



การใช้ Image processing ในการเบลอภาพ ด้วย Matrix

จัดทำโดย

นาย สุธิ์ สาระพันธ์ รหัสนักศึกษา 63015190

นาย รัฐศักดิ์ ประชุมรักษ์ รหัสนักศึกษา 63015149

รายงานเล่มนี้เป็นส่วนหนึ่งของการศึกษาในรายวิชา ELEMENTARY DIFFERENTIAL
EQUATIONS AND LINEAR ALGEBRA

ภาคเรียนที่ 2/2563 หลักสูตร วิศวกรรมศาสตร์

คณะวิศวกรรมศาสตร์คอมพิวเตอร์(ต่อเนื่อง)

ปีการศึกษา 2563

การใช้ Image processing ในการเบลอภาพ ด้วย Matrix

จุดประสงค์ในการประยุกต์ใช้งาน

1. เพื่อการเรียนรู้และปฏิบัติในด้านการใช้งาน Matrix
2. เพื่อเรียนรู้ในการทำ Image processing และนำมาประยุกต์ใช้
3. เพื่อการศึกษาเขียนโปรแกรมเพื่อนำไปประยุกต์ใช้กับ Matrix

Image Processing

การประมวลผลภาพ (Image processing) จากการศึกษาทฤษฎีการประมวลผลด้วยภาพนั้น จะต้องเริ่มจากการพิจารณาปัจจัยต่างๆ อาทิ เช่น ชนิดของรูปภาพ ความละเอียดของภาพ โหมดของสี โหมดของสีเทา การปรับค่าความเข้มสีให้อยู่ ในช่วงมาตรฐาน การกรองสัญญาณรบกวน เป็นต้น

ชนิดของรูปภาพ

- การจำแนกชนิดของรูปภาพสามารถแบ่งได้จากวิธีการจัดเก็บได้เป็น 2 ชนิดใหญ่ๆ ดังต่อไปนี้
- รูปภาพเวกเตอร์ (Vector Graphic) ที่โครงสร้างของรูปภาพเกิดจากลายเส้นต่างๆ อาทิ เช่น เส้นตรง เส้นโค้ง เป็นต้น ทำให้เป็นรูปภาพที่ไม่ขึ้นอยู่กับความละเอียดของภาพ เมื่อทำการย่อ ขยาย ภาพชนิดนี้แล้วยังทำให้ภาพคมชัดเสมอ รูปภาพชนิดนี้สามารถสร้างจากโปรแกรม Adobe Illustrator, Corel DRAW เป็นต้น
 - รูปภาพแบบบิตแมป (Bitmap Image) ที่โครงสร้างของรูปภาพเกิดมาจากจุดเล็กๆ ประกอบรวมตัวกันขึ้นมาจนเห็นเป็นรูปภาพขึ้นมา ดังนั้นคุณภาพของรูปภาพชนิดนี้จะขึ้นอยู่กับความ ละเอียดของภาพด้วย เมื่อนำภาพมาขยายด้วยคอมพิวเตอร์จะพบว่า ภาพที่ได้มีลักษณะที่หยاب และ เมื่อย่อรูปก็จะพบว่ารูปภาพนั้นมีความชัดเจนนขึ้น รูปภาพชนิดนี้สามารถสร้างจากโปรแกรม Adobe Photoshop, Corel PHOTO Paint เป็นต้น

โหมดของสี

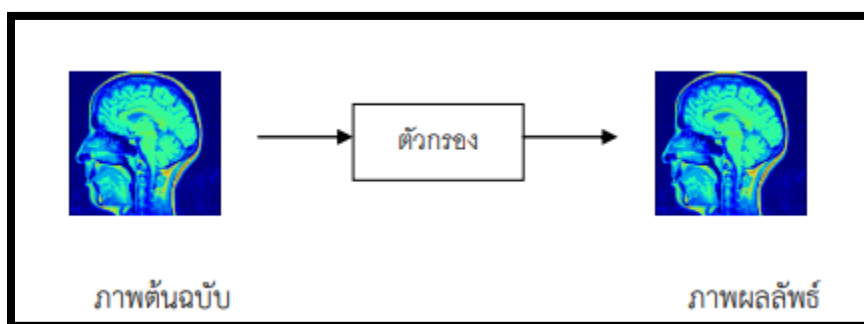
จากการพิจารณาชนิดของภาพ ความละเอียดของภาพ ที่กล่าวไปข้างต้นแล้ว การประมวลผลด้วยภาพยังได้มีการพิจารณาโหมดสีของภาพที่เกิดมาจากการผสมแม่สีจำนวน 3 สีคือ สีแดง สีเขียว และ น้ำเงินบนคอมพิวเตอร์หรือสามารถเรียกอีกอย่างหนึ่งว่า RGB colors (Red-Green-Blue) ซึ่งความเหมือนจริงของสีที่ผสมด้วยคอมพิวเตอร์ขึ้นอยู่กับในหนึ่งจุด (pixel) ของ การแสดงผลนั้นใช้ระดับของสีหรือค่าของ color depths ว่ามีค่าเป็นเท่าไร ตัวอย่างเช่นสี RGB มีค่า color depths เป็น 8 planes ทำให้ต้องใช้ 8 บิตเก็บข้อมูลหนึ่งสี ดังนั้นแม่สีแต่ละสีจะมีระดับสี จำนวน $2^8 = 256$ ระดับ เมื่อมีการผสมสีหนึ่งสีจาก แดง-เขียว-น้ำเงิน (RGB) จะต้องพิจารณาสีแต่ละ ส่วนจาก 0 ถึง 255 ส่วน ดังนั้น RGB ทั้งหมดจะมีจำนวนบิตเท่ากับ 24 บิต ($8+8+8$) ทำให้การแสดงผล สี RGB ในหนึ่งจุด (pixel) จะมีจำนวนสีมากถึง $256 \times 256 \times 256 = 16.7$ ล้านสี

การแปลงภาพสีให้เป็นภาพขาว-ดำ(Thresholding)

เป็นกระบวนการแปลงภาพสีให้มีการแสดงผลได้แค่ 2 ระดับ คือ ขาว และดำ โดยจะแปลงข้อมูลภาพให้เป็นภาพ binary (Binary Image) มีกระบวนการแปลงภาพที่ มีความเข้มหลายระดับ (Multilevel Image) ให้เป็นภาพที่ มีความเข้มเพียง 2 ระดับ หรือ 1 บิต (bit) คือ 0 และ 1 โดย 0 แทน ด้วยจุดที่มีภาพสีขาว และ 1 แทนด้วยจุดที่มีภาพสีดำ Thresholding Technique คือการพิจารณาจุด pixel ในภาพว่าจุดใดควรจะเป็นจุดขาว หรือจุด ใดควรจะเป็นจุดที่มีค่าเท่ากับ 1 (จุดดำ) โดยจะทำการเปรียบเทียบค่าของแต่ละ pixel ($f(x,y)$) กับค่าคงที่ ที่เรียกว่า Threshold (Threshold Value) เทคนิคนี้ นิยมใช้กันมากในกรณีที่ ความแตกต่างระหว่างวัตถุ (Object) และพื้นหลัง (Background) ค่า pixel ในภาพที่ มีค่าน้อยกว่าค่า Threshold จะถูกกำหนดเป็น 1 (จุดดำ) และถ้าค่าของ pixel ใดๆ ในภาพมีค่ามากกว่าหรือเท่ากับค่า Threshold จะถูกกำหนดให้ เป็น 0 (จุดขาว)

การกรองข้อมูลภาพ

การกรองข้อมูลภาพ (Image Filtering) คือการนำภาพไปผ่านตัวกรองสัญญาณเพื่อให้ได้ภาพผลลัพธ์ออกมา ภาพผลลัพธ์ที่ได้จะมีคุณสมบัติแตกต่างจากภาพเริ่มต้น วัตถุประสงค์หลักของการกรองข้อมูลภาพคือการเน้น (enhance) หรือลดทอน (attenuate) คุณสมบัติบางประการของภาพ เพื่อให้ได้ภาพที่มีคุณสมบัติตามต้องการ

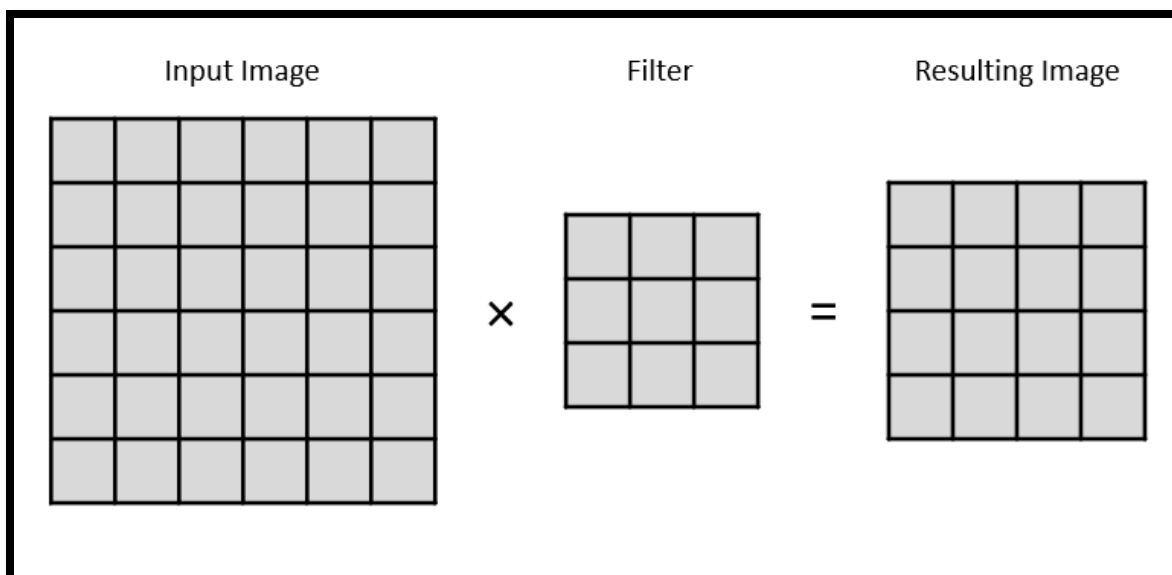


รูปที่ 1 การกรองข้อมูลภาพ

Convolution

Convolution เป็นหนึ่งในการประมวลผลสัญญาณและภาพ สามารถทำงานใน 1D (เช่นการประมวลผลเสียง), 2D (เช่นการประมวลผลภาพ) หรือ 3D (การประมวลผลวิดีโอ) Convolution ในเชิงพื้นที่ 2 มิติซึ่งส่วนใหญ่จะใช้ในการประมวลผลภาพสำหรับการแยกคุณลักษณะและยังเป็นบล็อกหลักของ Convolutional Neural Networks (CNN) โดยทั่วไปเราสามารถพิจารณาภาพเป็นเมทริกซ์ที่มีองค์ประกอบเป็นตัวเลขระหว่าง 0 ถึง 255 ขนาดของเมทริกซ์นี้คือ (ความสูงของภาพ) \times (ความกว้างของภาพ) \times (ช่องภาพ) ภาพระดับสีเทา มี 1 ช่องสัญญาณโดยที่ภาพสีมี 3 ช่องสัญญาณ (สำหรับ RGB)

เมทริกซ์เหล่านี้สามารถนำไปใช้กับรูปภาพเพื่อใช้เอฟเฟกต์ภาพซึ่งสามารถทำได้ผ่านการดำเนินการทางคณิตศาสตร์ที่เรียกว่าการแปลง นี่คือการเพิ่มแต่ละพิกเซลของรูปภาพในพื้นที่โดยให้น้ำหนักด้วยเคอร์เนล

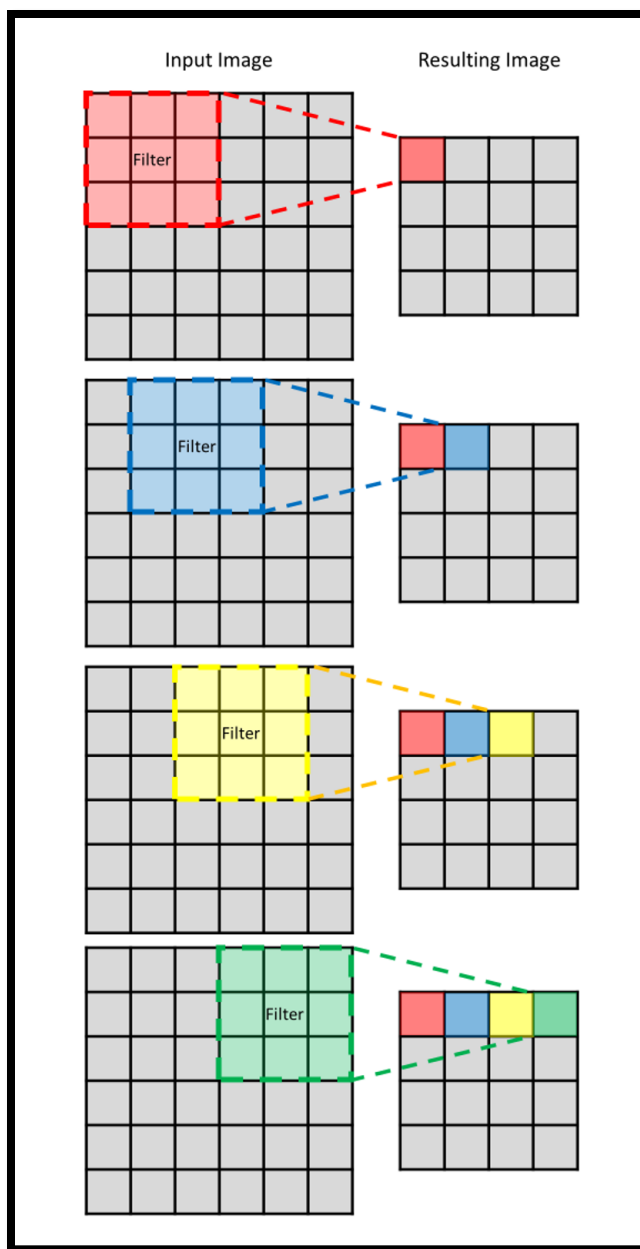


รูปที่ 2 รูปแบบ Matrix * Filter

ค่าของพิกเซลในภาพที่ได้จะคำนวณโดยการคูณค่า Matrix Kernel และค่า Matrix ของรูปภาพ กระบวนการนี้จะวนซ้ำจนกว่าเคอร์เนลจะเสร็จสิ้นการทำซ้ำการคูณนี้จนกว่าจะได้ Output ของ Matrix ครบตามจำนวน Rows และ Collums

Matrix รูปภาพ	Matrix Kernel	Matrix ผลลัพธ์																																																																																																				
Matrix Original จากรูป	Box Blur :	Box Blur :																																																																																																				
<table><tr><td>155</td><td>158</td><td>163</td><td>..</td><td>172</td><td>172</td><td>172</td></tr><tr><td>154</td><td>157</td><td>161</td><td>..</td><td>172</td><td>172</td><td>172</td></tr><tr><td>152</td><td>154</td><td>159</td><td>..</td><td>172</td><td>172</td><td>172</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr><tr><td>197</td><td>198</td><td>198</td><td>..</td><td>1</td><td>1</td><td>1</td></tr><tr><td>195</td><td>197</td><td>197</td><td>..</td><td>1</td><td>1</td><td>1</td></tr><tr><td>194</td><td>196</td><td>196</td><td>..</td><td>1</td><td>1</td><td>1</td></tr></table>	155	158	163	..	172	172	172	154	157	161	..	172	172	172	152	154	159	..	172	172	172	197	198	198	..	1	1	1	195	197	197	..	1	1	1	194	196	196	..	1	1	1	<table><tr><td>1/9</td><td>1/9</td><td>1/9</td></tr><tr><td>1/9</td><td>1/9</td><td>1/9</td></tr><tr><td>1/9</td><td>1/9</td><td>1/9</td></tr></table>	1/9	1/9	1/9	1/9	1/9	1/9	1/9	1/9	1/9	<table><tr><td>69.33333333</td><td>105.33333333</td><td>107.22222222</td><td>112.77777778</td><td>114.66666667</td><td>76.44444444</td></tr><tr><td>103.33333333</td><td>157</td><td>159.44444444</td><td>173.11111111</td><td>172</td><td>114.66666667</td></tr><tr><td>102.55555556</td><td>155.77777778</td><td>158.66666667</td><td>172</td><td>165.33333333</td><td>115.33333333</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr><tr><td>131.33333333</td><td>197.22222222</td><td>197.66666667</td><td>1</td><td>1</td><td>0.6666666667</td></tr><tr><td>130.77777778</td><td>196.44444444</td><td>197</td><td>1</td><td>1</td><td>0.6666666667</td></tr><tr><td>86.88888889</td><td>130.55555556</td><td>130.77777778</td><td>0.6666666667</td><td>0.6666666667</td><td>0.4444444444</td></tr></table>	69.33333333	105.33333333	107.22222222	112.77777778	114.66666667	76.44444444	103.33333333	157	159.44444444	173.11111111	172	114.66666667	102.55555556	155.77777778	158.66666667	172	165.33333333	115.33333333	131.33333333	197.22222222	197.66666667	1	1	0.6666666667	130.77777778	196.44444444	197	1	1	0.6666666667	86.88888889	130.55555556	130.77777778	0.6666666667	0.6666666667	0.4444444444
155	158	163	..	172	172	172																																																																																																
154	157	161	..	172	172	172																																																																																																
152	154	159	..	172	172	172																																																																																																
...																																																																																																
197	198	198	..	1	1	1																																																																																																
195	197	197	..	1	1	1																																																																																																
194	196	196	..	1	1	1																																																																																																
1/9	1/9	1/9																																																																																																				
1/9	1/9	1/9																																																																																																				
1/9	1/9	1/9																																																																																																				
69.33333333	105.33333333	107.22222222	112.77777778	114.66666667	76.44444444																																																																																																	
103.33333333	157	159.44444444	173.11111111	172	114.66666667																																																																																																	
102.55555556	155.77777778	158.66666667	172	165.33333333	115.33333333																																																																																																	
...																																																																																																	
131.33333333	197.22222222	197.66666667	1	1	0.6666666667																																																																																																	
130.77777778	196.44444444	197	1	1	0.6666666667																																																																																																	
86.88888889	130.55555556	130.77777778	0.6666666667	0.6666666667	0.4444444444																																																																																																	

รูปที่ 3 ตัวอย่าง Matrix * Matrix kernel โดยวิธีการ Convolution

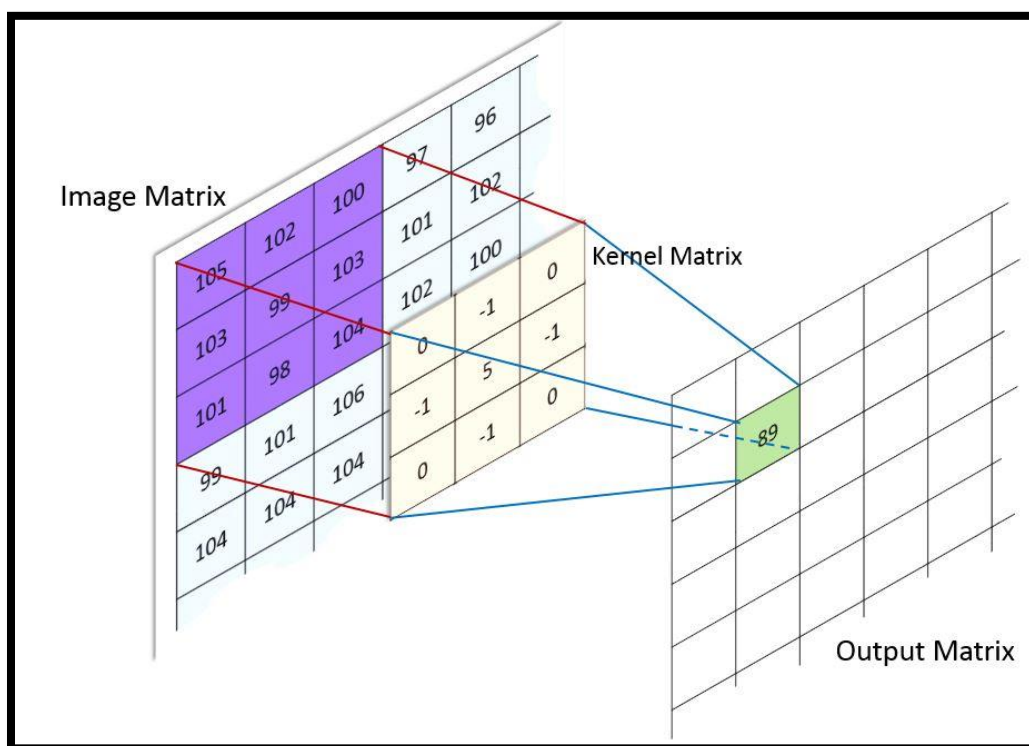


รูปที่ 4 การคูณ Matrix และ Matrix Kernel แต่ละตำแหน่ง

การใช้งาน Convolution แต่ละรายการมีเคอร์เนลซึ่งอาจเป็นเมทริกซ์ใดก็ได้ที่เล็กกว่าภาพต้นฉบับในด้านความสูงและความกว้าง แต่ละเคอร์เนลมีประโยชน์สำหรับงานเฉพาะเช่นการเลาการเบลอการตรวจจับขอบและอื่น ๆ

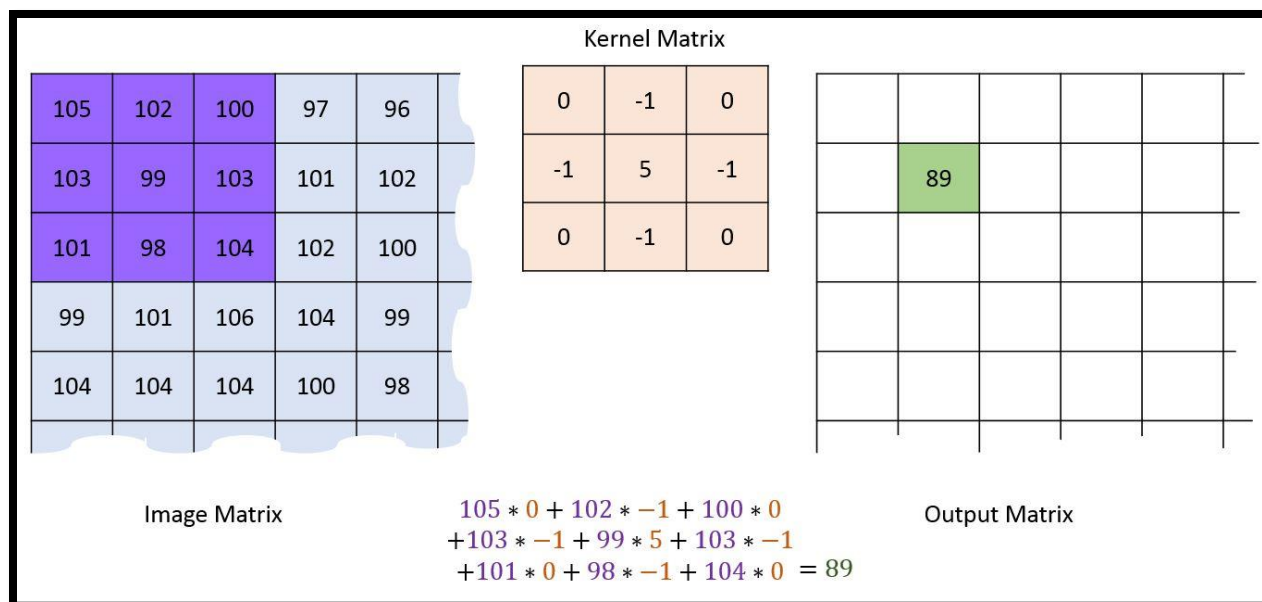
การใช้งาน Convolution

1. พลิ๊กเคอร์เนลทั้งแนวนอนและแนวตั้ง เนื่องจากเคอร์เนลที่เลือกจะเป็นแบบสมมาตร เคอร์เนลที่พลิ๊กจะเท่ากับเคอร์เนลดั้งเดิม
2. เลือกตำแหน่งที่ 1,1 ของเคอร์เนลใน Matrix และจะนำไปคูณกับค่า Matrix ของภาพ จากนั้นจะเลือกตำแหน่ง 1,2 , 1,3 ไปจนถึงตำแหน่งที่ 3,3 ตามลำดับ



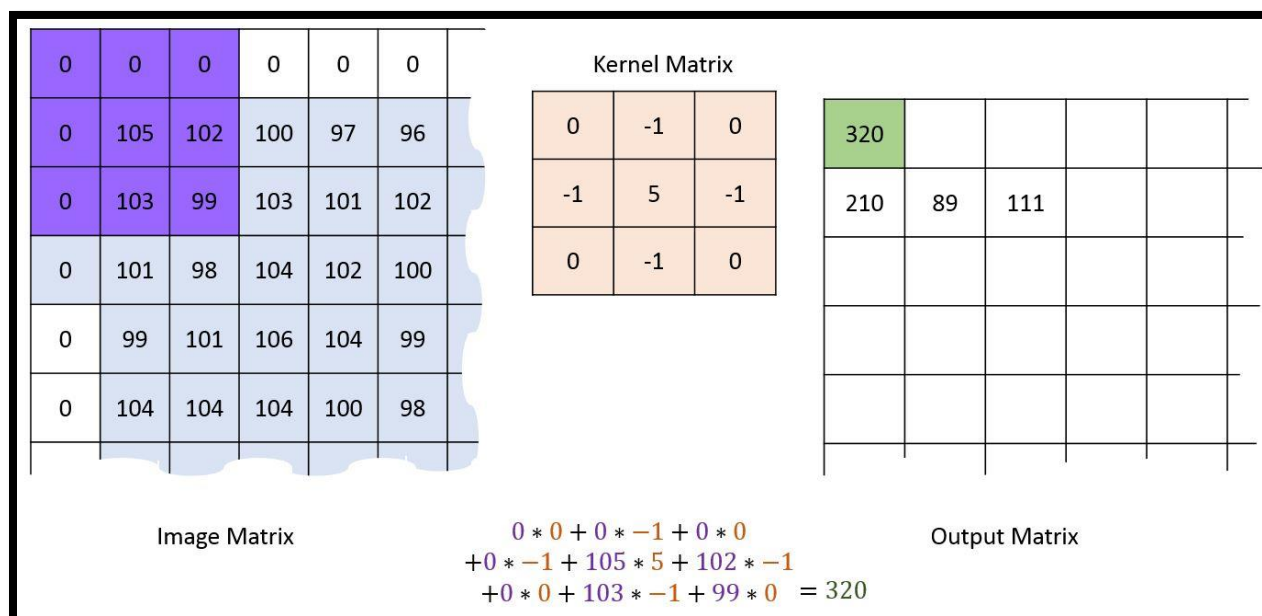
รูปที่ 5 เปรียบเทียบ Matrix * Matrix Kernel ตามตำแหน่ง

3. คูณแต่ละตำแหน่งของ Matrix Kernel และ Matrix ของรูปภาพเข้าด้วยกัน ค่าผลลัพธ์ที่ได้ ออกมาจะเท่ากับตำแหน่งนั้นๆ ของ Matrix Kernel
4. สรุปผลลัพธ์ Matrix Kernel * Matrix ของรูปภาพทั้งหมดและวางผลลัพธ์ไว้ที่ตำแหน่งเดียวกันในเมทริกซ์เอาต์พุตของ Matrix Kernel



รูปที่ 6 เปรียบเทียบ Matrix * Matrix Kernel ตามตำแหน่ง

5. สำหรับขอบด้านบนของ Output Matrix นั้น ในการหาค่าของมันจะต้องทำการเพิ่ม ค่าของ Matrix เข้าไปนั่นก็คือ 0 โดยเติมให้รอบของขนาด Matrix หลังจากนั้นก็จะทำการคูณโดยเริ่มที่ 0 ตัวแรก โดยจะนับเป็นตำแหน่งที่ 1,1



รูปที่ 7 เปรียบเทียบ Matrix * Matrix Kernel ตามตำแหน่ง

ขั้นตอนการคำนวณและ Code โปรแกรมที่ใช้

```
#load image
img = Image.open((resource_path("Peach.jpg")))
#image convert to black/white
imgGray = img.convert('L')
#save image to test_gray.png
imgGray.save('test_gray.png')
#load new image black/white
imgg = Image.open((resource_path("test_gray.png")))
#convert image to Matrix array
numpydata = asarray(imgg)
#convert Matrix array to Matrix
m = np.matrix(numpydata)
#Display Matrix
print("Matrix Form Original Image")
print(m)
```

รูปที่ 8 Code โปรแกรมสำหรับโหลดภาพเข้ามาในโปรแกรม

ในการ Processing ด้วย Convolution นั้นจะต้องทำการเปลี่ยนสีของภาพให้เป็นสีขาวดำ เพื่อในการคำนวณค่า Matrix ของภาพนั้นจะได้ตรงและไม่ผิดเพี้ยนจนเกินไปโดยได้ใช้โปรแกรม VS-Code ในการเขียนโปรแกรมคำนวณด้วยภาษา Python โดยจะทำการ Processing ทั้งหมด 3 แบบ

```
# Gaussian Blur
gaussian = (1 / 16.0) * np.array([[1., 2., 1.],
                                   [2., 4., 2.],
                                   [1., 2., 1.]])

# Box blur
box = (1 / 9.0) * np.array([[1, 1, 1],
                             [1, 1, 1],
                             [1, 1, 1]])

# Identity
identity = np.array([[0, 0, 0],
                     [0, 1, 0],
                     [0, 0, 0]])
```

รูปที่ 9 Code โปรแกรมของ Matrix Kernel

1. Matrix Box Blur

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

รูปที่ 10 Matrix Box Blur

Matrix Box Blur นั้นจะต้องทำการคูณ $1/9$ กับ Matrix เข้าด้วยกันก่อนนำไปทำการ Image processing

2. Matrix Identity Blur

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

รูปที่ 11 Matrix Identity Blur

Matrix Identity Blur นั้นจะเป็น Matrix Identity และไม่ต้องทำการหาค่า Matrix สามารถนำไปทำการ Image processing ได้เลย

3. Matrix Gaussian Blur

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

รูปที่ 12 Matrix Gaussian Blur

Matrix Gaussian Blur ต้องหาค่าเหมือนกันกับ Matrix Box Blur โดยการนำ $1/6$ ไปคูณกับ Matrix เข้าด้วยกันก่อนนำไป Image processing

Matrix ที่ใช้ ที่มา [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

ขั้นตอนการทำงานโปรแกรมมีดังนี้

1. เลือกรูปภาพที่ต้องการ
2. ทำการ Run โปรแกรมเพื่อเปลี่ยนค่าสีของภาพเป็นขาวดำและแสดงผล Matrix หลังจากนั้นโปรแกรมจะทำการ Convert Matrix เข้าไปในรูปโดยรับค่ามา คือ Matrix Kernel ที่ใช้ และจำนวนรอบจะวนลูปตามจำนวนรอบที่ใช้ หลังจากนั้นจะรวม Matrix เข้าเป็น Matrix แล้วแปลงกลับมาเป็นรูป และในส่วนของ Matrix ที่ขาดไปจะเติมด้วย 0

```
#ฟังก์ชันการ convert Matrix เข้าไปในรูป โดยรับค่ามา คือ Matrix ของรูป Matrix Kernel ที่ใช้ และจำนวนรอบ
def multi_convolver(image, kernel, iterations):
    #วนลูปตามจำนวนรอบที่ใช้
    for i in range(iterations):
        #ใช้ฟังก์ชัน convolve2d เพื่อรวมMatrixเข้าเป็น Matrix แล้วแปลงกลับมาเป็นรูป โดยที่จะใช้เลขMatrixเดิมตาม 'same' ที่กำหนดไว้ และในส่วนของ Matrix ที่ขาดไปจะเติมด้วย 0
        image = convolve2d(image, kernel, 'same', boundary = 'fill', fillvalue = 0)
    return image

#ขั้นตอนการการนำMatrixมารวมกัน โดยใช้ฟังก์ชัน convolve2d โดยการนำ Matrix ของรูป และ Matrix ที่เราต้องการจะใช้ และ การคำนวณเกรอบ เข้าไปในฟังก์ชัน
con = multi_convolver(imgg, gaussian, 1)
con3 = multi_convolver(imgg, box, 1)
con4 = multi_convolver(imgg, identity, 1)
```

รูปที่ 13 Code โปรแกรมหลังจากทำการโหลดรูปและทำการ Run โปรแกรม

ภาพ ขาว - ดำ - เทา (Grayscale) จะมีระดับความเข้มของสีคือ 0 - 255 (8 - bit) โดยการแปลงภาพสี RGB มาเป็นภาพ Grayscale นั้นมีการใช้สูตรทางคณิตศาสตร์ดังนี้

$$\text{Gray} = 0.299 * R + 0.587 * G + 0.114 * B$$

Gray = ค่าความเข้มของสีเทาโดยจะมีค่าระหว่าง 0 - 255

R = ค่าความเข้มของสีแดงโดยจะมีค่าระหว่าง 0 - 255

G = ค่าความเข้มของสีเขียวโดยจะมีค่าอยู่ที่ระหว่าง 0 - 255

B = ค่าความเข้มของสีน้ำเงินโดยจะมีค่าระหว่าง 0 - 255

โดยขั้นตอนการคำนวณจะทำได้ทั้ง rows และ columns ของ Matrix



รูปที่ 14 ตัวอย่าง ภาพสี และ Matrix

ตัวอย่าง ภาพสี RGB ที่โปรแกรมทำการแปลงเป็น Matrix แล้ว โดย Matrix นั้นจะเริ่มจากฝั่งซ้ายไปฝั่งขวาตามลูกศร



รูปที่ 15 ตัวอย่าง ภาพขาว - ดำ - เทา และ Matrix

ตัวอย่าง ภาพสี RGB ที่เปลี่ยนเป็น ภาพ ขาว - ดำ - เทา และ Matrix โดยการคำนวณแต่ละ rows และ collums ของ Matrix ตามสูตรคณิตศาสตร์ที่กล่าวไปข้างต้น โดยการนำ $\text{Gray} = 0.299 * R + 0.587 * G + 0.114 * B$

3. แสดงผล Matrix ด้วย โปรแกรม Command Prompt

```
D:\Users\yoesch\Desktop\img>img2mat.py
Matrix Form Original Image
[[155 158 163 ... 172 172 172]
 [154 157 161 ... 172 172 172]
 [152 154 159 ... 172 172 172]
 ...
 [197 198 198 ... 1 1 1]
 [195 197 197 ... 1 1 1]
 [194 196 196 ... 1 1 1]]
Lossy conversion from float64 to uint8. Range [0.0, 254.0625]. Convert image to uint8 prior to saving to suppress this warning.
Lossy conversion from float64 to uint8. Range [0.0, 253.7777777777778]. Convert image to uint8 prior to saving to suppress this warning.
D:\Users\yoesch\Desktop\img>img2mat.py:56: UserWarning: a4.jpg is a low contrast image
  imgsave('a4.jpg', con4)
Lossy conversion from int32 to uint8. Range [0, 255]. Convert image to uint8 prior to saving to suppress this warning.

Result Box Blur :
[[ 69.33333333 105.33333333 108.11111111 ... 114.66666667 114.66666667
  76.44444444]
 [103.33333333 157.         161.         ... 172.         172.
  114.66666667]
 [102.22222222 155.11111111 158.66666667 ... 172.         172.
  114.66666667]
 ...
 [131.55555556 197.55555556 197.44444444 ... 1.         1.
  0.66666667]
 [130.77777778 196.44444444 196.22222222 ... 1.         1.
  0.66666667]
 [ 86.88888889 130.55555556 130.33333333 ... 0.66666667 0.66666667
  0.44444444]]

Result Identity Blur :
[[155 158 163 ... 172 172 172]
 [154 157 161 ... 172 172 172]
 [152 154 159 ... 172 172 172]
 ...
 [197 198 198 ... 1 1 1]
 [195 197 197 ... 1 1 1]
 [194 196 196 ... 1 1 1]]

Result Gaussian Blur :
[[ 87.5625 118.5625 121.8125 ... 129.         129.         96.75 ]
 [116.         156.9375 161.0625 ... 172.         172.         129. ]
 [114.6875 154.875 158.6875 ... 172.         172.         129. ]
 ...
 [147.875 197.6875 197.625 ... 1.         1.         0.75 ]
 [146.875 196.5625 196.375 ... 1.         1.         0.75 ]
 [109.6875 146.875 146.6875 ... 0.75 0.75 0.5625]]
```

รูปที่ 16 Matrix ที่แสดงผลใน Command Prompt

4. หลังจากนั้นโปรแกรมจะทำการคำนวณ Matrix * Matrix Kernel เข้าด้วยกัน

G20												
	A	B	C	D	E	F	G	H	I	J	K	L
1	Matrix ที่ใช้					Matrix Original จากรูป						
2	ที่มา : https://en.wikipedia.org/wiki/Kernel_(image_processing)											
3	Box Blur :					155	158	163	..	172	172	172
4	1/9	1/9	1/9			154	157	161	..	172	172	172
5	1/9	1/9	1/9			152	154	159	..	172	172	172
6	1/9	1/9	1/9		
7						197	198	198	..	1	1	1
8	Identity :					195	197	197	..	1	1	1
9	0	0	0			194	198	198	..	1	1	1
10	0	1	0									
11	0	0	0									
12												
13	Gaussian :											
14	1/16	1	2	1								
15		2	4	2								
16		1	2	1								
17												

รูปที่ 17 ตัวอย่างการคำนวณ Matrix * Matrix Kernel ด้วยโปรแกรม Excel

จากรูปที่ 17 จะทำการนำ Matrix Original ที่ได้จากการเปลี่ยนค่าสีของภาพที่อยู่ด้านขวา (ตารางสีฟ้า) นำมาคูณกับ Matrix Kernel (ตารางด้านซ้าย) ตามตำแหน่งของ rows และ columns (รูปที่ 4 และ รูปที่ 7) ของ Matrix Kernel ทั้งสามแบบ เช่น Matrix Original * Box Blur , Matrix Original * Identity Blur และ Matrix Original * Gaussian Blur เป็นต้น Matrix ที่ขาดไปจะเติมด้วย 0 รอบ Matrix Original และ สัญลักษณ์ ที่คั่นอยู่ใน Matrix นั้นคือค่าประมาณของผลลัพธ์ตัวเลข Matrix โดยจะใกล้เคียง rows และ columns ที่อยู่ใกล้กัน

5. เมื่อทำการคำนวณ Matrix Original * Matrix Kernel ครบจำนวนตามตำแหน่งของ Matrix ใหม่ของแต่ละ Kernel Matrix

Matrix Original จากรูป				Result Matrix				ตรงที่มี คำนวณได้เลขประมาณกับผลลัพธ์ของรูป			
g)				Box Blur :							
155	158	163	...	69.33333333	105.3333333	107.2222222	...	112.7777778	114.6666667	76.4444444	...
154	157	161	...	103.3333333	157	159.4444444	...	173.1111111	172	114.6666667	...
152	154	159	...	102.5555556	155.7777778	158.6666667	...	172	165.3333333	115.3333333	...
...
197	198	198	...	131.3333333	197.2222222	197.6666667	...	1	1	0.666666667	...
195	197	197	...	130.7777778	196.4444444	197	...	1	1	0.666666667	...
194	196	196	...	86.88888889	130.5555556	130.7777778	...	0.666666667	0.666666667	0.444444444	...
				Identity :							
				155	158	163	...	172	172	172	...
				154	157	161	...	172	172	172	...
				152	154	159	...	172	172	172	...
			
				197	198	198	...	1	1	1	...
				195	197	197	...	1	1	1	...
				194	196	196	...	1	1	1	...
				Gaussian :							
				87.5625	118.5625	120.875	...	129	129	96.75	...
				116	156.9375	159.875	...	172	172	129	...
				114.9375	155.375	158.3125	...	172	172	129	...
			
				147.5625	197.4375	197.75	...	1	1	0.75	...
				146.875	196.5625	197	...	1	1	0.75	...
				109.6875	146.875	147.25	...	0.75	0.75	0.5625	...

รูปที่ 18 หลังจากคำนวณ Matrix * Matrix Kernel ตามตำแหน่งแล้ว

6. หลังจากคำนวณ Matrix * Matrix Kernel เสร็จแล้วจะทำการแปลงค่า Matrix กลับไปเป็นรูปภาพ แล้วทำการเซฟไว้ในตำแหน่งไฟล์ที่ตั้งไว้ของโปรแกรม

```
#save ภาพเก็บไว้จากผลลัพธ์ที่ออกมาหลังจากการคูณ Matrix เข้าไป
imsave('a1.jpg',con)
imsave('a3.png',con3)
imsave('a4.jpg',con4)
print("")

#แสดงผลเป็น Matrix ออกมา
print("Result Box Blur :")
print(con3)
print("")

print("Result Identity Blur :")
print(con4)
print("")

print("Result Gaussian Blur :")
print(con)
print("")
```

รูปที่ 19 Code โปรแกรมสำหรับแปลงค่าเป็นรูปภาพ

7. ผลลัพธ์ที่ได้จะเป็นรูปภาพ ขาว-ดำ และมีการ Processing ทั้งหมด 3 ครั้ง ยกตัวอย่างเช่น



รูปที่ 20 ภาพของ นาย สุธี สาระพันธ์ ที่ยังไม่ได้ทำการแปลงค่าสี



รูปที่ 21 ภาพของ นาย สุธี สาระพันธ์ หลังจากโปรแกรมทำการเปลี่ยนค่าสีและคำนวณ Matrix แล้ว



รูปที่ 22 ภาพของ นาย สุธิ สารพันธ์ ที่โปรแกรมคำนวณ Matrix Original * Box Blur

จากรูปที่ 22 จะสังเกตได้ว่าหลังจากคำนวณครั้งที่ 1 Matrix Original * Box Blur แล้วนั้น จะมีการเบลอภาพทั้งภาพและสว่างขึ้นเล็กน้อย



รูปที่ 23 ภาพของ นาย สุธิ สารพันธ์ ที่โปรแกรมคำนวณ Matrix Original * Identity Blur

จากรูปที่ 24 จะสังเกตได้ว่าหลังจากโปรแกรมคำนวณครั้งที่ 2 $\text{Matrix Original} * \text{Identity Blur}$ แล้วนั้น จะมีการเบลอภาพและจะชัดแค่ตรงกลางของรูปเป็นวงกลม และ ภาพสว่างขึ้นมา มากกว่า Box blur เล็กน้อย



รูปที่ 25 ภาพของ นาย สุธิ สารพันธ์ ที่โปรแกรมคำนวณ $\text{Matrix Original} * \text{Gaussian Blur}$

จากรูปที่ 25 จะสังเกตได้ว่าหลังจากโปรแกรมทำการคำนวณครั้งที่ 3 $\text{Matrix Original} * \text{Gaussian Blur}$ แล้วนั้น จะมีการเบลอภาพทั้งภาพ และสีของภาพยังคงเท่าเดิมไม่สว่างขึ้นหรือลดลง

วิธีการคำนวณ Matrix Original * Matrix Kernel

สมการที่ใช้จะเป็นสมการ Symmetric convolution

โดยสมการนี้เป็นการคำนวณด้วย kernel โดยการนำ Matrix * Matrix Kernel และการคำนวณแบบนี้ไม่ใช้การคำนวณแบบปกติถึงแม้ว่าจะใช้สัญลักษณ์ * เหมือนกัน ยกตัวอย่างตามหัวข้อต่างๆที่อยู่ด้านล่างต่อไปนี้

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \right) [2,2] = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9).$$

รูปที่ 24 ตัวอย่างสมการ Symmetric convolution

1. นำ Matrix Original และ Matrix Kernel แต่ละ Matrix มาคูณกันเช่น

Matrix Original * Box blur

Matrix Original จากรูป						
155	158	163	...	172	172	172
154	157	161	...	172	172	172
152	154	159	...	172	172	172
...
197	198	198	...	1	1	1
195	197	197	...	1	1	1
194	196	196	...	1	1	1

Box Blur :		
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

รูปที่ 26 ภาพยกตัวอย่าง Matrix Original * Matrix Box blur

เพื่อที่จะให้ได้ผลลัพธ์ ของ Matrix ออกมาเป็นขนาด 3*3 จำนวน 4 ชุด จะต้องเติมเลข 0 ที่รอบนอกของตาราง Matrix และ สัญลักษณ์ เป็นเลขนั้นคือค่าประมาณของผลลัพธ์ตัวเลข Matrix จะใกล้เคียงกับ rows และ collums ที่ติดกัน

0	0	0	0	0	0	0	0	0
0	155	158	163	172	172	172	0
0	154	157	161	172	172	172	0
0	152	154	159	172	172	172	0
0	0
0	197	198	198	1	1	1	0
0	195	197	197	1	1	1	0
0	194	196	196	1	1	1	0
0	0	0	0	0	0	0	0	0

ตารางที่ 1 ตัวอย่าง Matrix Original ที่เติม 0 รอบ Matrix แล้ว

2.ทำการคูณกันโดยการนำ Matrix Original (3*3) * Matrix Box blur (3*3)

0	0	0	0	0	0	0	0	0
0	155	158	163	172	172	172	0
0	154	157	161	172	172	172	0
0	152	154	159	172	172	172	0
0	0
0	197	198	198	1	1	1	0
0	195	197	197	1	1	1	0
0	194	196	196	1	1	1	0
0	0	0	0	0	0	0	0	0

Box Blur :		
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

รูปที่ 27 ภาพยกตัวอย่าง Matrix Original * Matrix Box blur

โดยการทำ Matrix 3*3 นั้นจะเริ่มจาก rows และ collums ที่ 1,1 ตามกรอบสี่เหลี่ยม โดย จะทำการเติมเลข 0 ไปในตำแหน่ง Matrix ที่อยู่นอกกรอบ เพื่อให้เป็น Matrix 3*3

3.นำมาคูณกับ Matrix Box blur ตามตำแหน่ง rows และ collums

0	0	0	0	0	0	0	0	0
0	155	158	163	172	172	172	0
0	154	157	161	172	172	172	0
0	152	154	159	172	172	172	0
0	0
0	197	198	198	1	1	1	0
0	195	197	197	1	1	1	0
0	194	196	196	1	1	1	0
0	0	0	0	0	0	0	0	0

Box Blur :		
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

รูปที่ 28 ภาพยกตัวอย่าง Matrix Original * Matrix Box blur แต่ละตำแหน่ง

จากรูปที่ 28 นำตำแหน่ง rows และ collums ของ Matrix Original ที่ทำเป็นขนาด 3*3 มาคูณกับตำแหน่งที่ 1,1 ของ Matrix Box blur (กรอบสี่เหลี่ยม) และ นำมาบวกกันผลลัพธ์ที่ได้จะเท่ากับตำแหน่ง rows และ collums ตำแหน่งที่ 1,1 (ตัวอย่างตามรูปที่ 4 และ รูปที่ 7)


$$= 0*1/9 + 0*1/9 + 0*1/9 + 0*1/9 + 155*1/9 + 158*1/9 + 0*1/9 + 154*1/9 + 157*1/9$$

$$= 69.33333333$$


Box Blur :		
69.33333333	105.3333333	107.2222222
103.3333333	157	159.4444444
102.5555556	155.7777778	158.6666667

รูปที่ 29 ผลลัพธ์ rows และ collums ตัวแรกที่ได้

4.ทำการคูณตามสูตรจาก รูปที่ 15 ไปเรื่อยๆโดยการขยับ Matrix ออกมาทีละช่อง ดังนี้




0	0	0	0	0	0	0	0	0
0	155	158	163	172	172	172	0
0	154	157	161	172	172	172	0
0	152	154	159	172	172	172	0
0	0
0	197	198	198	1	1	1	0
0	195	197	197	1	1	1	0
0	194	196	196	1	1	1	0
0	0	0	0	0	0	0	0	0




Box Blur :		
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

รูปที่ 30 ตัวอย่าง Matrix Original * Matrix Box blur ที่เลื่อนตำแหน่งไปด้านขวา 1 ช่อง

จากรูปที่ 30 จะเป็นการขยับ Matrix (3*3) ที่ละ 1 rows และ collums ไปทางด้านขวา และทำการคำนวณแบบเดียวกันกับ ข้อที่ 3 และผลลัพธ์ที่ได้จะเท่ากับตำแหน่ง 1,2 , 1,3 ตามลำดับ เมื่อคำนวณจนถึงตำแหน่งที่ 1,3 ของ Matrix Box blur แล้ว ก็จะวนลูปกลับไปเริ่มใหม่ โดยการขยับ collums ลงมา 1 collums ทั้ง Matrix Original และ Matrix Box blur โดยเริ่มที่ ตำแหน่ง 2,1 ตามรูปที่ 31



0	0	0	0	0	0	0	0	0
0	155	158	163	172	172	172	0
0	154	157	161	172	172	172	0
0	152	154	159	172	172	172	0
0	0
0	197	198	198	1	1	1	0
0	195	197	197	1	1	1	0
0	194	196	196	1	1	1	0
0	0	0	0	0	0	0	0	0



Box Blur :		
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

รูปที่ 31 ตัวอย่าง Matrix Original * Matrix Box blur ที่เลื่อนตำแหน่งลงมา 1 collums

เมื่อขยับ rows และ collums ลงมาแล้วก็จะทำการคำนวณแบบเดียวกับ **ข้อที่ 3** โดยการนำ Matrix (3*3) ที่เริ่มด้วยตำแหน่งที่ 2,1 มาคูณกับ Matrix Box blur ตำแหน่งที่ 2,1 หลังจากนั้นจะขยับไป 1 rows และ collums ไปทางด้านขวา (**รูปที่ 3**) และทำการคำนวณ

0	0	0	0	0	0	0	0	0
0	155	158	163	172	172	172	0
0	154	157	161	172	172	172	0
0	152	154	159	172	172	172	0
0	0
0	197	198	198	1	1	1	0
0	195	197	197	1	1	1	0
0	194	196	196	1	1	1	0
0	0	0	0	0	0	0	0	0

Box Blur :		
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

รูปที่ 32 ตัวอย่าง Matrix Original * Matrix Box blur ที่เลื่อนตำแหน่งแล้ว

เมื่อคำนวณถึงตำแหน่งที่ 2,3 จะวนลูปและเริ่มต้นใหม่ที่ตำแหน่ง 3,1 ทั้ง Matrix Original และ Matrix Box blur จะทำการคำนวณแบบนี้ไปเรื่อยๆจนได้ตำแหน่ง Matrix (3*3) ครบทั้ง 4 ชุด (สามารถดูตัวอย่างการคูณได้ตาม **รูปที่ 4** และ **รูปที่ 7**)

5.เมื่อทำการคำนวณ Matrix Original * Matrix Box blur ครบตาจำนวน rows และ collums ของ Matrix ทั้ง 4 ชุด จะได้ตำแหน่งตาม **รูปที่ 33**

Box Blur :					
89.33333333	105.3333333	107.2222222	112.7777778	114.6666667	78.44444444
103.3333333	157	159.4444444	173.1111111	172	114.6666667
102.5555556	155.7777778	156.6666667	172	165.3333333	115.3333333
131.3333333	197.2222222	197.6666667	1	1	0.666666667
130.7777778	196.4444444	197	1	1	0.666666667
86.66666667	130.5555556	130.7777778	0.666666667	0.666666667	0.4444444444

รูปที่ 33 ตำแหน่งของ Matrix Original * Matrix Box blur ที่ทำการคำนวณครบแล้ว

สรุป

- ในการแปลงรูปให้กลายเป็น Matrix นั้นสามารถทำได้ด้วยการเขียนโปรแกรมหลังจากนั้น โปรแกรมจะทำการแปลงรูปให้กลายเป็น Metrix จากตัวอย่างที่ได้อธิบายไปจะเป็นรูป ขาว-ดำ (Gray Scale) ซึ่งจะมีค่า Matrix ที่แตกต่างจาก รูปสี และเมื่อคำนวณออกมาค่าที่ได้จะไม่ผิดพลาดไปมาก
- การแปลงภาพให้เป็นขาวดำเป็นกระบวนการแปลงภาพสีให้มีการแสดงผลได้แค่ 2 ระดับ คือ ขาว และ ดำ โดยจะแปลง ข้อมูลภาพให้เป็นภาพ binary (Binary Image) มีกระบวนการแปลงภาพที่มีความเข้มหลายระดับ (Multilevel Image) ให้เป็นภาพที่มีความเข้มเพียง 2 ระดับ หรือ 1 บิต (bit) คือ 0 และ 1 โดย 0 แทน ด้วยจุดที่มีภาพสีขาว และ 1 แทนด้วยจุดที่มีภาพสีดำ
- Thresholding Technique คือการพิจารณาจุด pixel ในภาพว่าจุดใดควรจะเป็นจุดขาว หรือจุด ใดควรจะเป็นจุดที่มีค่าเท่ากับ 1 (จุดดำ) โดยจะทำการเปรียบเทียบค่าของแต่ละ pixel ($f(x,y)$) กับค่าคงที่
- การกรองข้อมูลภาพ (Image Filtering) คือการนำภาพไปผ่านตัวกรองสัญญาณเพื่อให้ได้ ภาพ ผลลัพธ์ออกมา ภาพผลลัพธ์ที่ได้จะมีคุณสมบัติแตกต่างจากภาพเริ่มต้น
- สมการ Symmetric convolution เป็นสมการที่คำนวณด้วย kernel โดยการนำ Matrix * Matrix Kernel และการคำนวณแบบนี้ไม่ใช่การคำนวณแบบปกติถึงแม้ว่าจะใช้สัญลักษณ์ * เหมือนกัน
- Kernel Matrix เป็น Matrix ที่ใช้สำหรับการทำ Image processing จากตัวอย่างใช้ทั้งหมด 3 Kernel Matrix คือ Box blur Matrix , Identity blur Matrix , Gaussian blur Matrix เป็นต้น

สรุปความน่าสนใจของการประยุกต์ใช้งานที่น่าสนใจ

- รูปภาพนั้นสามารถนำมาทำเป็น Matrix ได้
- Matrix สามารถนำมาทำเป็นรูปภาพได้
- สามารถนำรูปภาพและ Matrix มาคำนวณเข้าด้วยกันเพื่อให้ได้รูปภาพใหม่
- ระหว่างรูปภาพที่มีสี และ รูปภาพขาว-ดำ จะมีค่า Matrix ที่แตกต่างกัน
- โปรแกรมภาษา python มี library ที่สามารถนำมาใช้กับ Matrix ได้
- Symmetric convolution เป็นสมการที่นำ Matrix ของรูปภาพและ Matrix มาคำนวณกันเพื่อให้ได้ผลลัพธ์ใหม่

The screenshot shows a Google Sheets spreadsheet titled "Matrix". The spreadsheet is divided into three main sections: "Matrix Original", "Box Blur", and "Gaussian".

- Matrix Original:** A 3x3 matrix of values, all 1/9.
- Box Blur:** A 3x3 matrix of values, all 1/9.
- Gaussian:** A 3x3 matrix of values, all 1/9.
- Result Matrix:** A 3x3 matrix of values, all 1/9.

The spreadsheet also includes a "Box Blur" section with a 3x3 matrix of values, all 1/9. The "Gaussian" section shows the result of a Gaussian blur operation on the original matrix. The spreadsheet also includes a "Result Matrix" section with various numerical values.

การคำนวณ Matrix * Matrix kernel ด้วยโปรแกรม Excel

ลิงค์โปรแกรม Excel =

https://docs.google.com/spreadsheets/d/1MqIbMBwiZOj6_yLksXMgdv9jdnwMDzfeEpMs9v5uhuo/edit?usp=sharing > ใช้ mail สถาบันในการเข้า

ลิงค์ youtube = <https://youtu.be/wBgllfkBVNI>

```

img2matpy X
C:\Users\peech\Desktop>img > img2matpy > box
1  # Import the necessary libraries
2  from PIL import Image
3  from numpy import asarray
4  import numpy as np
5  import os
6  import sys
7  from scipy.signal import convolve2d
8  from skimage.io import imshow, imread, imsave
9
10 def resource_path(relative_path):
11     if hasattr(sys, '_MEIPASS'):
12         return os.path.join(sys._MEIPASS, relative_path)
13     return os.path.join(os.path.abspath('.'), relative_path)
14
15 #load image
16 img = Image.open((resource_path("Peach.jpg")))
17 #image convert to black/white
18 imgGray = img.convert('L')
19 #save image to test_gray.png
20 imgGray.save("test_gray.png")
21 #load new image black/white
22 imgg = Image.open((resource_path("test_gray.png")))
23 #convert image to Matrix array
24 numpydata = asarray(imgg)
25 #convert Matrix array to Matrix
26 m = np.matrix(numpydata)
27 #display Matrix
28 print("Matrix form Original Image")
29 print(m)
30
31 # Gaussian Blur
32 gaussian = (1 / 16.0) * np.array([[1., 2., 1.],
33                                     [2., 4., 2.],
34                                     [1., 2., 1.]])
35 # Box blur
36 box = (1 / 9.0) * np.array([[1, 1, 1],
37                               [1, 1, 1],
38                               [1, 1, 1]])
39
40 # Identity
41 identity = np.array([[0, 0, 0],
42                      [0, 1, 0],
43                      [0, 0, 0]])
44
45 def multi_convolver(image, kernel, iterations):
46     for i in range(iterations):
47         image = convolve2d(image, kernel, 'same', boundary = 'fill', fillvalue = 0)
48     return image
49
50 con = multi_convolver(imgg, gaussian, 2)
51 con3 = multi_convolver(imgg, box, 2)
52 con4 = multi_convolver(imgg, identity, 2)
53
54 imsave('a1.jpg', con)
55 imsave('a3.png', con3)
56 imsave('a4.jpg', con4)
57 print("")
58
59 print("Result Box Blur :")
60 print(con3)
61 print("")
62
63 print("Result Identity Blur :")
64 print(con4)
65 print("")
66
67 print("Result Gaussian Blur :")
68 print(con)
69 print("")
70

```

Code โปรแกรมภาษา Python ในการคำนวณ Matrix * Matrix kernel

อ้างอิง

ทฤษฎีที่เกี่ยวข้องกับการ Image processing

https://research.psu.ac.th/files/res_che2553/resche_files/334_chapter2.pdf

พื้นฐานรูปภาพที่จำเป็น

<https://blogwai.com/matlab-4/>

ลักษณะและการคำนวณของ Kernel

[https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

ตัวอย่าง Code โปรแกรมสำหรับการ Image processing

<https://medium.com/swlh/image-processing-with-python-convolutional-filters-and-kernels-b9884d91a8fd>

<https://towardsdatascience.com/image-processing-with-python-blurring-and-sharpening-for-beginners-3bcebec0583a>

ตัวอย่าง Code โปรแกรม และ ทฤษฎีที่เกี่ยวข้องกับ Convolution

<https://github.com/ashushekar/image-convolution-from-scratch>

<https://www.pyimagesearch.com/2016/07/25/convolutions-with-opencv-and-python/>

ตัวอย่าง Box Blur Algorithm

<https://www.geeksforgeeks.org/box-blur-algorithm-with-python-implementation/>