DTI 516 Multimedia Processing

Chapter: 5

Image Segmentation

Dr. Paween Khoenkaw
Digital Technology Innovation : Maejo University

Image Segmentation

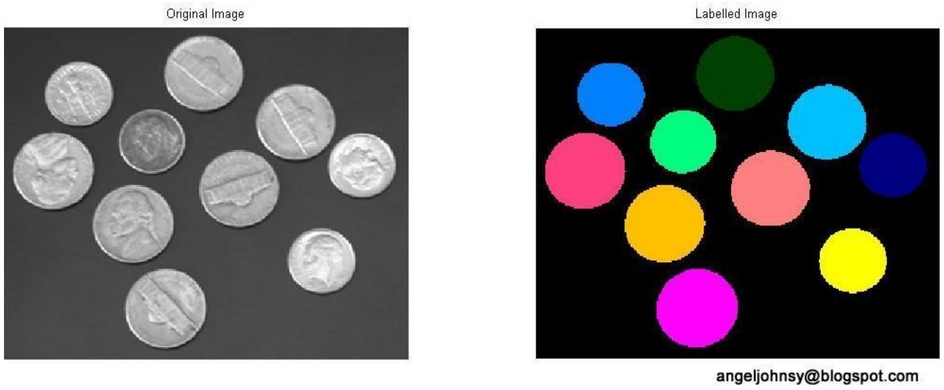
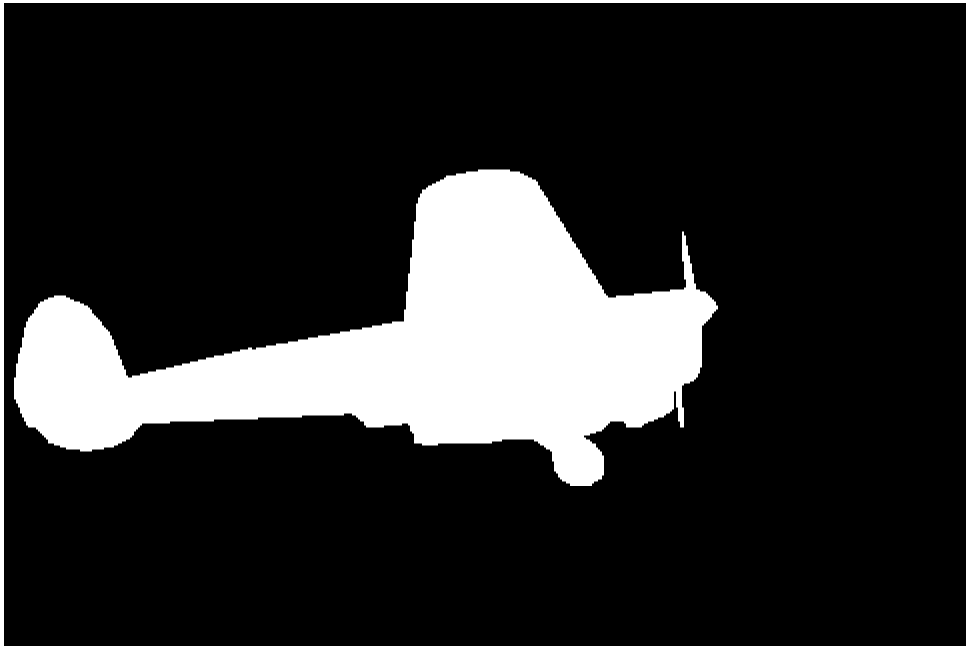


Image Segmentation

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as superpixels).



Thresholding Methods

Simple Threshold Segmentation

Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images

$$I(j) = \begin{cases} 1 \; where \; I(j) \geq \theta \\ 0 \; where \; I(j) < \theta \end{cases}$$

Thresholding

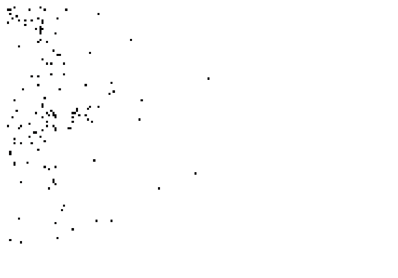
Simple Threshold Segmentation

$$I(j) = \begin{cases} 1 \; where \; I(j) \geq \theta \\ 0 \; where \; I(j) < \theta \end{cases}$$

What is the θ for snow segmentation

Simple Threshold Segmentation

$$I(j) = \begin{cases} 1 \; where \; I(j) \geq \theta \\ 0 \; where \; I(j) < \theta \end{cases}$$



θ=0

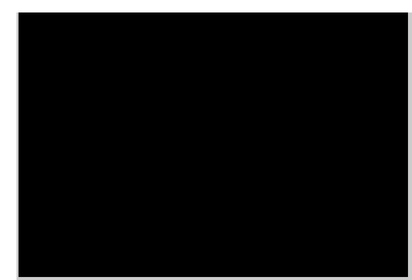
θ=10

θ=50

θ=100

θ=127

θ=200



θ=255

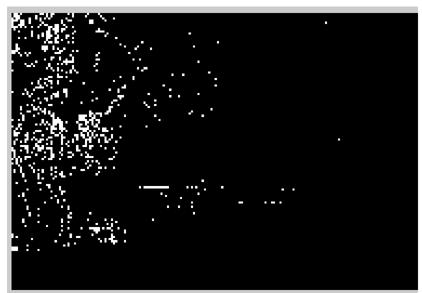
Simple Threshold Segmentation

$$I(j) = \begin{cases} 1\ where\ I(j) \geq \theta \\ 0\ where\ I(j) < \theta \end{cases}$$

What is the θ for trees segmentation

Simple Threshold Segmentation

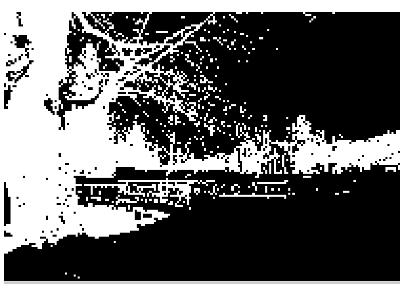
$$I(j) = \begin{cases} 0 \; where \; I(j) \geq \theta \\ 1 \; where \; I(j) < \theta \end{cases}$$



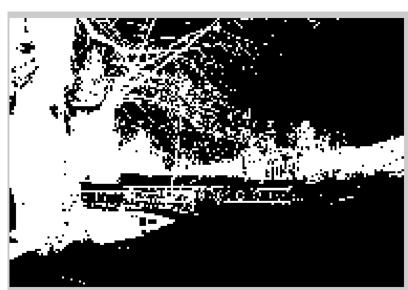
θ=1

θ=10

θ=50



θ=100



θ=127

θ=200

θ=254

Simple Threshold Segmentation

$$I(j) = \begin{cases} 0 \; where \; I(j) \geq \theta \\ 1 \; where \; I(j) < \theta \end{cases}$$

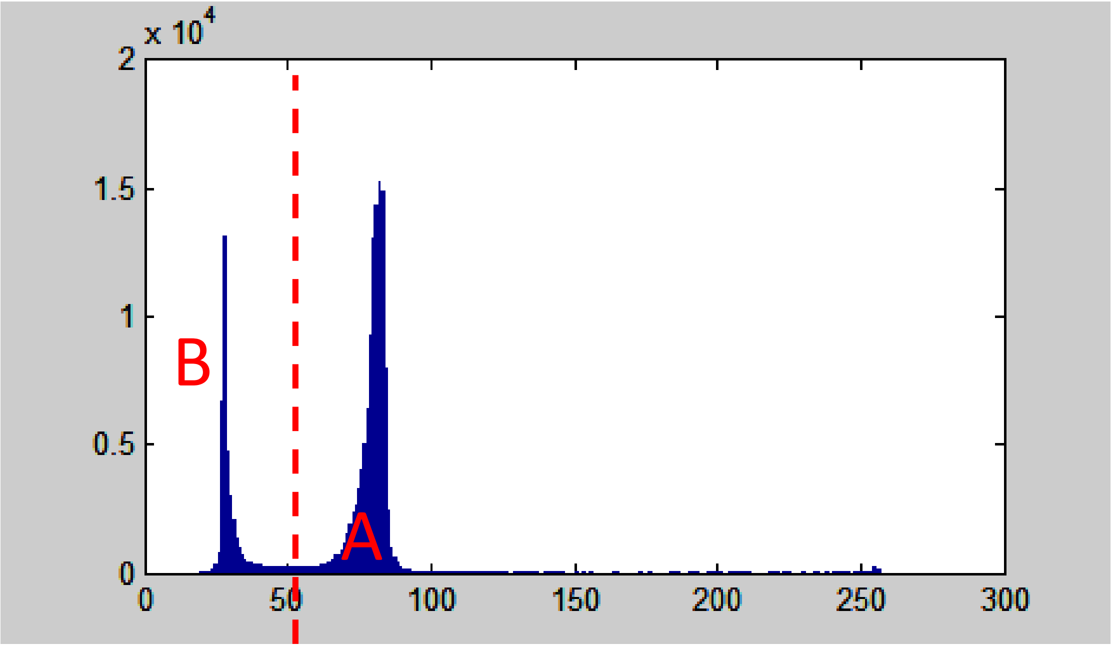
Can we calculate determine θ automatically?

Simple Threshold Segmentation

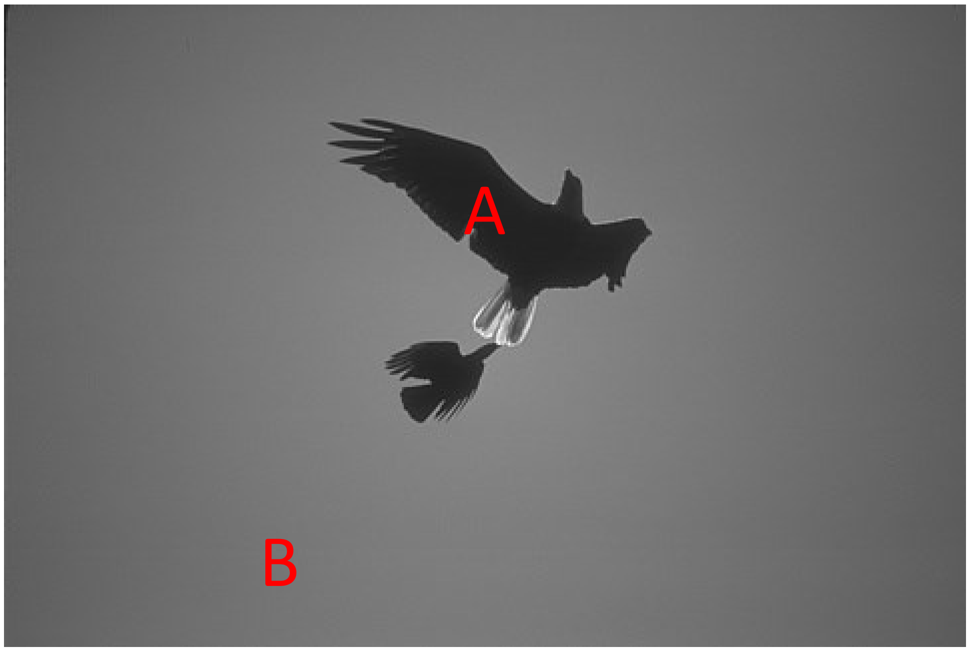
Otsu's method

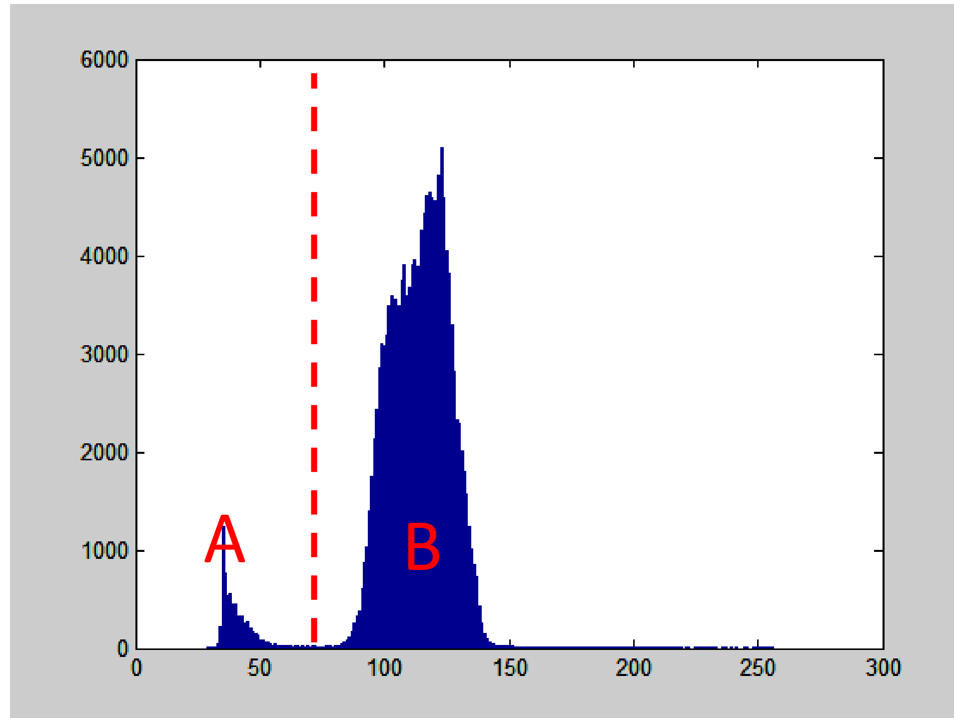
Nobuyuki Otsu (大津展之 *Ōtsu Nobuyuki*)

Otsu's method



Otsu's method

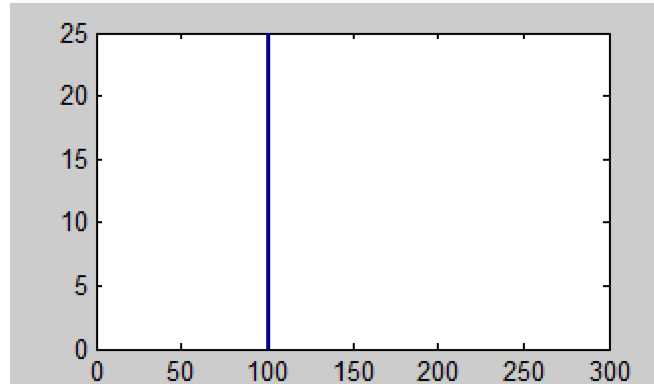




Understanding Population Variance

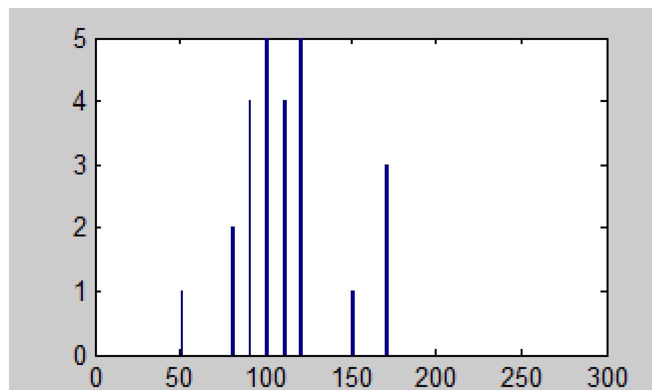
$$\sigma^2 = \frac{\sum(X - \bar{X})}{n}$$

100	**100**	**100**	**100**	**100**
100	**100**	**100**	**100**	**100**
100	**100**	**100**	**100**	**100**
100	**100**	**100**	**100**	**100**
100	**100**	**100**	**100**	**100**



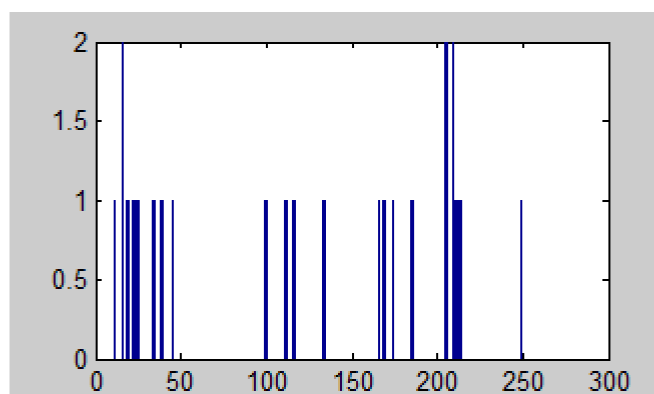
$$\boldsymbol{\sigma^2 = 0}$$

90	**100**	**110**	**80**	**120**
90	**100**	**110**	**120**	**170**
90	**100**	**110**	**120**	**170**
90	**100**	**110**	**120**	**170**
50	**100**	**80**	**120**	**150**



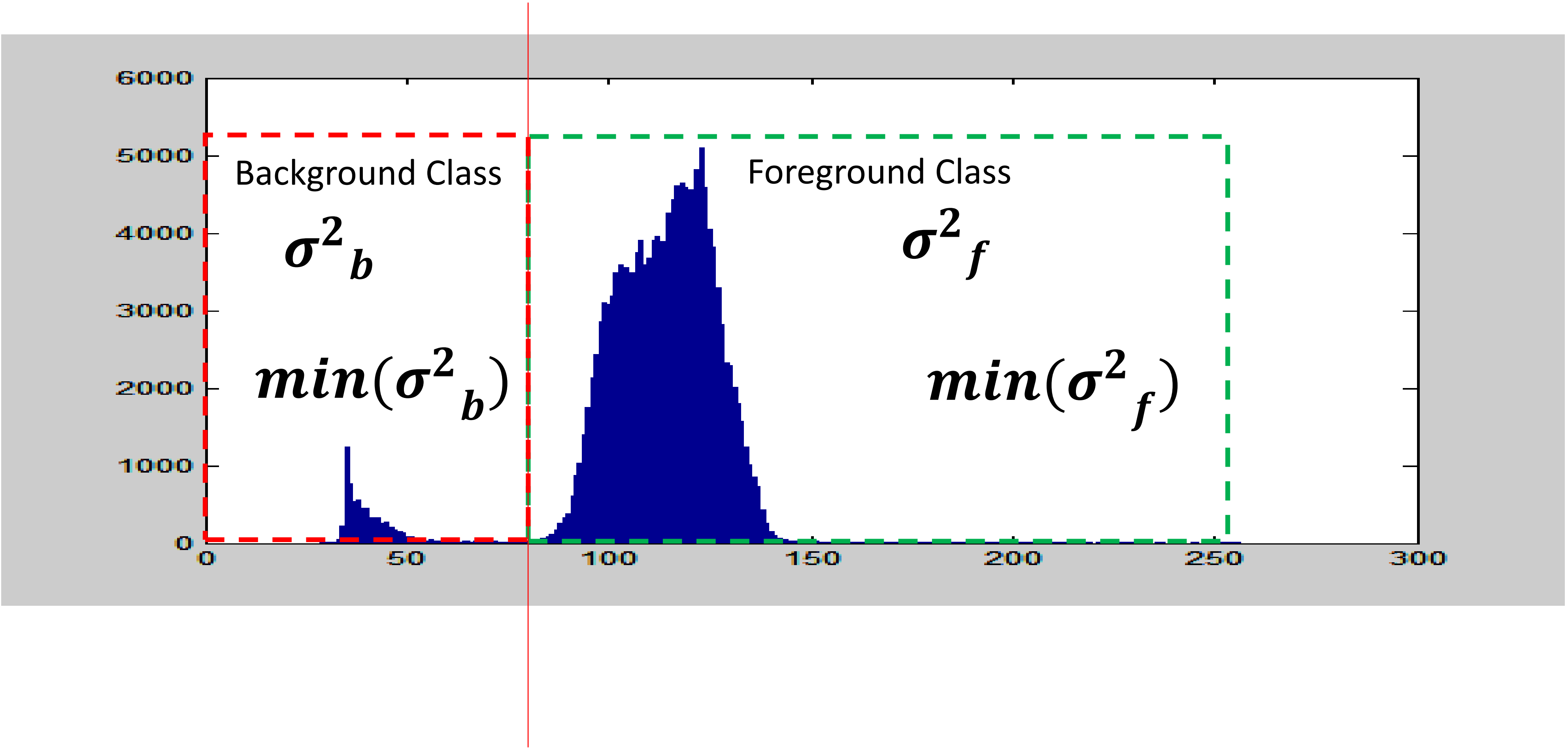
$$\boldsymbol{\sigma^2 = 815.36}$$

15	**24**	**168**	**115**	**44**
173	**208**	**132**	**110**	**99**
10	**208**	**248**	**210**	**212**
18	**184**	**165**	**21**	**204**
133	**38**	**204**	**33**	**15**



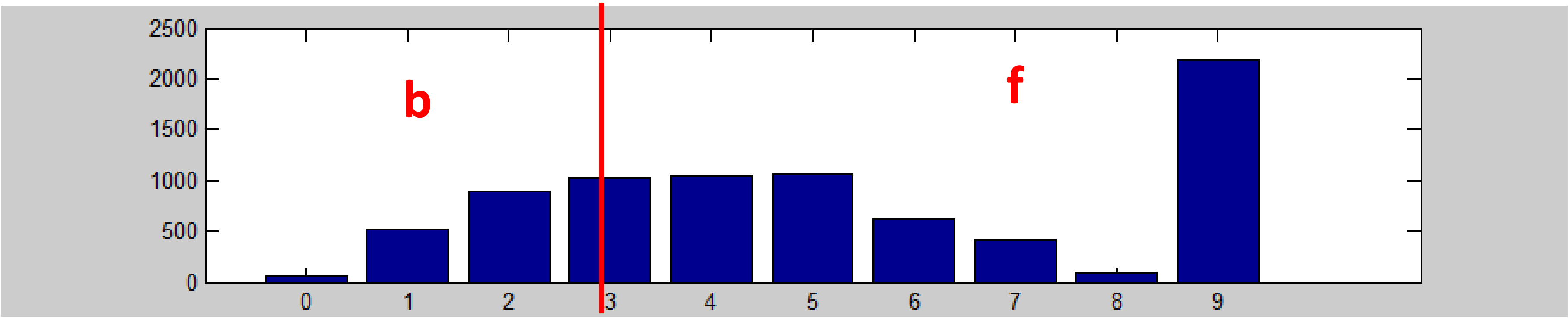
$$\boldsymbol{\sigma^2 = 6331.91}$$

Otsu's method



$min(\sigma^2{}_b + \sigma^2{}_f)$

Within-Class Variance



{67, 515, 898, 1025, 1040, 1057, 623, 413, 100, 2193}

Mean b $\mu_b = \frac{(0\times67)+(1\times515)+(2\times898)+(3\times1025)}{67+515+898+1025} = 2.1501$

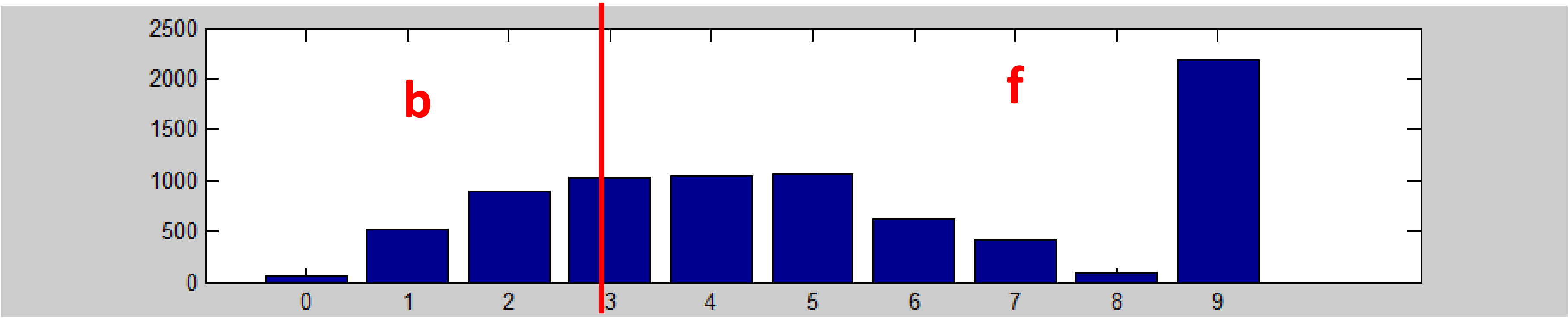
Variance b $\sigma_b^2 = \frac{((0-\mu_b)^2\times 67) + ((1-\mu_b)^2\times 515) + ((2-\mu_b)^2\times 898) + ((3-\mu_b)^2\times 1025)}{67+515+898+1025}$

= 0.6992

Weight b $w_b = \frac{67+515+898+1025}{67+515+898+1025+1040+1057+623+413+100+2193}$

= 0.3158

Within-Class Variance



{67, 515, 898, 1025, 1040, 1057, 623, 413, 100, 2193}

Mean f $$\mu_f = \frac{(4\times1040)+(5\times1057)+(6\times623)+(7\times413)+(8\times100)+(9\times2193)}{1040+1057+623+413+100+2193} = 6.747$$

Variance f $$\sigma_f^2 = \frac{((4-\mu_f)^2\times 1040) + ((5-\mu_f)^2\times 1057) + \ldots\ldots + ((9-\mu_f)^2\times 2193)}{1040+1057+623+413+100+2193}$$

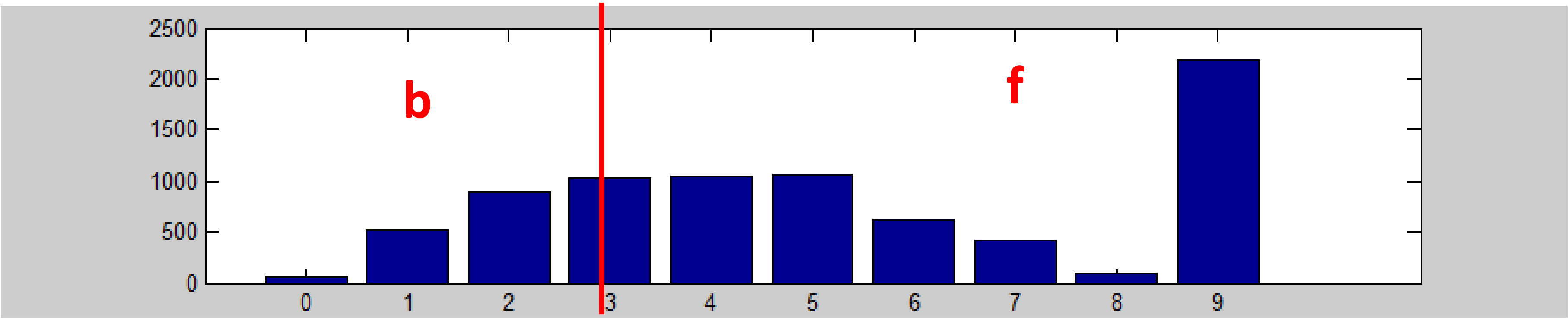
= 4.1903

Weight f $$w_f = \frac{1040+1057+623+413+100+2193}{67+515+898+1025+1040+1057+623+413+100+2193}$$

= 0.6842

Within-Class Variance $$\sigma_w^2 = \left(w_b \times \sigma_b^2\right) + \left(w_f \times \sigma_f^2\right)$$

Within-Class Variance



{67, 515, 898, 1025, 1040, 1057, 623, 413, 100, 2193}

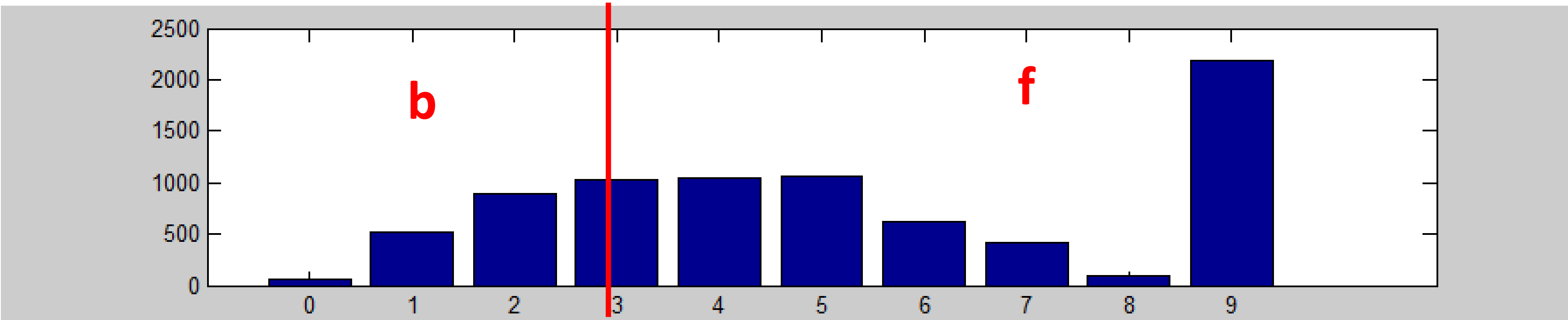
$\mu_b = 2.1501 \quad \sigma_b^2 = 0.6992 \quad w_b = 0.3158$

$\mu_f = 6.747 \quad \sigma_f^2 = 4.1903 \quad w_f = 0.6842$

Within-Class Variance $\quad \sigma_w^2 = \left(w_b \times \sigma_b^2\right) + \left(w_f \times \sigma_f^2\right)$

$$\sigma_w^2 = (0.3158 \times 0.6992) + (0.6842 \times 04.1903) = 3.0878$$

Within-Class Variance



{67, 515, 898, 1025, 1040, 1057, 623, 413, 100, 2193}

$$\mu_b = 2.1501 \qquad \sigma_b^2 = 0.6992 \qquad w_b = 0.3158$$

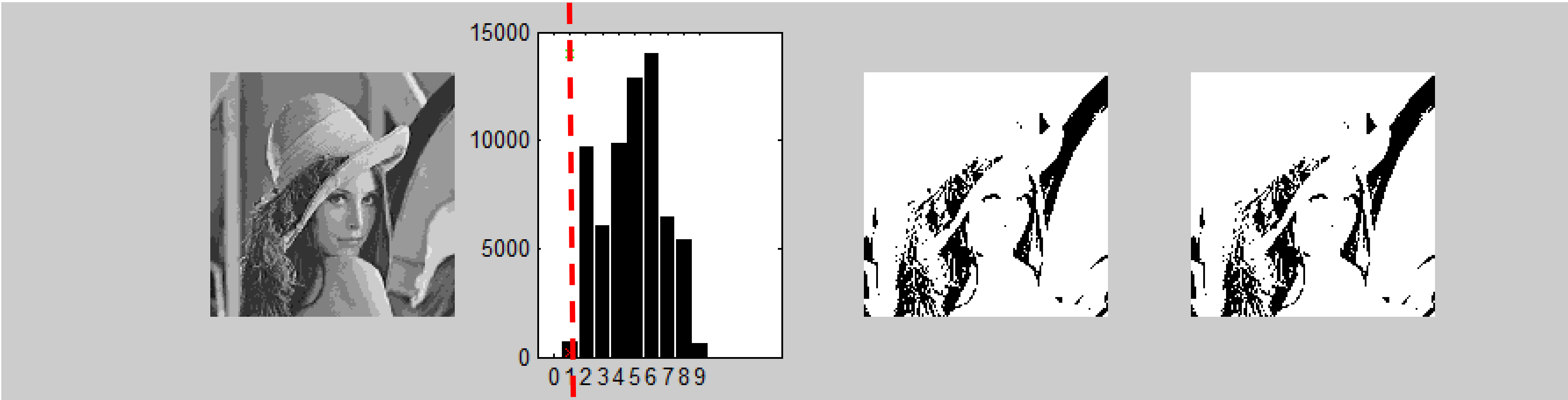
$$\mu_f = 6.747 \qquad \sigma_f^2 = 4.1903 \qquad w_f = 0.6842$$

Within-Class Variance $\quad \sigma_w^2 = \left(w_b \times \sigma_b^2\right) + \left(w_f \times \sigma_f^2\right)$

$$\sigma_w^2 = (0.3158 \times 0.6992) + (0.6842 \times 04.1903) = 3.0878$$

***Otsu algorithm move this, to find the smallest of this

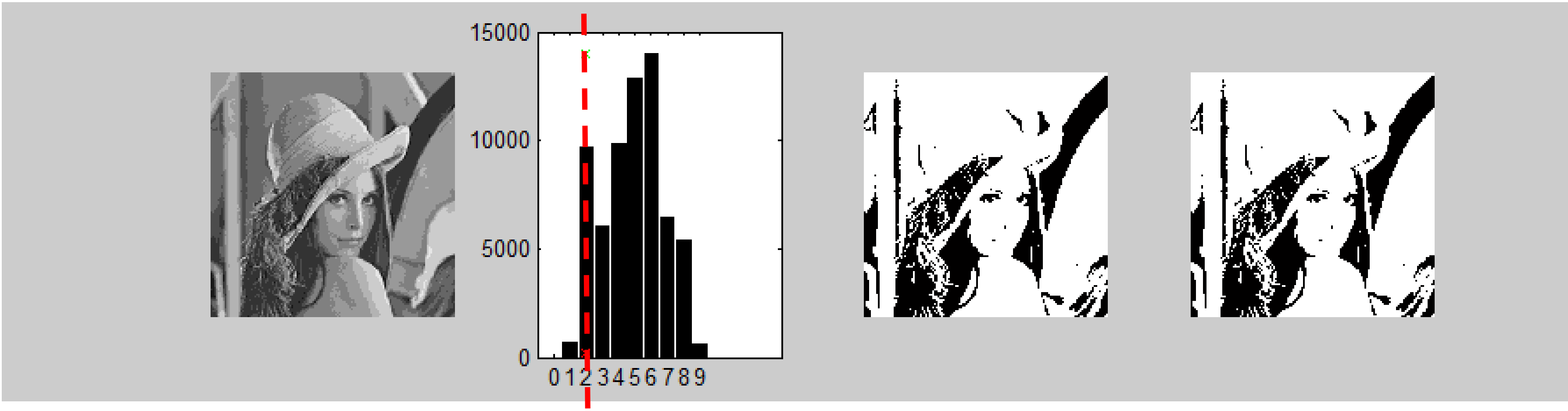
Otsu's method



Current threshold Best threshold

Iteration	μ_b	σ_b^2	w_b	μ_f	σ_f^2	w_f	σ_w^2	Bin
2	1.0000	0	0.0100	4.9127	3.3687	0.9900	3.3350	1
								2
								3
								4
								5
								6
								7
								8

Otsu's method

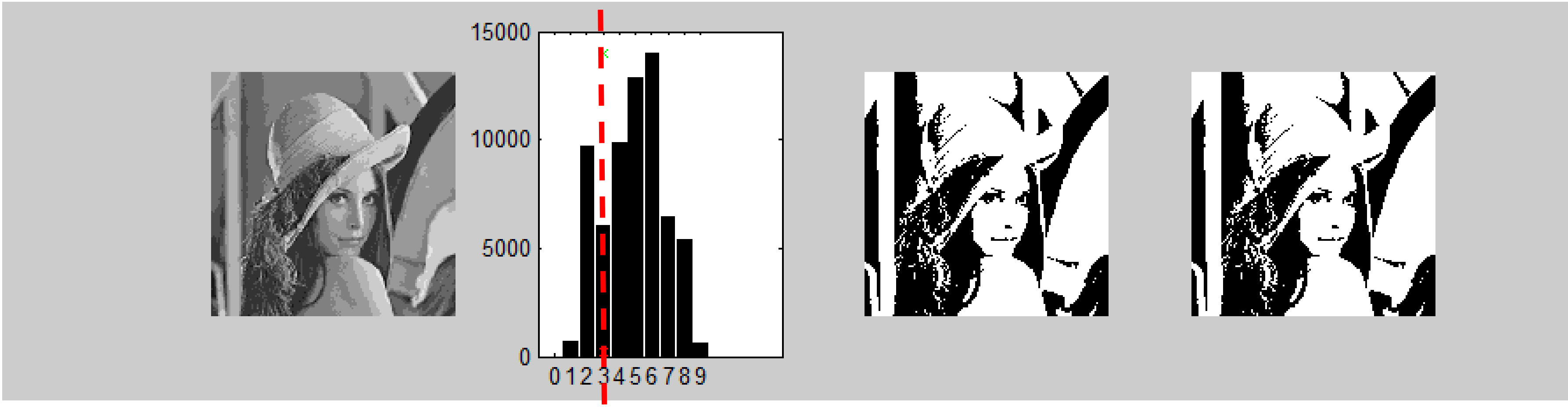


Current threshold

Best threshold

Iteration	μ_b	σ_b^2	w_b	μ_f	σ_f^2	w_f	σ_w^2	Bin
2	1.0000	0	0.0100	4.9127	3.3687	0.9900	3.3350	1
3	1.9368	0.0592	0.1582	5.4253	2.2057	0.8418	1.8662	2
								3
								4
								5
								6
								7
								8

Otsu's method

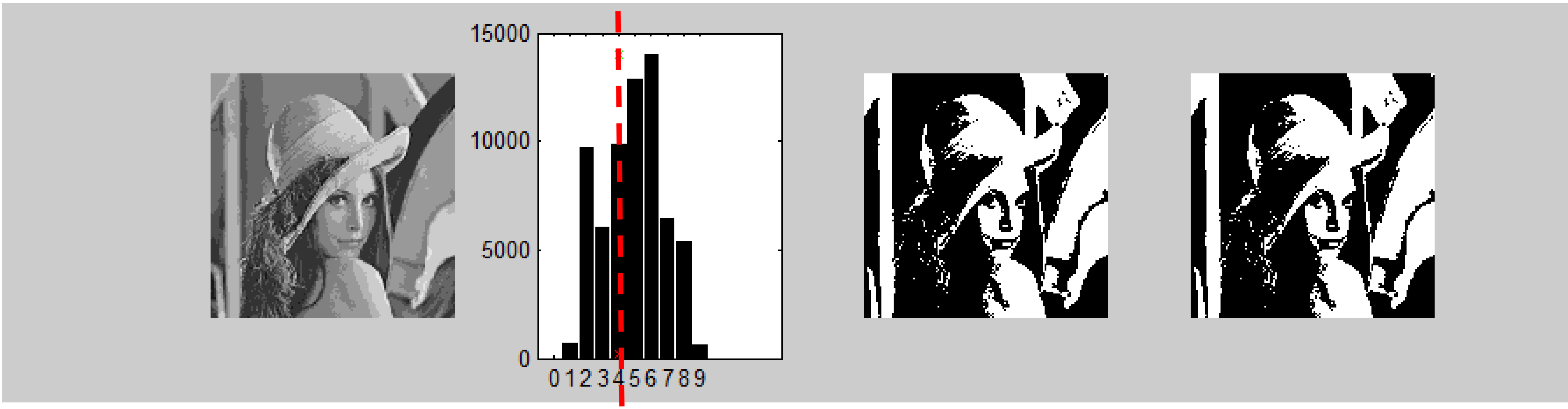


Current threshold

Best threshold

Iteration	μ_b	σ_b^2	w_b	μ_f	σ_f^2	w_f	σ_w^2	Bin
2	1.0000	0	0.0100	4.9127	3.3687	0.9900	3.3350	1
3	1.9368	0.0592	0.1582	5.4253	2.2057	0.8418	1.8662	2
4	2.3289	0.3005	0.2506	5.7244	1.6628	0.7494	1.3214	3
								4
								5
								6
								7
								8

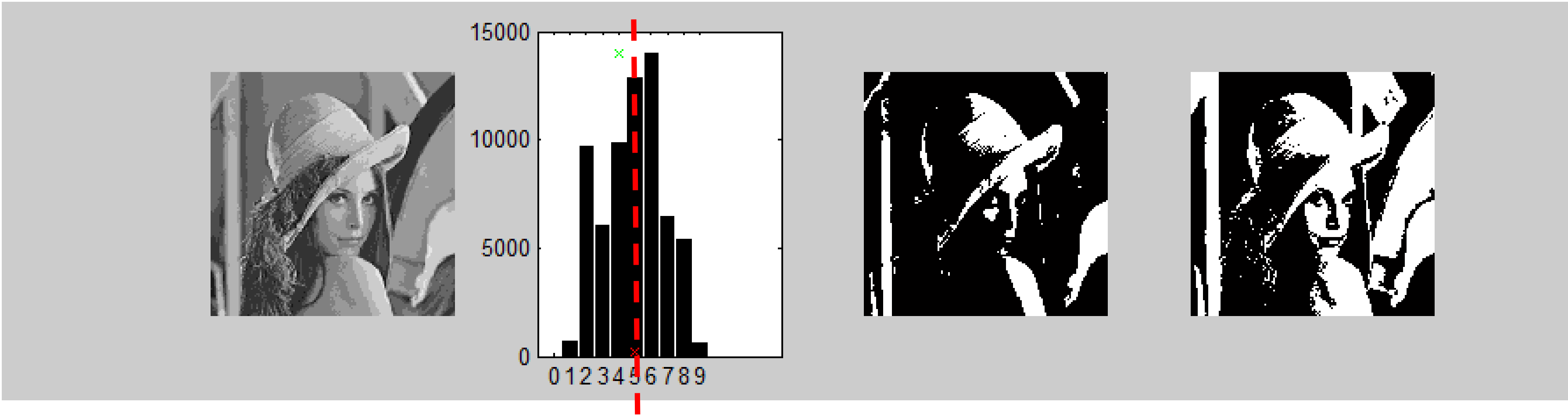
Otsu's method



Current threshold Best threshold

Iteration	μ_b	σ_b^2	w_b	μ_f	σ_f^2	w_f	σ_w^2	Bin
2	1.0000	0	0.0100	4.9127	3.3687	0.9900	3.3350	1
3	1.9368	0.0592	0.1582	5.4253	2.2057	0.8418	1.8662	2
4	2.3289	0.3005	0.2506	5.7244	1.6628	0.7494	1.3214	3
5	2.9543	0.8420	0.4004	6.1554	1.1495	0.5996	1.0263	4
								5
								6
								7
								8

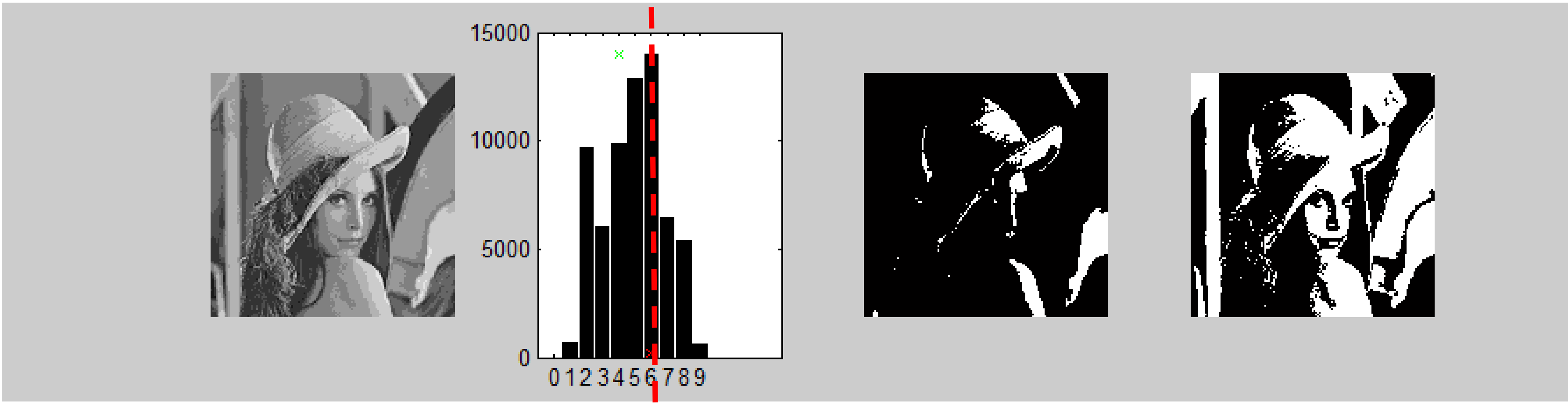
Otsu's method



Current threshold   Best threshold

Iteration	μ_b	σ_b^2	w_b	μ_f	σ_f^2	w_f	σ_w^2	Bin
2	1.0000	0	0.0100	4.9127	3.3687	0.9900	3.3350	1
3	1.9368	0.0592	0.1582	5.4253	2.2057	0.8418	1.8662	2
4	2.3289	0.3005	0.2506	5.7244	1.6628	0.7494	1.3214	3
5	2.9543	0.8420	0.4004	6.1554	1.1495	0.5996	1.0263	4
6	3.6273	1.4888	0.5967	6.7179	0.7428	0.4033	1.1880	5
								6
								7
								8

Otsu's method

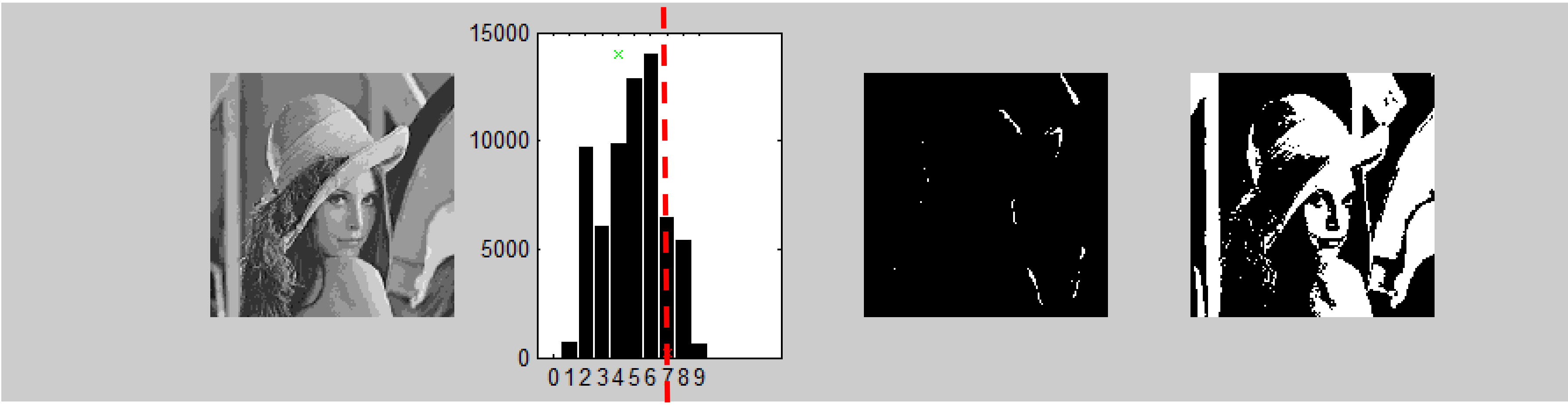


Current threshold

Best threshold

Iteration	μ_b	σ_b^2	w_b	μ_f	σ_f^2	w_f	σ_w^2	Bin
2	1.0000	0	0.0100	4.9127	3.3687	0.9900	3.3350	1
3	1.9368	0.0592	0.1582	5.4253	2.2057	0.8418	1.8662	2
4	2.3289	0.3005	0.2506	5.7244	1.6628	0.7494	1.3214	3
5	2.9543	0.8420	0.4004	6.1554	1.1495	0.5996	1.0263	4
6	3.6273	1.4888	0.5967	6.7179	0.7428	0.4033	1.1880	5
7	4.2535	2.1896	0.8107	7.5291	0.3417	0.1893	1.8397	6
								7
								8

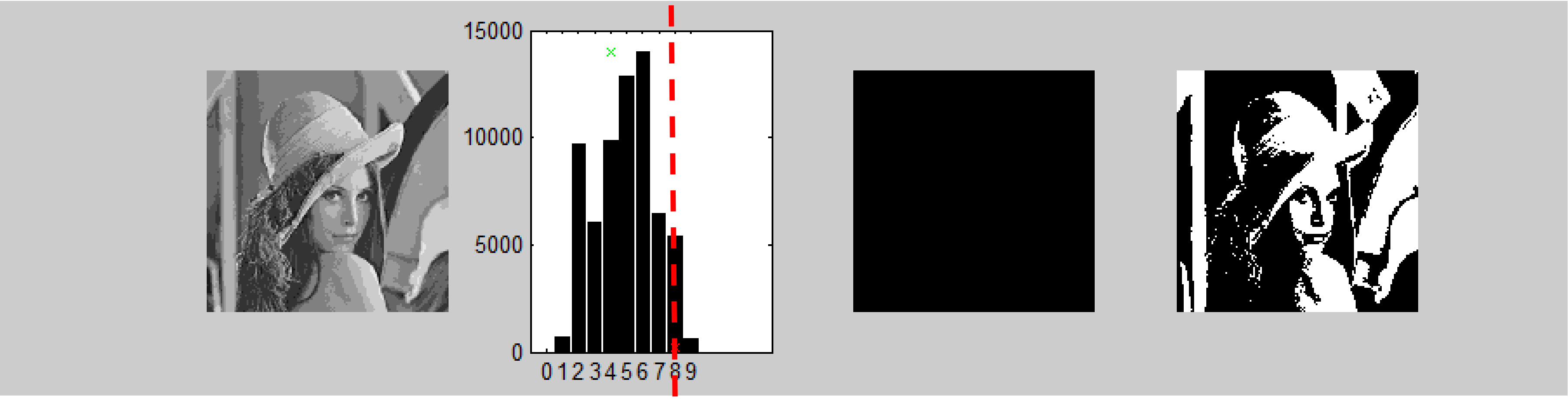
Otsu's method



Current threshold Best threshold

Iteration	μ_b	σ_b^2	w_b	μ_f	σ_f^2	w_f	σ_w^2	Bin
2	1.0000	0	0.0100	4.9127	3.3687	0.9900	3.3350	1
3	1.9368	0.0592	0.1582	5.4253	2.2057	0.8418	1.8662	2
4	2.3289	0.3005	0.2506	5.7244	1.6628	0.7494	1.3214	3
5	2.9543	0.8420	0.4004	6.1554	1.1495	0.5996	1.0263	4
6	3.6273	1.4888	0.5967	6.7179	0.7428	0.4033	1.1880	5
7	4.2535	2.1896	0.8107	7.5291	0.3417	0.1893	1.8397	6
8	4.5494	2.6789	0.9086	8.0958	0.0866	0.0914	2.4419	7
								8

Otsu's method

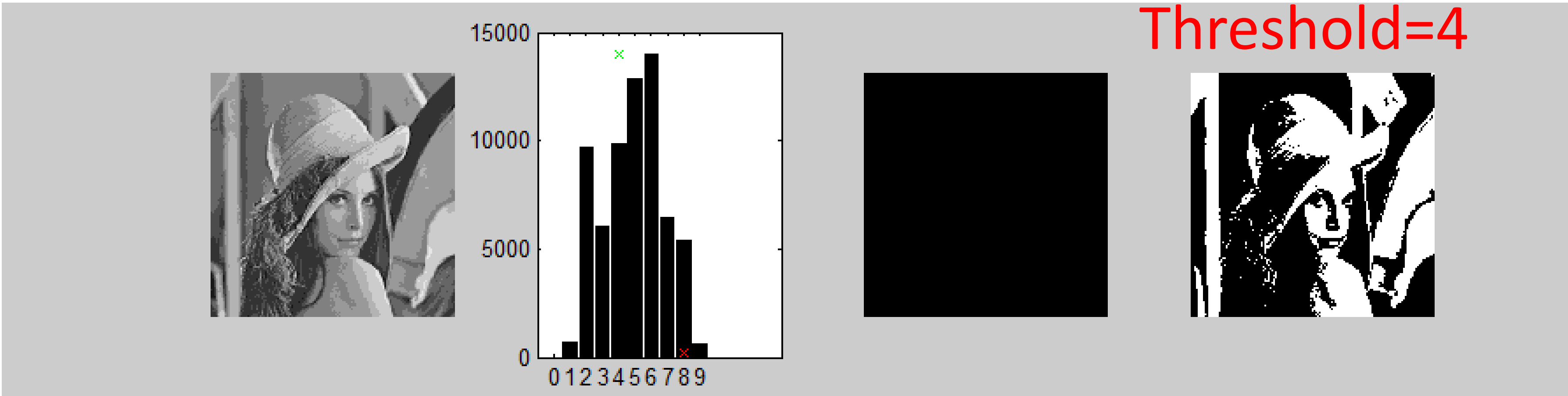


Current threshold

Best threshold

Iteration	μ_b	σ_b^2	w_b	μ_f	σ_f^2	w_f	σ_w^2	Bin
2	1.0000	0	0.0100	4.9127	3.3687	0.9900	3.3350	1
3	1.9368	0.0592	0.1582	5.4253	2.2057	0.8418	1.8662	2
4	2.3289	0.3005	0.2506	5.7244	1.6628	0.7494	1.3214	3
5	2.9543	0.8420	0.4004	6.1554	1.1495	0.5996	1.0263	4
6	3.6273	1.4888	0.5967	6.7179	0.7428	0.4033	1.1880	5
7	4.2535	2.1896	0.8107	7.5291	0.3417	0.1893	1.8397	6
8	4.5494	2.6789	0.9086	8.0958	0.0866	0.0914	2.4419	7
9	4.8371	3.3656	0.9912	9.0000	0	0.0088	3.3361	8

Otsu's method



Current threshold Best threshold

Iteration	μ_b	σ_b^2	w_b	μ_f	σ_f^2	w_f	σ_w^2	Bin
2	1.0000	0	0.0100	4.9127	3.3687	0.9900	3.3350	1
3	1.9368	0.0592	0.1582	5.4253	2.2057	0.8418	1.8662	2
4	2.3289	0.3005	0.2506	5.7244	1.6628	0.7494	1.3214	3
5	2.9543	0.8420	0.4004	6.1554	1.1495	0.5996	1.0263	4
6	3.6273	1.4888	0.5967	6.7179	0.7428	0.4033	1.1880	5
7	4.2535	2.1896	0.8107	7.5291	0.3417	0.1893	1.8397	6
8	4.5494	2.6789	0.9086	8.0958	0.0866	0.0914	2.4419	7
9	4.8371	3.3656	0.9912	9.0000	0	0.0088	3.3361	8

Otsu's method

A Faster Approach

Within-Class Variance $\sigma_w^2 = \left(w_b \times \sigma_b^2\right) + \left(w_f \times \sigma_f^2\right)$

Between Class Variance $\sigma_B^2 = \sigma^2 - \sigma_w^2$

However $\mu = (w_b \times \mu_b) + \left(w_f \times \mu_f\right)$

$$\sigma_B^2 = w_b(\mu_b - \mu)^2 + w_f(\mu_f - \mu)^2$$

$$\sigma_B^2 = w_b w_f(\mu_b - \mu_f)^2$$

$$\sigma_w^2 \propto \frac{1}{\sigma_B^2}$$

Otsu's method

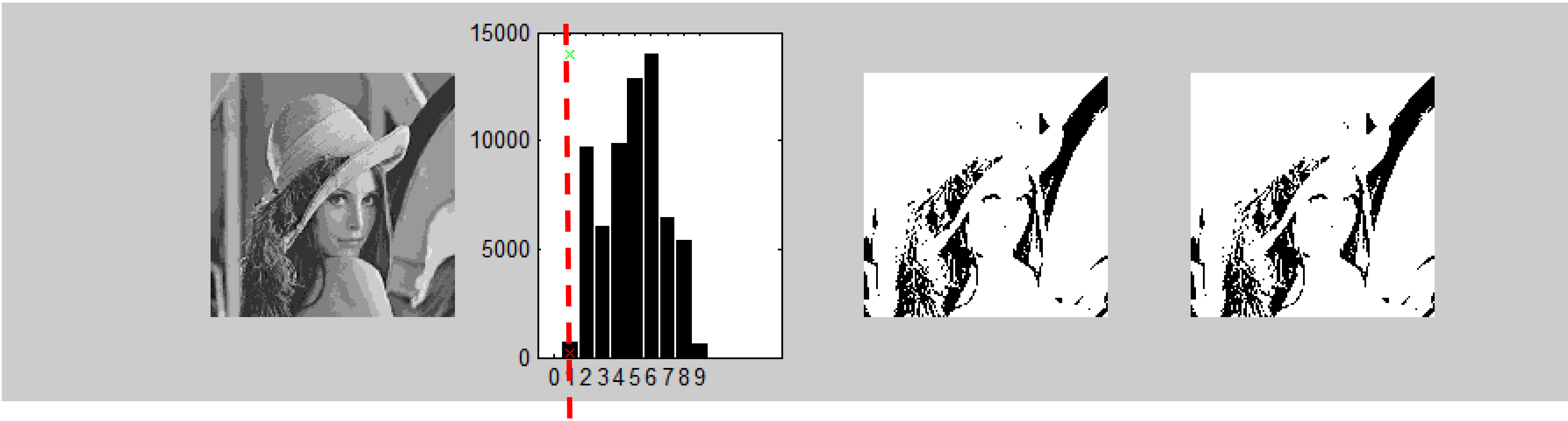
Within-Class Variance $\sigma_w^2 = \left(w_b \times \sigma_b^2\right) + \left(w_f \times \sigma_f^2\right)$

Traditional Otsu move threshold to minimize Within-class Variance

Traditional Otsu move threshold to maximize Between-class Variance

Between Class Variance $\sigma_B^2 = w_b w_f (\mu_b - \mu_f)^2$

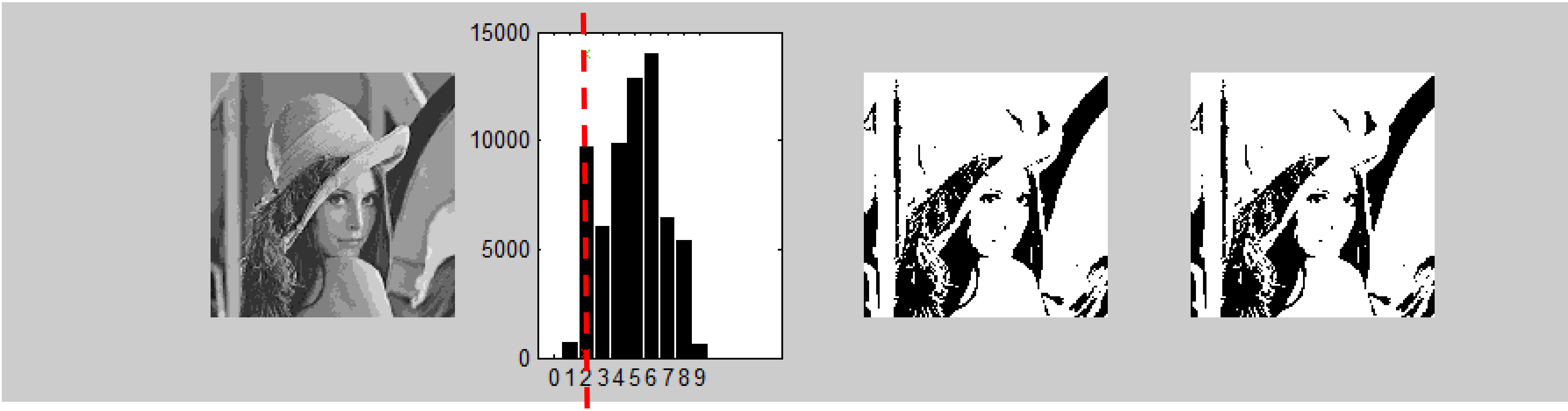
Otsu's method(Fast)



Current threshold   Best threshold

Iteration	μ_b	w_b	μ_f	w_f	σ_w^2	Bin
2	1.0000	0.0100	4.9127	0.9900	0.1515	1
						2
						3
						4
						5
						6
						7
						8

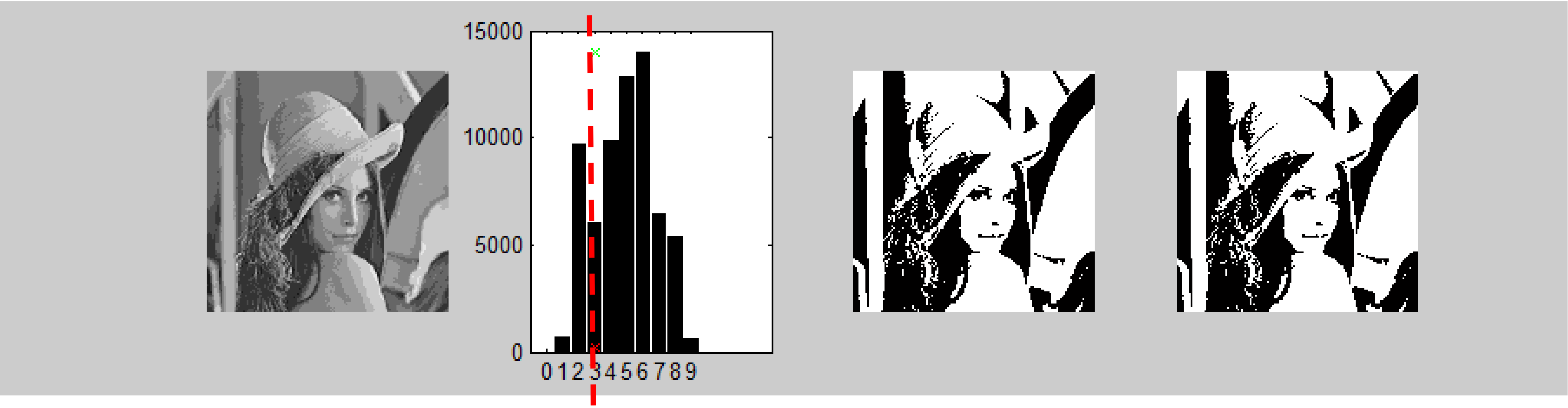
Otsu's method(Fast)



Current threshold Best threshold

Iteration	μ_b	w_b	μ_f	w_f	σ_w^2	Bin
2	1.0000	0.0100	4.9127	0.9900	0.1515	1
3	1.9368	0.1582	5.4253	0.8418	1.6203	2
						3
						4
						5
						6
						7
						8

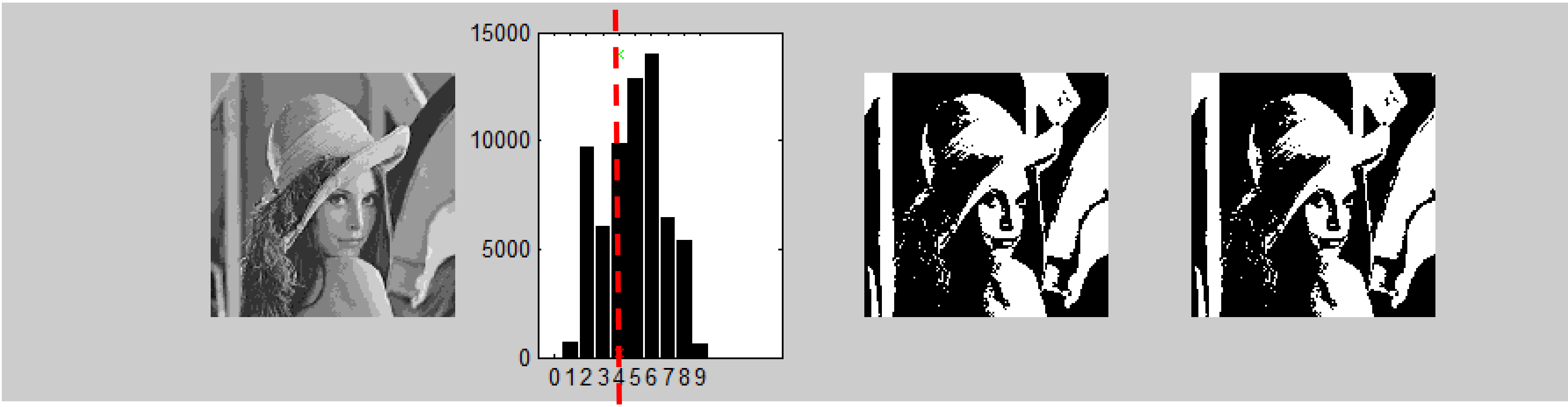
Otsu's method(Fast)



Current threshold   Best threshold

Iteration	μ_b	w_b	μ_f	w_f	σ_w^2	Bin
2	1.0000	0.0100	4.9127	0.9900	0.1515	1
3	1.9368	0.1582	5.4253	0.8418	1.6203	2
4	2.3289	0.2506	5.7244	0.7494	2.1651	3
						4
						5
						6
						7
						8

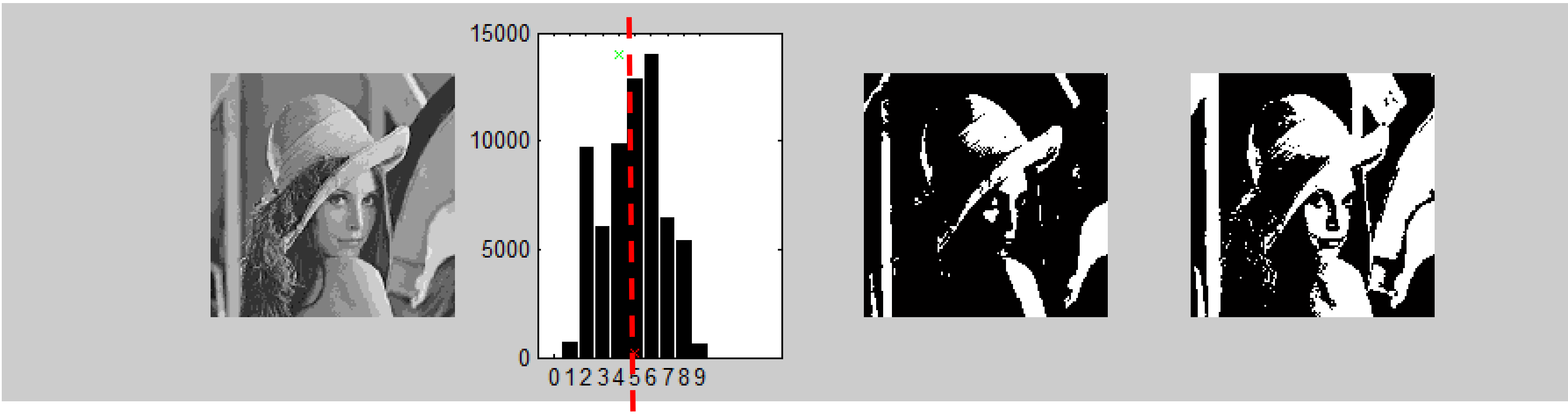
Otsu's method(Fast)



Current threshold   Best threshold

Iteration	μ_b	w_b	μ_f	w_f	σ_w^2	Bin
2	1.0000	0.0100	4.9127	0.9900	0.1515	1
3	1.9368	0.1582	5.4253	0.8418	1.6203	2
4	2.3289	0.2506	5.7244	0.7494	2.1651	3
5	2.9543	0.4004	6.1554	0.5996	2.4602	4
						5
						6
						7
						8

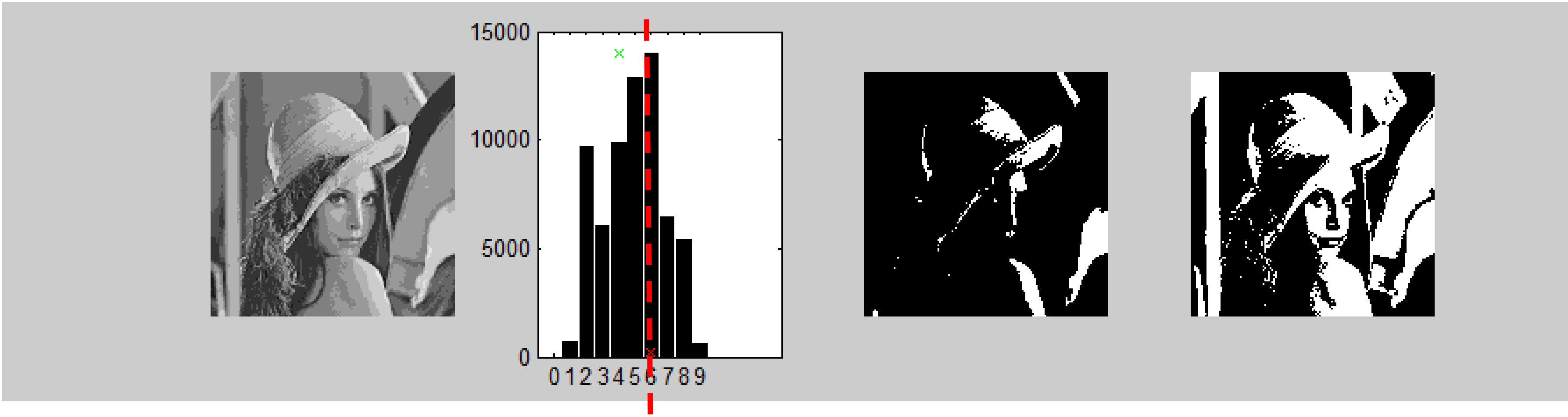
Otsu’s method(Fast)



Current threshold Best threshold

Iteration	μ_b	w_b	μ_f	w_f	σ_w^2	Bin
2	1.0000	0.0100	4.9127	0.9900	0.1515	1
3	1.9368	0.1582	5.4253	0.8418	1.6203	2
4	2.3289	0.2506	5.7244	0.7494	2.1651	3
5	2.9543	0.4004	6.1554	0.5996	2.4602	4
6	3.6273	0.5967	6.7179	0.4033	2.2985	5
						6
						7
						8

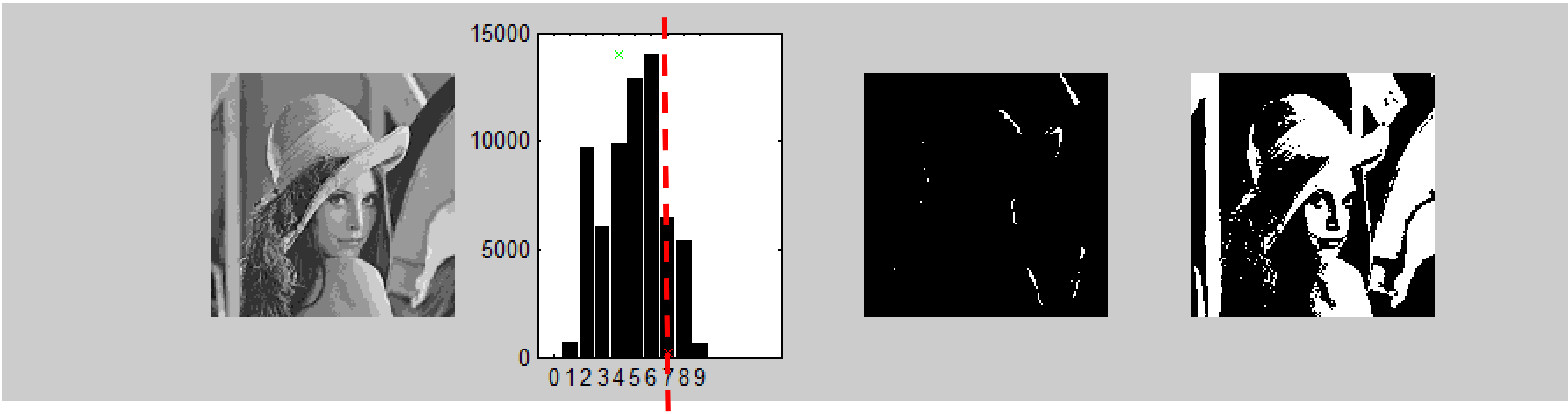
Otsu's method(Fast)



Current threshold　　Best threshold

Iteration	μ_b	w_b	μ_f	w_f	σ_w^2	Bin
2	1.0000	0.0100	4.9127	0.9900	0.1515	1
3	1.9368	0.1582	5.4253	0.8418	1.6203	2
4	2.3289	0.2506	5.7244	0.7494	2.1651	3
5	2.9543	0.4004	6.1554	0.5996	2.4602	4
6	3.6273	0.5967	6.7179	0.4033	2.2985	5
7	4.2535	0.8107	7.5291	0.1893	1.6468	6
						7
						8

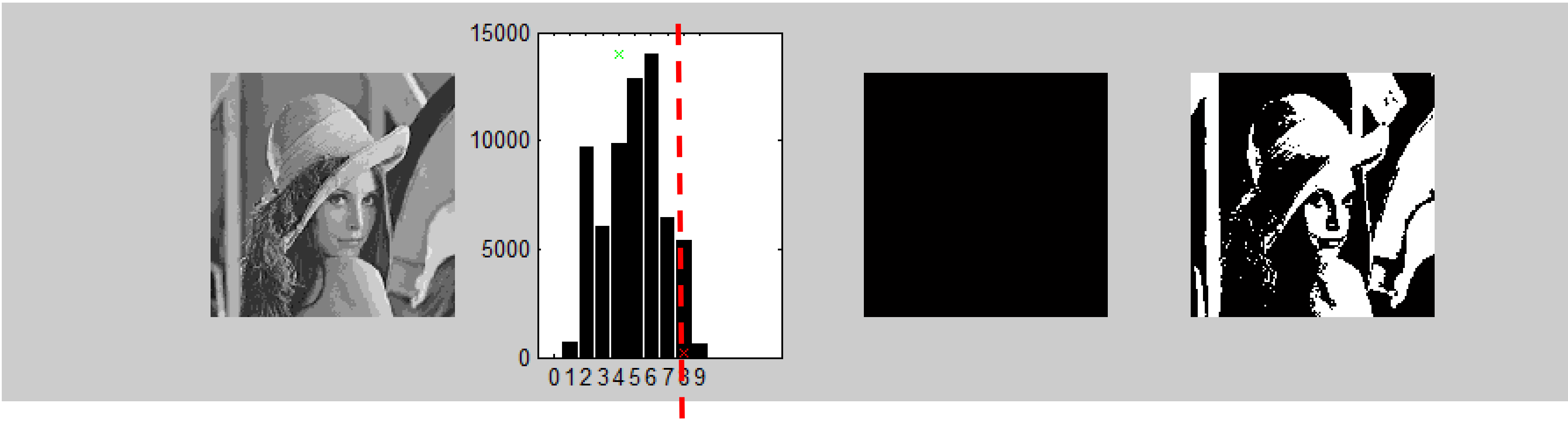
Otsu's method(Fast)



Current threshold   Best threshold

Iteration	μ_b	w_b	μ_f	w_f	σ_w^2	Bin
2	1.0000	0.0100	4.9127	0.9900	0.1515	1
3	1.9368	0.1582	5.4253	0.8418	1.6203	2
4	2.3289	0.2506	5.7244	0.7494	2.1651	3
5	2.9543	0.4004	6.1554	0.5996	2.4602	4
6	3.6273	0.5967	6.7179	0.4033	2.2985	5
7	4.2535	0.8107	7.5291	0.1893	1.6468	6
8	4.5494	0.9086	8.0958	0.0914	1.0446	7
						8

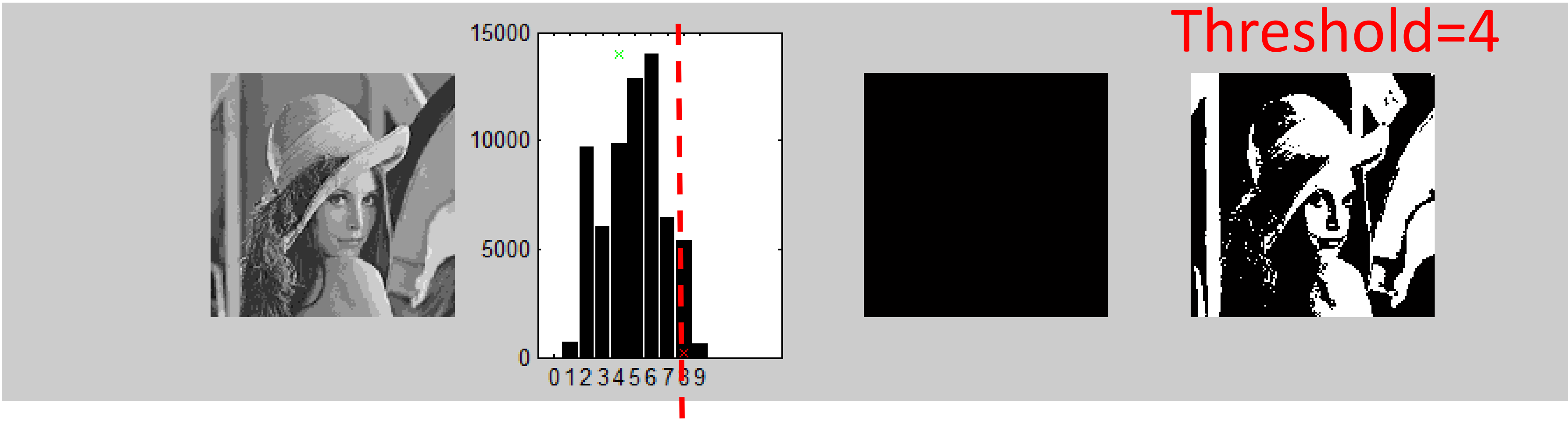
Otsu's method(Fast)



Current threshold   Best threshold

Iteration	μ_b	w_b	μ_f	w_f	σ_w^2	Bin
2	1.0000	0.0100	4.9127	0.9900	0.1515	1
3	1.9368	0.1582	5.4253	0.8418	1.6203	2
4	2.3289	0.2506	5.7244	0.7494	2.1651	3
5	2.9543	0.4004	6.1554	0.5996	2.4602	4
6	3.6273	0.5967	6.7179	0.4033	2.2985	5
7	4.2535	0.8107	7.5291	0.1893	1.6468	6
8	4.5494	0.9086	8.0958	0.0914	1.0446	7
9	4.8371	0.9912	9.0000	0.0088	0.1505	8

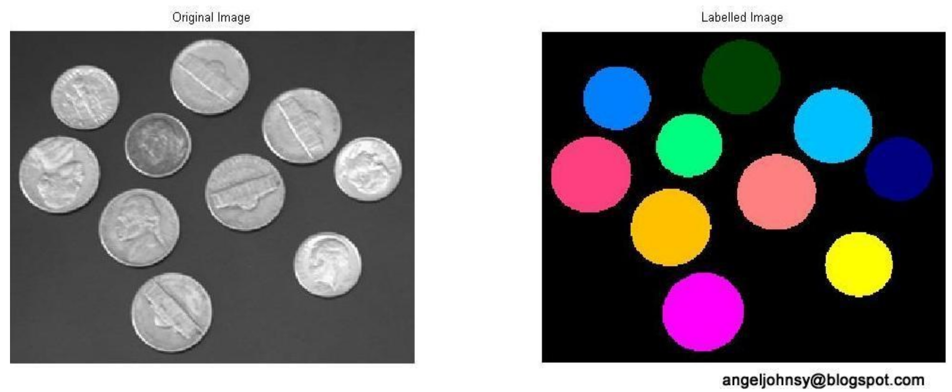
Otsu's method(Fast)



Current threshold　　Best threshold

Iteration	μ_b	w_b	μ_f	w_f	σ_w^2	Bin
2	1.0000	0.0100	4.9127	0.9900	0.1515	1
3	1.9368	0.1582	5.4253	0.8418	1.6203	2
4	2.3289	0.2506	5.7244	0.7494	2.1651	3
5	2.9543	0.4004	6.1554	0.5996	2.4602	4
6	3.6273	0.5967	6.7179	0.4033	2.2985	5
7	4.2535	0.8107	7.5291	0.1893	1.6468	6
8	4.5494	0.9086	8.0958	0.0914	1.0446	7
9	4.8371	0.9912	9.0000	0.0088	0.1505	8

Pixel Neighborhood



Pixel Neighborhood

1	2	3	4	5
6	7	8	9	10
11	12	**13**	14	15
16	17	18	19	20
21	22	23	24	25

1	2	3	4	5
6	7	**8**	9	10
11	12	**13**	14	15
16	17	18	19	20
21	22	23	24	25

4-connected neighbors

1	2	3	4	5
6	7	8	9	10
11	12	**13**	14	15
16	17	18	19	20
21	22	23	24	25

8-connected neighbors

Pixel Neighborhood

Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

V

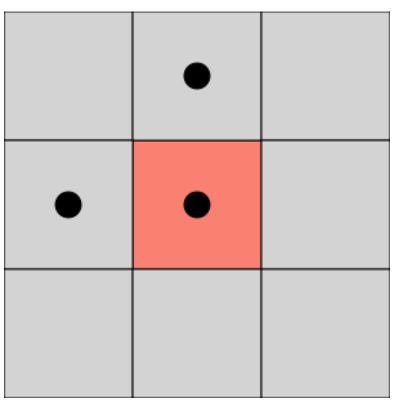
p is current pixel

If all neighbours are 0 then assign a new label to p
else
If only one neighbour has V={1} then assign its label to p
else
If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

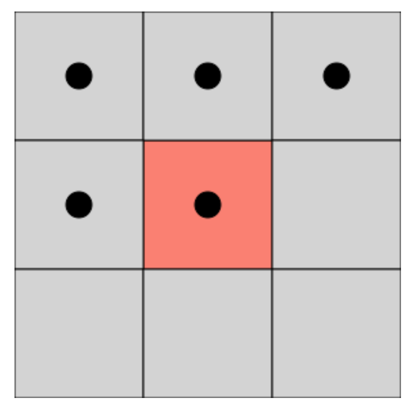
Pixel Neighborhood

Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0



4-connected neighbors



8-connected neighbors

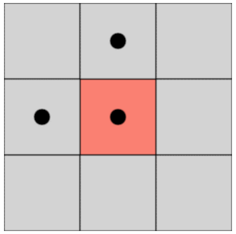
Pixel Neighborhood

Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	1	0	0	0	0	0	0

V

4-connected neighbors



If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

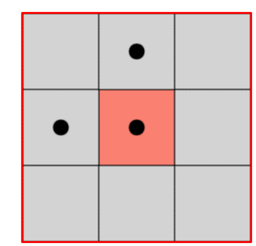
Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

V

X									

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

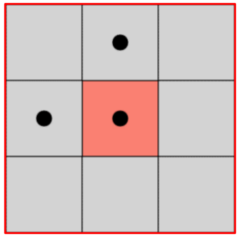
Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

V

	X								

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

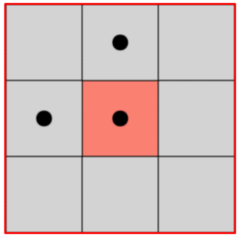
Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

V

	1								

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

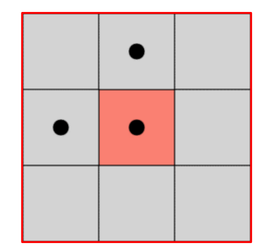
Connected-component labelling

0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

V

	1	**X**							

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

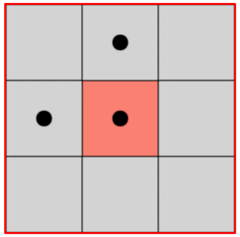
Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

V

	1		X						

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

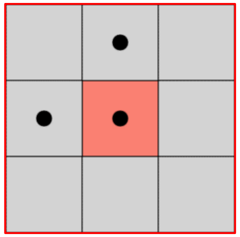
Connected-component labelling

0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

V

	1		**2**						

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

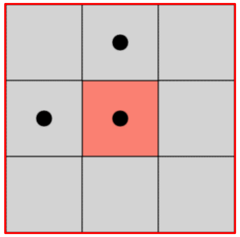
Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

V

	1		2						x

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

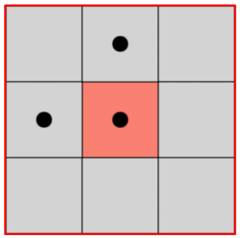
Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

V

	1		2						
	X								

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

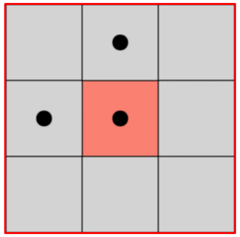
Connected-component labelling

0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

V

	1		**2**						
	1								

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

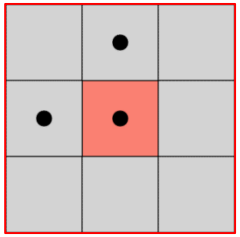
Connected-component labelling

0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

V

	1		**2**						
	1		**2**						

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

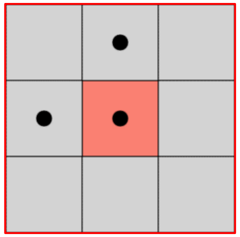
Connected-component labelling

0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

V

	1		**2**						
	1		**2**						
	1		**2**						
	1	**1**	**X**						

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

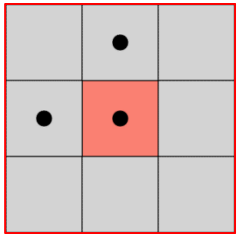
Connected-component labelling

0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

V

	1		**2**						
	1		**2**						
	1		**2**						
	1	**1**	**X**						

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

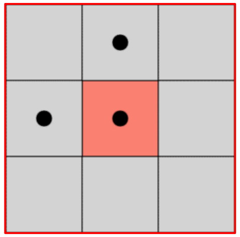
Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

V

	1		1						
	1		1						
	1		1						
	1	1	X						

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

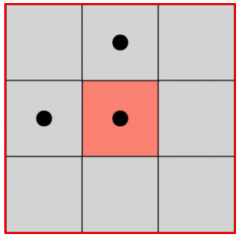
Connected-component labelling

0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

V

	1		**1**						
	1		**1**						
	1		**1**						
	1	**1**	**1**						

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

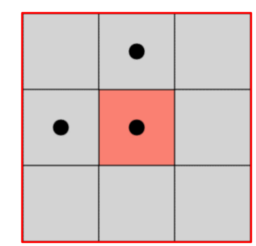
Connected-component labelling

0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

V

	1		**1**						
	1		**1**						
	1		**1**						
	1	**1**	**1**						
	1		**1**		**X**				

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

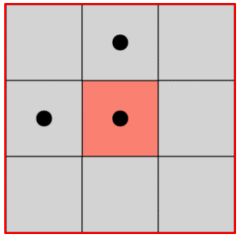
Connected-component labelling

0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

V

	1		**1**						
	1		**1**						
	1		**1**						
	1	**1**	**1**						
	1		**1**		**2**				

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

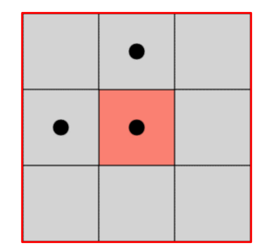
Connected-component labelling

0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

V

	1		**1**						
	1		**1**						
	1		**1**						
	1	**1**	**1**						
	1		**1**		**2**	**2**			
	1		**1**		**2**	**2**			

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Pixel Neighborhood

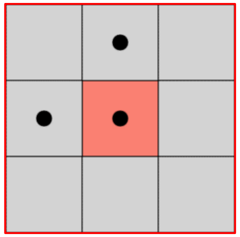
Connected-component labelling

0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

V

	1		1						
	1		1						
	1		1						
	1	1	1						
	1		1		2	2			
	1		1		2	2			

4-connected neighbors



If V={1} then

If all neighbours are 0 then assign a new label to p

else

If only one neighbour has V={1} then assign its label to p

else

If more than one of the neighbours have V={1} then assign one of the labels to p and make a note of the equivalences.

Homework !

0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	1	0	0	0	0
0	1	0	0	1	0	0	1	1	0	0	0
0	1	0	1	0	0	0	1	1	1	0	0
0	1	1	0	0	0	0	1	1	1	1	0
0	1	0	1	0	0	0	1	0	0	0	0
0	1	0	0	1	0	0	1	0	0	0	0
0	1	0	0	0	1	0	1	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Segment this image using

1) 4-connected neighbors

2) 8-connected neighbors

Matlab image processing function

Labelling:

L=bwlabel**(im)**;

Coordinate search:

[*r c***]**=find(**L**==*label*);

Coloring Label:

im_rgb=label2rgb(**L**);

Pixel connectivity

https://en.wikipedia.org/wiki/Pixel_connectivity