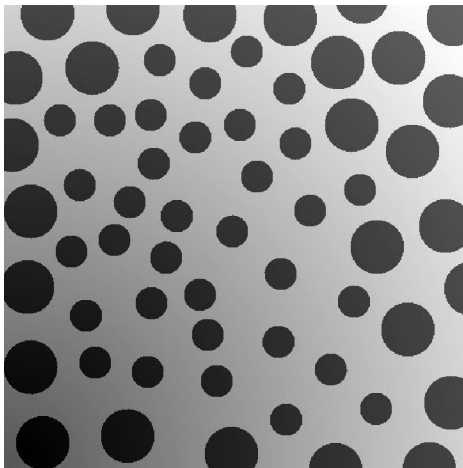


Kittipong Tapyou 65070501003

## Homework 09 : Variable Thresholding

Key characteristics on the given image



รูปภาพที่ได้เป็นรูปที่มีลักษณะเป็น gray level image โดยที่เป็นภาพวงกลมหลายๆวง ที่มีค่า intensity ต่ำ (ดำคือวงกลม, ขาวคือพื้นหลัง) ภาพมีลักษณะที่ไม่มี noise แต่มีการไล่เฉดของความเข้ม(illumination) โดยหากพิจารณาภาพที่ควรจะเป็นคือมีแค่วงกลมสีดำ และพื้นหลังสีขาว แต่ภาพที่ได้พื้นหลังมีการไล่เฉดของแสง

เมื่อทำภาพไปประมวลผลเช่นการทำ binarization เพื่อแยก object และ background อาจจะทำให้ได้ผลที่ไม่สมบูรณ์เนื่องจากพื้นหลังมีค่าสีใกล้เคียงกับวัตถุในบางพื้นที่ ทำให้เมื่อทำ thresholding โดยใช้ global threshold อาจทำให้ไม่สามารถแยก object และ background ได้ไม่ถูกต้อง background อาจจะถูก detect เป็น object ด้วย

ดังนั้น strategies ที่จะใช้ในงานนี้คือการทำ image segmentation โดยใช้หลักการของ thresholding method เพื่อแยก object กับ background ออกจากกัน

## Proposed Methodology

เนื่องจากภาพมี illumination ทำให้ background ของภาพนั้นมีหลาย intensity การใช้ global thresholding จะทำให้ object ที่ได้ไม่สมบูรณ์ ซึ่งวิธีที่จะสามารถแยก object กับ background ในภาพนี้นั้นคือการทำ thresholding โดยที่มีหลาย threshold ในหลายพื้นที่ (variable thresholding) ซึ่งแบ่งภาพเป็น partition ย่อยๆ จากนั้นทำการ binarize รูปภาพในแต่ละ partition จึงค่อยนำภาพมารวมกัน

โดยเขียนเป็นขั้นตอนได้ดังนี้

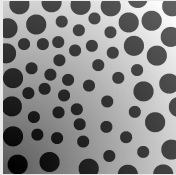
- ทำการสลับ black กับ white pixel เพื่อให้จุดที่เราสนใจซึ่งคือวงกลม กลายเป็น white pixel
- ทำการแบ่งภาพออกเป็น partition ที่ขนาดเท่ากัน โดยแบ่งออกเป็น  $4^n$  รูปภาพ

โดยที่แต่ละภาพมีขนาดเท่ากับ  $\{w, h\} / 2^n$

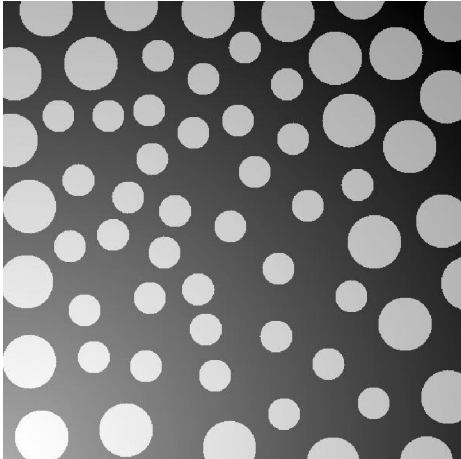
- หา threshold ของแต่ละรูปภาพที่ถูกแบ่งออกมาโดยใช้ Otsu's method
- นำ threshold ของแต่ละรูปที่ได้ มา binarize รูปภาพ
- นำภาพที่ถูก binarize มารวมกัน เพื่อให้ได้ภาพขนาดเท่าเดิม

(\*หมายเหตุ : ปกติแล้วจะให้ background เป็น 0 และ object เป็น 1 ซึ่งต้องนำค่าสีที่ได้จากภาพมา inverse ขั้นตอนนี้จะทำก่อนหรือหลัง thresholding ก็ได้ \*)

## Provided Code

```
In[*]:=
img = ;

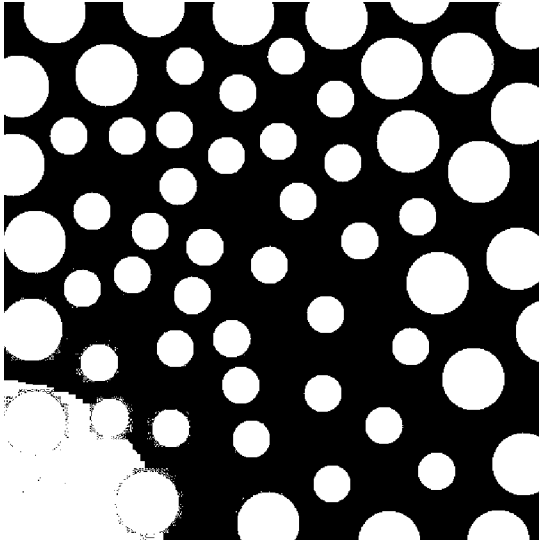
grayImg = ColorConvert[img, "Grayscale"];
grayImg = Image[Abs[ImageData[grayImg] - 1]]
```

Out[\*]= 

ทำการแปลงภาพให้อยู่ในรูปของ grayscale จากนั้นทำกับสลับ intensity (inverse dark pixel กับ white pixel)

```
In[ ]:= binImg = Binarize[grayImg]
```

```
Out[ ]:=
```



จะเห็นว่าเมื่อภาพเกิด illumination ทำให้เมื่อภาพถูก binarize โดยใช้ global threshold, background จะถูก classify เป็น object ด้วย

```
In[ ]:= Otsu[img_] := Module[{p1, p2, c1, c2, m1, m2, , c, nPixels, imgData, h, i, P1, P2, mk,
  imgData = ImageData[img, "Byte"];
  nPixels = Length[Flatten[imgData, 1]];
  h = BinCounts[Flatten[imgData, 1], {0, 256, 1}];
  p = # / nPixels & /@ h;
  mg = Sum[i p[[i]], {i, 1, 255}];
  {min, max} = MinMax[Flatten[imgData, 1]];
  For[i = min, i < max, i++,
    p1 = # / nPixels & /@ h[[i]];
    p2 = # / nPixels & /@ h[[i+1]];

    P1 = Total[p1];
    P2 = 1 - P1;

    mk = Sum[j p1[[j]], {j, 1, Length[p1]}];
    m1 = mk / P1;
    m2 = Sum[j p2[[j - i + 1]], {j, i, max - i}] / P2;

     $\sigma_i = (mg P1 - mk)^2 / (P1 (1 - P1))$ ;
    AppendTo[ $\sigma$ , i  $\rightarrow$   $\sigma_i$ ];

  ];
  idx = First @ Keys @ MaximalBy[ $\sigma$ , Last];
  Return[idx];
]
```

สร้าง thresholding โดยใช้ Otsu's method แทนที่การใช้คำสั่ง Binarize ซึ่งเป็น Built-in function

```

In[ ]:= GrayToBin[image_, thres_] := Module[{dim, img},
  img = ImageData[image, "Byte"];
  dim = Dimensions[img];
  Return[
    Table[If[img[[i, j]] > thres, 1, 0], {i, dim[[1]]}, {j, dim[[2]]}]
  ];
]

```

Function สำหรับการทำ binarize โดย input คือ image และ threshold

```

In[ ]:= VariableThresholding[img_, level_] := Module[{dim, partImgs, partDim, hist, binImgs, t},
  dim = ImageDimensions[img];
  partImgs = ImagePartition[img, dim/ (2^level)];
  partDim = Dimensions[partImgs];
  hist = ConstantArray[{}, partDim];
  binImgs = ConstantArray[{}, partDim];
  thresholdList = ConstantArray[{}, partDim];
  For[i = 1, i ≤ partDim[[1]], i++,
    For[j = 1, j ≤ partDim[[2]], j++,
      thres = N[Otsu[partImgs[[i, j]]]];
      thresholdList[[i, j]] = thres;
      binImgs[[i, j]] = Image[GrayToBin[partImgs[[i, j]], thres]];
      hist[[i, j]] = Histogram[
        Flatten[ImageData[partImgs[[i, j]], "Byte"], 1],
        255,
        Epilog->{Red, Thick, InfiniteLine[
          {{thres, 0}, {thres, 1}}
        ]}
      ];
    ];
  ];

  Return[{partImgs, binImgs, thresholdList, hist}]
]

```

Function สำหรับการทำ variable thresholding ซึ่ง จะแบ่งรูปภาพออกเป็นหลาย partition จากคำสั่ง ImagePartition โดยที่ภาพที่ได้จะมีขนาดเท่ากับ  $\{w, h\} / 2^n$  ซึ่งสามารถปรับเปลี่ยน  $n$  ได้ ทำให้ได้รูปภาพจำนวน  $4^n$  รูป จากนั้นทำการหา threshold ของแต่ละ partition โดยใช้ Otsu's method, เมื่อได้ threshold มาแล้วก็ทำการ binarize รูปภาพจาก threshold นั้นๆ อีกทั้งยังได้มีการ plot histogram ของ gray level ในแต่ละ partition

```

In[ ]:= result = VariableThresholding[grayImg, 1];

```

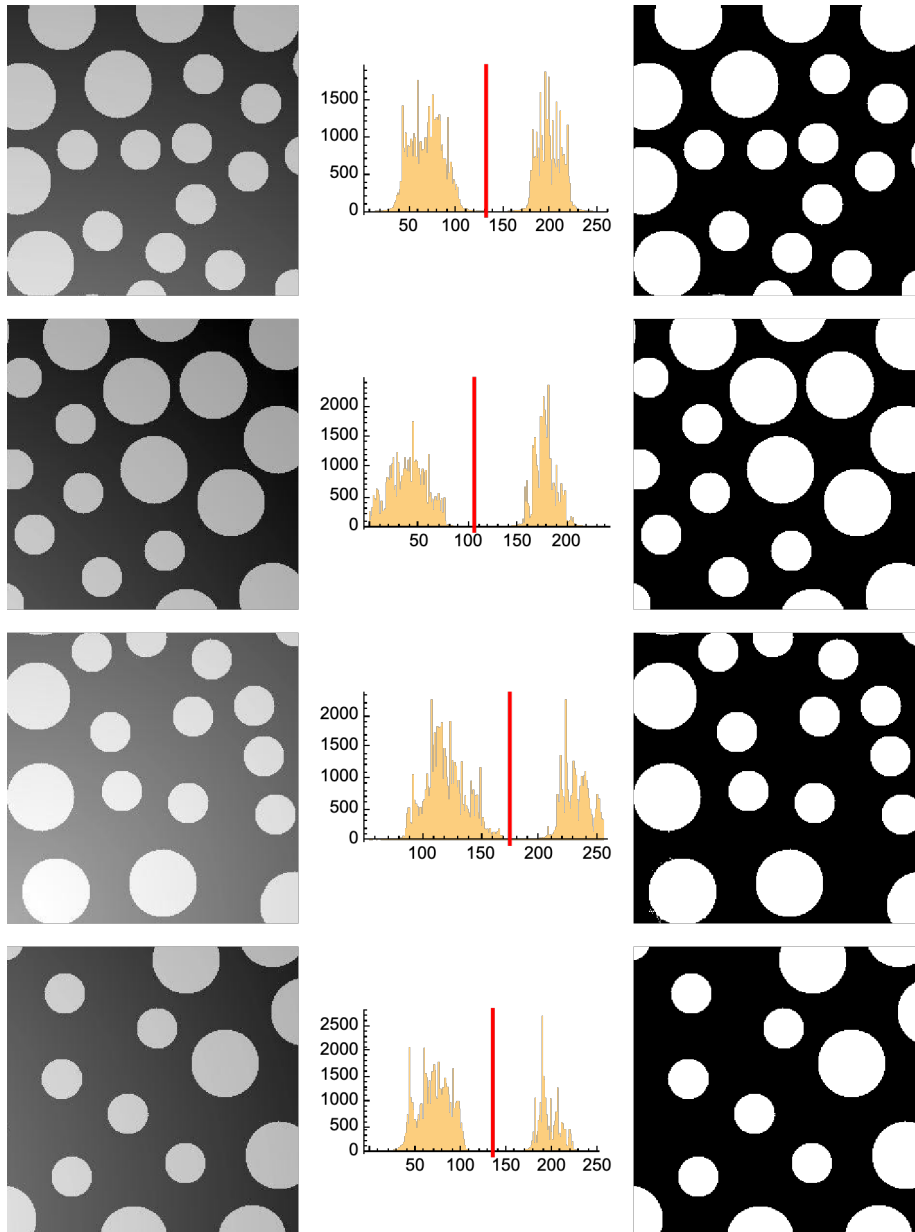
จากตัวอย่างคือการทำ variable thresholding จากวิธีการด้านบน โดย  $n = 1$  (แบ่งรูปภาพออกเป็น 4 รูป)

```

In[ ]:= GraphicsGrid[
  Transpose[{
    Flatten[result[[1]], 1],
    Flatten[result[[4]], 1],
    Flatten[result[[2]], 1]
  }]
]

```

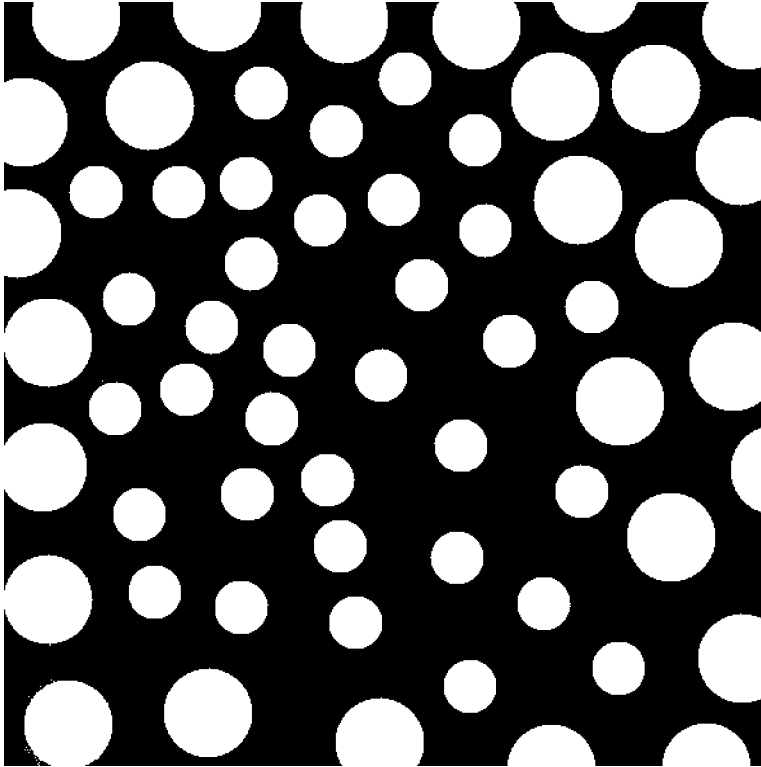
Out[ ]:=



จากตารางด้านบนจะเป็นการแสดงผลลัพธ์ที่ได้จากการทำ variable thresholding โดยที่จะแบ่งรูปภาพออกเป็น 4 รูป โดย column 1, 2, 3 แทนถึง gray level image, histogram & threshold , binary image ตามลำดับ โดยเราจะเห็นได้ว่าการแบ่งภาพออกเป็น partition ย่อยๆ จากนั้นทำการหา threshold จะทำให้ได้ threshold ที่เหมาะสมในแต่ละ sub region ตัวอย่างเช่นรูปที่ 2 มี threshold ประมาณ 100 แต่ภาพที่ 3 มี threshold ประมาณ 175 ดังนั้นหากใช้ global threshold จะไม่สามารถแยก object ออกจาก background ได้

```
In[ ]:= ImageAssemble[result[[2]]]
```

```
Out[ ]:=
```



ภาพผลลัพธ์หลังจากการทำ variable thresholding  
 โดยจะนำภาพจากขั้นตอนก่อนหน้ามารวมกันโดยใช้คำสั่ง image assemble เพื่อรวม partition  
 ต่างๆเข้าด้วยกัน จะเห็นได้ว่าสามารถแยก background และ object ออกจากกันอย่างชัดเจน โดยที่  
 object เป็นสีขาว และ background เป็นสีดำ

## Result:

เนื่องจากภาพที่มีลักษณะของการเกิด illumination นั้นจะทำให้เกิดการไล่ระดับของ intensity  
 ดังนั้นแต่ละพื้นที่ จะมี intensity ของ background ที่แตกต่างกัน จากผลลัพธ์ที่ได้จาก variable threshold-  
 ing โดยการแบ่งภาพออกเป็น partition ย่อย และทำ binarize ในแต่ละ partition ด้วยค่า threshold  
 ที่ต่างกันในแต่ละรูป จากวิธี Otsu's method เราสามารถ classify object และ background  
 ได้อย่างแม่นยำจากผลลัพธ์ที่แสดงด้านบน

เพิ่มเติม : ในกรณีของการศึกษานี้ ภาพที่ได้นั้นไม่มีองค์ประกอบของ noise และ เห็นขอบของ object  
 อย่างชัดเจน, แต่หากเป็นภาพที่มี noise หรือขอบไม่ชัดเจน อาจจะต้องมี blur รูปภาพ หรือ ทำ high pass  
 เพื่อให้เห็นขอบของภาพได้อย่างชัดเจน