Summary of Lecture 3

- Simple histogram based image segmentation and its limitations.
- Continuous and discrete amplitude random variables $\overset{\text{properties}}{\Longrightarrow}$ the histogram equalizing point function.
- Images as matrices containing "$N \times M$ random outcomes" of a random variable.
- The relationship between image histograms and sample probability mass functions of images.

Histogram Equalization

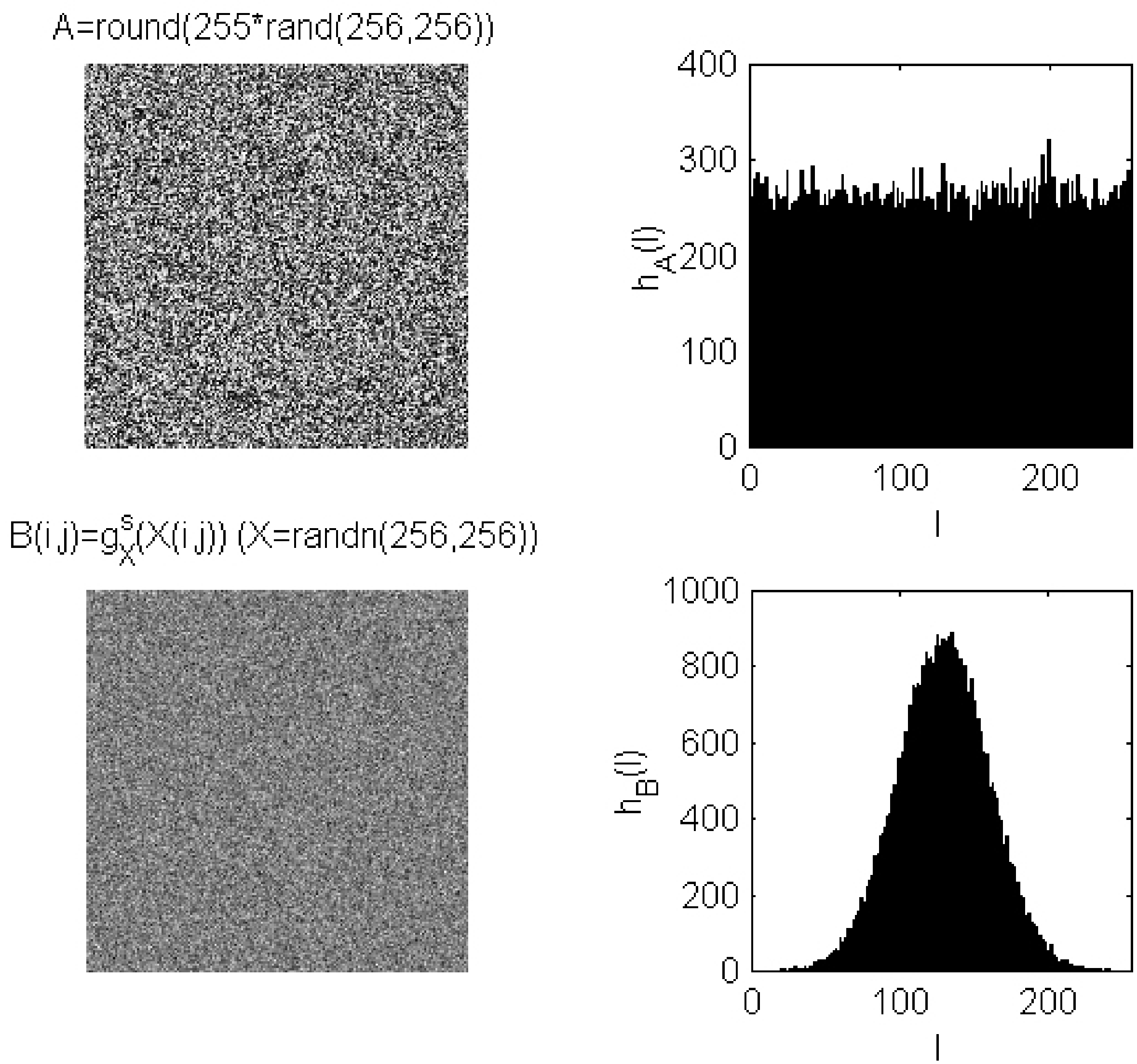
- $g_1(l) = \Sigma_{k=0}^{l} p_A(k) \Rightarrow g_1(l) - g_1(l-1) = p_A(l) = \frac{h_A(l)}{NM}$ $(l = 1, \ldots, 255)$.
- $g_A^e(l) = \mathrm{round}(255 g_1(l))$ is the histogram equalizing point function for the image $\mathbf{A}$.
- $B(i,j) = g_A^e(A(i,j))$ is the histogram equalized version of $\mathbf{A}$.
- In general, histogram equalization stretches/compresses an image such that:
  - Pixel values that occur frequently in $\mathbf{A}$ occupy a bigger dynamic range in $\mathbf{B}$, i.e., get stretched and become more visible.
  - Pixel values that occur infrequently in $\mathbf{A}$ occupy a smaller dynamic range in $\mathbf{B}$, i.e., get compressed and become less visible.
- Histogram equalization is not ideal, i.e., in general $\mathbf{B}$ will have a "flatter" histogram than $\mathbf{A}$, but $p_B(l)$ is not guaranteed to be uniform (flat).

Random Images - Images of White Noise

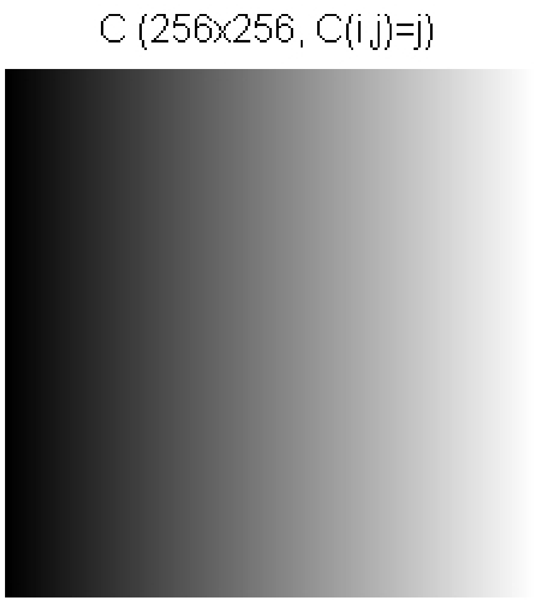
- A single outcome of a continuous amplitude uniform random variable $\chi \in [0,1]$ in matlab: $>> \ x = \mathrm{rand}(1,1);$
- An $N \times M$ matrix of outcomes of a continuous amplitude uniform random variable $\chi \in [0,1]$ in matlab: $>> \ X = \mathrm{rand}(N,M);$
- An $N \times M$ image matrix of outcomes of a discrete amplitude uniform random variable $\Theta \in \{0,1,\ldots 255\}$ in matlab: $>> \ A = \mathrm{round}(255 * X);$

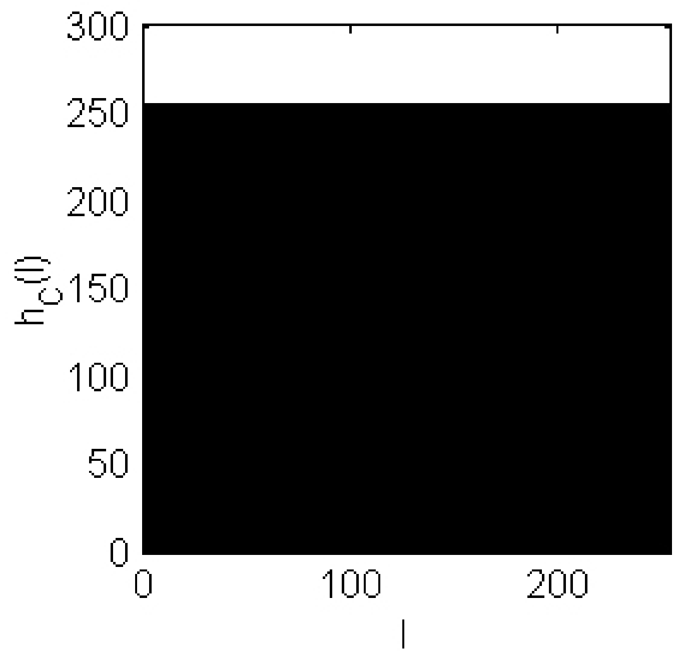
- A single outcome of a continuous amplitude gaussian random variable $\chi \ (\mu = 0, \sigma^2 = 1)$ in matlab: $>> \ x = \mathrm{randn}(1,1);$
- An $N \times M$ matrix of outcomes of a continuous amplitude gaussian random variable $\chi \ (\mu = 0, \sigma^2 = 1)$ in matlab: $>> \ X = \mathrm{randn}(N,M);$
- An $N \times M$ image matrix of outcomes of a discrete amplitude "gaussian" random variable $\Theta \in \{0,1,\ldots 255\}$: $A(i,j) = g_X^s(X(i,j))$.

Example


A=round(255*rand(256,256))
B(i,j)=g_X^s(X(i,j)) (X=randn(256,256))
h_A(l)
h_B(l)
l
400
300
200
100
0
1000
800
600
400
200
0
100
200

Warning





- Remember, two totally different images may have very similar histograms.

Histogram Matching - Specification

- Given *images* $\mathbf{A}$ and $\mathbf{B}$, using point processing we would like to generate an image $\mathbf{C}$ from $\mathbf{A}$ such that $h_C(l) \sim h_B(l), \ (l = 0, \ldots, 255)$.
- More generally, given an image $\mathbf{A}$ and a histogram $h_B(l)$ (or sample probability mass function $p_B(l)$), we would like to generate an image $\mathbf{C}$ such that $h_C(l) \sim h_B(l), \ (l = 0, \ldots, 255)$.
- Histogram matching/specification enables us to "match" the grayscale distribution in one image to the grayscale distribution in another image.

Derivation for Continuous Amplitude R.V.s

- We have already seen that for a continuous amplitude random variable χ with strictly increasing and continuous $F_\chi(x)$, the random variable $Y = F_\chi(\chi)$ has the uniform probability density/distribution function.
- *Equivalently*, for a continuous amplitude random variable $Y \in [0, 1]$ which has the uniform probability density function, $\chi = F_\chi^{-1}(Y)$ has the probability density [distribution] function $f_\chi(x)$ $[F_\chi(x)]$.

$$\chi \;[F_\chi(x)] \;\Rightarrow\; Y = F_\chi(\chi) \;[\text{uniform}]$$
$$Y \;[\text{uniform}] \;\Rightarrow\; \chi = F_\chi^{-1}(Y) \;[F_\chi(x)]$$

- Now, assume we have a continuous amplitude random variable Z with strictly increasing and continuous $F_Z(z)$. Then:

$$Y \;(\text{uniform}) \;\Rightarrow\; W = F_Z^{-1}(Y) \;[F_Z(w)] \tag{1}$$

but we can generate the required uniform random variable Y from χ via $Y = F_\chi(\chi)$ which means W can be generated from χ via:

$$W = F_Z^{-1}(Y) = F_Z^{-1}(F_\chi(\chi)) \tag{2}$$

Main Result

- Given a continuous amplitude random variable χ with strictly increasing and continuous $F_\chi(x)$, let $F_Z(z)$ be the *specified* distribution ($F_Z(z)$ strictly increasing and continuous).
- Then, $W = F_Z^{-1}(F_\chi(\chi))$ is a random variable that is a function of χ with $F_W(w) = F_Z(w)$.
- For discrete amplitude random variables this derivation does not work exactly in general. However, similar to the histogram equalizing point function, we will generate a point function that operates on an image $\mathbf{A}$ to "match" its histogram to that of image $\mathbf{B}$.

Algorithm

- In general we will not be able to calculate inverses of the distribution functions of discrete amplitude random variables.
- Let $p_A(l),\ p_B(l)\ (l = 0, \ldots, 255)$ be the sample probability mass functions of images $\mathbf{A}$ and $\mathbf{B}$ respectively.
- Let $g_1(l) = \Sigma_{k=0}^{l} p_A(k)$ and $g_2(l) = \Sigma_{k=0}^{l} p_B(k)$.
- Generate the "histogram matched" $\mathbf{C}$ as $C(i,j) = g_3(A(i,j))$ where:

$$
\begin{aligned}
g_3(l) &= m \qquad (m \in \{0, 1, \ldots, 255\}) \\
m &= \min\{k | g_2(k) - g_1(l) \geq 0,\ \ k = 0, \ldots, 255\}
\end{aligned}
\tag{3}
$$

- Assuming g_2 and g_1 are precomputed:

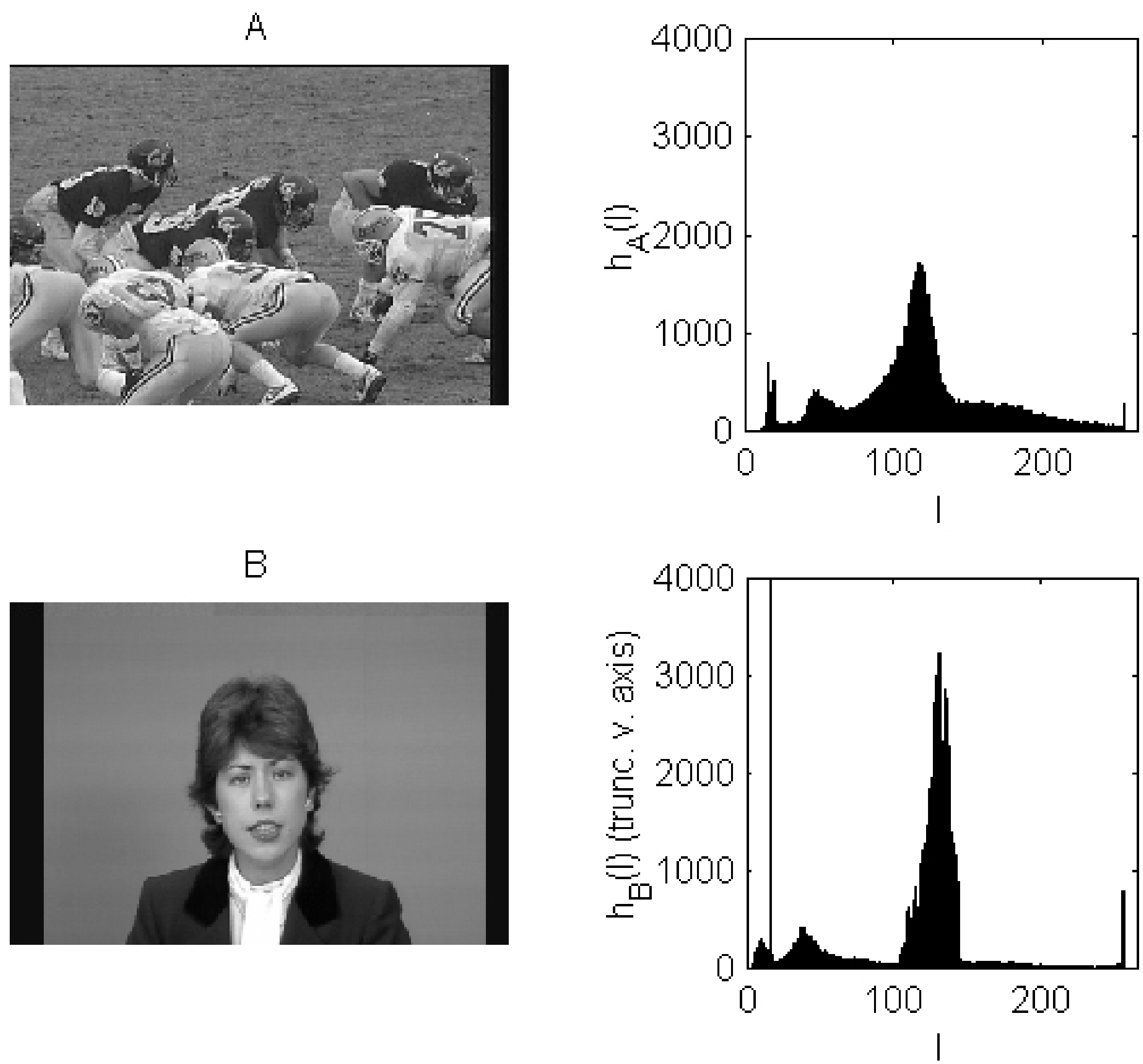
$$
\begin{aligned}
>> \ \ \text{for}\ \ i\ &= 1 : 256 \\
&g3(i) = 256 - \text{sum}(g2 >= g1(i)); \\
&\text{end};
\end{aligned}
$$

Example I

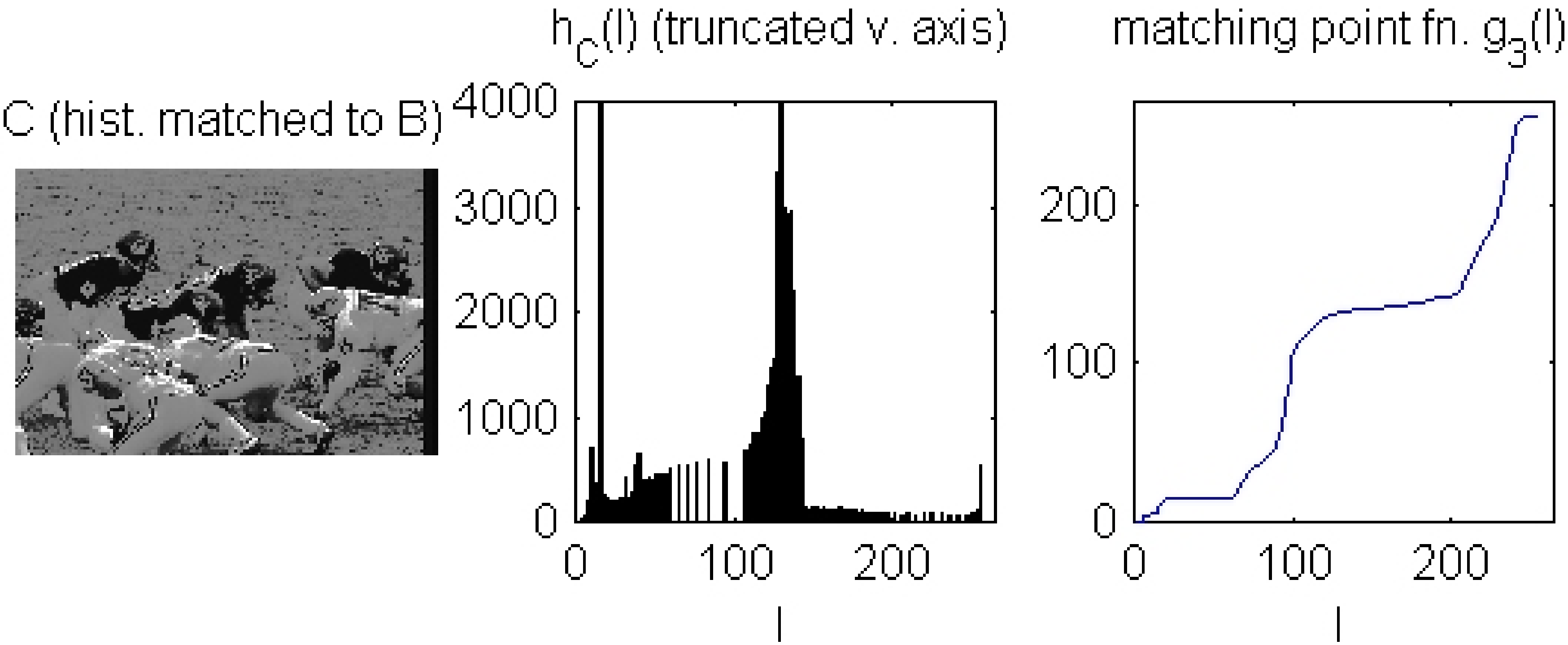
Example 7.1 in textbook (P. 244) in "our notation":

l	$p_A(l)$	$g_1(l)$	$g_3(l) = \min\{k \| g_2(k) - g_1(l) \geq 0\}$	$g_2(k)$	$p_B(k)$	k
0	0.25	0.25	1	0	0	0
1	0.25	0.5	1	0.5	0.5	1
2	0.25	0.75	2	1.0	0.5	2
3	0.25	1.0	2	1.0	0	3

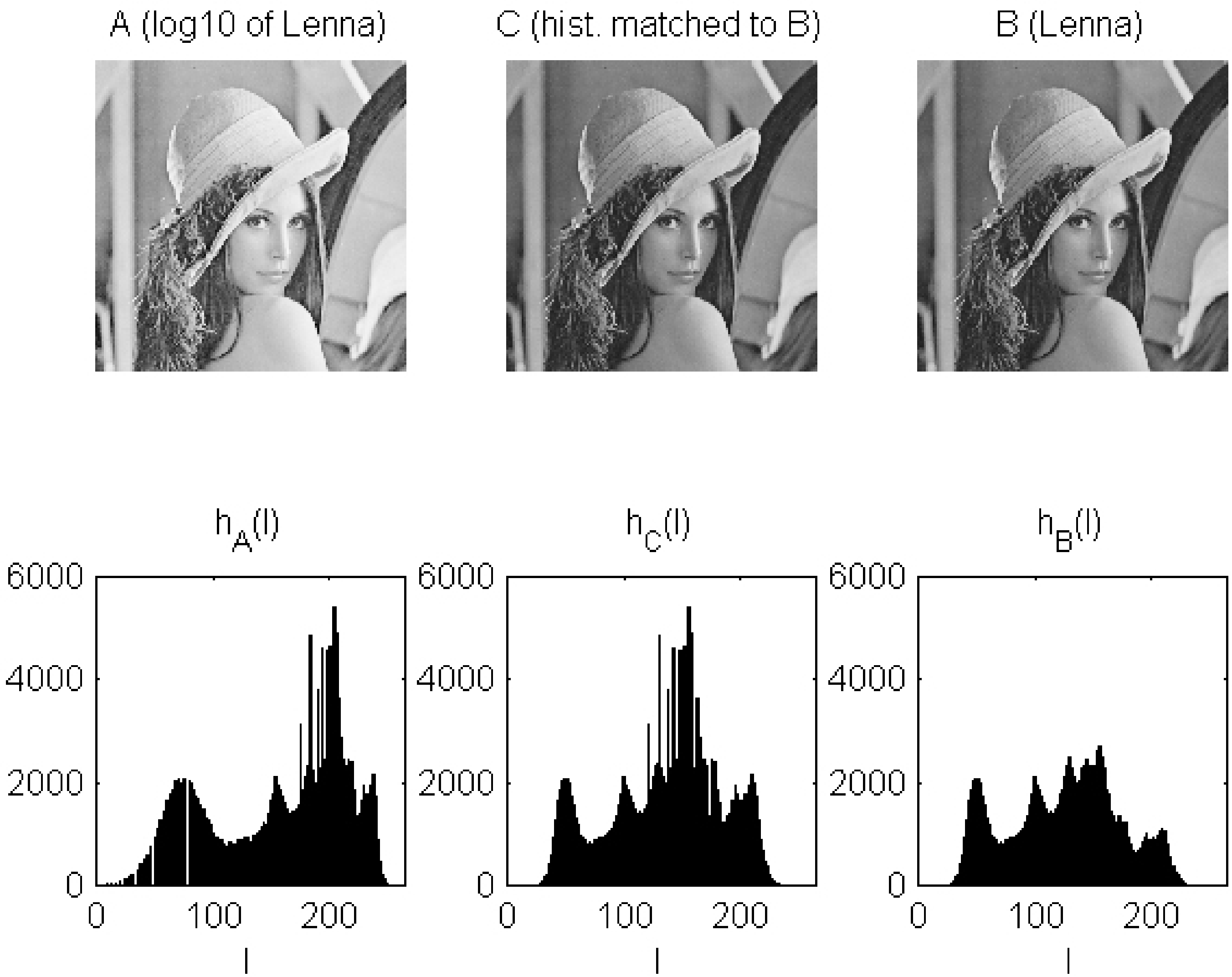
Example II - Histogram Matching Different Images



Example II - contd.



Example III - “Undoing” via Histogram Specification



Quantization

- Let $t_n \in \{0, 1, \ldots, 255\}$ denote a sequence of thresholds $(n = 0, \ldots, P-1)$.
- Consider the P "half-open, discrete intervals" $R_n = [t_n, t_{n+1})$ $(t_0 = 0,\ \ t_P = 256)$.
- Let $r_n \in R_n$ be the reproduction level of the interval R_n.
- Define the quantizing point function or the P-level quantizer $Q(l)$ in terms of the $R_n,\ r_n$ (or equivalently in terms of $t_n,\ r_n$) as follows:

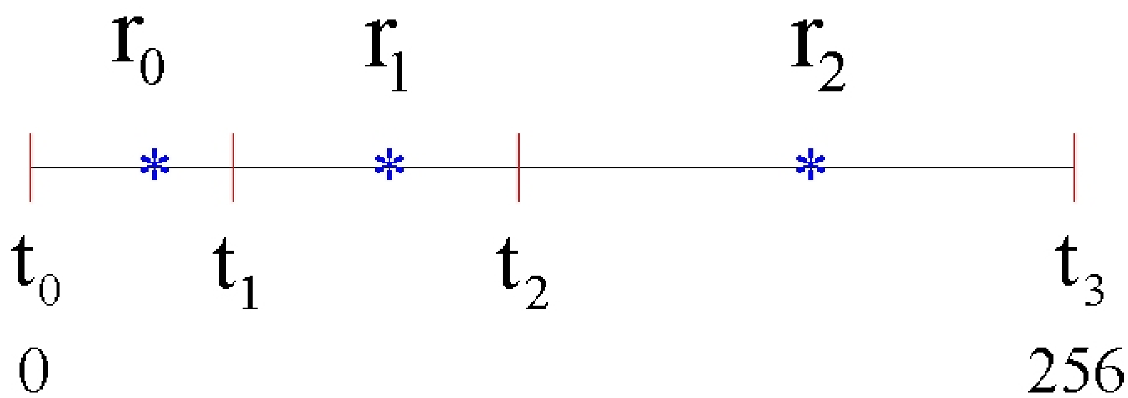
$$Q(l) = \{r_k | l \in R_k,\ \ k = 0, \ldots, P-1\} \tag{4}$$

i.e., $l \in R_k \Leftrightarrow Q(l) = r_k$.

- Quantizing an image $\mathbf{A}$ in matlab:

$$\begin{array}{lll} >> & Q & = \text{zeros}(256,1);\ \ x = (0:255)'; \\ >> & \text{for} & i = 1:P \\ & & Q = Q + r(i) * ((x >= t(i)) \& (x < t(i+1))); \\ & & \text{end;} \qquad \%\ t(P+1) = 256 \\ >> & B & = Q(A+1); \end{array}$$

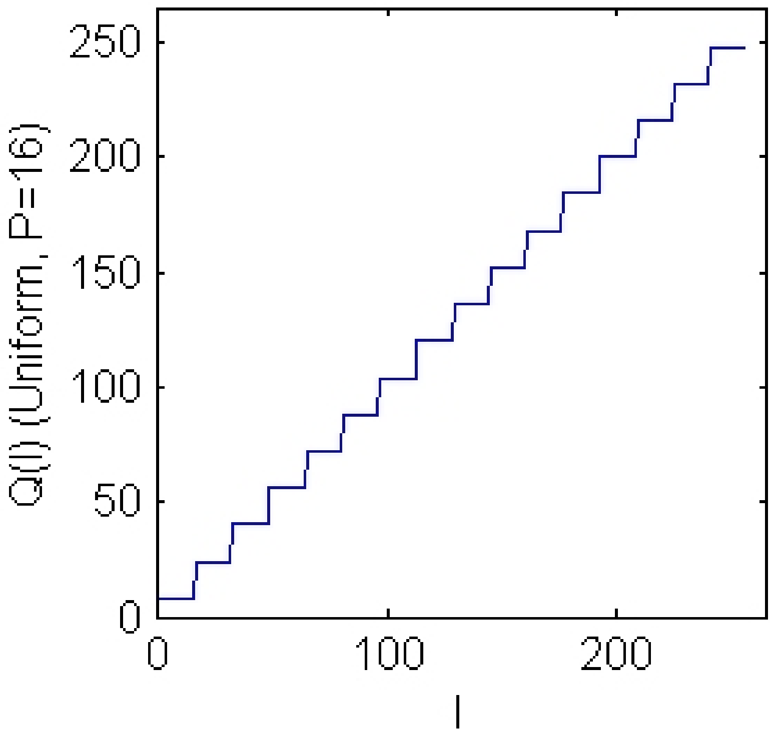
The Interval Partition View of a Quantizer

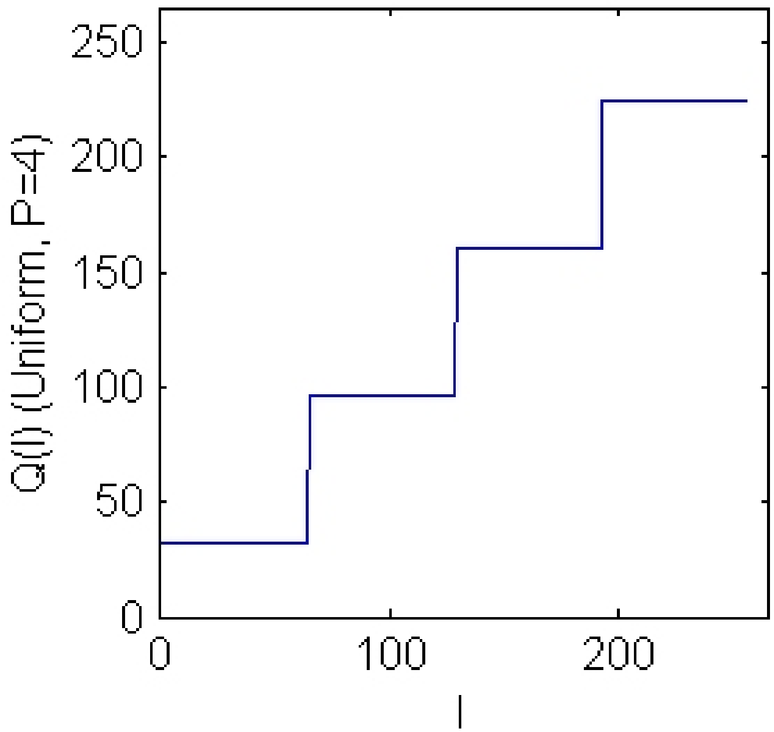


- $R_n = [t_n, t_{n+1})$.

Uniform Quantization

- In uniform quantization $P = 256/\Delta$, $t_{n+1} - t_n = \Delta$, $\forall n$ and $r_n = \frac{t_n + t_{n+1}}{2}$.
- Δ is the stepsize of the uniform quantizer $(r_n = n\Delta + \Delta/2)$.





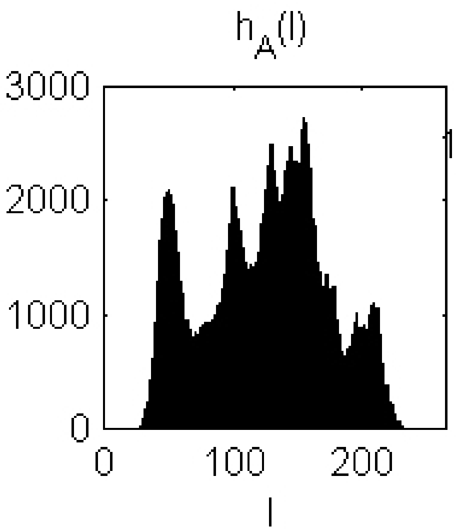
Easy uniform quantization: $>> B = delta * \text{floor}(A/delta) + delta/2;$

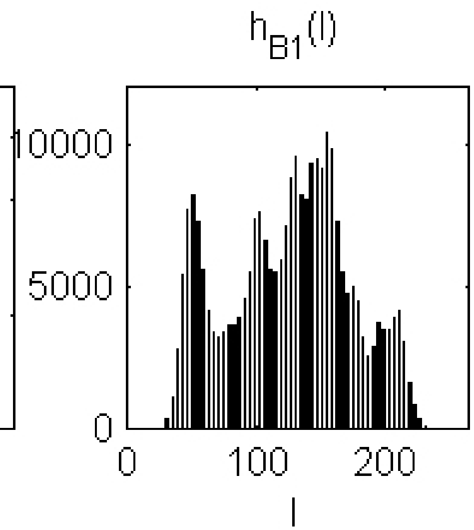
Example

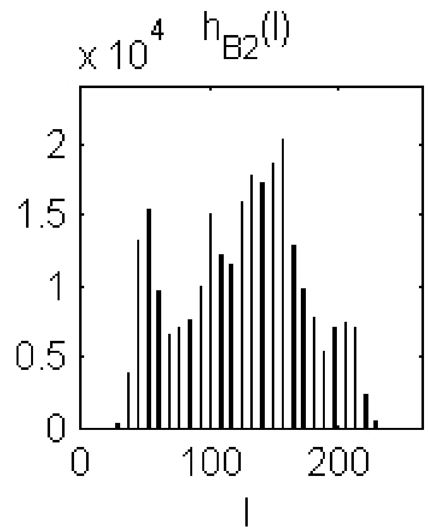
A

B1, P=64 (Δ=4)

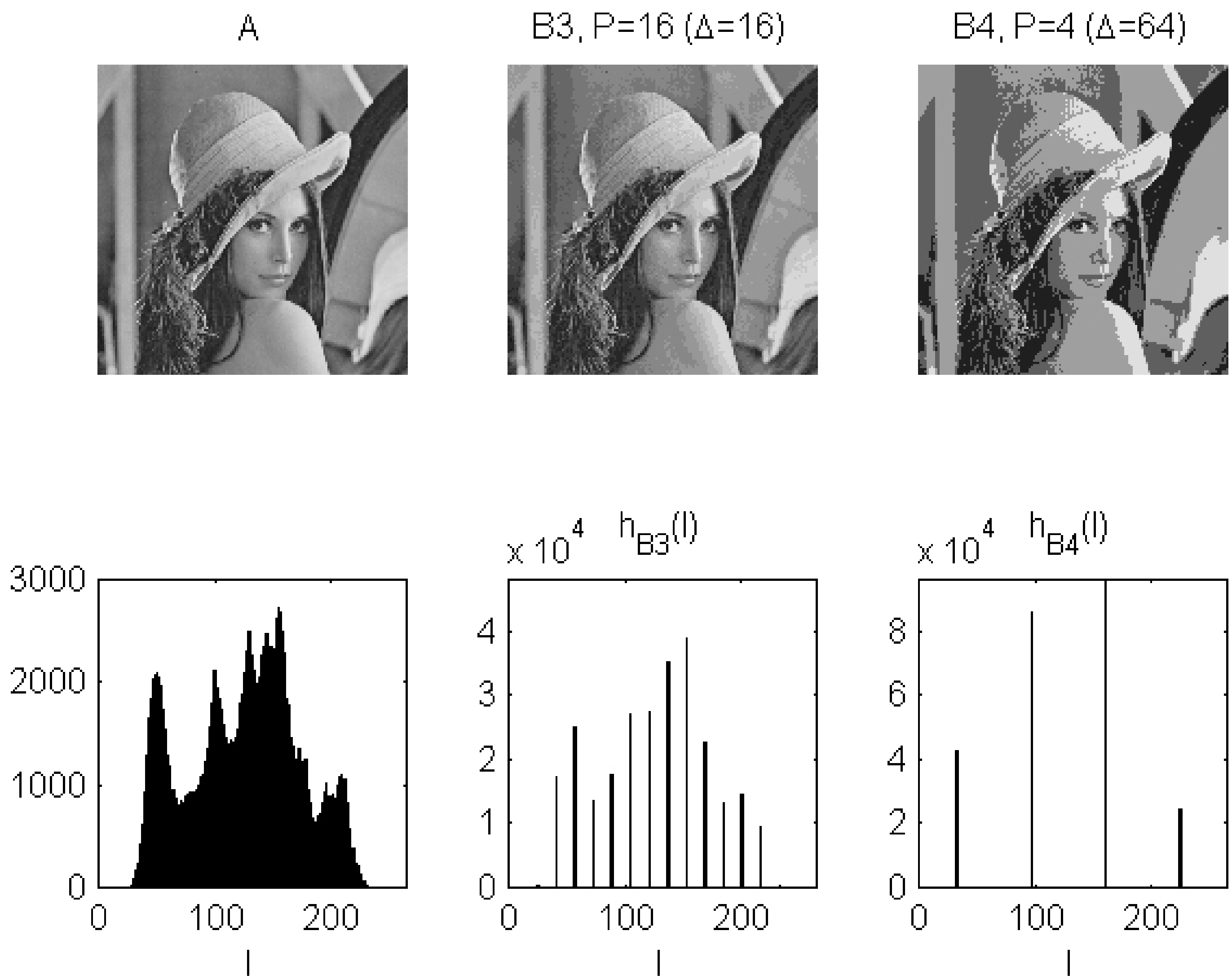
B2, P=32 (Δ=8)





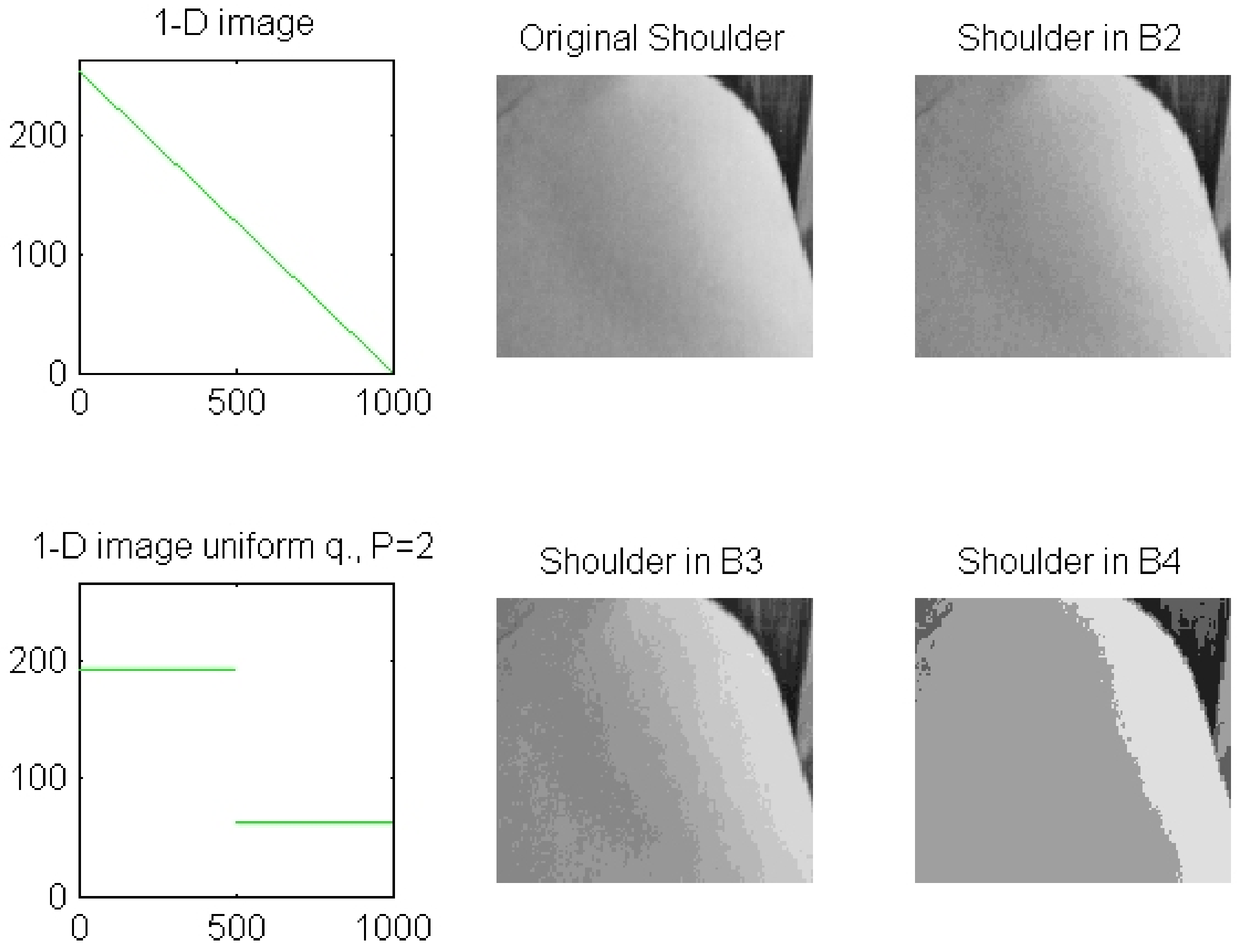


Example - contd.



Quantization Artifacts - False Contours

False Contours or “False Edges” on a 1-D image and earlier example:



Quantization Statistics

- The quantization error matrix is defined as $\mathbf{E} = \mathbf{A} - Q(\mathbf{A})$.
- The sample mean squared quantization error (MSQE) is:

$$\begin{aligned} \text{MSQE} &= \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(E(i,j))^2}{NM} && (5) \\ &= \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(A(i,j) - Q(A(i,j)))^2}{NM} \\ &= \sum_{l=0}^{255}(l - Q(l))^2 p_A(l) && (6) \end{aligned}$$

Example

For the earlier example:

Δ	Quantized Image	MSQE
4	B1	1.50
8	B2	5.49
16	B3	22.18
64	B4	334.77

Designing Good Quantizers

- We would like to design the R_n, r_n (or equivalently t_n, r_n) such that the MSQE is as small as possible.
- Repeating Equations 4 and 6:

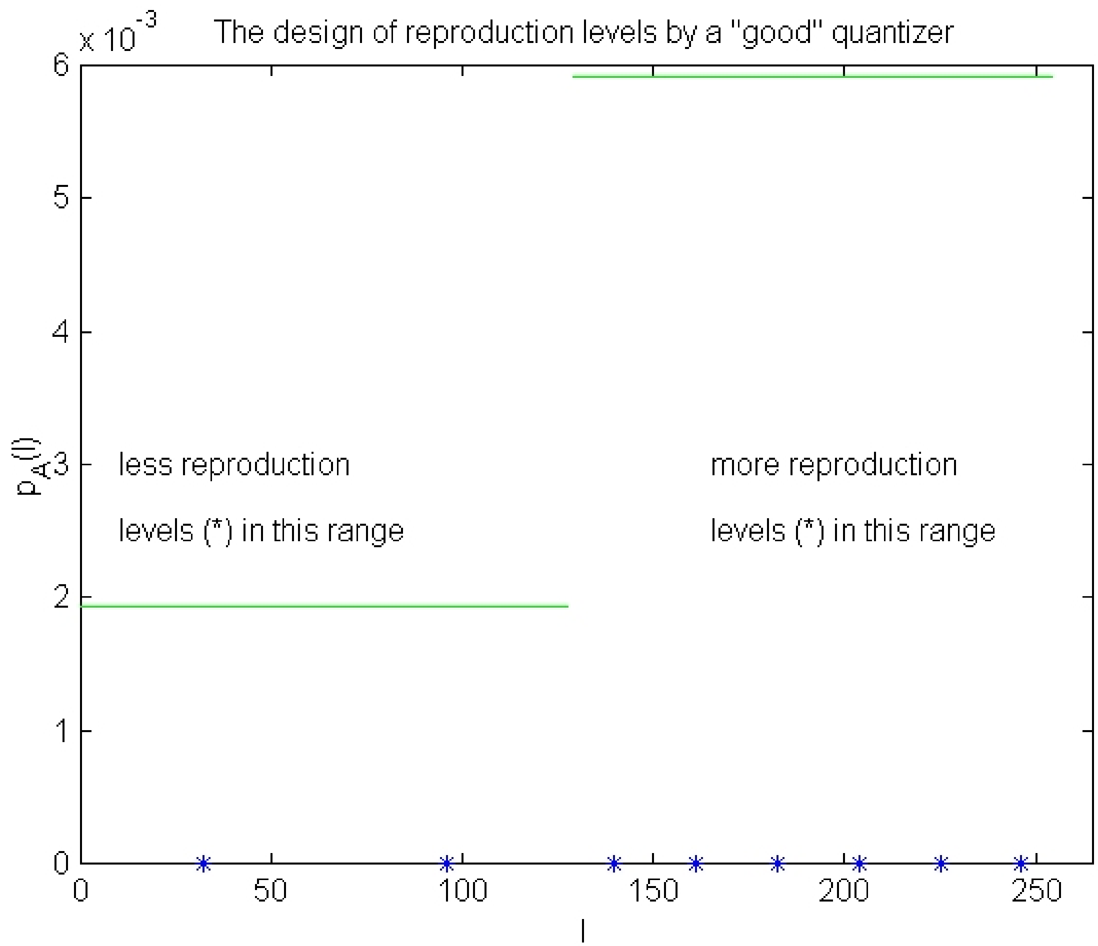
$$\begin{aligned} Q(l) &= \{r_k | l \in R_k,\ \ k = 0, \ldots, P-1\} \\ \text{MSQE} &= \sum_{l=0}^{255} (l - Q(l))^2 p_A(l) \end{aligned}$$

  we can make the following important observations assuming P is fixed:

  – Around ranges of l where $p_A(l)$ is large, a good quantizer should have many small R_n, i.e., since we can have at most P discrete intervals, most of these intervals should be around ranges of l where $p_A(l)$ is large.

  – Equivalently, a good quantizer should not "waste" many reproduction levels around ranges of l where $p_A(l)$ is *small*.

Example



The reproduction levels $*$ are shown in the interval partition view.

Designing the Thresholds for Given Reproduction Levels

- Assume P is given and we "picked" the reproduction levels r_n in locations where $p_A(l)$ is large.
- How do we *optimally* pick the thresholds t_n ?
- By definition $r_{n-1} < t_n \leq r_n$ except for $t_0 = 0, \;\; t_{P+1} = 255$.
- Suppose $r_{n-1} < l \leq r_n$. Let $d_n = (r_{n-1} + r_n)/2$ be the midpoint. Then in order to minimize MSQE, $Q(l)$ must be:

$$Q(l) = \begin{cases} r_{n-1} & l < d_n \\ r_n & l \geq d_n \end{cases} \tag{7}$$

- For MSQE optimality

$$t_n = \text{round}(\frac{r_{n-1} + r_n}{2}) \tag{8}$$

- Note that this result is *independent* of $p_A(l)$ and only depends on r_n.

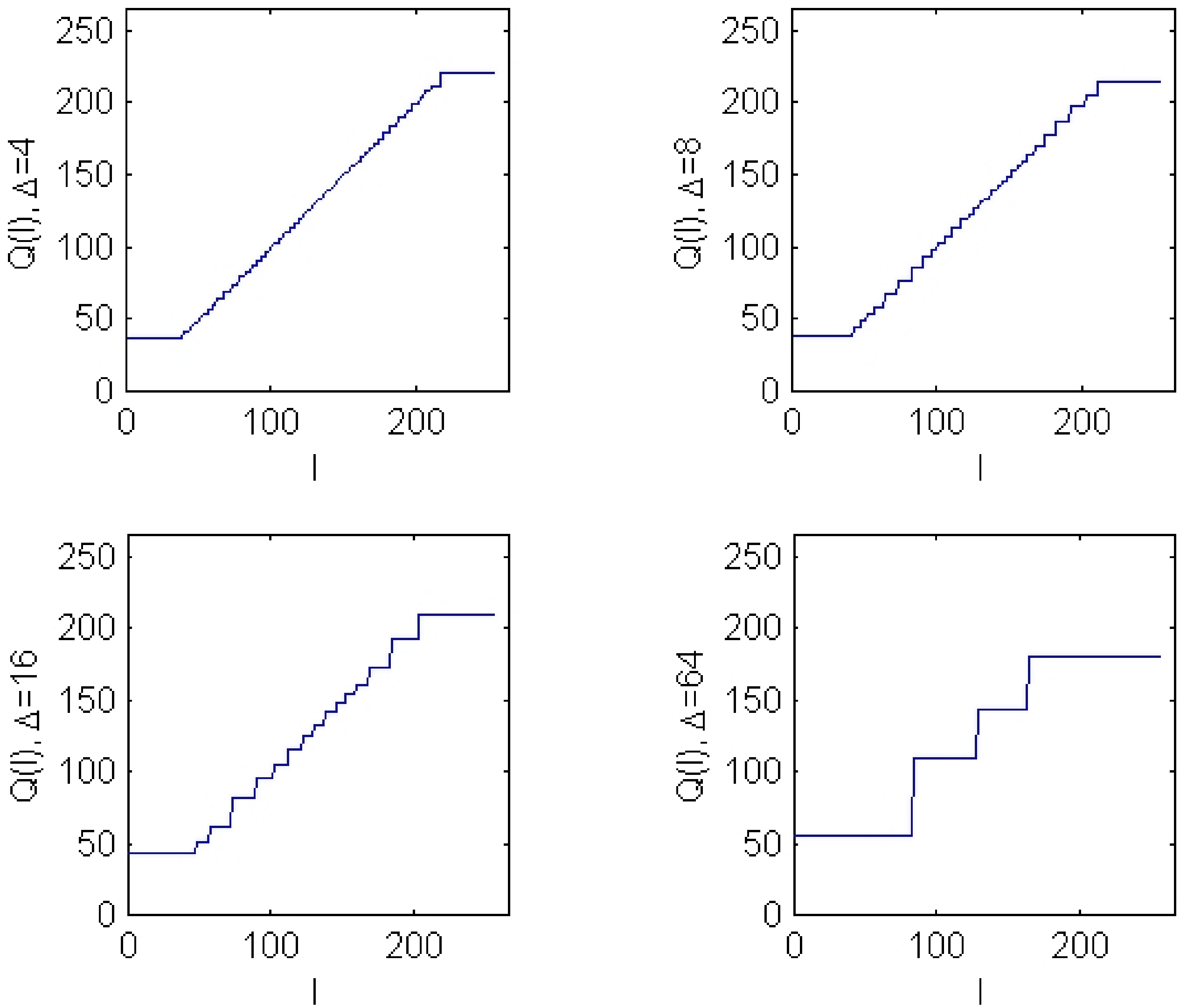
Companding

- Given the r_n we know how to choose the t_n.
- We also know that we should pick r_n close apart where $p_A(l)$ is large and far apart where $p_A(l)$ is small.
- Assume $p_A(l)$ is uniform. Then, clearly, we can pick r_n "uniformly", i.e., use the r_n that correspond to a uniform quantizer.
- In general $p_A(l)$ is not uniform, but "hopefully" $p_B(l)$ for $B(i,j) = g_A^e(A(i,j))$ is.
- Let $g_1(l) = \Sigma_{k=0}^{l} p_A(k)$. Let $\Delta = 256/P$. pick r_n $(n = 0, \ldots, P-1)$ via:

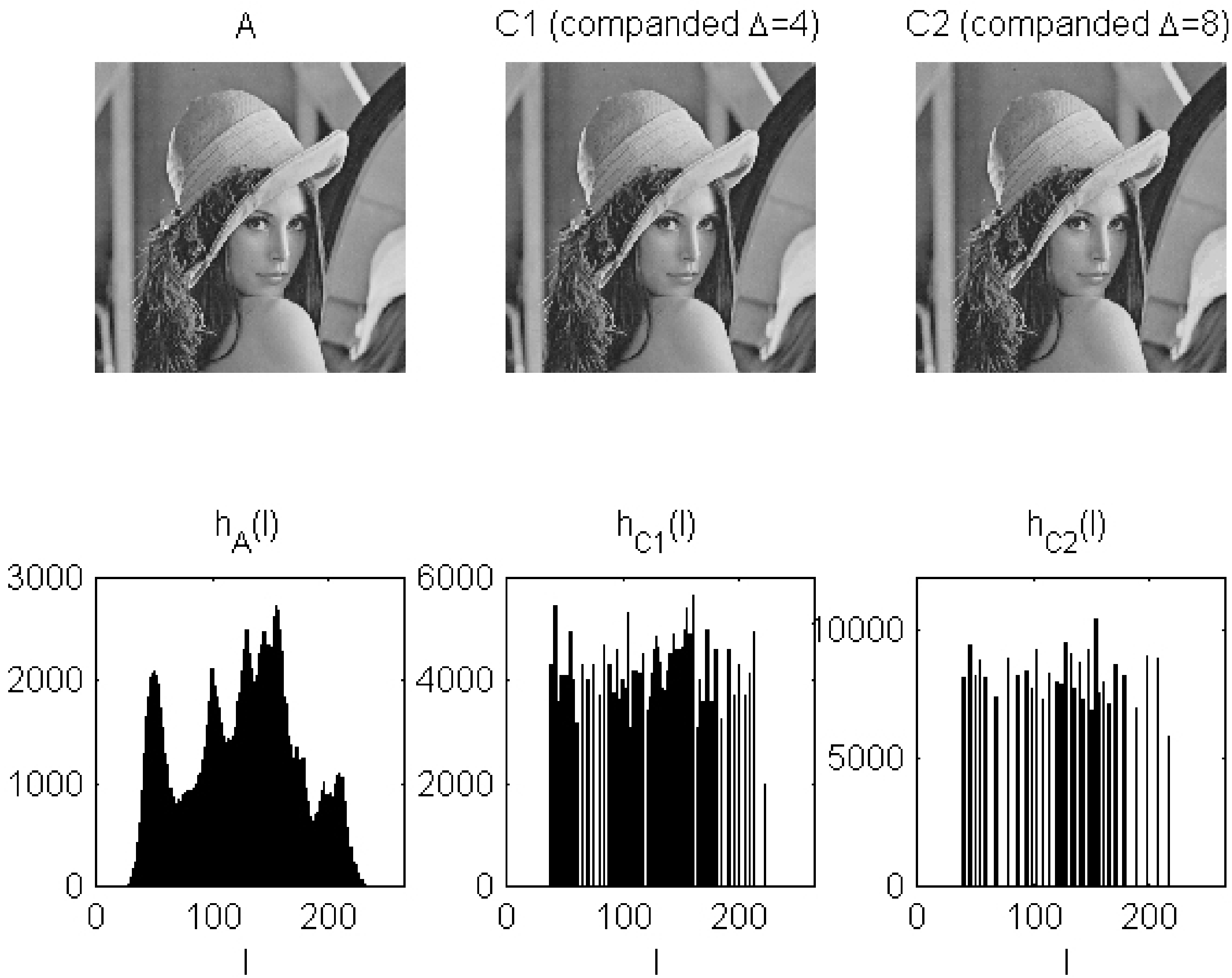
$$r_n = \min\{k | 255 g_1(k) - (n\Delta + \Delta/2) \geq 0, \; k = 0, \ldots, 255\} \tag{9}$$

(Note the similarity to Equation 4 as we are again calculating a "discrete inverse")

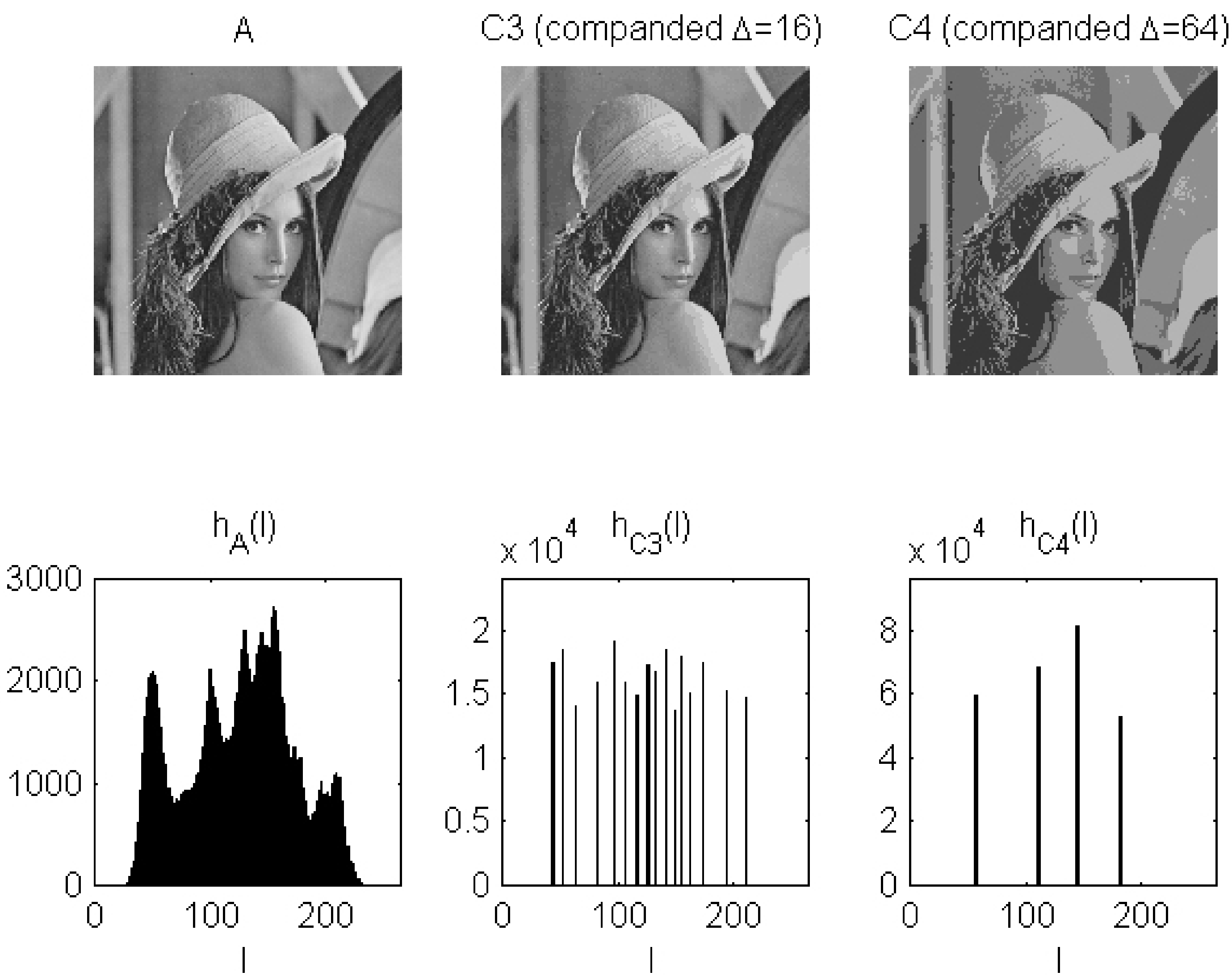
Companding Quantizer Point Functions



Example


A
C1 (companded Δ=4)
C2 (companded Δ=8)
hA(l)
hC1(l)
hC2(l)
3000
2000
1000
0
6000
4000
2000
0
10000
5000
0
0
100
200
l

Example - contd.



Example - contd.

Uniform $\Delta = 16$

Companded $\Delta = 16$

Δ	Companded Image	MSQE
4	C1	1.56
8	C2	4.28
16	C3	13.84
64	C4	186.27

Compare to uniform quantization results.

Summary

- In this lecture we learnt how to generate random images.
- We learnt about histogram matching which enabled us to "match" the histogram of a given image to another image's histogram.
- We learnt how to calculate the "inverses" of discrete functions.
- We learnt about quantization, simple uniform quantization and companding.
  - Calculating errors.
  - Simple quantization statistics.
  - Choosing thresholds optimally.
  - Please read the textbook pages 243-244, 99-118.

Homework IV

1. Generate a 256×256 matrix $\mathbf{A}$ of outcomes of a continuous amplitude Gaussian random variable with $\mu = 2$ and $\sigma^2 = 3$. Calculate its sample mean and variance. Note that $\mathbf{A}$ is not an image matrix. Normalize $\mathbf{A}$ to obtain $\mathbf{B}$. Calculate the sample mean, variance, probability mass function as well as the histogram of $\mathbf{B}$. Show $\mathbf{B}$ and all calculated quantities.

2. Using histogram modification, modify *your image* so that the resulting image has a histogram that matches $h_B(l)$ as in 1 above. Show your image, the modified image, their histograms and the matching point function. Briefly compare the modified image's histogram to $h_B(l)$.

3. Do the processing I did in the "undoing example" on your image.

4. Uniform quantize your image using $\Delta = 4, 8, 16, 64$. Show the quantized image, its histogram and MSQE in each case.

5. Compand your image using $\Delta = 4, 8, 16, 64$. Show the quantized image, its histogram and MSQE in each case. Compare the results to those obtained in 4.

6. $p_A(l) = [.1, 0, .3, .2, 0, 0, .3, .1]$ and $p_B(k) = [.2, 0, 0, .1, .4, .3]$ for two *images* $\mathbf{A}$ and $\mathbf{B}$. Calculate a point function $g(l)$ such that $C(i,j) = g(A(i,j))$ has histogram $h_C(l)$ that "matches" $h_B(l)$. Assume all images have a total of 10 pixels. Calculate the histogram $h_C(l)$.

References

[1] A. K. Jain, *Fundamentals of Digital Image Processing.* Englewood Cliffs, NJ: Prentice Hall, 1989.