Hough Transform for detecting lines แบบที่ 1

สมการ $y = mx + b$ เมื่อ m = ความชัน (slope)
b = จุดตัดของเส้นบนแกน y
x = coordinate แกน x (จุด x ของ pixel ที่อยู่บนภาพ)
y = coordinate แกน y (จุด y ของ pixel ที่อยู่บนภาพ)

Algorithm

1.) หาขอบของภาพ และ thinning ให้ ภาพมีเส้นบางที่สุด และปรับให้ภาพเป็น binary image

2.) สร้าง Accumulator vote สำหรับค่าของ m และ b

โดย m เป็นขนาดความชันที่เป็นไปได้ อาจจะกำหนดตั้งแต่ -5 ถึง 5

b เป็นค่าจุดตัดแกน y ที่เป็นไปได้ ซึ่งหาได้จาก
ความสูงของภาพ (height) * 2 เช่น ถ้า height = 15
b = (height * 2) + 1 = (15 * 2)+1 = 31 ซึ่งจะได้ค่า b ตั้งแต่ -15,..,0,..,15

3.) ปรับสมการให้เป็น **$b = y - mx$**

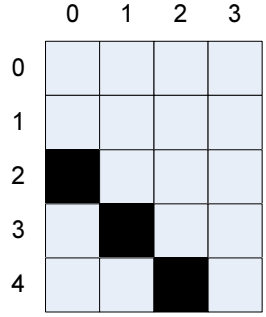
ทำการแทนค่าพิกัด(Cooridate) x และ y เฉพาะ pixel ที่เป็นขอบ(edge) เท่านั้นในสมการ $b = y - mx$ โดยทำการแทนค่า x, y และ m ในสมการไปเรื่อย ๆ ซึ่งได้ค่า b เป็นเท่าไร ให้นำไป vote ใน Accumulator ที่สร้างไว้ แล้วทำการแทน x, y ในทุกความชันไปเรื่อย ๆ

4.) ลองกำหนดค่า x ให้สมการ แล้วหาค่า y และทำการ plot จุดให้เป็นเส้นตรง ลงไปในภาพเดิม ก็จะได้ตำแหน่งของเส้นตรงนั้น

ข้อเสีย

- ถ้าเส้นตรงที่มีความชันเป็น ∞ (infinity) หมายความว่าเส้นตรงนั้นเป็นเส้นตรงที่ตั้งฉาก หรือ vertical line จะไม่สามารถหาสมการของเส้น vertical line นั้นได้
- และถ้าต้องการ vote accumulator ให้มีความชันละเอียดมากขึ้นนั้น จำเป็นจะต้องกำหนดค่า m ให้ละเอียดขึ้น ซึ่งจะทำให้ Accumulator ใหญ่ อาจทำให้เปลือง memory และใช้เวลานาน

Example of Hough Line 1



จาก Coordinate เป็น edge (หรือ pixel สีดำเท่านั้น) จากภาพจะได้ Coordinate ดังนี้

(0, 2) (1, 3) (2, 4)

ปรับสมการ $y = mx + b$ เป็น $b = y - mx$

<u>จุด (0, 2) แทนสมการดังนี้</u>

$b = 2 - (-2*0) = 2$ ดังนั้น $m = -2$ และ $b = 2$

$b = 2 - (-1*0) = 2$ " $m = -1$ และ $b = 2$

$b = 2 - (0*0) = 2$ " $m = 0$ และ $b = 2$

$b = 2 - (1*0) = 2$ " $m = 1$ และ $b = 2$

$b = 2 - (2*0) = 2$ " $m = 2$ และ $b = 2$

<u>จุด (1, 3) แทนสมการดังนี้</u>

$b = 3 - (-2*1) = 5$ ดังนั้น $m = -2$ และ $b = 5$

$b = 3 - (-1*1) = 4$ " $m = -1$ และ $b = 4$

$b = 3 - (0*1) = 3$ " $m = 0$ และ $b = 3$

$b = 3 - (1*1) = 2$ " $m = 1$ และ $b = 2$

$b = 3 - (2*1) = 1$ " $m = 2$ และ $b = 1$

<u>จุด (2, 4) แทนสมการดังนี้</u>

$b = 4 - (-2*2) = 8$ ดังนั้น $m = -2$ และ $b = 8$

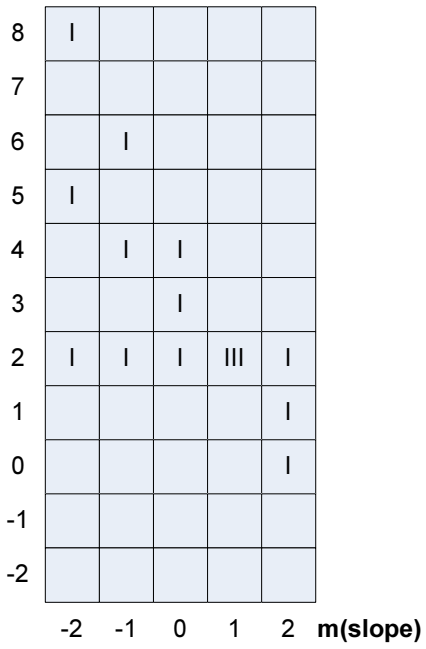
$b = 4 - (-1*2) = 6$ " $m = -1$ และ $b = 6$

$b = 4 - (0*2) = 4$ " $m = 0$ และ $b = 4$

$b = 4 - (1*2) = 2$ " $m = 1$ และ $b = 2$

$b = 4 - (2*2) = 0$ " $m = 2$ และ $b = 0$

<u>Accumulator Vote</u>



เพราะฉะนั้น Maximum vote คือ **b=2** และ **m=1**

ดังนั้นจะได้สมการเส้นตรงจาก $y = mx + b$ เป็น

$y = 1x + 2$

เมื่อได้สมการ $y = 1x + 2$ มาแล้วทำการแทนค่า x โดยแทนกลับไปยังสมการเพื่อหาค่า y เมื่อได้ค่า x และ y แล้ว ให้ทำการ plot เส้นตรงทับลงบนภาพเดิม ก็จะได้ตำแหน่งที่อยู่ของเส้นนั้น ๆ

x	0	1	2	3
y	2	3	4	5

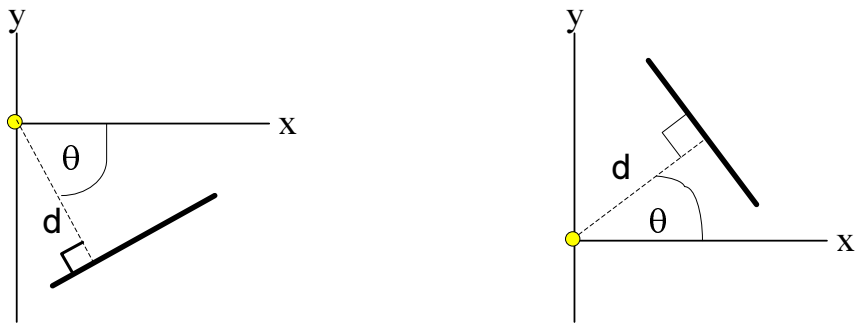
Hough Transform for detecting lines แบบที่ 2

สมการ d = xcosθ + ysinθ เมื่อ d = ระยะทางจากจุด origin เมื่อลากไปตั้งฉากกับเส้นตรง

θ = มุมหรือองศาของเส้นที่ลากจากจุด origin ไปตั้งฉากกับเส้นตรงเมื่อวัดจากแกน x

x = coordinate แกน x (จุด x ของ pixel ที่อยู่บนภาพ)

y = coordinate แกน y (จุด y ของ pixel ที่อยู่บนภาพ)



Algorithm

1.) หาขอบของภาพ และ thinning ให้ ภาพมีเส้นบางที่สุด และปรับให้ภาพเป็น binary image

2.) สร้าง Accumulator vote สำหรับค่า d และ θ ที่เป็นไปได้ โดย

$d = \sqrt{width^2 + height^2}$ จะได้ขนาดของ d ที่เป็นไปได้

θ = ขนาดขององศาที่เป็นไปได้ คือ 0 ถึง 360 องศา

3.) ทำการแทนค่าพิกัด x, y เฉพาะ pixel ที่เป็นขอบ(edge) ในสมการ d = xcosθ + ysinθ และทำการปรับค่า θ ไปเรื่อย ๆ ตั้งแต่ 0 ถึง 360 องศา เพื่อหาค่า d และทำการ vote ใน Accumulator ที่สร้างไว้จนครบทุก ๆ ค่า θ

4.) เมื่อ vote[d][theta] สำหรับทุก x, y (เฉพาะ edge) ครบแล้ว ให้ทำการหา Maximum vote แล้วนำไปแทนในสมการ d = xcosθ + ysinθ

5.) ลองกำหนดค่า x ให้สมการแล้วหาค่า y และทำการ plot จุดให้เป็นเส้นตรง ก็จะได้บริเวณของภาพที่มีเส้นตรงนั้นอยู่

ข้อดี ค่า θ (theta) คงที่ คือ 0 ถึง 360 ซึ่งจะปรับเฉพาะ ค่า d เท่านั้น ซึ่งจะใหญ่ตามขนาดของภาพ

หมายเหตุ สามารถหาเส้นตรงได้หลายเส้นจากภาพหนึ่งภาพ ซึ่งก็คือทำการหา Maximum vote ตัวที่มากที่สุด และตัวที่รองลงมาเรื่อย ๆ แล้วนำค่า d และ θ มาเป็นคำตอบ

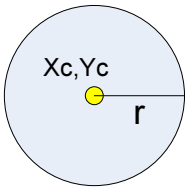
Hough Transform for detecting circles

สมการ $Xc = x + r\cos\theta$ จุดประสงค์ในการตำแหน่งของวงกลมในภาพ คือการหาจุด center หรือจุด

$Yc = y + r\sin\theta$ กึ่งกลางของวงกลม และรัศมีของวงกลม เมื่อ

Xc, Yc = จุดกึ่งกลางของวงกลม

r = รัศมีของวงกลม



Algorithm

1.) หาขอบของภาพ และ thinning ให้ ภาพมีเส้นบางที่สุด และปรับให้ภาพเป็น binary image

2.) สร้าง Accumulator vote สามมิติสำหรับค่า Xc, Yc และ r ซึ่งสามารถกำหนดค่าได้ ดังนี้

- Xc = 0 ถึง width(ความกว้างของภาพ) และ Yc = 0 ถึง height(ความสูงของภาพ)
- ค่า r ให้ทำการกำหนดตามขนาดเส้นทแยงมุมของภาพหารสอง หรือทำการกำหนดให้อยู่ในช่วง rmin และ rmax เพื่อทำการคำนวณเร็วขึ้น เช่น อาจจะกำหนดให้เป็นจาก 10 ถึง 50 ดังนั้นจะสามารถกำหนดให้ r มีขนาดคือ (50-10)+1 = 41 ตัว
- ค่า θ ให้ทำการแทนค่าตั้งแต่ 0 ถึง 360 องศา ซึ่งอาจเขียนตัวอย่างของโปรแกรมได้ ดังนี้

```
……….
for(int x = 0; x<width; x++)
    for(int y = 0; y<height; y++)
        for(int r = rmin; r<=rmax; r++)
            for(int theta=0; theta<=360; theta++){
                xc = x + (r * Math.cos(theta));
                yc = y + (r * Math.sin(theta));
                vote[xc][yc][r] += 1;
            }
……..
```

3.) เลือกค่า Maximum vote จาก vote[xc][yc][r] ก็จะได้จุดกึ่งกลางคือ พิกัด Xc, Yc และ รัศมีเท่ากับ r

หมายเหตุ สามารถหาวงกลมได้หลายวงจากภาพหนึ่งภาพ ซึ่งก็คือทำการหา Maximum vote ตัวที่มากที่สุด และตัวที่รองลงมาเรื่อย ๆ แล้วนำค่า Xc, Yc และ r มาเป็นคำตอบ

Homework 3 (Hough Transform)

1. ให้นักศึกษาทำการเขียนโปรแกรมเพื่อทำการ Detect Line โดยใช้ Algorithm การ Detect Line แบบที่ 2 โดยเมื่อทำการ detect แล้วให้ทำการแสดงเส้นสีแดงขีดทับบนภาพเดิมเพื่อแสดงตำแหน่งของ Line ที่ Detect เจอ พร้อมทั้งแสดงสมการของเส้นนั้น ๆ (ภาพที่ทดสอบ ควรมี เส้นตรงมากกว่า 1 เส้นขึ้นไป)

2. ให้นักศึกษาทำการเขียนโปรแกรมเพื่อทำการ Detect Circle โดยเมื่อทำการ detect แล้วให้ทำการแสดงเส้นวงกลมสีแดง โดยเขียนวงกลมทับบนภาพเดิมเพื่อแสดงตำแหน่งของ วงกลมที่ Detect เจอ พร้อมทั้งแสดงสมการของวงกลมนั้น ๆ (ภาพที่ทดสอบควรมีวงกลมมากกว่า 1 วงขึ้นไป)