ภาพเกรย์สเกลหรือภาพระดับสีเทา

พูดง่ายๆก็ภาพขาว-ดำ-เทา โดยจะมีระดับความเข้มของสีเท่าคือ 0-255 (8 bit) ภาพเกรย์สเกลเกิดจากการแปลงภาพสี RGB มาเป็นภาพ Grayscale โดยใช้สูตรทางคณิตศาสตร์ดังนี้

$$Gray=0.299\times R+0.587\times G+0.114\times B$$

Gray : ค่าความเข้มของสีเทาโดยจะมีค่าระหว่าง 0-255
R : ค่าความเข้มของสีแดงโดยจะมีค่าระหว่าง 0-255
G : ค่าความเข้มของสีเขียวโดยจะมีค่าระหว่าง 0-255
B : ค่าความเข้มของสีน้ำเงินโดยจะมีค่าระหว่าง 0-255

RGB IMAGE

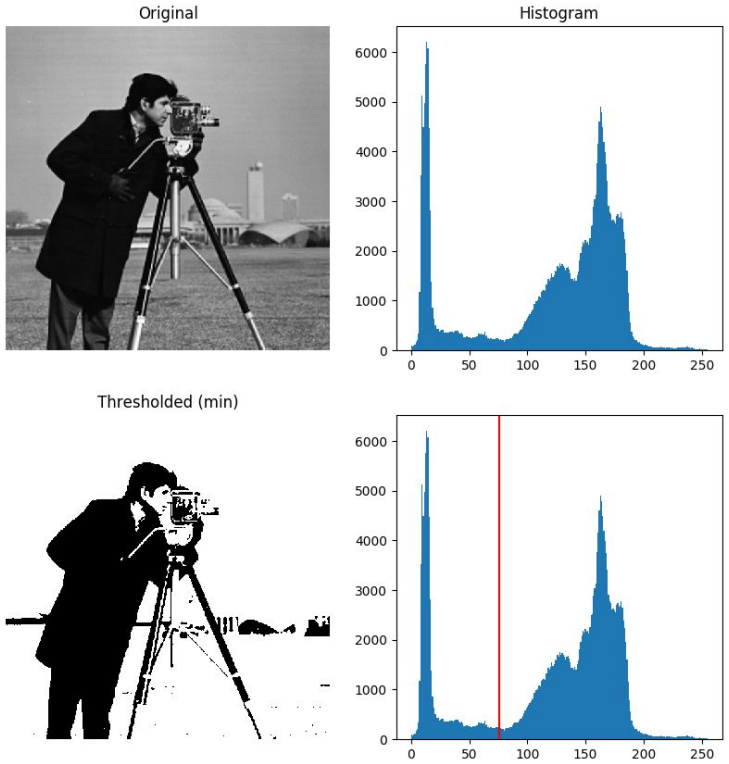
ภาพในระบบสี RGB คือค่าความเข้มแสงในสีต่างๆ คือ สีแดง(R),สีเขียว(G),สีน้ำเงิน(B)

ภาพสีในระบบ RGB จะมีจำนวนบิตต่อพิกเซลคือ 24 บิต หมายความว่าสามารถแสดงสีได้ถึง 16,777,216

สี จำนวนบิตของข้อมูลในแต่ละพิกเซลนั้นจะเป็นตัวกำหนดความแตกต่างของระดับสีซึ่งถ้าข้อมูลในพิกเซลนั้นมี8บิตก็หมายความว่า 8 บิต = 2^8=256 สี (0-255)

- Red 8 bit
- Green 8 bit
- Blue 8 bit
- RGB = 24 bit

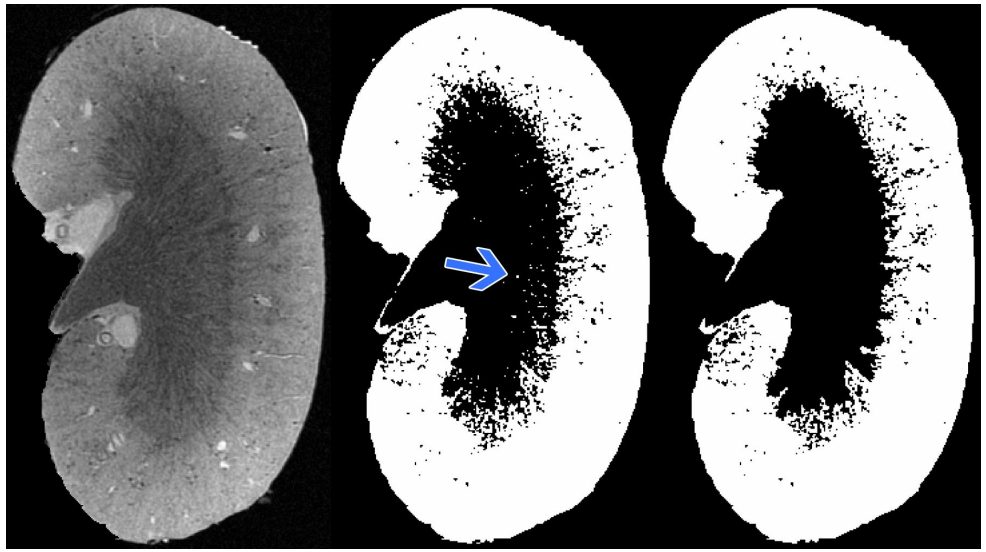
Thresholding

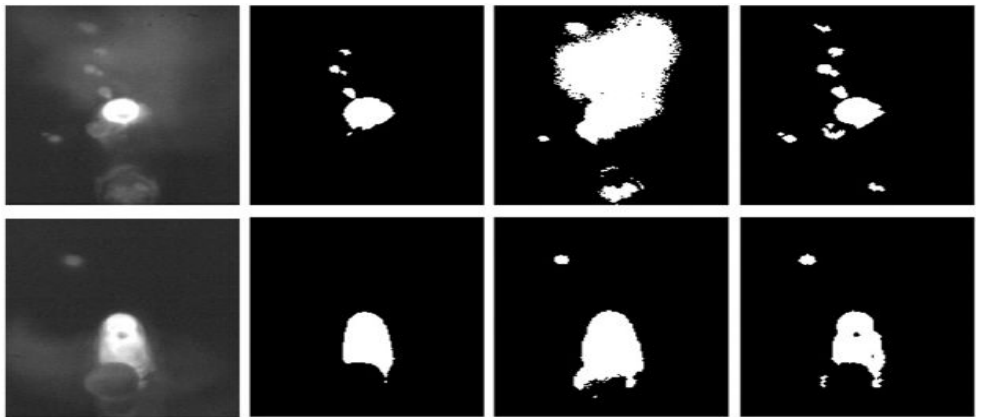


ความรู้พื้นฐานก่อนตะลุย Thresholding

- การทำ Thresholding หลักๆจะทำงานร่วมกับภาพแบบ GrayScale และ Binary ซึ่งมีรายละเอียดโครงสร้างของภาพดังนี้
- GrayScale มีระดับสีหรือระดับความสว่างได้ 256 แบบ
  - **0 (ดำ) - 255 (ขาว)**
- Binary มีระดับสี 2 แบบ คือ ขาว - ดำสนิท
  - **แทนด้วยค่า 0 (ดำ) กับ 1 (ขาว) หรือบางที่ใช้ 0 กับ 255**

ความรู้พื้นฐานก่อนตะลุย Thresholding



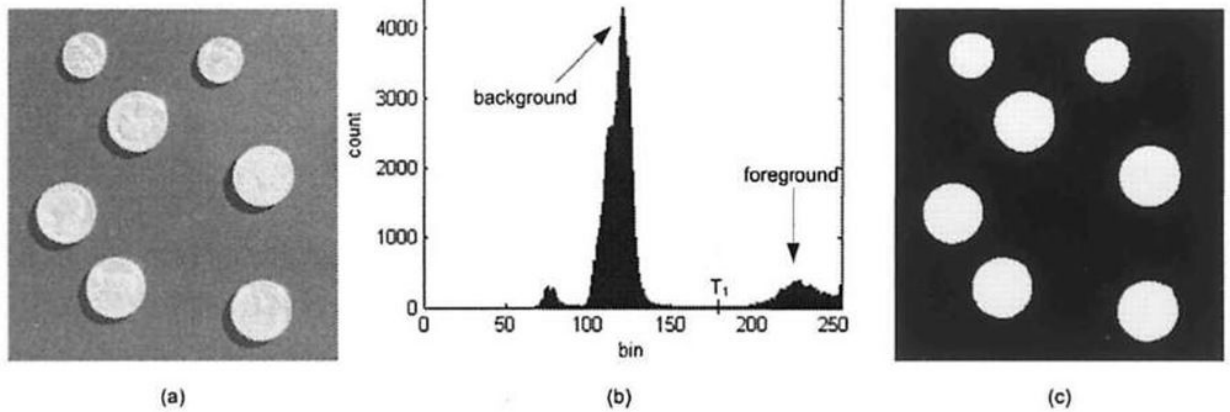


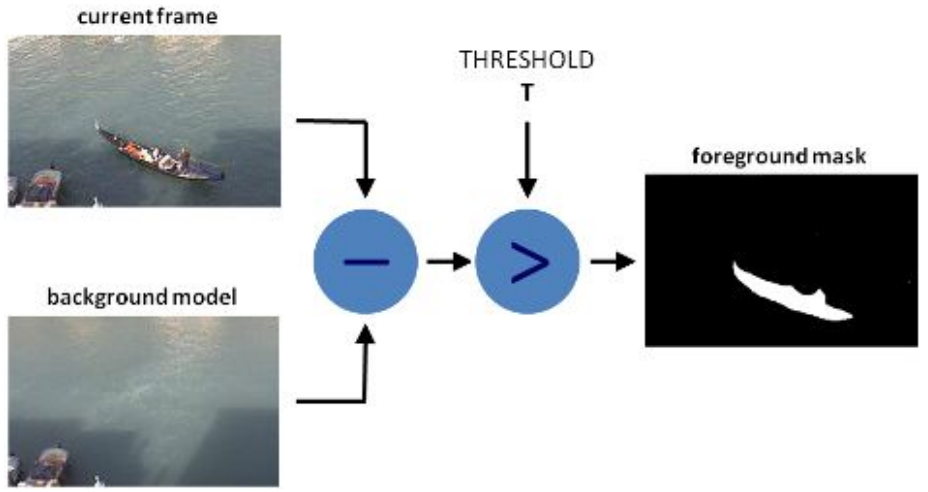
- โดยทั่วไปคอมพิวเตอร์ประมวลผลภาพจำเป็นต้องใช้งานภาพแบบไบนารี่ซึ่งมีค่าสี 2 ค่า **คือ ขาว - ดำ (0 กับ 1) หรือบางที่ใช้ (0 กับ 255)**

- แต่วิธีการส่วนใหญ่หรือที่ใช้งานจริงจะสนับสนุนแค่ส่วนของภาพระดับเทาเท่านั้น (GrayScale)

ทำไมต้องใช้ภาพขาวดำ

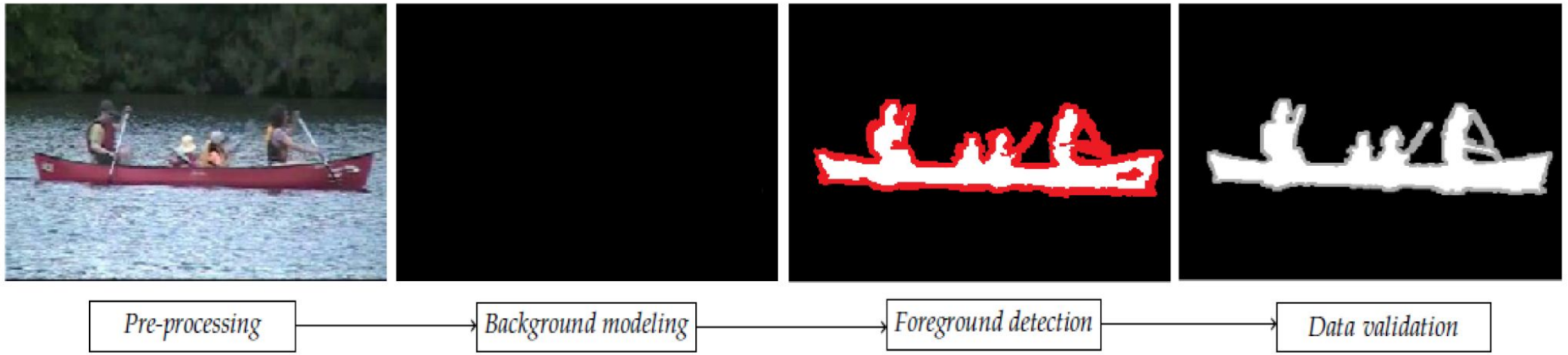
- เพราะมีรูปแบบการเก็บข้อมูล 2 ส่วนประกอบด้วย Foreground และ Background หรือภาพพื้นหลังและสิ่งที่เราสนใจในภาพ คอมพิวเตอร์จะจำแนกข้อมูลได้ง่ายกว่าภาพสี





ทำไมต้องใช้ภาพขาวดำ

- สะดวกในการนำมาวิเคราะห์ข้อมูลหรืองานด้านอื่นๆ เช่น ใช้ในงานอุตสาหกรรม ใช้ตรวจสอบลายนิ้วมือ วิเคราะห์เอกสาร ด้านการแพทย์ เป็นต้น



สิ่งที่ต้องเริ่มต้นทำอันดับแรก คือ

- แปลงภาพสี เป็น ภาพระดับเทา GrayScale
- แปลงภาพ GrayScale เป็นภาพขาวดำ (Binary)

Thresholding

การทำเทรชโฮลด์ คือ ขั้นตอนง่ายๆ ในการแยกส่วนประกอบของภาพ โดยเกิดจากการนำภาพสีมาทำการแปลงให้เป็นภาพสีเทา แล้วทำภาพเทามาทำให้เป็นภาพไบนารีหรือภาพขาวดำ โดยใช้คำสั่งการทำเทรชโฮลด์เพื่อปรับปรุงภาพขาวดำให้มีความชัดเจนตามที่ต้องการ โดย**พิจารณาค่าและเปรียบเทียบกับค่าจุดแบ่ง (Threshold Value) และตัดสินการแบ่งประเภท**

Thresholding

- การแบ่งประเภท ก็คือ การทำภาพขาวดำ (Binary) นั่นเอง
- โดยเราจะเลือกค่ามา 1 ค่า
  - ค่ากลางที่นิยมใช้จะอยู่ที่ 128 โดยค่านี้จะมีผลดีเมื่อภาพนั้นไม่มีสัญญาณรบกวน (Noise) และมีภาพพื้นหลังสม่ำเสมอ
  - ถ้าค่าระดับเทามีค่ามากกว่าค่าที่เลือกมา จะให้ค่าจุดนั้นเป็น **สีขาว**
  - แต่ถ้าน้อยกว่า จะให้ค่าจุดนั้นเป็น **สีดำ**

    เรียกการแบ่งประเภทจากค่าที่เลือกมาว่า การทำเทรดโฮลด์

คำสั่งสำหรับทำ Threshold ง่ายๆ

cv2.threshold(1,2,3,4)

1 คือ ภาพ Grayscale

2 คือ ค่าเกณฑ์หรือจุดแบ่งในการจำแนก Pixel

3 คือ maxValue ค่าสูงสุด

4 คือ รูปแบบการแบ่งประเภท

คำสั่ง Threshold

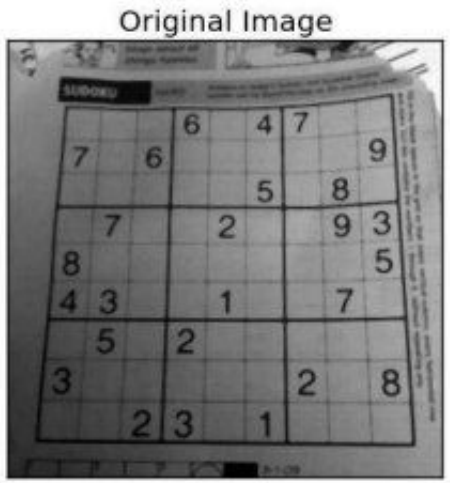
ชื่อรูปแบบการแบ่งประเภท

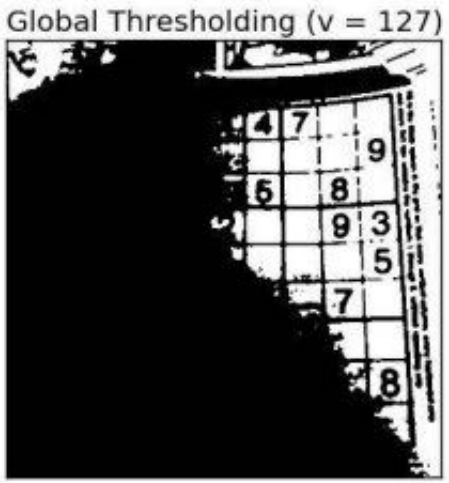
- cv2.THRESH_BINARY
- cv2.THRESH_BINARY_INV
- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO
- cv2.THRESH_TOZERO_INV

คำสั่ง Threshold

รูปแบบการแบ่งประเภท	ความหมาย
cv2.THRESH_BINARY	ถ้าค่าสูงกว่าค่าจุดแบ่งจะเปลี่ยนเป็น 255 ที่เหลือเป็น 0
cv2.THRESH_BINARY_INV	ถ้าค่าสูงกว่าค่าจุดแบ่งจะเปลี่ยนเป็น 0 ที่เหลือเป็น 255
cv2.THRESH_TRUNC	ถ้าค่าสูงกว่าค่าจุดแบ่งจะเปลี่ยนให้มีค่าเท่ากับค่าจุดเปลี่ยนที่เหลือมีค่าเท่าเดิม
cv2.THRESH_TOZERO	ถ้าค่าต่ำกว่าค่าจุดแบ่งจะเปลี่ยนเป็น 0 ที่เหลือคงเดิม
cv2.THRESH_TOZERO_INV	ถ้าค่าสูงกว่าค่าจุดแบ่งจะเปลี่ยนเป็น 0 ที่เหลือคงเดิม

Adaptive Thresholding





จากการใช้งานคำสั่ง **cv2.threshold()** จะสังเกตุว่าต้องมีกำหนดค่าจุดแบ่งขึ้นมาใช้ซึ่งเป็นค่าคงที่และใช้ค่าดังกล่าวทั่วทุกจุดของภาพ

Adaptive Thresholding

วิธีดังกล่าวมีข้อเสียคือไม่สามารถใช้งานได้ผลดีกับภาพที่มีค่าความสว่างไม่สม่ำเสมอกันได้เอง ดังนั้นเพื่อแก้ปัญหาดังกล่าวเราจะใช้ส่วนที่เรียกว่า

“ Adaptive Thresholding”

หมายถึง กำหนดจุดแบ่งให้สามารถปรับเปลี่ยนหรือยืนหยุ่นค่าได้โดยอ้างอิงตามบริเวณพื้นที่ใกล้เคียงกันว่าควรกำหนดค่าจุดแบ่งแต่ละพื้นที่มีค่าเท่าใด (ปรับให้เหมาะสม)

คำสั่ง Adaptive Thresholding

cv2.adaptiveThreshold(1,2,3,4,5,6)

1 คือ ภาพ Grayscale

2 คือ maxValue ค่าสูงสุด

3 คือ AdaptiveMethod (รูปแบบการปรับค่าจุดแบ่ง)

4 คือ ThresholdType (รูปแบบการพิจารณาจุดเปลี่ยน)

5 คือ BlockSize (ขนาด Block) พื้นที่สำหรับพิจารณาค่าเหมาะสม

6 คือ ค่า C (ค่า C)

ค่า Adaptive Method

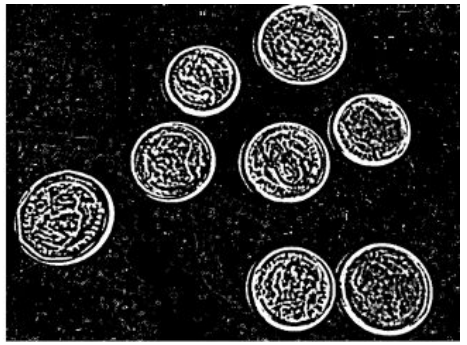
ชื่อรูปแบบ	ความหมาย
cv2.ADAPTIVE_THRESH_MEAN_C	คำนวณค่าเฉลี่ยรอบๆ BlockSize แล้วหักจากค่า C
cv.ADAPTIVE_THRESH_GAUSSIAN_C	คำนวณค่าเฉลี่ยรอบๆ BlockSize แล้วอ้างอิงทำการทำงานกับฟังก์ชั่น Gaussian และค่า C

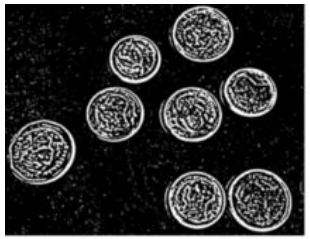
Morphological

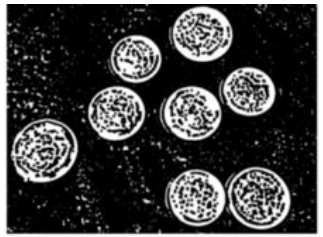
https://www.youtube.com/c/KongRuksiamOfficial/ https://www.facebook.com/KongRuksiamTutorial/

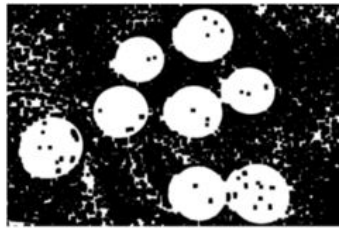
เป็นเทคนิคสำหรับการวิเคราะห์และประมวลผลภาพ โดยอาศัยโครงสร้างทางเรขาคณิตบนพื้นฐานของทฤษฎีเซต ทฤษฎีตาข่ายโครงสร้างของเครือข่ายและฟังก์ชั่นแบบสุ่ม เทคนิคนี้นิยมนำมาใช้งานกับภาพดิจิตอล**เพื่อวิเคราะห์พื้นผิว ขนาด รูปร่าง** เป็นต้น

Morphological







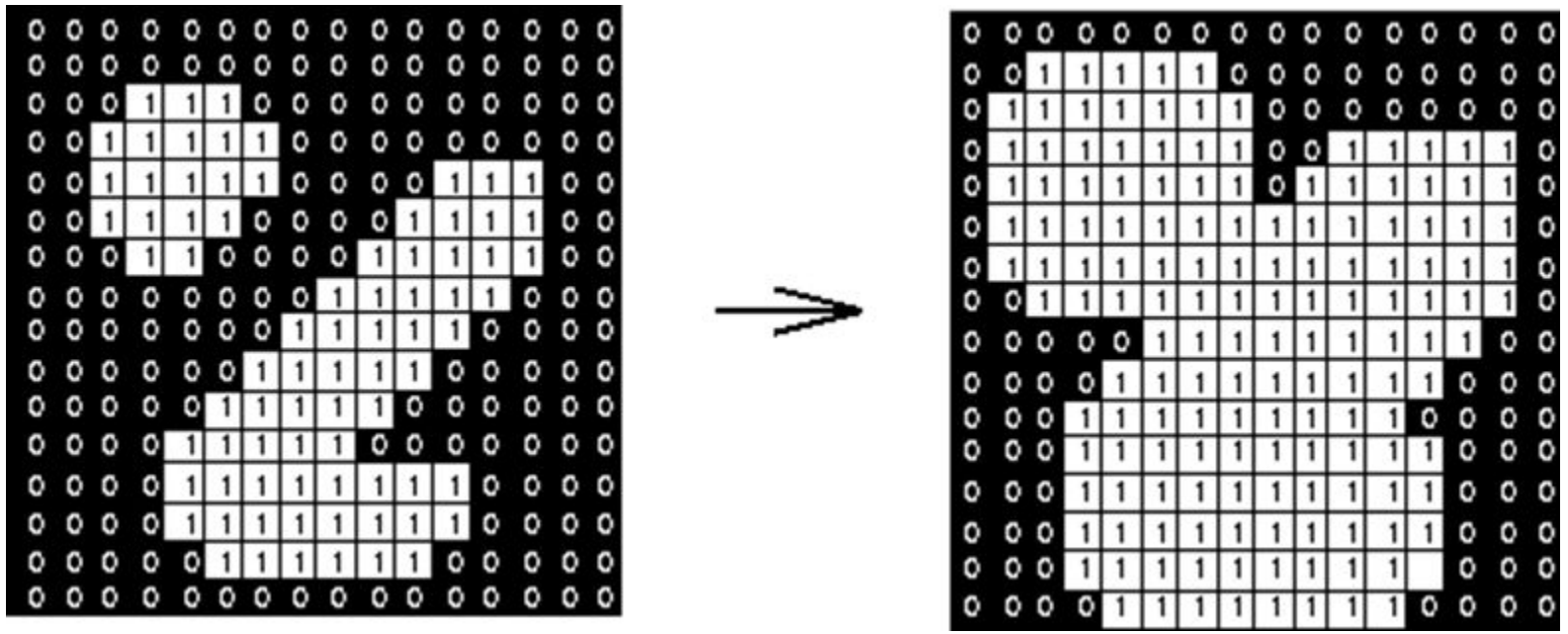


1 x 1 2 x 2 3 x 3

การประมวลผลรูปร่างและโครงสร้างภาพนั้นจะขึ้นอยู่กับรูปร่างของค่า Pixels ในภาพ ส่วน Output ขึ้นอยู่กับการเปรียบเทียบของ Pixels ในภาพ Input กับพื้นที่ใกล้เคียงโดยการเลือกขนาดและรูปร่างของพื้นที่ใกล้เคียงนั้น จะอาศัยตัวดำเนินการ 4 รูปแแบบ

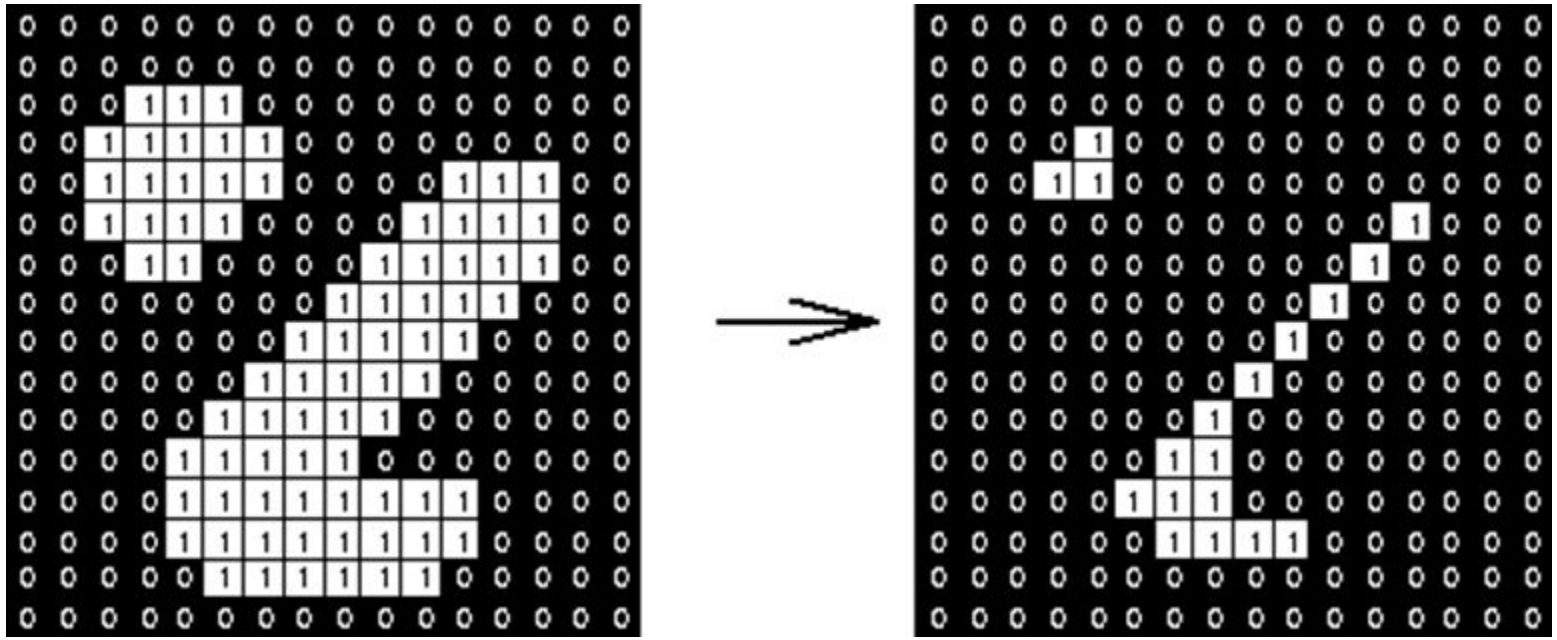
- **การขยายภาพ (Dilation)**
- **การกร่อนภาพ (Erosion)**
- **การเปิดภาพ (Opening)**
- **การปิดภาพ (Closing)**

1.การขยายภาพ (Dilation) สำหรับตรวจสอบและขยายรูปทรงที่มีอยู่ในภาพ Input

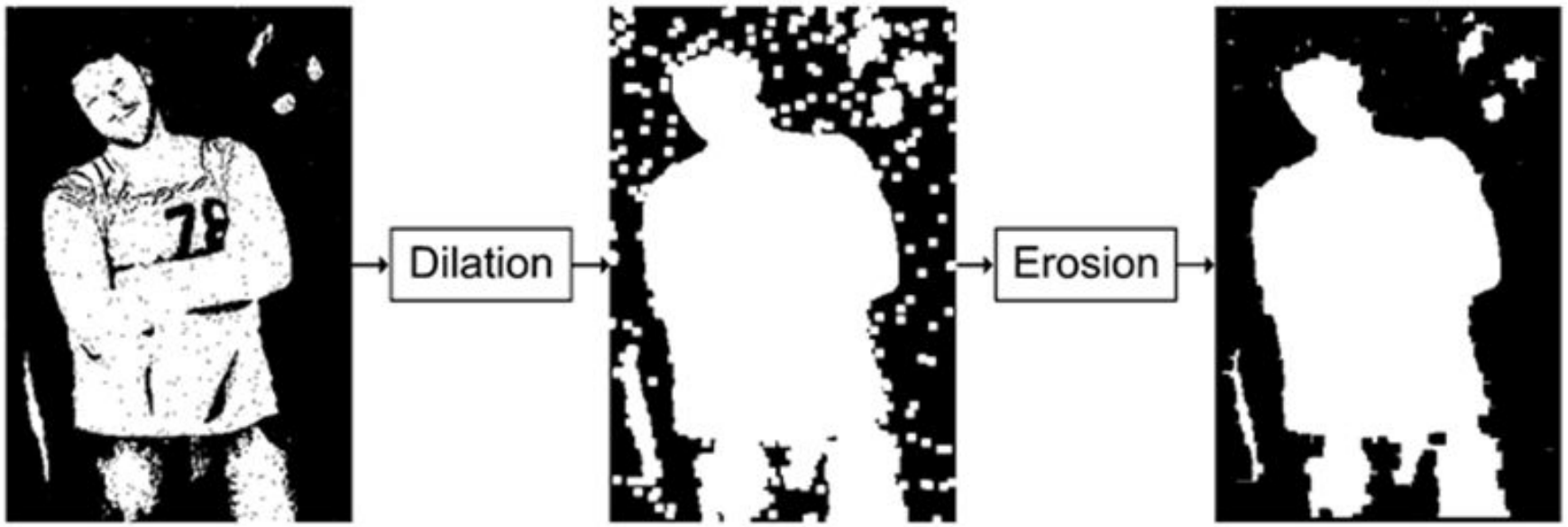


- ถ้าเป็นภาพสีเทา จะขยายภาพและเพิ่มความสว่างของวัตถุโดยใช้พื้นที่ใกล้เคียงสูงสุด
- ถ้าเป็นภาพไบนารี จะขยายภาพและเชื่อมต่อพื้นที่ที่แยกออกจากกันด้วยช่องว่างที่มีขนาดเล็กกว่าองค์ประกอบโครงสร้างและเพิ่ม Pixels เข้าไปที่ขอบด้านนอกของแต่ละวัตถุในภาพ

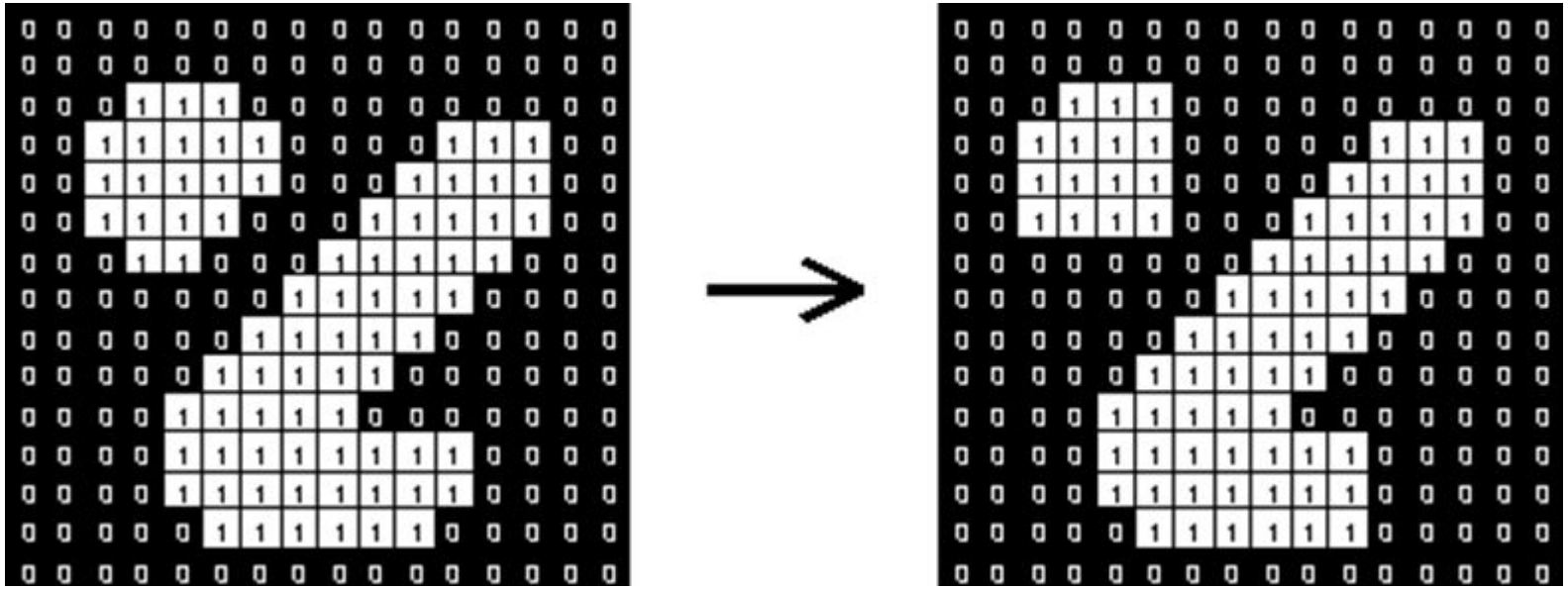
2.การกร่อนภาพ (Erosion) => สำหรับลดขนาดของวัตถุและความผิดปกติเล็กๆ โดยลบวัตถุที่มีรัศมีเล็กกว่าองค์ประกอบของโครงสร้าง



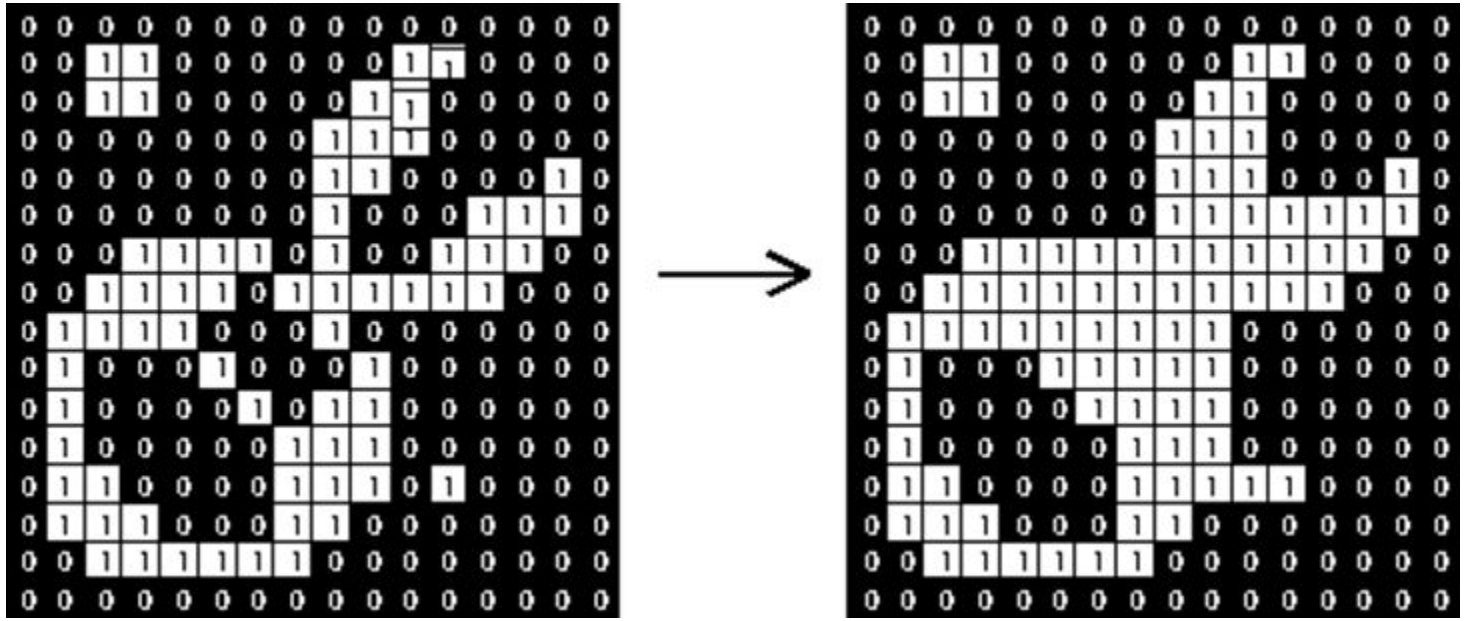
- ถ้าเป็นภาพสีเทา จะลดขนาดวัตถุและลดความสว่างของวัตถุบนพื้นหลังสีเข้ม โดยใช้พื้นที่ใกล้เคียงต่ำสุด
- ถ้าเป็นภาพไบนารี ลบวัตถุที่มีขนาดเล็กกว่าองค์ประกอบของโครงสร้างออกและลบ Pixels ที่ขอบด้านนอกออกจากวัตถุในภาพ



การทำงานร่วมกันระหว่าง Dilation และ Erosion



3.การเปิดภาพ (Opening) ใช้สำหรับกำจัดสัญญานรบกวนในรูปร่างและโครงสร้างของภาพกำจัดวัตถุขนาดเล็กออกไปจากพื้นที่มืดของภาพและนำไปวางไว้ในพื้นหลังของภาพ ทำให้ Pixels ภาพถูกเปิดกว้างมากขึ้น การเปิดภาพนิยมใช้ในการค้นหาองค์ประกอบของโครงสร้าง เช่น ขอบภาพและมุมภาพ



4.การปิดภาพ (Closing) => กระทำในทางตรงกันข้ามกับ Opening โดยเป็นการทำให้ Pixels ของภาพเชื่อมต่อกันมากขึ้น

สรุปการเปิดและปิดภาพเป็นการกำจัดสัญญาน รบกวนในรูปร่างและโครงสร้างของภาพ

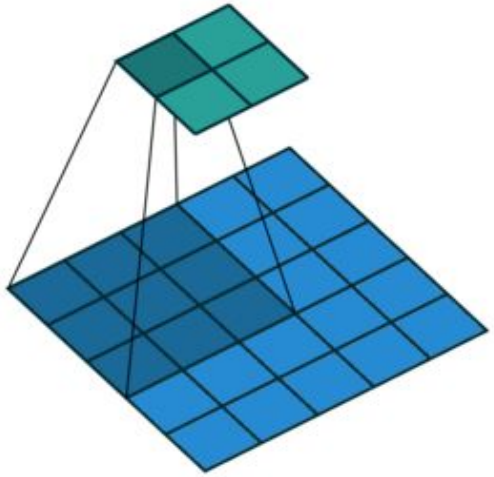
- **การเปิดภาพ (Opening)** = > กำจัดวัตถุขนาดเล็ก
- **การปิดภาพ (Closing)** => กำจัดช่องโหว่เล็กๆและเชื่อมต่อ

การใช้งาน Dilation (Dilate)

1 .สร้างตัวกรองข้อมูล (Kernel) ขึ้นมา 1 ชุด (Structuring Element)

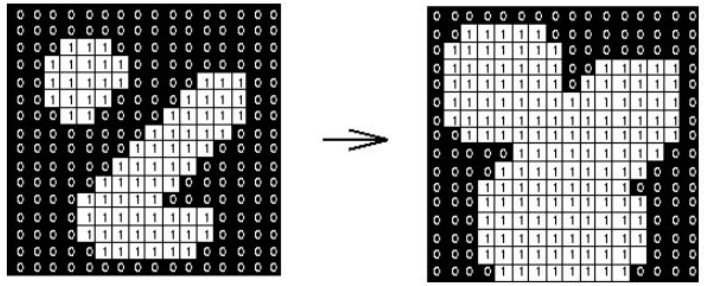
เป็นกลุ่มตัวเลข “ 1 ” (2x2 หรือ 3x3)

2.นำไปกรองข้อมูลในภาพ



- **Dilation** is an important morphological operation

$A \oplus B$



- Applied **Structuring Element**:

1	1	1
1	1	1
1	1	1

Set of coordinate points =

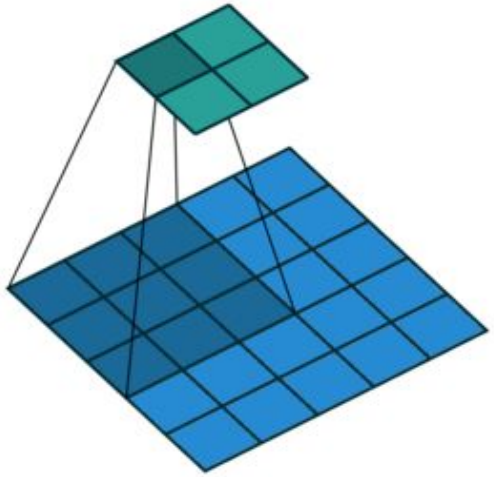
{ (-1, -1), (0, -1), (1, -1),
(-1, 0), (0, 0), (1, 0),
(-1, 1), (0, 1), (1, 1) }

การใช้งาน Erosion (erode)

1 .สร้างตัวกรองข้อมูล (Kernel) ขึ้นมา 1 ชุด (Structuring Element)

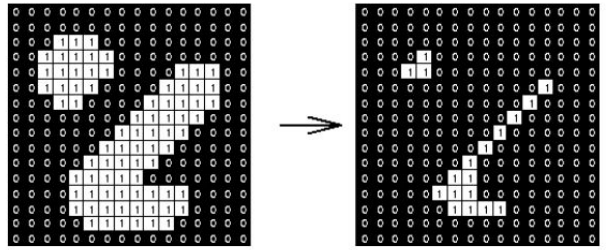
เป็นกลุ่มตัวเลข “ 1 ” (2x2 หรือ 3x3)

2.นำไปกรองข้อมูลในภาพ



- **Erosion** is an important morphological operation

$A \ominus B$



- Applied **Structuring Element**:

1	1	1
1	1	1
1	1	1

Set of coordinate points =

{ (-1, -1), (0, -1), (1, -1),
(-1, 0), (0, 0), (1, 0),
(-1, 1), (0, 1), (1, 1) }

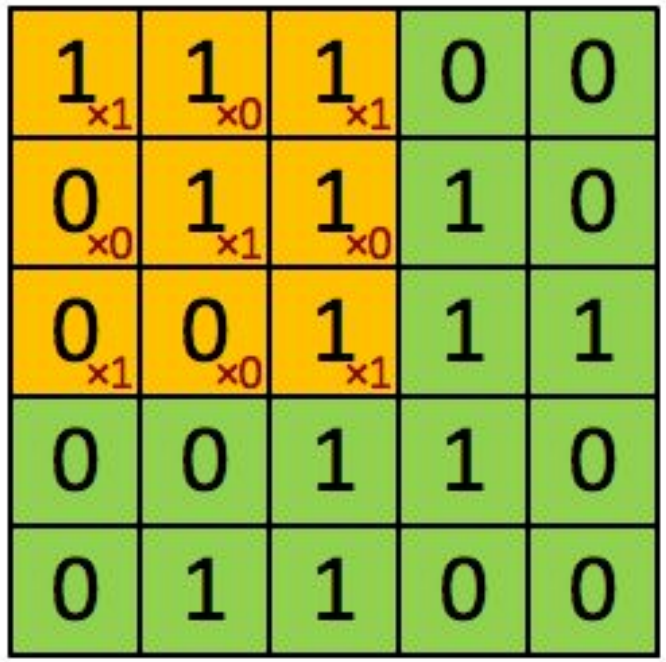
คอนโวลูชั่น - Convolution

หมายถึง วิธีการปรับปรุงคุณภาพของภาพโดยใช้วิธีกรองข้อมูลภาพเพื่อกำจัดสิ่งรบกวนออกจากภาพ โดยอาศัยหลักคณิตศาสตร์ที่เรียกว่า **Convolution**

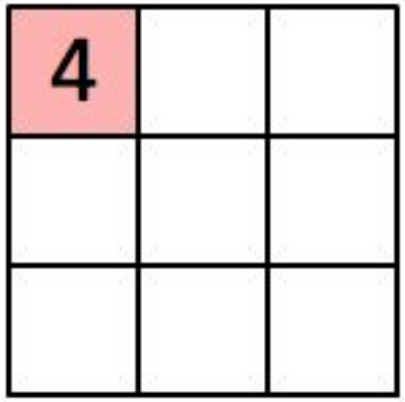
คอนโวลูชั่น - Convolution

การกรองข้อมูลภาพ (Image Filtering) คือ การนำภาพไปผ่านตัวกรองสัญญาณเพื่อให้ได้ภาพผลลัพธ์ที่ได้จะมีคุณสมบัติแตกต่างจากภาพเริ่มต้น วัตถุประสงค์หลักของการกรองข้อมูลภาพ คือการเน้นหรือลดทอนคุณสมบัติบางประการของภาพผ่าน **ตัวกรอง (Kernel) ในลักษณะ 2 มิติ** โดยเลื่อนหน้าต่างในตารางไปวิ่งผ่านทีละจุดบนภาพ เพื่อให้ได้ภาพที่มีคุณสมบัติตามต้องการ

สร้าง Kernel เพื่อนำไปกรองรูปภาพ

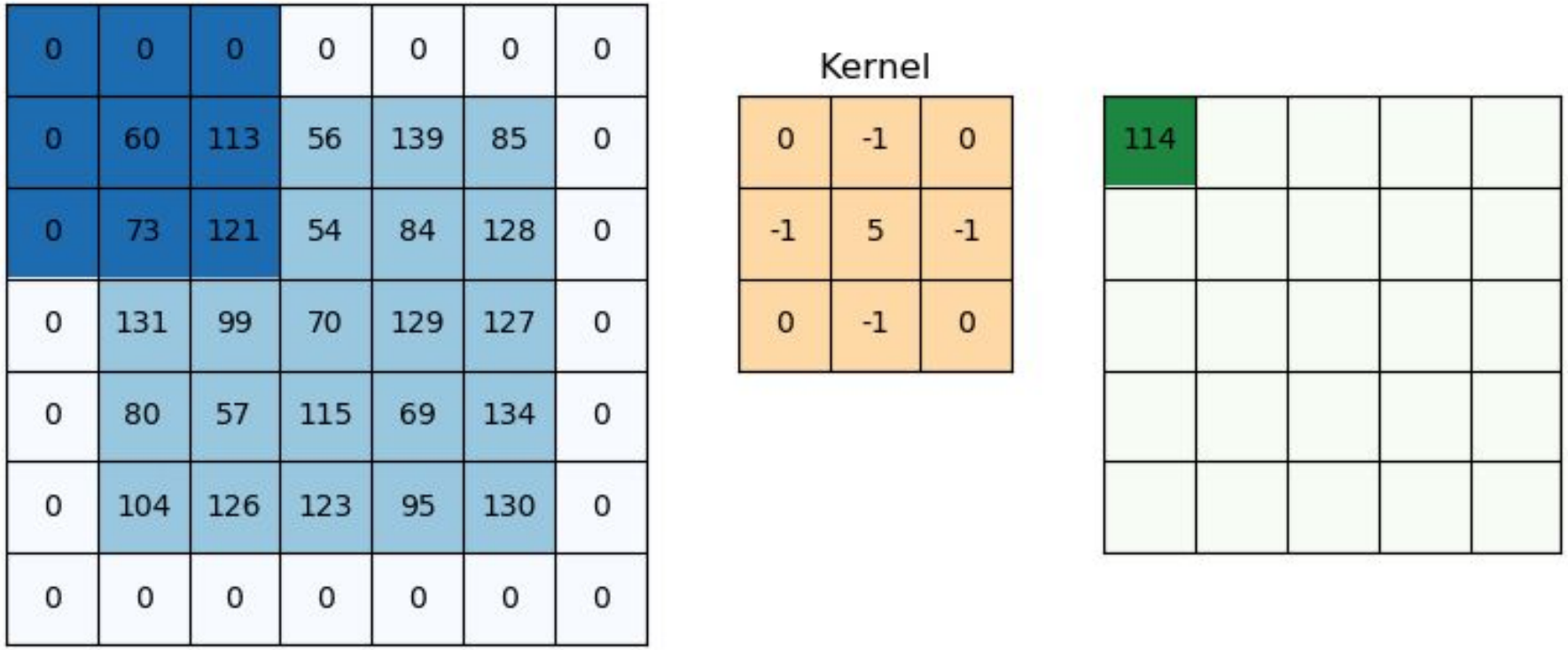


Image

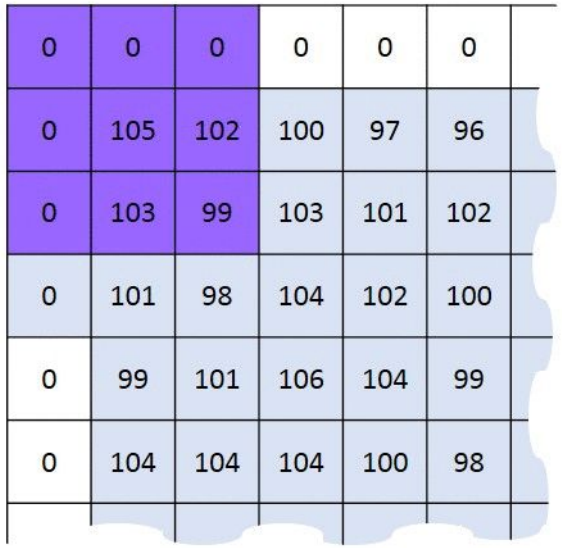


Convolved Feature

สร้าง Kernel เพื่อนำไปกรองรูปภาพ



สร้าง Kernel เพื่อนำไปกรองรูปภาพ



0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 \quad = 320$$

320					

Output Matrix

Convolution with horizontal and vertical strides = 1

ทำคอนโวลูชันภาพด้วย ฟังก์ชัน filter2D

cv2.filter2D(1,2,3)

1.รูปภาพ (Numpy Array)

2.ชนิดตัวแปรในอาร์เรย์ (กำหนดค่าเป็น -1 เพื่อให้อ้างอิงกับ Array ของภาพต้นฉบับที่ใส่เข้าไป)

3.Kernel ตัวกรองภาพ

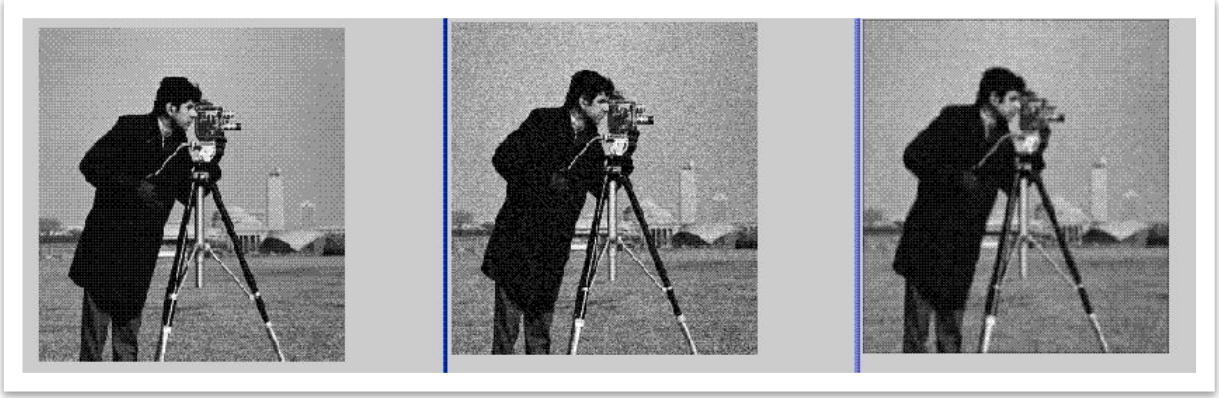
(กว้าง x สูง = ones 3x3 หรือ ones 5x5)

ตัวกรองค่าเฉลี่ยภาพเบลอ (Mean Filtering)

เป็นการลดระดับความเข้มต่างๆ ระหว่าง Pixel ใช้เพื่อลดสัญญาณรบกวน หรือ Noise ในภาพแต่ทำให้ภาพเบลอได้ ตัวกรองค่าเฉลี่ยจะใช้การแทนที่ค่า Pixel แต่ละ Pixel ในภาพ ด้วยค่าเฉลี่ยของพื้นที่ใกล้เคียงรวมทั้งตัวมันเองและมักจะใช้ตัวกรองคอนโวลูชั่นที่มีขนาดหน้าต่างขนาด 3x 3 และขนาด 5x5 เพื่อกรองให้ภาพเรียบมากยิ่งขึ้น **(ยิ่งใช้ขนาดตัวกรองใหญ่มากก็จะยิ่งเบลอมาก)**

คำสั่งที่ใช้ คือ

cv2.blur(ภาพ , (ขนาดตัวกรอง))



ตัวกรองค่าเฉลี่ยภาพเบลอ (Mean Filtering)

$$\frac{1}{25} \times$$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

$$\frac{1}{9} \times$$

1	1	1
1	1	1
1	1	1

ตัวกรองค่ามัธยฐาน (Median Filtering)

ตัวกรองค่ามัธยฐานจะใช้การกำจัดสัญญาณรบกวนแบบสุ่ม สามารถลดความเบลอของขอบภาพได้ดี จึงไม่ลดความคมชัดของภาพ ใช้ได้ดีเป็นพิเศษกับสัญญาณรบกวนที่มีลักษณะเป็นจุดสีขาวและสีดำกระจายอยู่บนภาพ และด้วยเหตุนี้มักจะใช้ในการใช้งานการมองเห็นของคอมพิวเตอร์

*** มัธยธาน (median) คือค่าที่อยู่ในลำดับกึ่งกลาง เมื่อนำข้อมูลทั้งหมดที่พิจารณามาเรียงต่อกัน**

คำสั่งที่ใช้ คือ

cv2.medianBlur(ภาพ , (ขนาดตัวกรอง))

Original Image

with Median Filter

ตัวกรองแบบเกาส์เซียน – (Gaussian Filter)

การใช้ฟังก์ชั่นแบบเกาส์เซียน จะเป็นผลให้ภาพมีความเบลอ ซึ่งใช้กันอย่างแพร่หลายใน Software Graphic โดยทั่วไปเป็นการลดสัญญานรบกวนและลดรายละเอียดของภาพ จะใช้ค่าตัวกรองที่มีค่าเท่ากันเป็นค่าเฉลี่ยทั้งหมด

คำสั่งที่ใช้ คือ

cv2.GaussianBlur(ภาพ , (ขนาดตัวกรอง),ค่า Sigma)

ค่า Sigma ยิ่งมีค่าเยอะ ยิ่งใกล้เคียงค่าใน blur()

การตรวจจับขอบภาพ (Edge Detection)

การหาขอบภาพวัตถุในภาพเป็นวิธีการแยกส่วนของข้อมูลภาพและเป็นขั้นตอนพื้นฐานของการแยกส่วนเทคนิคอื่นๆ เช่น การวิเคราะห์ภาพ การจดจำรูปแบบภาพ และเทคนิคการมองเห็นของคอมพิวเตอร์ โดยที่ขอบเขตพื้นที่และขอบภาพจะมีความสัมพันธ์อย่างใกล้ชิด เมื่อหาเส้นรอบรูปของภาพได้แล้ว เราสามารถปรับความคมชัดและความเข้มที่รอยต่อของขอบเขตพื้นที่ได้ เพื่อให้ได้ภาพที่มีความสมบูรณ์และมีคุณภาพมากยิ่งขึ้น

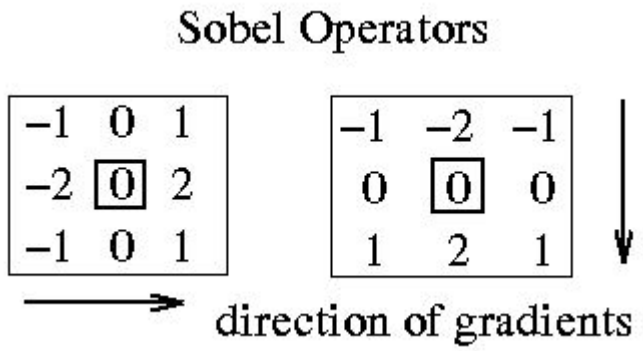
การตรวจจับขอบภาพ (Edge Detection)

การตรวจจับขอบภาพเพื่อหาเส้นรอบรูปของวัตถุในภาพ เป็นหนึ่งในขั้นตอนพื้นฐานในการประมวลผลภาพ การวิเคราะห์ภาพ การจดจำรูปแบบภาพและเทคนิคการมองเห็นของเครื่องจักรและการมองเห็นของคอมพิวเตอร์ โดยเฉพาะอย่างยิ่งในการตรวจสอบคุณสมบัติและการแยกคุณลักษณะ ซึ่งมีอัลกอริทึมหลายแบบ

- **การตรวจจับขอบภาพวิธีโซเบล (Sobel Method)**
- **การตรวจจับขอบภาพวิธีลาปลาเซียน (Laplacian Method)**
- **การตรวจจับขอบภาพวิธีแคนนี (Canny Method)**

ตรวจจับขอบภาพวิธีโซเบล (Sobel Method)

วิธีการโซเบล คือ วิธีการในการหาขอบภาพโดยใช้คอนโวลูชั่นในลักษณะทิศทางในแนวตั้งและทิศทางในแนวนอนและรวมค่าทั้ง 2 ทิศทางเข้าด้วยกัน



ตรวจจับขอบภาพวิธีโซเบล (Sobel Method)

ให้ Gx แทนค่าทิศทางในแนวนอน และ Gy แทนค่าทิศทางในแนวตั้ง

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

G_x

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

G_y

ตรวจจับขอบภาพวิธีโซเบล (Sobel Method)

คำสั่งที่ใช้

cv2.Sobel(1,2,3,4)

1 คือ Array รูปภาพ

2 คือ ชนิดตัวแปรในอาร์เรย์ (กำหนดค่าเป็น -1 เพื่อให้อ้างอิงกับ Array ของภาพต้นฉบับที่ใส่เข้าไป)

3 คือ ขนาดตัวกรองแกน X

4 คือ ขนาดตัวกรองแกน Y

ตรวจจับขอบภาพวิธีลาปลาเซียน (Laplacian Method)

วิธีการลาปลาเซียนคือ ใช้วิธีการกรองภาพ ปรับปรุงภาพและหาขอบภาพในเวลาเดียวกัน

คำสั่งที่ใช้

cv2.Laplacian(1,2)

1 คือ Array รูปภาพ

2 คือ ชนิดตัวแปรในอาร์เรย์ (กำหนดค่าเป็น -1 เพื่อให้อ้างอิงกับ Array ของภาพต้นฉบับที่ใส่เข้าไป)

ตรวจจับขอบภาพวิธีแคนนี (Canny Method)

เป็นวิธีการหาขอบภาพที่ได้รับความนิยม เนื่องจากใช้การเปรียบเทียบค่าจริงในภาพ โดยใช้ค่าเทรชโฮลด์ 2 ค่าในการคำนวณหาพื้นที่ขอบภาพที่มีความเข้มและความจางต่างกันไป (จะมีค่าเพี้ยนเมื่อภาพนั้นมีสัญญาณรบกวนเยอะเกินไป)

คำสั่งที่ใช้

cv2.Canny(1,2,3)

1 คือ Array รูปภาพ

2 คือ เทรชโฮลด์ค่าที่ 1

3 คือ เทรชโฮลด์ค่าที 2

ตรวจจับขอบภาพวิธีแคนนี (Canny Method)

เป็นวิธีการหาขอบภาพที่ได้รับความนิยม เนื่องจากใช้การเปรียบเทียบค่าจริงในภาพ โดยใช้ค่าเทรชโฮลด์ 2 ค่าในการคำนวณหาพื้นที่ขอบภาพที่มีความเข้มและความจางต่างกันไป (จะมีค่าเพี้ยนเมื่อภาพนั้นมีสัญญาณรบกวนเยอะเกินไป)

คำสั่งที่ใช้

cv2.Canny(1,2,3)

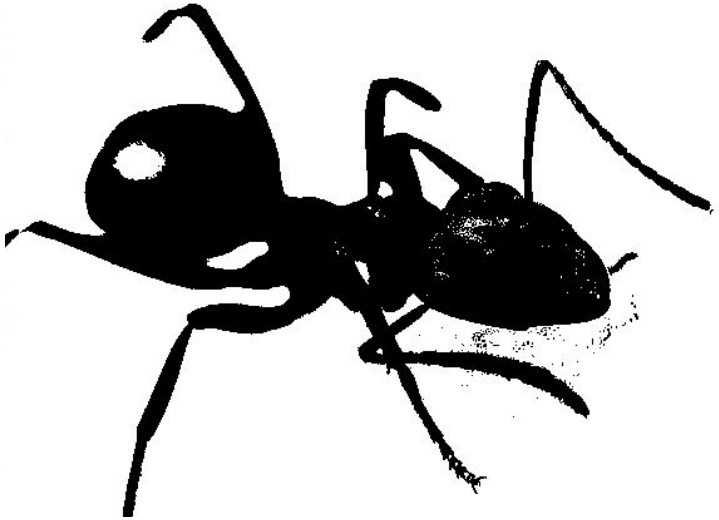
1 คือ Array รูปภาพ

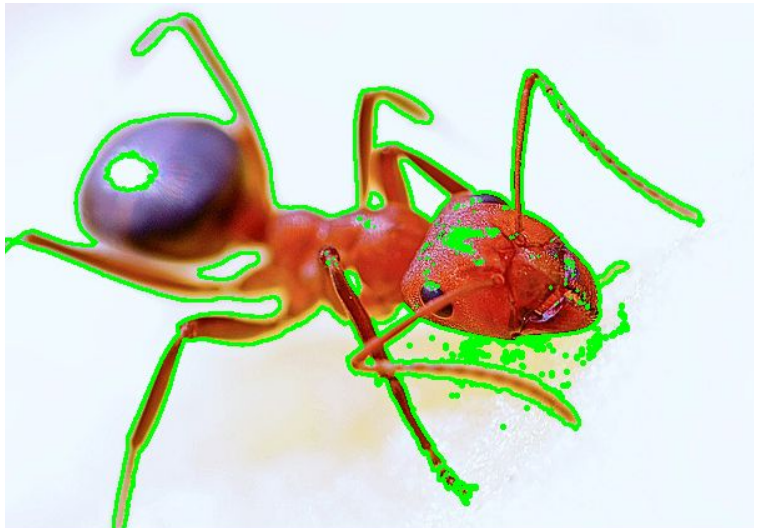
2 คือ เทรชโฮลด์ค่าที่ 1

3 คือ เทรชโฮลด์ค่าที 2

การหาเส้นเค้าโครงของภาพ (Contours)

เป็นวิธีการหาเส้นแบ่งพื้นที่ในภาพที่มีสีที่ต่างกันไป โดยปกติจะเป็นภาพ 2 สีหรือ 2 พื้นที่คือ ขาว - ดำ (ภาพแบบไบนารี่) โดยให้บอกพื้นที่ผ่านเส้นแบ่งดังกล่าวว่าพื้นที่ใดคือพื้นที่สีขาวและพื้นที่ใดคือพื้นที่สีดำ เป็นต้น **(พิจารณาเฉพาะพื้นที่สีดำ)**





การหาเส้นเค้าโครงของภาพ (Contours)

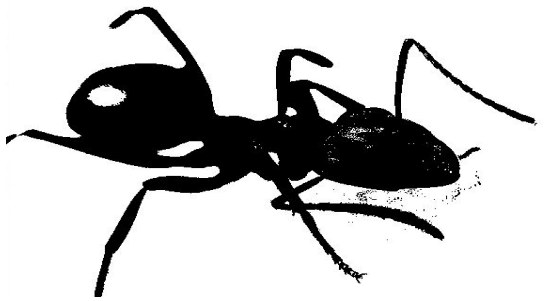
คำสั่งที่ใช้

cv2.findContours(1,2,3)

1 คือ Array รูปภาพ

2 คือ รูปแบบการพิจารณาลำดับชั้นของเส้นเค้าโครง

3 คือ รูปแบบการหาเส้นเค้าโครง



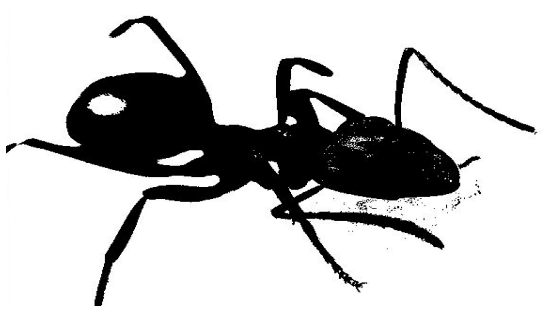
การหาเส้นเค้าโครงของภาพ (Contours)

1.รูปแบบการพิจารณาลำดับชั้นของเส้นเค้าโครง

- cv2.RETR_TREE คือ กำหนดการหาเส้นเค้าโครงจากชั้นด้านในสุดมาด้านนอกสุด

2.รูปแบบการหาเส้นเค้าโครง

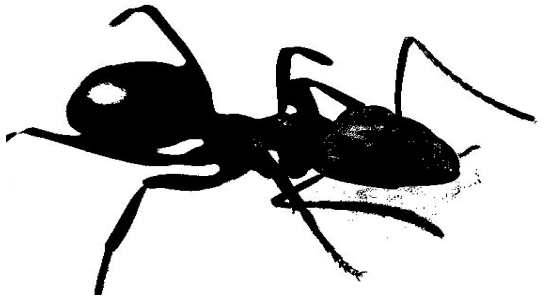
- cv2.CHAIN_APPROX_NONE คือ ค้นหาเส้นเค้าโครงแบบจุดต่อจุด



ผลลัพธ์จากการหาเส้นเค้าโครง

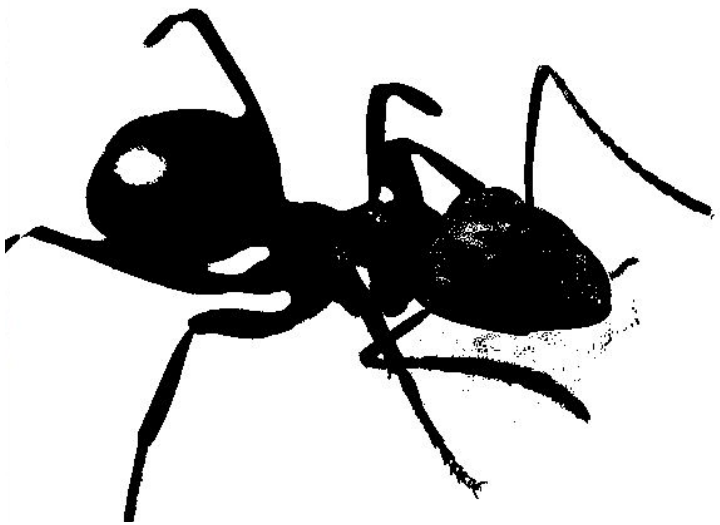
contours , hierarchy = cv2.findContours(1,2,3)

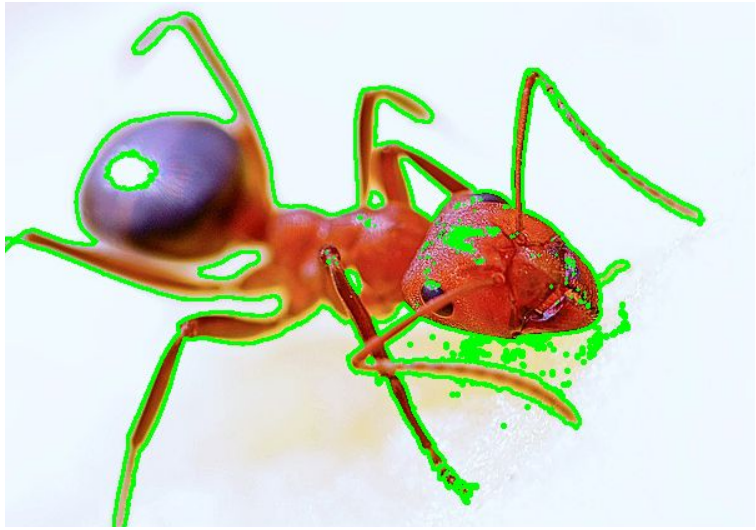
- contours คือ ข้อมูลเส้นเค้าโครง
- hierarchy คือ ลำดับชั้น



การคำนวณหาจำนวนเส้นเค้าโครง

- len(contours)





การวาดเส้นเค้าโครงบนภาพ (Draw Contours)

คำสั่งที่ใช้

cv2.drawContours(1,2,3,4,5)

1 คือ Array รูปภาพ

2 คือ เส้นเค้าโครง

3 คือ ลำดับชั้นของเส้นเค้าโครง **(-1 คือเอาทั้งหมด)**

4 คือ สีเส้นเค้าโครง

5 คือ ความหนาของเส้น