บทที่ 2
ทฤษฎีที่นำมาใช้ในงานวิจัย

2.1 คำศัพท์เฉพาะและความหมาย

เนื้อหาที่จะกล่าวถึงในบทนี้และบทต่อ ๆ ไปจะมีการอ้างถึงศัพท์เฉพาะหลายแห่ง เพื่อให้เข้าใจความหมายตรงกันจึงได้รวบรวมและนำมาอธิบายไว้ตอนต้นดังนี้

ค่าความเข้มแสงของจุดภาพ (ค่าระดับเทา) เป็นตัวเลขซึ่งบอกว่าจุดภาพจะมีความสว่างมากน้อยเพียงใด สำหรับงานวิจัยนี้กำหนดให้จุดภาพมีค่าความเข้มแสงอยู่ระหว่าง 0-255 โดยจุดภาพจะมีค่าความเข้มแสงน้อยที่สุดคือ 0 (สีดำ) และมากที่สุด 255 (สีขาว)

ภาพขาวดำ (binary image) คือภาพซึ่งจุดภาพมีค่าความเข้มแสง 2 ค่าคือ 0 หรือ 255

ภาพระดับเทา (gray scale image) คือภาพซึ่งจุดภาพมีค่าความเข้มแสงตั้งแต่ 0-255

ภาพต้นฉบับ ภาพก่อนการประมวลผลภาพดิจิตอล

ภาพผลลัพธ์ ภาพหลังการประมวลผลภาพดิจิตอล

ทิศทางในการประมวลผลภาพดิจิตอลกับจุดภาพ การประมวลผลภาพดิจิตอลซึ่งใช้ในงานวิจัยนี้จะกระทำกับหน่วยความจำที่ตำแหน่งต่ำกว่าไปยังตำแหน่งที่สูงกว่า อย่างไรก็ตามในการแสดงผลภาพบิตแมปของระบบปฏิบัติการวินโดวส์จะแสดงผลจากตำแหน่งมุมล่างซ้ายของภาพไปยังมุมบนขวาของภาพ

ค่าขีดแบ่ง เป็นค่าความเข้มแสงที่ใช้แยกประเภทของจุดภาพของการกระมวลผลภาพโดยวิธีกำหนดค่าขีดแบ่ง (ดูรายละเอียดในหัวข้อ 2.2.1)

ภาพขีดแบ่ง ภาพที่เกิดการประมวลผลภาพดิจิตอลด้วยวิธีกำหนดค่าขีดแบ่ง (ดูรายละเอียดใน 2.2.1)

ภาพการยุบตัว ภาพที่ได้จากการประมวลผลการยุบตัว (ดูรายละเอียดใน 2.2.3)

ภาพเส้นโครงร่าง ภาพที่ได้จากการประมวลผลด้วยวิธีการทำให้บาง (ดูรายละเอียดใน 2.2.4)

แมสก์ (mask) เป็นกรอบจัตุรัสขนาดเล็ก เช่น 3x3, 5x5 กำหนดไว้เพื่อใช้สำหรับเป็นตัวกระทำการของการประมวลผลภาพดิจิตอล เช่น การทำ convolution เป็นต้น

2.2 ทฤษฎีที่เกี่ยวข้อง

งานวิจัยนี้ได้นำวิธีการประมวลผลภาพดิจิตอลและคอมพิวเตอร์กราฟิกมาประยุกต์ใช้กับการแยกบริเวณสมองอันได้แก่ การกำหนดค่าขีดแบ่ง การระบายบริเวณ การยุบตัว การทำให้บาง และเกาเซียนฟิลเตอร์ รายละเอียดของวิธีการที่นำมาใช้มีดังนี้

2.2.1 การกำหนดค่าขีดแบ่ง (Thresholding)

การกำหนดค่าขีดแบ่ง [2] เป็นการแยกบริเวณโดยการพิจารณาค่าความเข้มแสงของจุดภาพเพื่อจำแนกจุดภาพออกเป็น 2 จำพวกได้แก่ จุดภาพที่เป็นวัตถุ และจุดภาพที่เป็นพื้นหลัง (background) การแยกจุดภาพที่เป็นวัตถุจะกำหนดค่าขีดแบ่ง 2 ค่าซึ่งมีช่วงครอบคลุมค่าความเข้มแสงของจุดภาพที่เป็นวัตถุโดยที่การกำหนดค่าขีดแบ่งมักพิจารณาร่วมกับฮีสโตแกรมเพื่อดูลักษณะร่วมและการเกาะกลุ่มกันของค่าความเข้มแสงของจุดภาพ

กำหนดให้

$f(x,y)$ เป็นค่าความเข้มแสงของจุดภาพ ณ ตำแหน่ง (x,y) ของภาพต้นฉบับ

$g(x,y)$ เป็นค่าความเข้มแสงของจุดภาพ ณ ตำแหน่ง (x,y) ของภาพผลลัพธ์

T_1, T_2 เป็นค่าความเข้มแสงใด ๆ โดยที่ $T_1 < T_2$

จะได้ว่า

$$g(x,y) = \begin{cases} 255 & \text{ถ้า } T_1 \le f(x,y) \le T_2 \\ 0 & \text{กรณีอื่นนอกเหนือจากข้างบน} \end{cases} \quad \ldots(2.1)$$

ค่าของ T_1 และ T_2 จะกำหนดช่วงค่าขีดแบ่งสำหรับจำแนกจุดภาพที่เป็นวัตถุออกจากจุดภาพที่เป็นพื้นหลัง หลังจากการประมวลผลจะได้ภาพซึ่งมีจุดภาพ 2 จำพวกได้แก่จุดภาพที่เป็นส่วนของวัตถุจะปรากฏเป็นสีขาวและจุดภาพที่เป็นพื้นหลังจะปรากฏเป็นสีดำ ในบางกรณีหากต้องการเห็นเค้าโครงเดิมของภาพเป็นภาพระดับเทาแทนที่จะเป็นภาพขาวดำอาจใช้สมการที่ (2.2) ข้างล่างซึ่งจะให้ภาพผลลัพธ์ที่มีวัตถุปรากฏเป็นภาพระดับเทาบนพื้นหลังสีดำ

$$g(x,y) = \begin{cases} f(x,y) & \text{ถ้า } T_1 < f(x,y) \le T_2 \\ 0 & \text{กรณีอื่นนอกเหนือจากข้างบน} \end{cases} \quad \ldots(2.2)$$

2.2.2 การระบายบริเวณ (Boundary fill)

การระบายบริเวณ [1] คือ การเปลี่ยนค่าความเข้มแสงของจุดภาพที่อยู่ในขอบเขตเดียวกันให้มีค่าความเข้มแสงเดียวกัน การระบายบริเวณจะต้องกำหนดตำแหน่งของจุดภาพในบริเวณที่ต้องการระบายซึ่งเรียกว่า *จุดเริ่มต้นของการแผ่ขยาย (seed point)* ค่าความเข้มแสงของจุดภาพที่เป็นขอบ (boundary) และค่าความเข้มแสงสำหรับจุดภาพที่ถูกเปลี่ยน (fill value) การประมวลผลจะเริ่มจากการเปลี่ยนค่าความเข้มแสงของจุดภาพ ณ ตำแหน่งจุดเริ่มต้นของการแผ่ขยายไปยังจุดภาพอื่นโดยรอบทุกทิศทาง และสิ้นสุดที่จุดภาพที่มีค่าความเข้มแสงที่กำหนดให้เป็นขอบ ขั้นตอนวิธีการระบายบริเวณ [1] อาจแสดงได้ดังนี้

กำหนดให้

x,y	เป็นตำแหน่งของจุดเริ่มต้นการแผ่ขยาย
BoundaryValue	เป็นค่าความเข้มแสงซึ่งกำหนดขอบเขตของบริเวณ
NewValue	เป็นค่าความเข้มแสงของจุดภาพซึ่งอยู่ในบริเวณเดียวกัน
BoundaryFill4	เป็นขั้นตอนวิธีการแผ่ขยายของบริเวณโดยพิจารณาจุดรอบข้างที่เป็น 4-connected [2]

```
procedure BoundaryFill4(
   x,y : integer;              { seeding point coordinate }
   BoundaryValue,              { value that defined boundary }
   NewValue:color)             { replacement value }

variable
   c : interger;

begin
   c = readpixel( x,y );
   if c <> BoundaryValue and c <> NewValue then
      begin
      WritePixel( x,,y, NewValue );
      BoundaryFill4( x, y-1, NewBoundaryValue, NewValue );
      BoundaryFill4( x, y+1, NewBoundaryValue, NewValue );
      BoundaryFill4( x-1, y, NewBoundaryValue, NewValue );
      BoundaryFill4( x+1, y, NewBoundaryValue, NewValue );
      end
end
```

รูปที่ 4 ขั้นตอนวิธีการระบายบริเวณ

ขั้นตอนวิธีที่แสดงข้างบนทำงานในลักษณะโปรแกรมเรียกตัวเองซ้ำ (recursive) สำหรับภาพซึ่งมีขนาดใหญ่ หรือภาพที่มีบริเวณกว้างมากอาจพบปัญหาเรื่องสแตกไม่พอ (stack overflow) ซึ่งเกิด

จากโปรแกรมเรียกตัวเองซ้ำหลายครั้งเกินไป ในทางปฏิบัติจะมีการปรับปรุงขั้นตอนวิธีเพื่อแก้ปัญหาหน่วยความจำโดยลดจำนวนครั้งในการใช้เนื้อที่หน่วยความจำและจะไม่ใช้วิธีเรียกตัวเองซ้ำ วิธีใหม่ทำงานในลักษณะสแปน **สแปน** (span) **หมายถึงเส้นแนวนอนซึ่งปลายทั้งสองข้างถูกกำหนดขอบเขตโดยจุดภาพซึ่งมีค่าความเข้มแสงเป็น** BoundaryValue **หรือ** NewValue **ในขณะทำการประมวลผล** ทุกจุดภาพในทุกสแปนจะถูกเปลี่ยนค่าเป็น NewValue ความซับซ้อนของวิธีนี้คือการหาสแปนที่ถูกต้องให้ครบสำหรับบริเวณที่กำหนด การหาสแปนเริ่มจากจุดตั้งต้นที่กำหนด ถือเสมือนว่าจุดตั้งต้นที่กำหนดเป็นจุดในสแปนเส้นหนึ่ง ให้มองหาขอบเขตที่ปลายสองด้านของสแปนเส้นนั้น (ค่าความเข้มแสงที่ปลายสองด้านอาจเป็นค่า BoundaryValue หรือ NewValue ค่าใดค่าหนึ่งแล้วแต่ว่าจะเจอค่าใดก่อน) เมื่อได้ขอบเขตสแปนจึงทำการเปลี่ยนค่าความเข้มแสงจุดภาพในสแปนให้เป็นค่า NewValue เนื่องจากว่าสแปนเส้นอื่น ๆ ในบริเวณเดียวกันอาจเรียงตัวติดกันทั้งบนและล่างของสแปนเส้นปัจจุบัน จึงพิจารณาสแปนเส้นอื่น ๆ ที่เป็นไปได้โดยดูจุดภาพที่อยู่บนและล่างตามแนวสแปนเส้นปัจจุบัน และหาขอบเขตของสแปนเส้นบน และล่าง เมื่อได้สแปนใหม่แล้วจึงทำการเปลี่ยนค่าความเข้มแสงของจุดภาพในสแปนให้เป็น NewValue ทำเช่นนี้เรื่อย ๆ สำหรับสแปนเส้นใหม่ที่พบ จนกระทั่งครบทุกสแปนในบริเวณเดียวกัน

มีข้อสังเกตว่าวิธีใหม่จะใช้การสแกนจุดภาพเพื่อหาสแปนและเก็บตำแหน่งสแปนเข้าสแตก (stack) เพื่อการประมวลผลครั้งต่อไป การเก็บเฉพาะตำแหน่งสแปนจะใช้สแตกจะน้อยกว่าขั้นตอนแบบเรียกตัวเองซ้ำซึ่งเก็บทุกตำแหน่งจุดภาพในทิศทางที่ตรวจสอบเข้าสแตกสำหรับการประมวลผลครั้งต่อไป วิธีการใหม่ที่กล่าวมาสามารถแสดงขั้นตอนวิธีได้ดังนี้

```
procedure BoundaryFilling(
    Seedx,Seedy,                { seeding point coordinate }
    BoundaryValue,              { value that defined boundary }
    NewValue : integer )        { replacement value }

variable
    sx,sy,fx1,dx2,fy    : integer;
    scan                : ingeger;
    foundspan           : boolean;
    fraglist            : list[6];
    seedstack           : stack[7];

begin
sx = Seedx;
sy = Seedy;

while true do
    begin
```

---

[6] list หมายถึงโครงสร้างข้อมูลที่สามารถเก็บและสืบค้นข้อมูลได้

[7] stack หมายถึงโครงสร้างข้อมูลซึ่งทำงานในลักษณะ last in first out

```
{ at the given seed point (sx,sy), find the span of current scan line }
{ which include (sx,sy) and every pixel in this span is between }
{ boundary value edge }
y := sy;
for x := sx downto 0 do                    { scan to left }
    begin
    if ReadPixel( x, y ) = BoundaryValue then
        break;
    else
        WritePixel( x, y, NewValue );
    end
fx1 := x+1;                                { keep left-x position }
for x := sx+1 to 255 do                    { scan to right }
    begin
    if ReadPixel( x, y ) = BoundaryValue then
        break;
    else
        WritePixel( x, y, NewValue );
    end
fx2 := x-1;                                { keep right-x position }
fy := y;                                   { keep y position }

{ add span (fx1,fx2,fy) to list which indicated that the span is scanned }
fraglist.add( fx1, fx2, fy );

{ scan downword and upward of the current scanned span }
for scan := 0 to 2 do
    begin
    if  scan = 0 then y = sy-1; else y = sy+1;
    if  y < 0 or y > 255 continue;         { do not scan beyond image edge }
    { scan the adjacent span until found edge }
    for x := fx1 to fx2 do
        begin
        if ReadPixel( x, y ) <> BoundaryValue then
            foundspan := true;
        else
            begin
            { if the adjacent span has reached boundary and any pixel in }
            { that span has never been filled then add seed point to stack }
            if foundspan = true then
                if fraglist.findseed( x-1, y ) = false then
                    seedstack.put( x-1, y );
                foundspan = false;
            end
        end
    end
{ check stack and find whether any seed are to be processed }
if seedstack.notempty() = true then
    begin
    seedstack.get( x, y );
    continue;
    end
else
    break;
end { while }
```

end { BoundaryFilling }

รูปที่ 5 ขั้นตอนวิธีการระบายบริเวณซึ่งได้แก้ปัญหาเรื่องสแตก

2.2.3 การยุบตัว (Erosion)

การยุบตัว (erosion) [2] เป็นการประมวลผลภาพดิจิตอลที่กระทำกับภาพระดับเทาหรือภาพขาวดำเพื่อให้บริเวณซึ่งมีค่าความเข้มแสงมากกว่าลดลงหรืออาจกล่าวอีกนัยหนึ่งว่าเป็นการทำให้บริเวณที่มีค่าความเข้มแสงน้อยกว่าขยายตัวไปในทิศทางของบริเวณที่มีค่าความเข้มแสงมากกว่า การยุบตัวมีนิยามดังนี้

- ให้ Z^2 เป็นเซตสเปซ 2 มิติของเลขจำนวนเต็มซึ่งสมาชิกทุกตัวเป็น coordinate ของจุดภาพสีดำในภาพต้นฉบับ หรืออีกนัยหนึ่ง $Z^2 = \{ (x_1,y_1), (x_2,y_2), (x_3,y_3), \ldots \}$ ซึ่ง $(x_1,y_1), (x_2,y_2), (x_3,y_3) \ldots$ เป็นตำแหน่งของจุดภาพสีดำในภาพต้นฉบับ

- ให้ B เป็นเซตใน Z^2 มีสมาชิกคือ $b = (b_1,b_2)$ และกำหนดให้ **การเลื่อนตำแหน่ง** (translation) ของ B ไป $x = (x_1,x_2)$ ซึ่งเขียนแทนด้วย $(B)_x$ นิยามโดย

$$(B)_x = \{c \mid c = b + x, \; for \; b \in B\} \qquad \ldots(2.3)$$

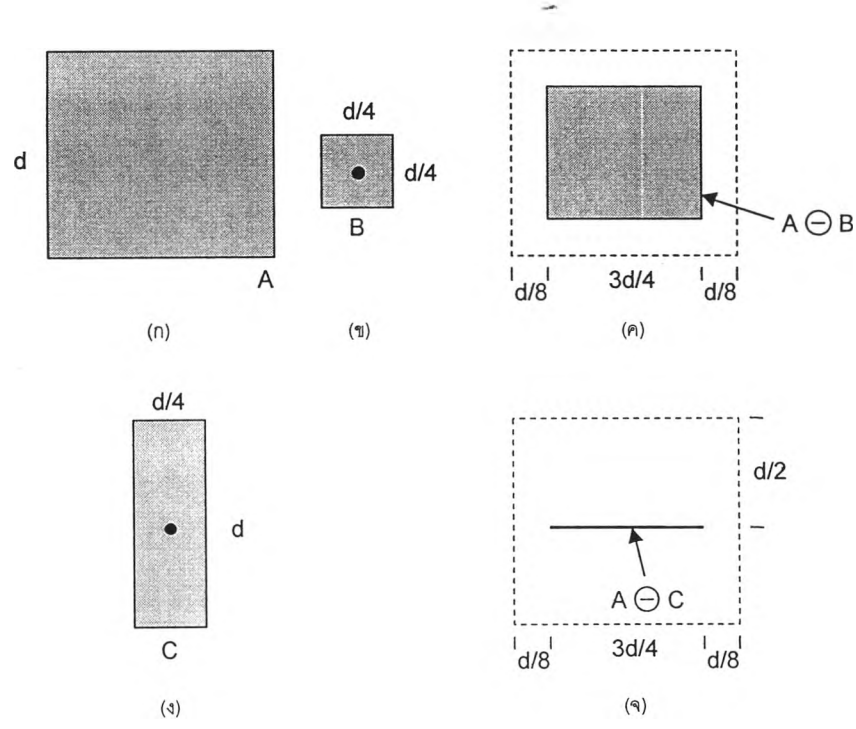
- สำหรับเซต A และ B ใน Z^2 **การยุบตัว** (erosion) ของ A โดย B เขียนแทนด้วย $A \ominus B$ และ

$$A \ominus B = \{x \mid (B)_x \subseteq A\} \qquad \ldots(2.4)$$

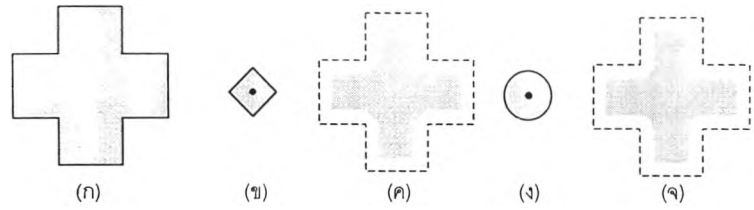
จากนิยามจะได้ว่าการยุบตัวอาศัยนิยามทางคณิตศาสตร์เกี่ยวกับเรื่องเซตและการกระทำบนเซต และได้ผลลัพธ์เป็นเซตของจุดภาพ การยุบตัวจะได้ผลลัพธ์เป็นเซตซึ่งมีจำนวนสมาชิกน้อยกว่าหรือเท่ากับเซตเดิม ในที่นี้เซตจะมีลักษณะพิเศษที่สมาชิกเป็นตำแหน่งจุดภาพ แทนด้วยคู่อันดับ (x,y) สำหรับโดเมนของเซตซึ่งใช้ในการยุบตัวกำหนดไว้เป็น Z^2 เป็นเป็นเซตของตำแหน่งจุดภาพสีดำเท่านั้น จุดภาพสีดำเป็นวัตถุในภาพที่อยู่ในความสนใจ และเป็นวัตถุที่ต้องการทำให้ยุบตัว

สมการ (2.3) กำหนดขึ้นสำหรับการเปลี่ยนตำแหน่งของสมาชิกใน B ไป x สมาชิกทุกตัวของ B จะถูกเปลี่ยนตำแหน่งทั้งในแนวแกน x และ y ด้วย $x = (x_1, y_1)$ เดียวกัน เซตซึ่งต้องมีการเปลี่ยนตำแหน่งในขณะประมวลผลนี้มีลักษณะคล้ายกับ convolution mask

สมการ (2.4) เป็นนิยามของเซตซึ่งมีผลลัพธ์ของการกระทำคือเซตของตำแหน่งจุดภาพซึ่งยุบตัว จะเรียก B ว่า *structure element* ซึ่งเป็นเซตที่กระทำไปบนเซต A เพื่อให้เกิดผลลัพธ์เป็นเซตใหม่ จากนิยาม A เป็นเซตของตำแหน่งจุดภาพสีดำของภาพต้นฉบับซึ่ง A ย่อมเป็นสับเซตของตำแหน่งจุดภาพทั้งหมดในภาพต้นฉบับ และ B เป็นเซตของตำแหน่งจุดภาพสีดำใน *แมสก์ (mask)* ซึ่ง B ย่อมเป็นสับเซตของตำแหน่งจุดภาพทั้งหมดใน mask ด้วย ผลการกระทำการยุบตัวแสดงได้ดังรูปที่ 6 [2]



รูปที่ 6 การประมวลผลด้วยการยุบตัว (ก) เซตต้นแบบ A (ข) structure element B (ค) การยุบตัวของ A โดย B (ง) structure element C (จ) การยุบตัวของ A โดย C



รูปที่ 7 รูปร่างใน structure element ลักษณะต่าง ๆ และผลการประมวลผลด้วยการยุบตัว (ก) เซตต้นแบบ (ข) structure element ซึ่งมีรูปร่างเป็นสี่เหลี่ยมตะแคง (ค) ผลลัพธ์ของการทำการยุบตัวโดยใช้ structure element รูป ข. (ง) structure element ซึ่งมีรูปร่างเป็นวงกลม และ (จ) ผลลัพธ์ของการทำการยุบตัวโดยใช้ structure element รูป ง.

รูปร่างของวัตถุใน structure element จะมีผลต่อผลลัพธ์ที่ได้ดังแสดงในรูปที่ 7 สำหรับงานวิจัยนี้จะใช้ structure element ดังรูป (ง) และกำหนดจุดภาพใน structure element ให้เรียงตัวใกล้เคียงวงกลมมากที่สุดเพื่อให้การยุบตัวมีความสมมาตรในทุกทิศทาง ขั้นตอนวิธีของการยุบตัวแสดงดังรูปที่ 8

```
procedure Erosion (
   OriginalImage  : image;        { original image }
   ResultImage    : image;        { result }
   )

variable
   StructElem : array[3][3] of byte = { 0,1,0, 1,1,1, 0,1,0 };
   x, y, sx, sy, MininWindow, TestValue : integer;

begin
   for y := 0 to 255 do
      for x := 0 to 255 do
         begin
         MininWindow := 255;
         for sy := -1 to 1 do
            for sx := -1 to 1 do
               begin
               if (y-sy) >= 0 and (y-sy) <256 and
                  (x-sx)>=0 and (x-sx)<256 then
                  begin
                  if StructElem[sy+1][sx+1] <> 0 then
                     TestValue := ReadPixel( OriginalImage, y-sy, x-sy );
                  else
                     TestValue := MininWindow;
                  end
               if TestValue < MininWindow then
                  MininWindow := TestValue;
               end
         MininWindow := max( MininWindow, 0 );
         WritePixel( ResultImage, x, y, MininWindow );
         end { for }
```

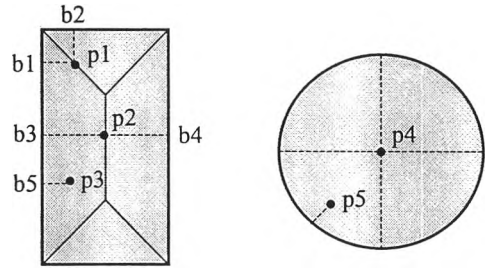
end { Erosion }

รูปที่ 8 ขั้นตอนวิธีการยุบตัว

ขั้นตอนวิธีที่แสดงกำหนดให้ structure element เป็นแมสก์ขนาด 3 x 3 ภาพต้นฉบับจะถูกมองภายใต้กรอบของ structure element ทั่วทั้งภาพ ตำแหน่งหนึ่ง ๆ ของการวางกรอบจะให้ผลลัพธ์ซึ่งเป็นค่าน้อยที่สุดภายในกรอบโดยพิจารณาเฉพาะจุดภาพตำแหน่งตรงกับ structure element ซึ่งมีค่าเป็น 1

2.2.4 การทำให้บาง (Thinning)

การทำให้บาง [2] คือการทำให้บริเวณยุบตัวลงจนเหลือเป็นเส้นโครงร่างของบริเวณ (skeleton of region) ซึ่งมีขนาดความกว้าง 1 จุดภาพโดยที่เส้นโครงร่างยังแสดงความเป็นเค้าโครงเดิมของบริเวณได้ครบถ้วน คำจำกัดความของเส้นโครงร่างของบริเวณ R ซึ่งมี B เป็นขอบของบริเวณคือจุดที่มีระยะห่างน้อยที่สุดระหว่างจุดนั้นและจุดบนเส้นขอบอย่างน้อย 2 จุดขึ้นไป



รูปที่ 9 เส้นโครงร่างของบริเวณสี่เหลี่ยมผืนผ้า และวงกลม เส้นทึบที่เห็นในรูปสี่เหลี่ยมผืนผ้าเป็นเส้นโครงร่าง สำหรับวงกลมนั้นเส้นโครงร่างเป็นจุดและอยู่ที่จุดศูนย์กลางวงกลม

รูปที่ 9 แสดงให้เห็นเส้นโครงร่างของบริเวณสี่เหลี่ยมผืนผ้าและวงกลม พิจารณารูปสี่เหลี่ยมผืนผ้าจะเห็นว่าจุด p1 เป็นจุดบนเส้นโครงร่างเนื่องจากอยู่ห่างจากจุดที่อยู่บนเส้นขอบ b1 และ b2 ตั้งแต่ 2 จุดขึ้นไป และมีระยะห่างใกล้ที่สุด ทำนองเดียวกันกับจุด p2 ซึ่งอยู่ห่างจากจุดที่ใกล้ที่สุด 2 จุดคือ b3 และ b4 ตามลำดับ สำหรับจุด p3 ไม่เป็นจุดบนเส้นโครงร่างเพราะมีจุดที่อยู่ใกล้ p3 ที่สุดอยู่จุดเดียวคือ b5 ในกรณีวงกลมนั้นจะพบว่าเส้นโครงร่างมีลักษณะเป็นจุด ๆ เดียวและอยู่ที่ตำแหน่งจุดศูนย์กลางวง

กลมคือ p4 เพราะตำแหน่งจุดศูนย์กลางวงกลมอยู่ห่างจากจุดบนเส้นขอบทุกจุดเป็นระยะทางเท่ากัน (คือระยะรัศมีวงกลม) สำหรับจุด p5 ไม่อยู่บนเส้นโครงร่างเพราะอยู่ใกล้กับจุดบนเส้นขอบจุดเดียวเท่านั้น

การหาเส้นโครงร่างจากโดยยึดตามคำจำกัดความ [2] ไม่อาจนำมาใช้ปฏิบัติได้โดยตรงเพราะจะมีการคำนวณเกิดขึ้นจำนวนมากซึ่งเป็นการคำนวณระยะห่างของจุดทุกจุดในบริเวณและจุดทุกจุดบนขอบเพื่อตรวจสอบเงื่อนไขว่าจุดใดเป็นจุดบนเส้นโครงร่าง ในทางปฏิบัติจริงจะใช้วิธีการอีกลักษณะหนึ่ง [2] คือ การลบจุดที่เป็นขอบของบริเวณจำนวนหลายรอบจนกระทั่งเหลือส่วนที่เป็นเส้นโครงร่าง การลบจุดจะอยู่ภายใต้เงื่อนไข 3 ข้อคือ (1) ต้องไม่ลบจุดที่เป็นจุดปลาย (2) ต้องไม่ทำให้เส้นขาดความต่อเนื่อง (3) ต้องไม่ทำให้บริเวณยุบตัวมากเกินไป ขอให้สังเกตว่าขอบก็คือเส้นบางที่มีความกว้าง 1 จุดภาพ การประมวลผลครั้งแรกจะทำให้ส่วนที่เป็นขอบจะถูกกำจัดออกไป (โดยยังคงเป็นไปตามเงื่อนไขทั้ง 3 ข้อด้วย) ส่วนนอกสุดที่ไม่ได้ถูกกำจัดออกจะถือเป็นขอบสำหรับการประมวลผลในครั้งที่สองและครั้งต่อไป การประมวลผลแต่ละครั้งจะทำให้บริเวณบางลง 1 จุดภาพ และการประมวลผลจะกระทำซ้ำแล้วซ้ำเล่าจนกระทั่งไม่มีจุดเหลือให้ถูกกำจัดออกไปอีกตามเงื่อนไข

วิธีการหาเส้นโครงร่างโดยการทำให้บริเวณบางลง (thinning) ตามที่ได้กล่าวมาสำหรับภาพขาวดำซึ่งกำหนดให้จุดภาพในบริเวณวัตถุมีค่าเป็น 1 และจุดภาพในพื้นหลังมีค่าเป็น 0 จะกระทำเป็นรอบ ๆ โดยแต่ละรอบแบ่งเป็น 2 ขั้นตอนดังนี้

ขั้นที่ 1

จุดภาพที่จะลบจะต้องเป็นจุดที่อยู่บนขอบของบริเวณและจุดจะอยู่บนเส้นขอบหากจุดนั้นมีค่าเป็น 1 และจุดโดยรอบทั้ง 8 (8-neighbour) [2] มีค่าเป็น 0 อย่างน้อย 1 จุด และจุดที่อยู่บนเส้นขอบของบริเวณจะถูกลบถ้า

(1) $2 \leq N(p1) \leq 6$

(2) $S(p1) = 1$

(3) $p2 \bullet p4 \bullet p6 = 0$

(4) $p4 \bullet p6 \bullet p8 = 0$

โดยที่ N(p1) คือจำนวนจุดรอบ p1 ที่มีค่าไม่เป็น 0 นั่นคือ
$N(p1) = p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9$
(ตำแหน่งของจุดเป็นดังรูปที่ 10)

S(p1) คือจำนวนครั้งในการเปลี่ยนค่าจาก 0 – 1 ตามลำดับของ p2, p3, ... , p8, p9, p2

$\bullet$ คือการ and ทางตรรกะ

p9	p2	P3
p8	p1	p4
p7	p6	p5

รูปที่ 10 แสดงการจัดเรียงตำแหน่งของจุดซึ่งใช้สำหรับขั้นตอนวิธีการหาเส้นโครงร่าง

ขั้นที่ 2

ทำเช่นเดียวกับขั้นที่ 1 โดยคง (1) และ (2) ไว้ แต่เปลี่ยน (3) และ (4) เป็น

(3') p2 ● p4 ● p8 = 0

(4') p2 ● p6 ● p8 = 0

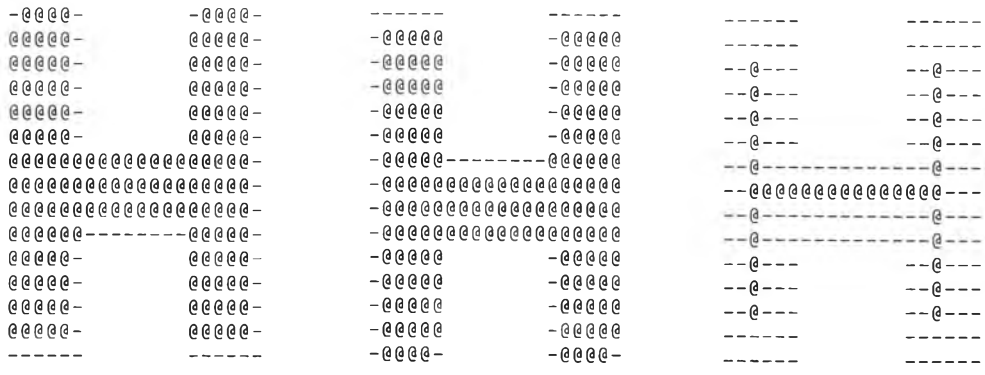
เงื่อนไข (1) กำหนดไว้เพื่อตรวจสอบว่าเป็นจุดในบริเวณหรือจุดปลายหรือไม่ ถ้าจำนวนจุดรอบ p1 เป็น 1 แสดงว่า p1 เป็นจุดปลาย ถ้าจำนวนจุดรอบ p1 เป็น 8 แสดงว่าเป็นจุดในบริเวณ ดังนั้นจะไม่ลบจุดออก และการลบจุด p1 ที่มีจำนวนจุดโดยรอบเป็น 7 จะทำให้เกิดการยุบตัวเข้าไปในบริเวณจึงต้องไม่ลบออกเช่นกัน เงื่อนไข (2) กำหนดไว้เพื่อป้องกันการลบจุดบนเส้นที่มีความหนา 1 จุดภาพผ่าน p1 เพราะมิฉะนั้นจะทำให้เส้นขาดความต่อเนื่องได้ เงื่อนไข (3), (4), (3') และ (4') สำหรับตรวจสอบว่าจุดนั้นเป็นส่วนของด้านหรือมุมซึ่งสามารถลบออกได้

ในขั้นที่ 1 ทุกจุดภาพในบริเวณที่กำลังพิจารณาจะผ่านการตรวจสอบคุณสมบัติ (1) – (4) ถ้าหากขาดคุณสมบัติข้อใดข้อหนึ่งหรือมากกว่าจะถือว่าจุดที่กำลังพิจารณานั้นไม่สามารถลบออกได้ ถ้าจุดที่กำลังพิจารณามีคุณสมบัติครบถ้วน (1) - (4) จุดนั้นจะถูกหมายว่าเป็นจุดที่ลบได้ แต่จะยังไม่ถูกลบจนกว่าจุดอื่น ๆ ในบริเวณได้ถูกตรวจสอบคุณสมบัติเสียก่อน ทั้งนี้เพื่อป้องกันไม่ให้โครงสร้างของบริเวณเสียรูปทรงในขณะที่กำลังประมวลผลจุดหลัง ๆ เมื่อจุดทุกจุดผ่านการตรวจสอบคุณสมบัติแล้วจึงจะมีการลบจุดที่ถูกหมายไว้ และเริ่มประมวลผลในขั้นที่ 2 ด้วยหลักการแบบเดียวกับขั้นที่ 1 เป็นอันครบรอบ

สรุปได้ว่าในแต่ละรอบของการประมวลผลเพื่อหาเส้นโครงร่างหรือเส้นโครงร่างประกอบด้วย

1) การประมวลผลตามขั้นที่ 1 เพื่อหมายจุดที่เข้าข่ายจะถูกลบออก
2) การลบจุดที่ถูกหมายในขั้นที่ 1
3) การประมวลผลตามขั้นที่ 2 เพื่อหมายจุดที่เข้าข่ายจะถูกลบออก
4) การลบจุดที่ถูกหมายในขั้นที่ 2

5) ทำซ้ำข้อ 1) – 4) จนกระทั่งไม่มีจุดใดถูกลบออกอีก จะได้บริเวณที่เหลือในภาพเป็นเส้นโครงร่างของบริเวณต่าง ๆ



(ก) (ข) (ค)

รูปที่ 11 ผลในแต่ละขั้น [2] ของเทคนิคการทำให้บาง (ก) ผลเมื่อผ่านขั้นที่ 1 (ข) ผลเมื่อผ่านขั้นที่ 2 (ค) ผลเมื่อการประมวลผลสิ้นสุด

```
procedure Thining (
    OriginalImage : image;
    OutputImage : image)

variable
    markdelete   : boolean;
    retry        : integer;
    x,y,np1,sp1  : integer;
    mask         : array[9] of byte;

begin

    Set the outer most border of the original image to background;
    retry := 100; { retry limit 100 }
    markdelete := true;

    { repeat until retry limit or no more pixel deleted }
    while retry > 0 and markdelete = true do
        begin
        for step := 1 to 2 do
            begin
            markdelete := false;
            for y := 1 to 254 do
                begin
                for x := 1 to 254 do
                    begin
                    OutputImage[y][x] := OriginalImage[y][x];  --
                    ReadPixeltoMask( x, y, OriginalImage, mask );
                    if step = 1 then
```

```
                    begin
                    if ( mask[1] and mask[5] and mask[7] ) <> 0 then
                        continue;
                    if ( mask[5] and mask[7] and mask[3] ) <> 0 then
                        continue;
                    end
                else
                    begin
                    if ( mask[1] and mask[5] and mask[3] ) <> 0 then
                        continue;
                    if ( mask[1] and mask[7] and mask[3] ) <> 0 then
                        continue;
                    end
                np1 :=  count number of nonzero element in mask in position 0, 1,
                        2, 3, 5, 6, 7, 8;
                if np1 < 2 or np1 > 6 then continue;
                sp1 :=  count number of 0->1 transition change in direction 0-1-2-
                        5-8-7-6-3-0;
                if sp1 <> 1 then continue;
                if OutputImage[y][x] <> 0 then
                    begin
                    OutputImage[y][x] := 0;
                    markdelete := true;
                    end
                end { for}
            end { for }
        end { for }
        retry := retry-1;
    end { while }
end { thining }
```

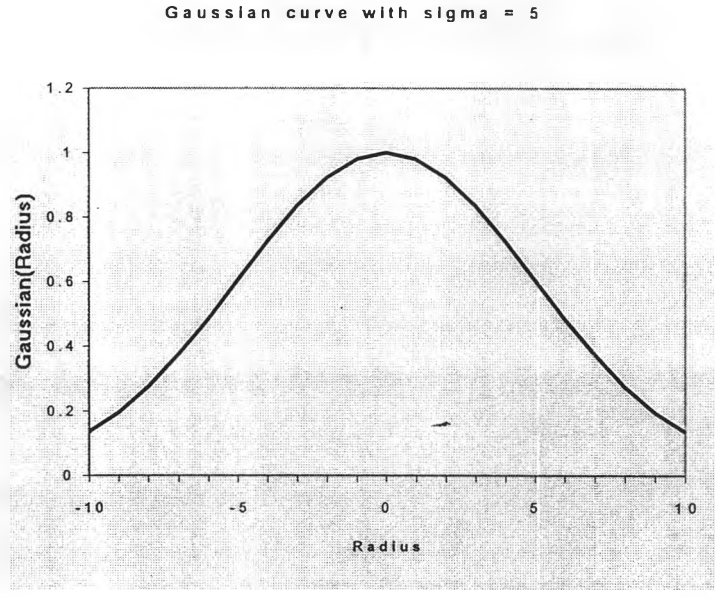
รูปที่ 12 ขั้นตอนวิธีการทำให้บาง

2.2.5 เกาเซียนฟิลเตอร์ (Gaussian filter)

เกาเซียนฟิลเตอร์ [4] เป็นคอนโวลูชันฟิลเตอร์ (convolution filter) ซึ่งค่าถ่วงน้ำหนัก (weight) ในแมสก์มีลักษณะสมมาตรเชิงวงกลม (circularly symmetrical or isotropic) กล่าวคือค่าถ่วงน้ำหนักในแนวตัดขวางใด ๆ ผ่านจุดศูนย์กลางของแมสก์จะอยู่ในรูปแบบของฟังก์ชันเกาส์หรือเส้นโค้งปกติ (normal curve) ตามสมการ (2.5)

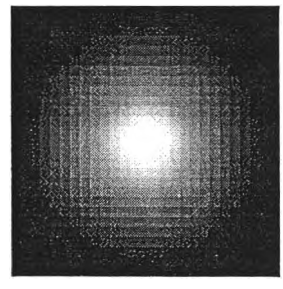
$$G(x) = e^{-\frac{x^2}{2\sigma^2}} \qquad \text{...(2.5)}$$

เส้นโค้งของเกาส์มีลักษณะดังในรูปที่ 13



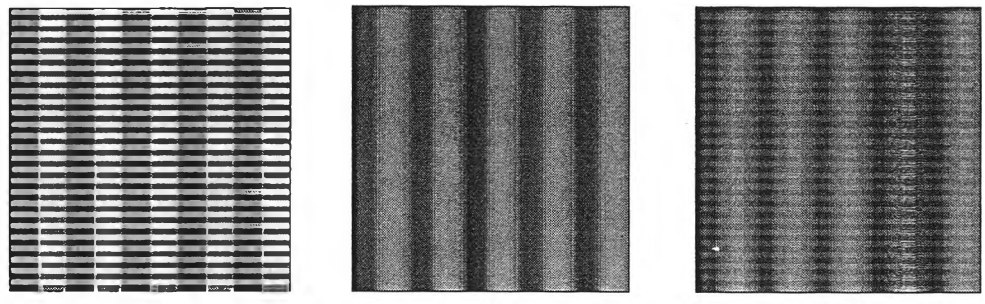
รูปที่ 13 เส้นโค้งของเกาส์ กำหนดค่า $\sigma = 5$

เส้นโค้งของเกาส์จะมีค่าน้อยลงเรื่อย ๆ เมื่อระยะห่างจากจุดศูนย์มากขึ้น แต่จะไม่เป็น 0 เสียทีเดียว ค่าถ่วงน้ำหนักบริเวณขอบของแมสก์จะเข้าใกล้ 0 เมื่อขนาดของแมสก์เพิ่มมากขึ้น ในทางปฏิบัติขนาดของแมสก์จะถูกกำหนดไว้คงที่และจะเลือกค่า σ ที่เหมาะสมเพื่อให้ค่าถ่วงน้ำหนักบริเวณขอบของแมสก์จะมีค่าน้อยพอที่จะอนุมานได้ว่าเป็น 0 เราสามารถสร้างคอนโวลูชันฟิลเตอร์แบบเกาเซียนได้จากอะเรย์ 1 มิติของเกาเซียนฟังก์ชัน ซึ่งค่าถ่วงน้ำหนักที่ตำแหน่งต่าง ๆ ในแมสก์ได้จากผลคูณคาร์ทีเซียนของสมาชิกในอะเรย์



รูปที่ 14 ลักษณะแมสก์ของคอนโวลูชันฟิลเตอร์แบบเกาเซียน

ลักษณะผิว (texture) ของวัตถุที่ปรากฏบนภาพใด ๆ อาจจำแนกได้เป็นลายผิวขนาดเล็ก (small scale texture) และลายผิวขนาดใหญ่ ลายผิวขนาดเล็กหมายถึงบริเวณในภาพซึ่งมีความแปรปรวนของค่าความเข้มแสงของจุดภาพสูงกว่าลายผิวขนาดใหญ่ คอนโวลูชันฟิลเตอร์แบบเกาเซียนมีลักษณะพิเศษที่สามารถกำจัดสิ่งรบกวนประเภทลายผิวขนาดเล็กได้อย่างมีประสิทธิภาพในขณะที่คงไว้ซึ่งความสมบูรณ์ของลายผิวขนาดใหญ่ [4] แมสก์ของคอนโวลูชันฟิลเตอร์แบบเกาเซียนมีความต่อเนื่องในตัวเอง จึงสามารถจำแนกความแปรปรวนโดยปราศจากการใช้ข้อมูลในบริเวณที่มากกว่าเกินจำเป็น ผลของการใช้คอนโวลูชันฟิลเตอร์แบบเกาเซียนแสดงดังรูปที่ 15 ภาพ (ข) และ (ค) มีความแตกต่างกันอย่างชัดเจน ภาพ (ข) ให้ความต่อเนื่องของแถบสีในแนวดิ่งได้ดีกว่าภาพ (ค) ความแตกต่างนี้เป็นเพราะคอนโวลูชันแบบเกาเซียนได้หลีกเลี่ยงการเปลี่ยนค่าถ่วงน้ำหนักอย่างฉับพลันที่บริเวณขอบของแมสก์ (ค่าที่บริเวณขอบค่อย ๆ ลาดลงจนกระทั่งเกือบเป็น 0) ในขณะที่ยูนิฟอร์มฟิลเตอร์[8] (uniform filter) มีค่าถ่วงน้ำหนักในแมสก์เท่ากันทุกค่า จึงมีการเปลี่ยนค่าถ่วงน้ำหนักบริเวณขอบของแมสก์อย่างฉับพลัน คอนโวลูชันแบบเกาเซียนมีความสำคัญทั้งในแง่ทฤษฎีของภาพทางชีววิทยา (biological vision) และในระบบคอมพิวเตอร์เพื่อการแสดงภาพ (computer vision system) [4]



(ก) (ข) (ค)

รูปที่ 15 การใช้คอนโวลูชันฟิลเตอร์แบบเกาเซียน (ก) ภาพต้นฉบับประกอบด้วยแถบสีขาวสลับดำทอดตัวในแนวดิ่ง และมีแถบเส้นสีดำตัดขวางในแนวนอนด้วยความถี่ที่สูงกว่า (ข) ภาพซึ่งใช้คอนโวลูชันฟิลเตอร์แบบเกาเซียน กำหนด σ = 1.8 และขนาดแมสก์ 7x7 จะพบว่าความต่อเนื่องของแถบสีขาวสลับดำในแนวดิ่งดีมาก เส้นรบกวนในแนวนอนถูกกำจัดออกเกือบหมด (ค) ภาพซึ่งใช้ยูนิฟอร์มฟิลเตอร์ขนาด 7x7 ซึ่งมีค่าถ่วงน้ำหนักเป็น 1

---

[8] ยูนิฟอร์มฟิลเตอร์เป็นฟิลเตอร์ซึ่งมีค่าถ่วงน้ำหนักในแมสก์เท่ากันทุกค่าซึ่งในหนังสือบางเล่มอาจเรียกว่า low pass filter ก็ได้