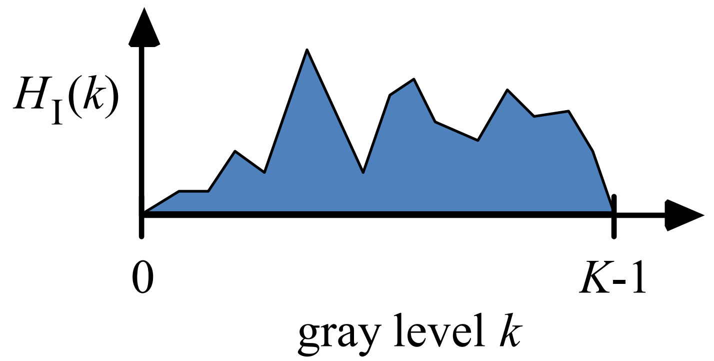
Module 3
Point Operations

- Linear Point Operations
- Nonlinear Point Operations
- Histogram Shaping and Matching
- Arithmetic Image Operations
- Geometric Image Operations

Simple Histogram Operations

- Recall: the **gray-level histogram** $\boldsymbol{H}_{\mathbf{I}}$ of an image $\mathbf{I}$ is a graph of the frequency of occurrence of each gray level in $\mathbf{I}$.
- $\boldsymbol{H}_{\mathbf{I}}$ is a one-dimensional function with domain $0, \ldots, K\text{-}1$.
- $\boldsymbol{H}_{\mathbf{I}}(k) = n$ if gray-level k occurs exactly n times in $\mathbf{I}$ (for each $k = 0, \ldots, K\text{-}1$).



- $\boldsymbol{H}_\mathbf{I}$ contains **no spatial information** - only the relative frequency of intensities (first-order information only).
- Much useful info is obtainable from $\boldsymbol{H}_\mathbf{I}$.
- Image quality is affected (enhanced, modified) by altering $\boldsymbol{H}_\mathbf{I}$.

Average Optical Density (AOD)

- The average intensity of the $M \times N$ image $\mathbf{I}$:

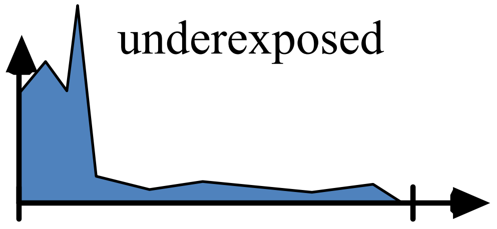
$$\mathrm{AOD}(\mathbf{I}) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathrm{I}(m, n)$$

- Can compute it from the histogram as well:

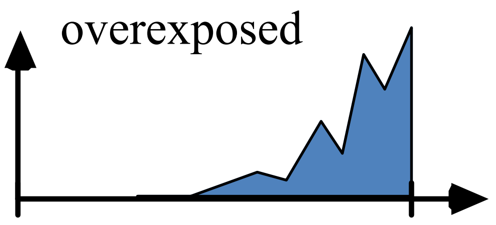
$$\mathrm{AOD}(\mathbf{I}) = \frac{1}{MN} \sum_{k=0}^{K-1} k H_{\mathbf{I}}(k)$$

Average Optical Density

- Examining the histogram can reveal possible errors in the imaging process:



Low AOD



High AOD

- By operating on the histogram, such errors can be ameliorated.

Point Operations

- **Point operation:** a **function** f on **single pixels** in $\mathbf{I}$:
  $\mathrm{J}(m,n) = \mathrm{f}[\mathrm{I}(m,n)],\ 0 \le m \le M\text{-}1,\ 0 \le n \le N\text{-}1$

- The same function f is applied at every (m,n).

- Unlike **local operation**s (e.g. OPEN), does not use the **neighbors** of $\mathrm{I}(m,n)$.

- Point operations **don't** modify **spatial relationships**. They **do** change the **histogram**, and therefore the appearance of the image.

Linear Point Operations

- The simplest class of point operations. They **offset** and **scale** the image intensities.
  - **offset**, also called **bias**, is like the “brightness” control on your television.
  - **scale**, also called **gain**, is like the “contrast” control on your television.
- **offset (or “bias”):**

$$\mathrm{J}(m,n) = \mathrm{I}(m,n) + L$$

- **gain (or “scale”):**

$$\mathrm{J}(m,n) = P \cdot \mathrm{I}(m,n)$$

Image Offset

- If $L > \mathbf{0}$, then $\mathbf{J}$ is a **brightened** version of $\mathbf{I}$. If $L < \mathbf{0}$, then it is a **dimmed** version of $\mathbf{I}$.
- The histogram is shifted by an amount L:

$$\boldsymbol{H}_{\mathbf{J}}(k) = \boldsymbol{H}_{\mathbf{I}}(k\text{-}L)$$

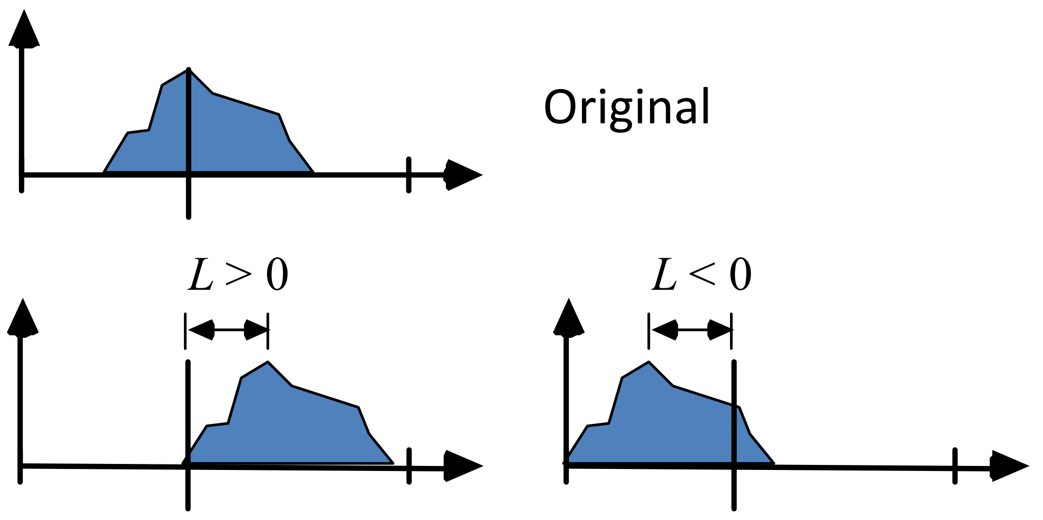


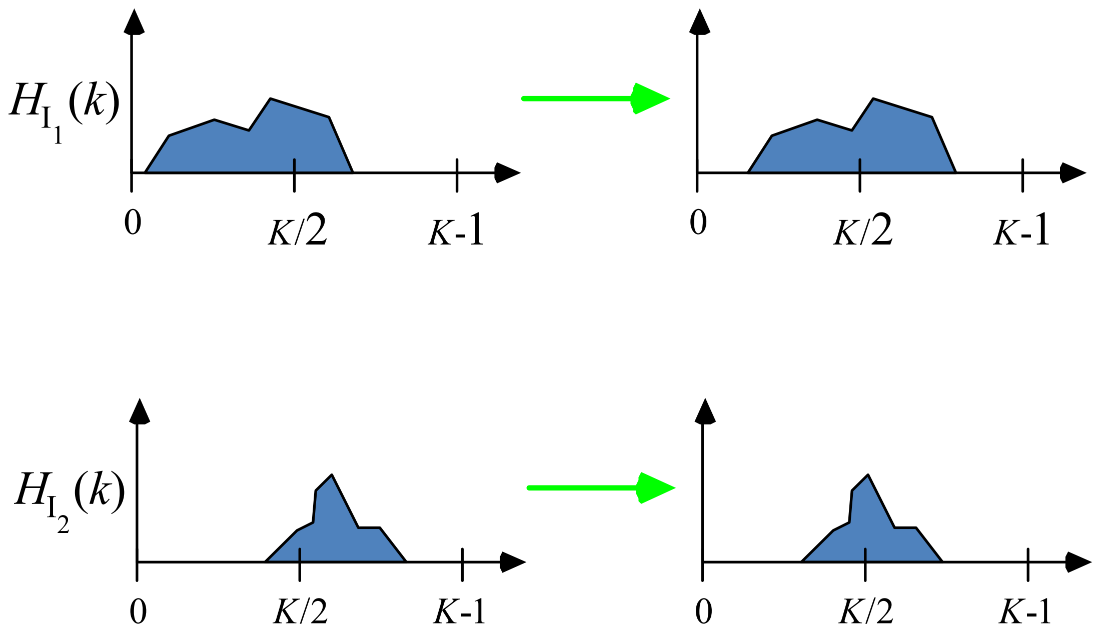
Image Offset Example

- **Compare** multiple images $\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_p$ of the same scene taken with different exposures or lighting conditions.

- A solution: **equalize** the image AODs, e.g., set them all to $K/2$ (for gray scale range $0, \ldots, K\text{-}1$).

- Let $L_q = \text{AOD}(\mathbf{I}_q)$, $q = 1, \ldots, p$. Then equalize via

$$\text{J}_q(m,n) = \text{I}_q(m,n) - L_q + K/2$$



AOD Equalization

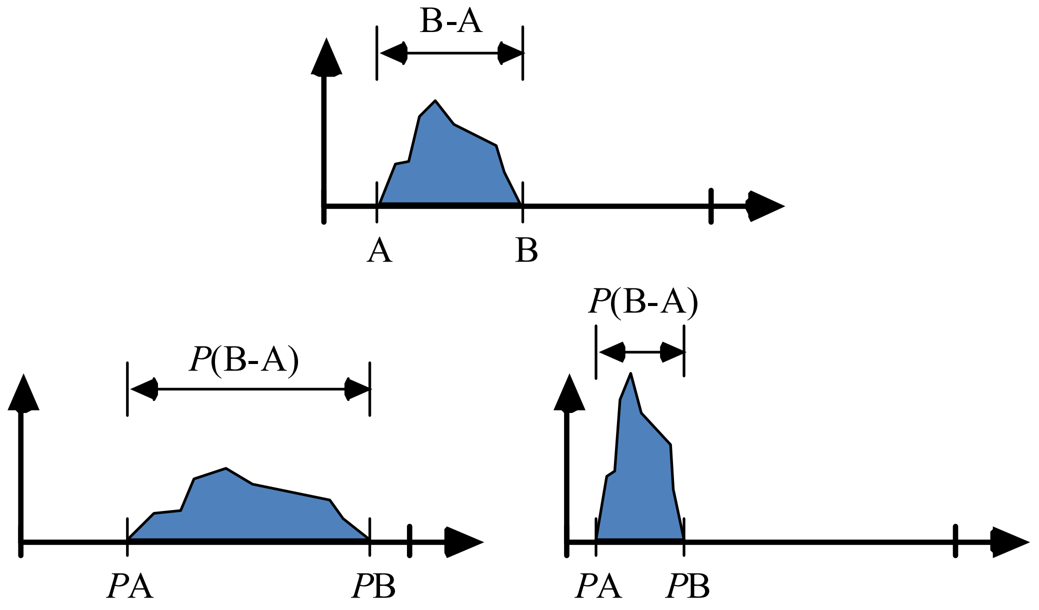
Image Scaling

- $\mathrm{J}(m,n) = P \cdot \mathrm{I}(m,n)$
- If $|P| > \mathbf{1}$, the **intensity range** is **widened**.
- If $|P| < \mathbf{1}$, the **intensity range** is **narrowed**.
- Multiplying by P **stretches** or **compresses** the image histogram by a factor P:

$$\boldsymbol{H}_{\mathbf{J}}(k) = \boldsymbol{H}_{\mathbf{I}}(k/P) \qquad \text{(continuous)}$$

$$\boldsymbol{H}_{\mathbf{J}}(k) = \boldsymbol{H}_{\mathbf{I}}[\mathrm{INT}(k/P)] \quad \text{(discrete)}$$

Image Scaling



- An image with a compressed gray level range generally has a **poor appearance** – washed out, not visually striking.

<u>Linear Point Operations: Offset & Scaling</u>

- Given reals L and P, a **linear point operation** on $\mathbf{I}$ is a function

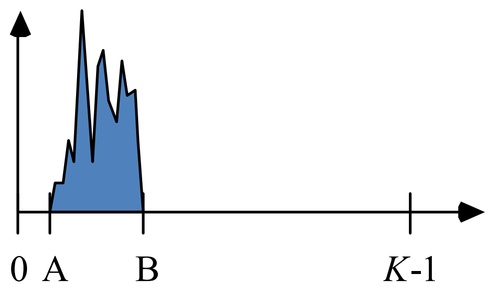
  $$\mathrm{J}(m,n) = P \cdot \mathrm{I}(m,n) + L$$

  comprising both offset and scaling.
- If $P < 0$, the histogram is **reversed**, creating a **negative** image. In this case, usually $P = -1$ and $L = K-1$:

  $$\mathrm{J}(m,n) = (K-1) - \mathrm{I}(m,n) \quad \text{(negative)}$$

Full-Scale Contrast Stretch

- The **most common** linear point operation. Suppose $\mathbf{I}$ has a compressed histogram:



- Let A and B be the min and max gray levels in $\mathbf{I}$.
- Define

$$\mathrm{J}(m,n) = P\cdot \mathrm{I}(m,n) + L$$

such that $P\mathrm{A}+L = 0$ and $P\mathrm{B}+L = (K\text{-}1)$.

- This maps the old gray level A to the new gray level 0 and the old gray level B to the new gray level K-1.

Full-Scale Contrast Stretch

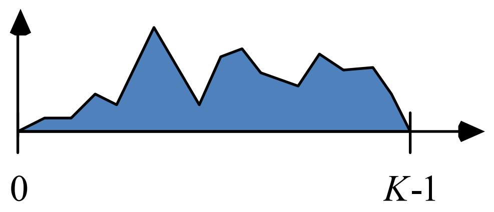
- Solving these **2 equations in 2 unknowns** yields:

$$P = \frac{K-1}{\mathrm{B}-\mathrm{A}} \text{ and } L = -\mathrm{A}\frac{K-1}{\mathrm{B}-\mathrm{A}}$$

or

$$\mathrm{J}(m,n) = \frac{(K-1)}{(\mathrm{B}-\mathrm{A})}\left[\mathrm{I}(m,n) - \mathrm{A}\right]$$

- The result is an image **J** with a full-range histogram:



Full-Scale Contrast Stretch

- **NOTE:** If the resulting values $\mathrm{J}(m,n)$ are **floating point**, then each pixel should be rounded to the nearest **integer** as a final step.
  - This will happen, for example, when we talk about histogram flattening and histogram shaping later in this module.
- **Many** image processing operations will produce an output image having pixels with floating point values and/or pixels with values that are **not** normalized to the range [0, 255].
  - Examples: LTI filters, nonlinear point operations, etc.
  - After such an operation, a full-scale contrast stretch (FSCS) is almost always applied to map the pixels of the output image back into the range [0, 255] for display.

Nonlinear Point Operations

- Now consider **nonlinear point functions** f

$$\mathrm{J}(m,n) = \mathrm{f}\,[\mathrm{I}(m,n)].$$

- A very broad class of functions!
- Commonly used:

$\mathrm{J}(m,n) = \lvert\mathrm{I}(m,n)\rvert$	(magnitude)
$\mathrm{J}(m,n) = [\mathrm{I}(m,n)]^2$	(square-law)
$\mathrm{J}(m,n) = [\mathrm{I}(m,n)]^{1/2}$	(square root)
$\mathrm{J}(m,n) = \log[1+\mathrm{I}(m,n)]$	(logarithm)
$\mathrm{J}(m,n) = \exp[\mathrm{I}(m,n)] = \mathrm{e}^{\mathrm{I}(m,n)}$	(exponential)

- Most of these are special-purpose, for example…

Logarithmic Range Compression

- Small groupings of **very bright pixels** may dominate the perception of an image at the expense of other rich information that is **less bright and less visible**.

- Astronomical images of faint nebulae and galaxies with dominating stars are an excellent example.

The Rosette Nebula

Logarithmic Range Compression

- **Logarithmic transformation**

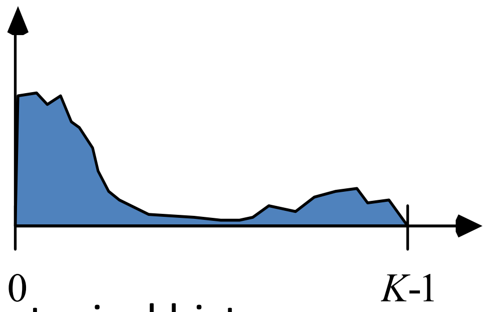
$$\mathrm{J}(m,n) = \log[1+\mathrm{I}(m,n)]$$

  nonlinearly **compresses** and **equalizes** the gray-scales.

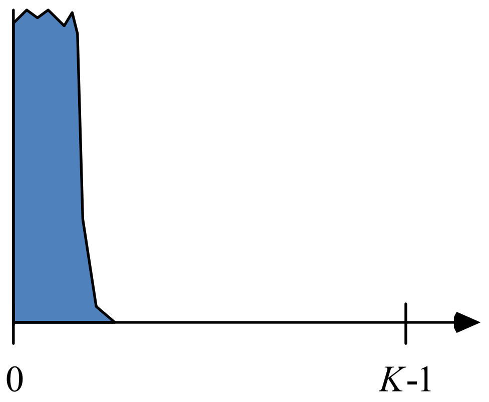
- Bright intensities are compressed much more heavily - thus **faint details** emerge.

Logarithmic Range Compression

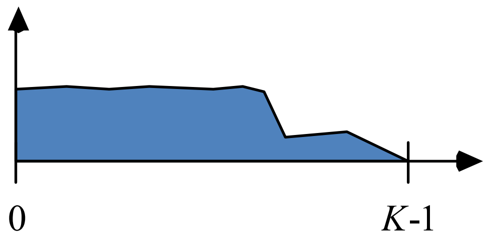
- A full-scale contrast stretch then utilizes the full gray-scale range:



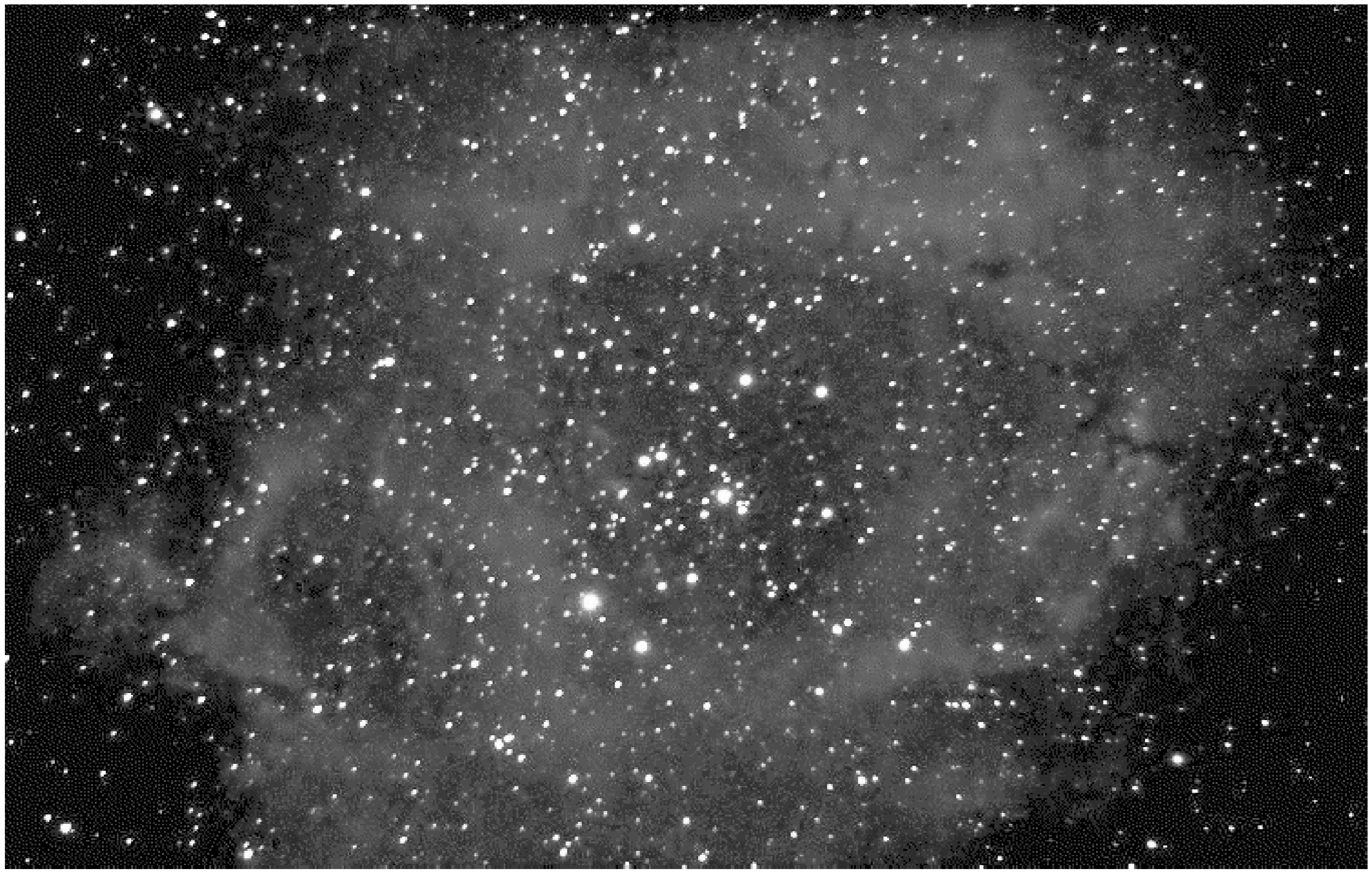
typical histogram



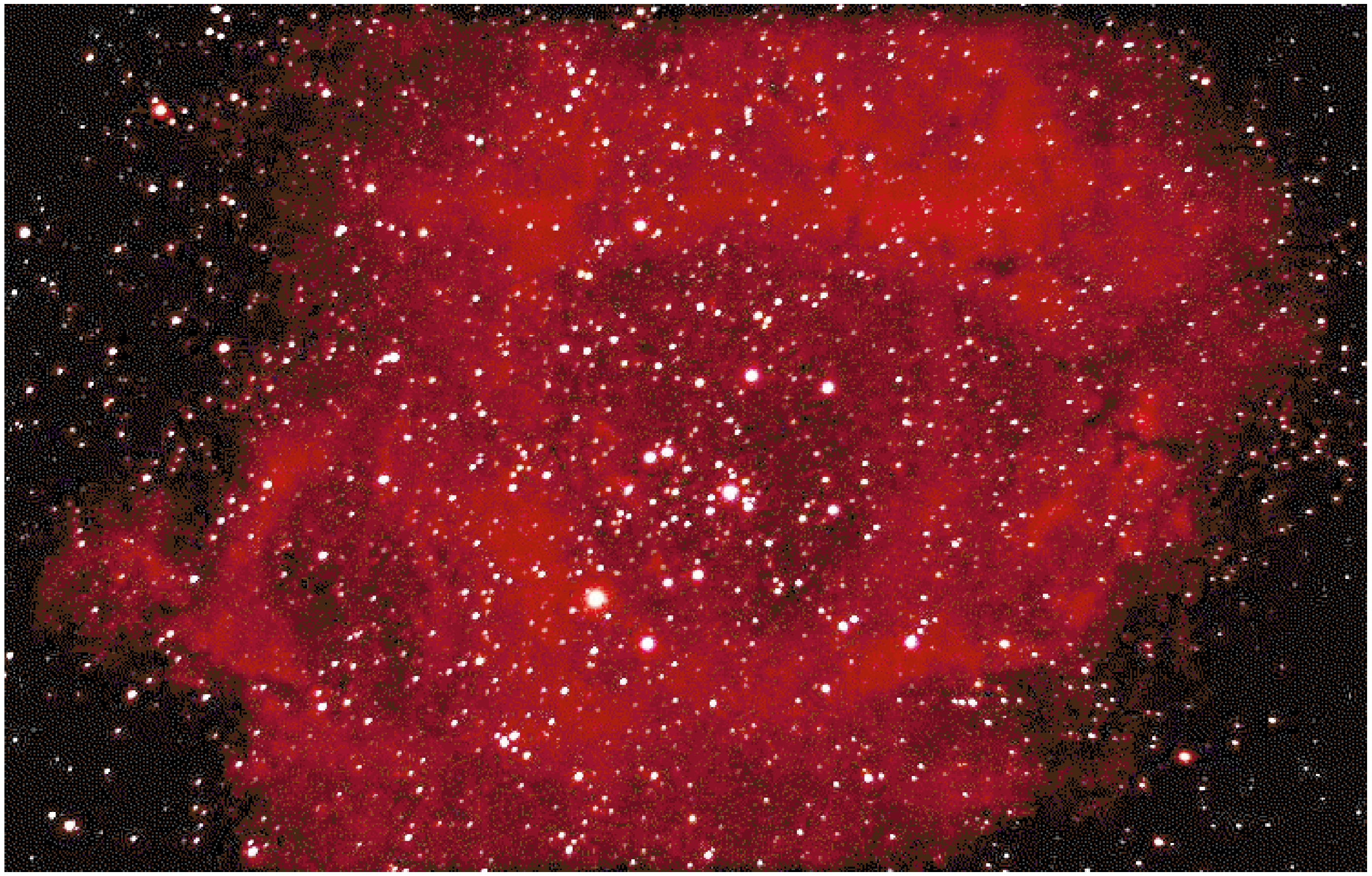
logarithmic transformation



stretched contrast



Contrast Stretched Rosette



Rosette in color

Gamma Correction

- **Monitors** that **display images** and **videos** often have a **nonlinear response**.
- Commonly an **exponential nonlinearity**

$$\text{Display}(m,n) = [\text{I}(m,n)]^{\gamma}$$

- **Gamma correction** is (digital) pre-processing to **correct the nonlinearity**:

$$\text{J}(m,n) = [\text{I}(m,n)]^{(1/\gamma)}$$

- Then

$$\text{Display}(m,n) = [\text{J}(m,n)]^{\gamma} = \text{I}(m,n)$$

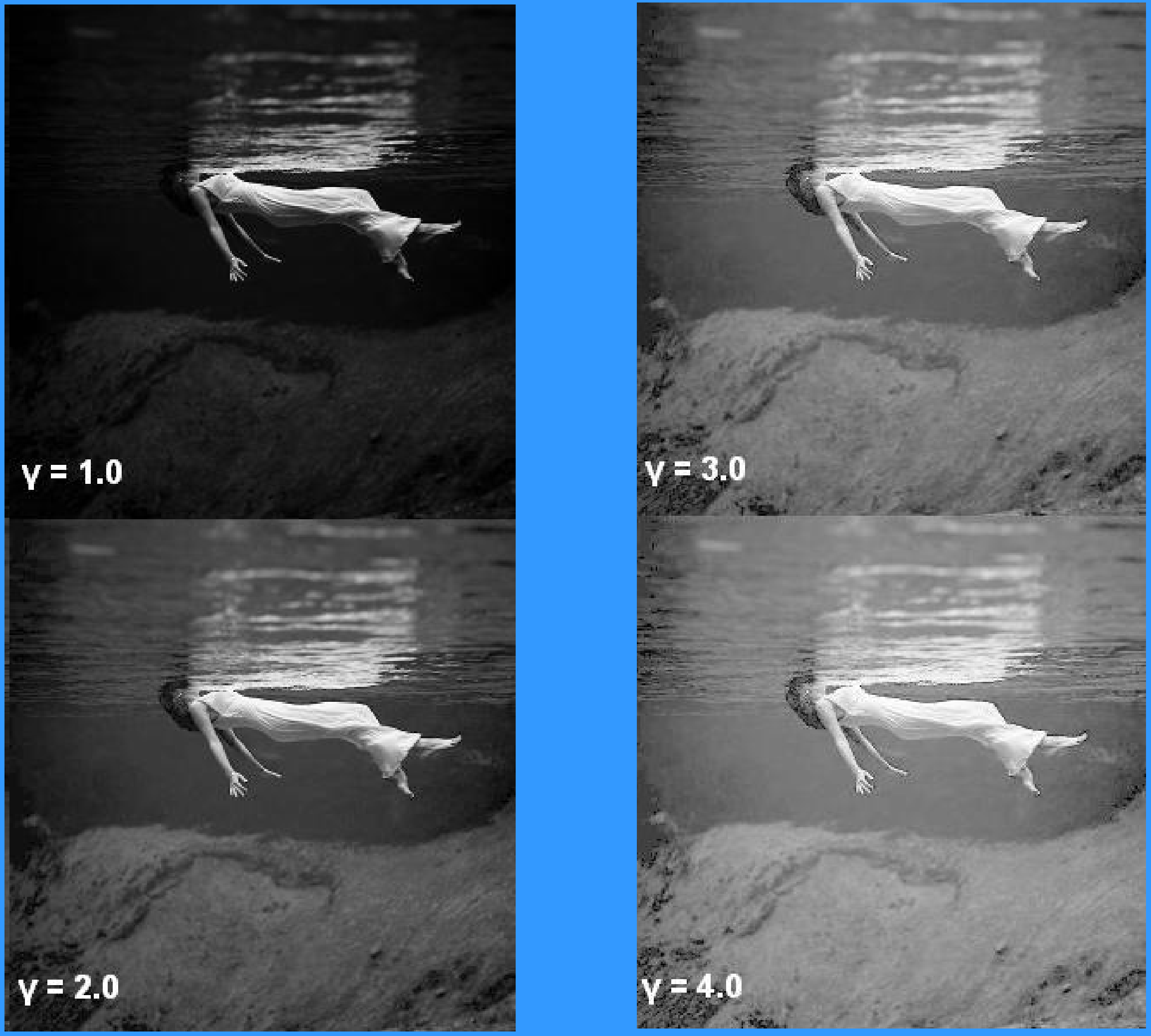
Gamma Correction

- For a CRT (e.g., analog NTSC TV), typically

$$\gamma = 2.2$$

- This is accomplished by **mapping** all **luminances** (or chrominances) to a $[0, 1]$ range.
- Hence **black** (0) and **white** (1) are unaffected.
- **Plasma** and **LCD** have **linear** characteristics**,** hence do not need gamma correction. But many devices that feed them still gamma correct; hence reverse nonlinearity often needed.

Gamma Correction



Histogram Equalization or "Flattening"

- An image with a **flat** histogram makes rich use of the available gray-scale range. This might be an image with

  - Smooth changes in intensity across many gray levels

  - Lots of texture covering many gray levels
- We can obtain an image with an **approximately** flat histogram using nonlinear point operations.
- This is called "histogram flattening" or "histogram equalization."
- It is a special case of "histogram shaping."

Normalized Histogram

- Define the **normalized histogram**:

$$p_{\mathbf{I}}(k) = \frac{1}{MN} H_{\mathbf{I}}(k)\ ;\ k = 0, \ldots, K-1$$

- The normalized histogram **sums to one**:

$$\sum_{k=0}^{K-1} p_{\mathbf{I}}(k) = 1$$

- Note that $p_{\mathbf{I}}(k)$ is the **probability** that gray level k occurs in the image (at any given coordinate): can be interpreted as an empirical probability density function (pdf) for the image.

Cumulative Histogram

- The **cumulative histogram** is

$$P_{\mathbf{I}}(r) = \sum_{k=0}^{r} p_{\mathbf{I}}(k) \; ; \; r = 0, \ldots, K-1$$

  which is **non-decreasing**; also, $P_{\mathbf{I}}(K\text{-}1) = 1$.
- Probabilistic interpretation: at any (m,n):

$$P_{\mathbf{I}}(r) = \Pr\{\mathrm{I}(m,n) \le r\}$$

$$p_{\mathbf{I}}(r) = P_{\mathbf{I}}(r) - P_{\mathbf{I}}(r\text{-}1) \quad ; r = 0, \ldots, K\text{-}1$$

Relation to Probability Theory

- The functions $p_{\mathbf{I}}(x)$ and $P_{\mathbf{I}}(x)$ may be rigorously interpreted as the probability density function (pdf) and cumulative distribution function (cdf) of the image $\mathbf{I}$.
- As with any pdf & cdf, we have $p_{\mathbf{I}}(x) = \mathrm{d}P_{\mathbf{I}}(x)/\mathrm{d}x$
- For theoretical work, it is useful to assume that:
  - the grayscales x are continuous: $x \in \mathbb{R}$.
  - $P_{\mathbf{I}}(x)$ is one-to-one, so that $P_{\mathbf{I}}^{-1}(x)$ exists.
- We'll describe histogram flattening and shaping for these assumptions, then adapt them for the more practical discrete case.

Histogram Equalization

- Let $\mathbf{I}$ be an image with continuous gray scales $x \in \mathbb{R}$.
- Let $p_{\mathbf{I}}(x)$ and $P_{\mathbf{I}}(x)$ be the pdf and cdf of $\mathbf{I}$.
- Assume that $P_{\mathbf{I}}^{-1}(x)$ exists.
- Define the histogram equalized image $\mathbf{J}$ by

$$\mathbf{J} = P_{\mathbf{I}}(\mathbf{I}) \quad \Big(\mathrm{J}(m,n) = P_{\mathbf{I}}[\mathrm{I}(m,n)] \ \forall \ (m,n) \Big)$$

- **Claim:** $\mathbf{J}$ has a flat histogram, i.e.,

$$p_{\mathbf{J}}(x) = 1$$

Proof: Since $p_{\mathbf{I}}(x) = \frac{d}{dx} P_{\mathbf{I}}(x),$ we have from the fundamental theorem of calculus that

$$\mathrm{J}(m,n) = P_{\mathbf{I}}[\mathrm{I}(m,n)] = \int_{-\infty}^{\mathrm{I}(m,n)} p_{\mathbf{I}}(\theta)\, d\theta.$$

Now, the cdf of image $\mathbf{J}$ is given by

$$\begin{aligned} P_{\mathbf{J}}(x) &= \Pr\{\mathrm{J}(m,n) \le x\} \\ &= \Pr\{P_{\mathbf{I}}[\mathrm{I}(m,n)] \le x\}. \qquad (1) \end{aligned}$$

If we do a one-to-one positive semidefinite mapping to both sides of the inequality in (1), it will not change any of the probabilities. Since $P_{\mathbf{I}}^{-1}(x)$ is such a mapping, we can rewrite (1) as

$$\begin{aligned} P_{\mathbf{J}}(x) &= \Pr\{P_{\mathbf{I}}^{-1}[P_{\mathbf{I}}[\mathrm{I}(m,n)]] \le P_{\mathbf{I}}^{-1}(x)\} \\ &= \Pr\{\mathrm{I}(m,n) \le P_{\mathbf{I}}^{-1}(x)\}. \qquad (2) \end{aligned}$$

To make this easier to understand, let’s temporarily write ☺ for $P_{\mathbf{I}}^{-1}(x)$.

Then (2) becomes $P_{\mathbf{J}}(x) = \Pr\{\mathrm{I}(m,n) \le ☺\}$.

But from the definition of the cdf on p. 29, this is the same as

$$P_{\mathbf{J}}(x) = P_{\mathbf{I}}(☺).$$

Plugging back in the definition of ☺ from p. 32, we have

$$P_{\mathbf{J}}(x) = P_{\mathbf{I}}[P_{\mathbf{I}}^{-1}(x)] = x.$$

Since the pdf is the derivative of the cdf, we have then that

$$p_{\mathbf{J}}(x) = \frac{d}{dx} P_{\mathbf{J}}(x) = \frac{d}{dx} x = 1.$$

So $p_{\mathbf{J}}(x) = 1$ and the histogram of $\mathbf{J}$ is flat, as claimed. ■

Histogram Shaping

- More generally, suppose we would like to transform the image $\mathbf{I}$ into a new image $\mathbf{J}$ with pdf $p_{\mathbf{J}}(x)$ and cdf $P_{\mathbf{J}}(x)$ that are given by some particular desired functions $q(x)$ and $Q(x)$ with $q(x) = \mathrm{d}Q(x)/\mathrm{d}x$.
- This is called **histogram shaping**.
- Histogram equalization (flattening) is the special case where $q(x) = 1$ and $Q(x) = x$
  - which also means that $Q^{-1}(x) = Q(x) = x$.

- For **arbitrary** $q(x)$ and $Q(x)$, the histogram shaping algorithm is given by

$$\mathbf{J} = Q^{-1}[P_{\mathbf{I}}(\mathbf{I})]$$

$$\left(\mathrm{J}(m,n) = Q^{-1}\{P_{\mathbf{I}}[\mathrm{I}(m,n)]\} \ \forall \ (m,n)\right)$$

- This works since the cdf of $\mathbf{J}$ is:

$$\begin{aligned}\Pr\{\mathrm{J}(m,n) \le x\} &= \Pr\{Q^{-1}[P_{\mathbf{I}}[\mathrm{I}(m,n)]] \le x\} \\ &= \Pr\{P_{\mathbf{I}}[\mathrm{I}(m,n)] \le Q(x)\} \\ &= \Pr\{\mathrm{I}(m,n) \le P_{\mathbf{I}}^{-1}[Q(x)]\} \\ &= P_{\mathbf{I}}\{P_{\mathbf{I}}^{-1}[Q(x)]\} = Q(x)\end{aligned}$$

- For a practical digital image $\mathbf{I}$ with a discrete histogram, all of this can only be **approximated**.

Practical Histogram Flattening

- To approximately **flatten** the histogram of the digital image $\mathbf{I}$:
- Define the **cumulative histogram image**

$$\mathbf{J} = P_{\mathbf{I}}(\mathbf{I})$$

so that

$$\mathrm{J}(m,n) = P_{\mathbf{I}}[\mathrm{I}(m,n)] = \sum_{k=0}^{\mathrm{I}(m,n)} p_{\mathbf{I}}(k)$$

- This is the cumulative histogram of image $\mathbf{I}$ evaluated at the gray level of pixel (m,n).

Practical Histogram Flattening

- Note that

$$0 \le \mathrm{J}(m,n) \le 1$$

- The elements of $\mathbf{J}$ are **approximately** linearly distributed between 0 and 1.
- Finally, let $\mathbf{K} = \mathrm{FSCS}(\mathbf{J})$ to get the histogram equalized image with full scale contrast.

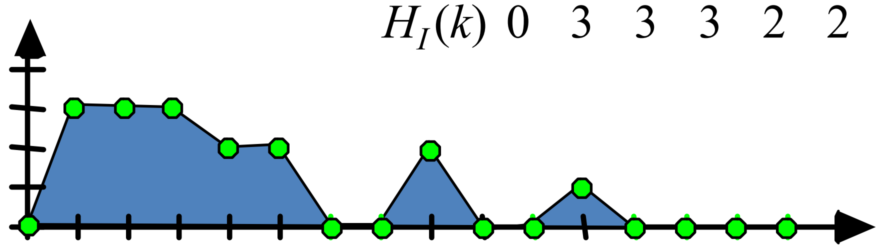
Histogram Flattening Example

- Let $\mathbf{I}$ be a 4×4 image with gray-level range $\{0, ..., 15\}$ (K-1 = 15):

$$\mathbf{I} = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 3 & 4 \\ \hline 2 & 5 & 3 & 2 \\ \hline 8 & 1 & 8 & 2 \\ \hline 4 & 5 & 3 & 11 \\ \hline \end{array}$$

- The histogram:

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$H_I(k)$	0	3	3	3	2	2	0	0	2	0	0	1	0	0	0	0



- The normalized histogram…

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$p_I(k)$	0	$\frac{3}{16}$	$\frac{3}{16}$	$\frac{3}{16}$	$\frac{2}{16}$	$\frac{2}{16}$	0	0	$\frac{2}{16}$	0	0	$\frac{1}{16}$	0	0	0	0
$P_I(k)$	0	$\frac{3}{16}$	$\frac{6}{16}$	$\frac{9}{16}$	$\frac{11}{16}$	$\frac{13}{16}$	$\frac{13}{16}$	$\frac{13}{16}$	$\frac{15}{16}$	$\frac{15}{16}$	$\frac{15}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{16}{16}$

- The equalized image $\mathbf{J}$ is computed and contrast stretched to obtain the final result $\mathbf{K}$ (as on p. 3.15 and after **rounding**):

$\mathbf{J} =$

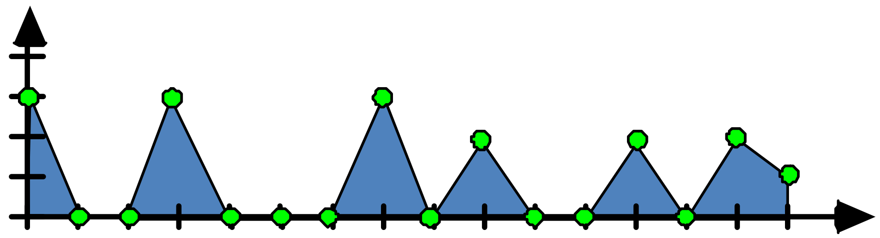
3/16	3/16	9/16	11/16
6/16	13/16	9/16	6/16
15/16	3/16	15/16	6/16
11/16	13/16	9/16	16/16

$\mathbf{K} =$

0	0	7	9
3	12	7	3
14	0	14	3
9	12	7	15

- The new **flattened** histogram looks like this:

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$H_I(k)$	3	0	0	3	0	0	0	3	0	2	0	0	2	0	2	1



- This is an **approximation** of the continuous algorithm.
- What it basically does is move the bins around to space them out more equally – making the histogram **more flat**.
- Because of the rounding, some bins may also get combined.
- The **spaces** that appear between bins in the new histogram are characteristic of equalization; for a typical image like Lena, the new histogram will often have a “comb-like” appearance.

Histogram Shaping

- The idea is to create a new image $\mathbf{K}$ with an approximate **specified** histogram shape, such as a triangle or bell-shaped curve.

- Let $H_{\mathbf{K}}(k)$ be the **desired** histogram shape, with corresponding normalized values (probabilities) $p_{\mathbf{K}}(k)$.

Histogram Shaping

- Define the cumulative histogram image as before:

$$\mathrm{J}(m,n) = P_{\mathbf{I}}\left[\mathrm{I}(m,n)\right] = \sum_{k=0}^{\mathrm{I}(m,n)} p_{\mathbf{I}}(k)$$

- Also define the cumulative probabilities

$$P_{\mathbf{K}}(r) = \sum_{k=0}^{r} p_{\mathbf{K}}(k)$$

Histogram Shaping Algorithm

- Let $r(m,n)$ denote the **minimum value** of r such that

$$P_{\mathbf{K}}(r) \geq \mathrm{J}(m,n)$$

- Then take $\mathrm{K}(m,n) = r(m,n)$.
- This is a **convention** for approximating

$$\mathrm{K}(m,n) = P_{\mathbf{K}}{}^{-1}[\mathrm{J}(m,n)]$$

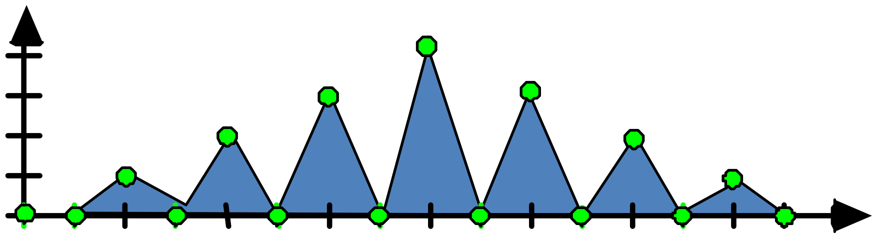
Histogram Shaping Example

- Consider the same image as in the last example. We had

$$\mathbf{I} = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 3 & 4 \\ \hline 2 & 5 & 3 & 2 \\ \hline 8 & 1 & 8 & 2 \\ \hline 4 & 5 & 3 & 11 \\ \hline \end{array} \qquad \mathbf{J} = \begin{array}{|c|c|c|c|}\hline 3/16 & 3/16 & 9/16 & 11/16 \\ \hline 6/16 & 13/16 & 9/16 & 6/16 \\ \hline 15/16 & 3/16 & 15/16 & 6/16 \\ \hline 11/16 & 13/16 & 9/16 & 16/16 \\ \hline \end{array}$$

- Fit this to a (triangular) histogram:

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$H_{\mathbf{K}}(k)$	0	0	1	0	2	0	3	0	4	0	3	0	2	0	1	0
$p_{\mathbf{K}}(k)$	0	0	$\frac{1}{16}$	0	$\frac{2}{16}$	0	$\frac{3}{16}$	0	$\frac{4}{16}$	0	$\frac{3}{16}$	0	$\frac{2}{16}$	0	$\frac{1}{16}$	0



- The cumulative probabilities:

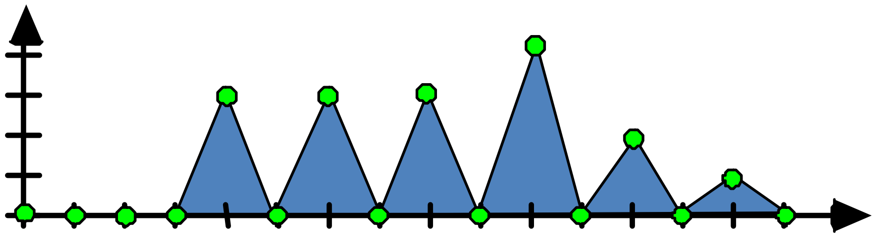
r	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P_{\mathbf{K}}(r)$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{3}{16}$	$\frac{3}{16}$	$\frac{6}{16}$	$\frac{6}{16}$	$\frac{10}{16}$	$\frac{10}{16}$	$\frac{13}{16}$	$\frac{13}{16}$	$\frac{15}{16}$	$\frac{15}{16}$	$\frac{16}{16}$	$\frac{16}{16}$

- Using this table, a **careful** visual inspection of $\mathbf{J}$ then lets us form the new image $\mathbf{K}$.

$$\mathbf{K} = \begin{array}{|c|c|c|c|}\hline 4 & 4 & 8 & 10 \\ \hline 6 & 10 & 8 & 6 \\ \hline 12 & 4 & 12 & 6 \\ \hline 10 & 10 & 8 & 14 \\ \hline \end{array}$$

- Here's the new histogram:

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$H_{\mathbf{K}}(k)$	0	0	0	0	3	0	3	0	3	0	4	0	2	0	1	0



Histogram Matching

- A **special case** of histogram shaping.
- The histogram of image $\mathbf{I}$ is matched to the histogram of **another** image $\mathbf{I}'$.
- The procedure is identical, once the cumulative probabilities are computed for the model image $\mathbf{I}'$.
- Useful application: **Comparing** similar images of the same scene obtained under different conditions (e.g., lighting, time of day). Extends the concept of equalizing AOD described earlier.

Arithmetic Image Operations

- Suppose we have two $M \times N$ images $\mathbf{I}_1$ and $\mathbf{I}_2$. The basic arithmetic operations are:

Pointwise Addition

$$\mathbf{J} = \mathbf{I}_1 + \mathbf{I}_2 \text{ ; if } \mathrm{J}(m,n) = \mathrm{I}_1(m,n) + \mathrm{I}_2(m,n)$$

Pointwise Subtraction

$$\mathbf{J} = \mathbf{I}_1 - \mathbf{I}_2 \text{ ; if } \mathrm{J}(m,n) = \mathrm{I}_1(m,n) - \mathrm{I}_2(m,n)$$

Pointwise Multiplication

$$\mathbf{J} = \mathbf{I}_1 \otimes \mathbf{I}_2 \text{ ; if } \mathrm{J}(m,n) = \mathrm{I}_1(m,n) \times \mathrm{I}_2(m,n)$$

Pointwise Division

$$\mathbf{J} = \mathbf{I}_1 \Delta \mathbf{I}_2 \text{ ; if } \mathrm{J}(m,n) = \mathrm{I}_1(m,n) / \mathrm{I}_2(m,n)$$

- The operations $\otimes$ and Δ are mainly useful when manipulating **Fourier Transform matrices**.

Applying Arithmetic Operations

- We will look at two simple but **important** applications of arithmetic operations on images:

  - Frame averaging for noise reduction
  - Frame differencing for motion detection

- Note: the individual images in a video sequence are usually called **frames**.

Frame Averaging for Noise Reduction

- An image $\mathbf{J}$ is often corrupted by **additive noise**:

  Surface radiation scatter

  Noise in the camera

  Thermal noise in a computer circuit

  Channel transmission noise
- **Model:** The received image $\mathbf{J}$ is the sum of a **desired** image $\mathbf{I}$ and an **undesired** noise image $\mathbf{N}$:

$$\mathbf{J} = \mathbf{I} + \mathbf{N},$$

- the elements $\mathrm{N}(m,n)$ of $\mathbf{N}$ are **random variables**.

Zero-Mean Noise Images

- We won't explore the mathematics of 2D random processes in any detail yet.
- For now, just assume that each pixel $\mathrm{N}(m,n)$ is a random variable and that they all have **zero mean**.
- A sequence of noise images $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_M$ can be considered as a set of realizations of these random variables… just like repeatedly tossing a coin can be considered as realizations of a single random variable.
- If M is large enough, then the time average of each pixel (m,n) is expected to converge towards the mean of $\mathrm{N}(m,n)$, which is zero.
- So pointwise time averaging of the noise images $\mathbf{N}_i$ should give

$$\frac{1}{M}\sum_{i=1}^{M}\mathbf{N}_i \approx \mathbf{0} \quad \text{(image of zeroes)}$$

Frame Averaging for Noise Reduction

- Suppose we acquire M images $\mathbf{J}_1, \ldots, \mathbf{J}_M$ of the **same scene**

  in rapid succession, so that there is **little motion** between frames;

  or there is **no motion** in the scene.

- However, the frames are **noisy:**

$$\mathbf{J}_i = \mathbf{I}_i + \mathbf{N}_i \text{ for } i = 1, \ldots, M.$$

Frame Averaging for Noise Reduction

- Suppose we **average** the frames:

$$\mathbf{J} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{J}_i = \frac{1}{M}\sum_{i=1}^{M}\left[\mathbf{I}_i + \mathbf{N}_i\right] = \frac{1}{M}\sum_{i=1}^{M}\mathbf{I}_i + \frac{1}{M}\sum_{i=1}^{M}\mathbf{N}_i$$

- Since $\mathbf{I}_1 \approx \cdots \approx \mathbf{I}_M \approx \mathbf{I}$, we have that

$$\frac{1}{M}\sum_{i=1}^{M}\mathbf{I}_i \approx \mathbf{I} \quad \text{and also} \quad \frac{1}{M}\sum_{i=1}^{M}\mathbf{N}_i \approx \mathbf{0}$$

Frame Averaging for Noise Reduction

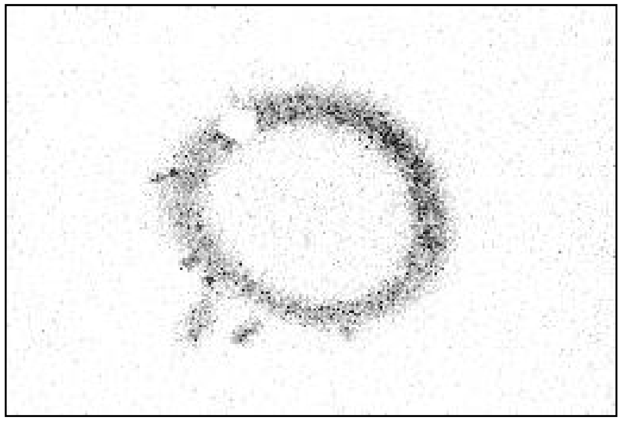
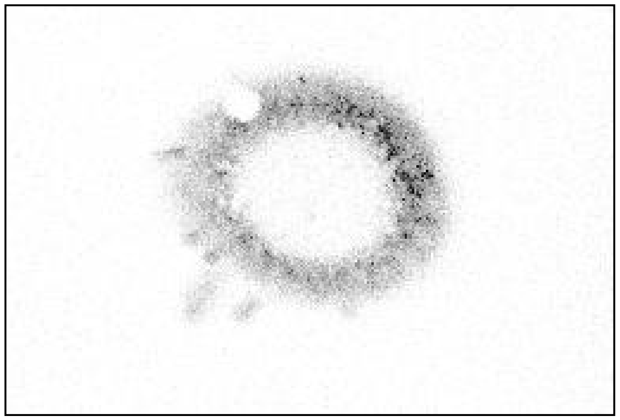
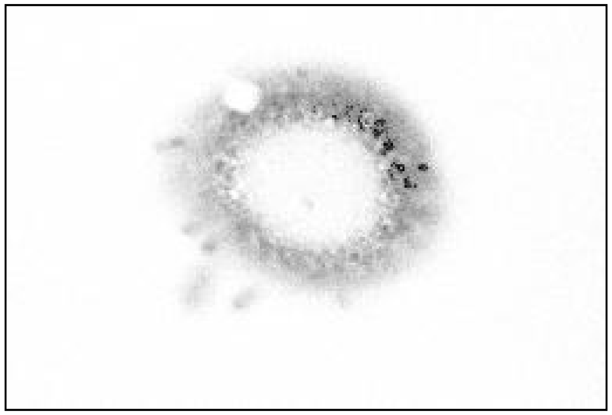
- So

$$\mathbf{J} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{I}_i + \frac{1}{M}\sum_{i=1}^{M}\mathbf{N}_i \approx \mathbf{I} + \mathbf{0} = \mathbf{I}$$

if enough frames (M) are averaged together.

Frame Averaging Example

Macroalga Valonia microphysa, imaged with laser scanning confocal microscope (LSCM). The ring is chlorophyll fluorescing under Ar laser excitation.

Single frame (no averaging)

Average of four frames

Average of sixteen frames.

Motion Detection by Frame Differencing

- Often it is of interest to detect **object motion** between frames.
- **Many applications**: video compression, target recognition, tracking, security, surveillance, automated inspection, etc.
- Frame differencing is a **simple** but **effective** approach.

Motion Detection by Frame Differencing

- Let $\mathbf{I}_1, \mathbf{I}_2$ be consecutive frames in close time proximity, e.g., from a video camera.
- Assumptions: there is a moving object and the background doesn't change much.
- Form the absolute difference image.

$$\mathbf{J} = |\mathbf{I}_1 - \mathbf{I}_2|$$

- The background is approximately subtracted out.
- Large values typically remain at both the old **and** new locations of the object.
- Presence/absense of motion can be detected by thresholding $\mathbf{J}$.

Geometric Image Operations

- Certain geometric image operations are widely used in image processing.
- Many concepts also overlap **computer graphics**.
- **Image processing** is primarily concerned with **correcting** or **improving** images of the **real world**.
- **Computer graphics** is primarily concerned with **creating** images of an **unreal world**.

Geometric Image Operations

- In a sense, geometric image operations are the **opposite** of point operations: they modify **spatial positions** but not **gray levels**.
- A geometric operation generally requires **two steps:**

  - A spatial mapping of image coordinates

  - Interpolation

Spatial Mapping

- The image coordinates are mapped to create the new image:

$$J(m,n) = I(m',n') = I[a(m,n), b(m,n)]$$

- But the coordinates $a(m,n)$ and $b(m,n)$ **might not be integers!**
- For example if

$$a(m,n) = m/3.5 \text{ and } b(m,n) = n/4.5$$

then $J(m,n) = I(m/3.5, n/4.5)$, which are undefined image coordinates!

Interpolation

- It is necessary to **interpolate** non-integer coordinates $\mathrm{a}(m,n)$ and $\mathrm{b}(m,n)$ to integer values.
- We will look at two methods:

  - Nearest neighbor interpolation

  - Bilinear interpolation

Nearest Neighbor Interpolation

- Simple-minded.
- The geometrically transformed coordinates are mapped to the **nearest integer coordinates**:

  $J(m,n) = I\{\mathrm{INT}[a(m,n)+0.5], \mathrm{INT}[b(m,n)+0.5]\}$
- **Serious drawback**: Pixel replication can occur, creating a “jagged” edge effect in non-smooth regions.

Nearest Neighbor Interpolation

Caveat

- If for some coordinate (m,n) either

  $$\text{INT}[\text{a}(m,n)+0.5] < 0 \quad \text{or} \quad \text{INT}[\text{b}(m,n)+0.5] < 0$$

  or

  $$\text{INT}[\text{a}(m,n)+0.5] > \text{N-1} \quad \text{or} \quad \text{INT}[\text{b}(m,n)+0.5] > \text{N-1}$$

  then

  $$\text{J}(m,n) = \text{I}\{\text{INT}[\text{a}(m,n)+0.5], \text{INT}[\text{b}(m,n)+0.5]\}$$

  is not defined.

- We usually set $\text{J}(m,n) = 0$ in these cases.

Bilinear Interpolation

- Produces a **much smoother** interpolation than nearest neighbor approach.
- Given four neighboring image coordinates $\mathrm{I}(m_0,n_0)$, $\mathrm{I}(m_1,n_1)$, $\mathrm{I}(m_2,n_2)$, and $\mathrm{I}(m_3,n_3)$, the new image $\mathrm{J}(m,n)$ is computed as

$$\mathrm{J}(m,n) = \mathrm{A}_0 + \mathrm{A}_1 \cdot m + \mathrm{A}_2 \cdot n + \mathrm{A}_3 \cdot m \cdot n$$

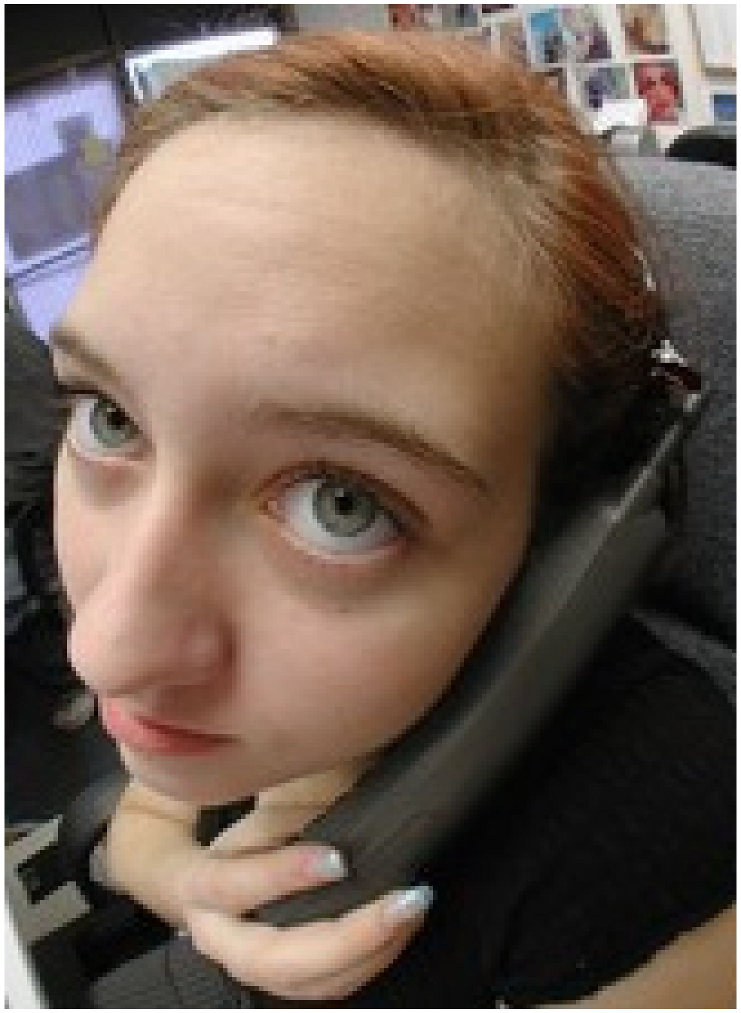
where the **bilinear weights** A_0, A_1, A_2, A_3 are found by solving

$$\begin{bmatrix} \mathrm{A}_0 \\ \mathrm{A}_1 \\ \mathrm{A}_2 \\ \mathrm{A}_3 \end{bmatrix} = \begin{bmatrix} 1 & m_0 & n_0 & m_0 n_0 \\ 1 & m_1 & n_1 & m_1 n_1 \\ 1 & m_2 & n_2 & m_2 n_2 \\ 1 & m_3 & n_3 & m_3 n_3 \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{I}(m_0,n_0) \\ \mathrm{I}(m_1,n_1) \\ \mathrm{I}(m_2,n_2) \\ \mathrm{I}(m_3,n_3) \end{bmatrix}$$

A **linear combination** of the four closest values. The **best planar fit** to the four nearest values.

Basic Geometric Transformations

- The basic geometric transformations are
  - Translation
  - Rotation
  - Zoom



“Fish-Eye” geometric distortion
(more complex)

Translation

- The simplest geometric operation - requires no interpolation. Let

$$\mathrm{a}(m,n) = m - m_0, \qquad \mathrm{b}(m,n) = n - n_0$$

where (m_0,n_0) are **constants**. In this case

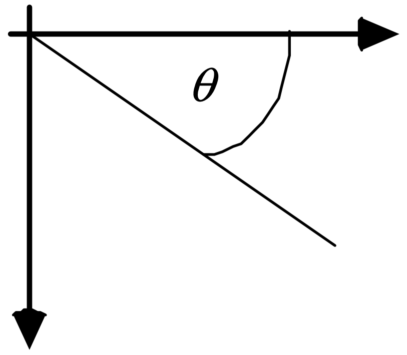
$$\mathrm{J}(m,n) = \mathrm{I}(m - m_0, n - n_0)$$

a shift of the image by amounts (m_0,n_0) in the (row, column) directions.

Rotation

- Clockwise rotation of an image by an angle θ relative to the horizontal axis is accomplished by the following transformation:

$$a(m,n) = m\cos\theta - n\sin\theta$$
$$b(m,n) = m\sin\theta + n\cos\theta$$



- Simplest cases:

$\theta = 90° : [a(m,n), b(m,n)] = (-n, m)$

$\theta = 180° : [a(m,n), b(m,n)] = (-m, -n)$

$\theta = -90° : [a(m,n), b(m,n)] = (n, -m)$

- A translation is **usually** require afterwards to obtain coordinate values in the nominal range, e.g.

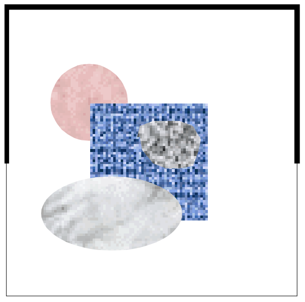
$(M-n, m)$, $(M-m, N-n)$, $(n, N-m)$

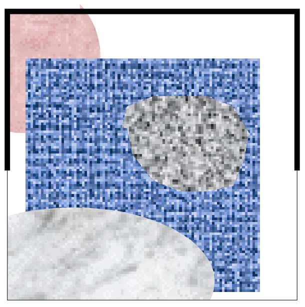
Zoom

- Zooming **magnifies** an image by the mapping functions

  $$a(m,n) = m/c \quad \text{and} \quad b(m,n) = n/d$$

  where $c \geq 1$ and $d \geq 1$.
- For large magnifications, a zoom image will look "blotchy" if nearest neighbor interpolation is used. A bilinear interpolation works quite well.


original


2x zoomed

Comments

- We will next look at **spatial frequency** analysis and processing of images.
- These methods effect both gray-levels and shapes… onward to **Module 4**!