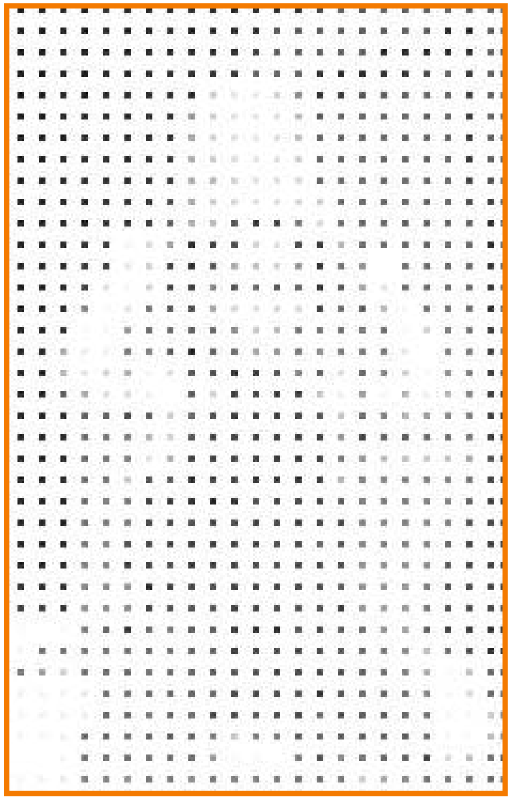
Image Processing

COS 426

What is a Digital Image?

A digital image is a discrete array of samples representing a continuous 2D function

Continuous function



Discrete samples

Limitations on Digital Images

- Spatial discretization
- Quantized intensity
- Approximate color (RGB)
- (Temporally discretized frames for digital video)

Image Processing

- Changing intensity/color
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Add random noise
- Filtering over neighborhoods
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph

Similar to Analog / Continuous

- Changing intensity/color
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Add random noise
- Filtering over neighborhoods
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph

Account for Limitations of Digital

- Changing intensity/color
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Add random noise
- Filtering over neighborhoods
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph

Inherently New Digital Operations

- Changing intensity/color
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Add random noise
- Filtering over neighborhoods
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering

Digital Image Processing

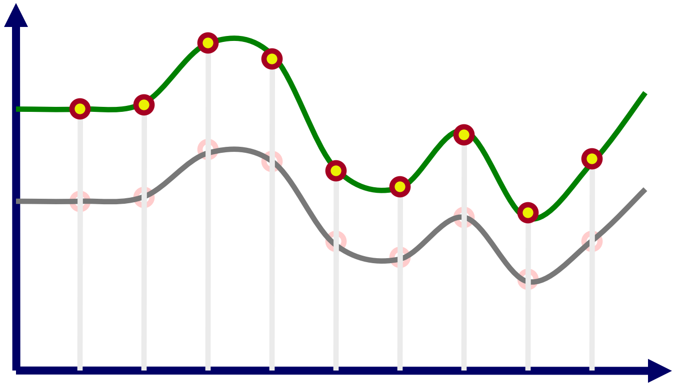
- Changing intensity/color
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Add random noise
- Filtering over neighborhoods
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering

Adjusting Brightness

- Simply scale pixel components
  - o Must clamp to range, e.g. [0..1] or [0..255]


Original


Brighter



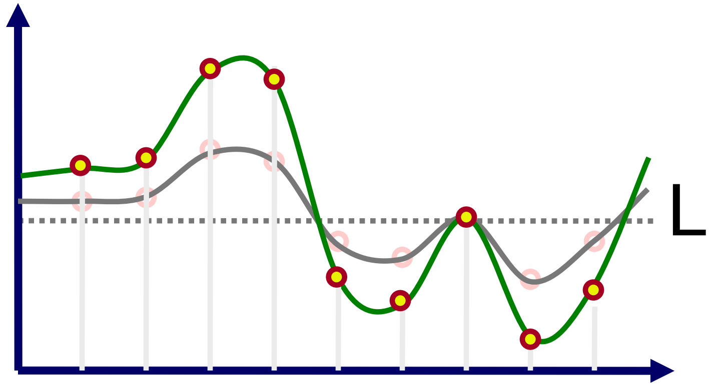
Note: this is “contrast” on your monitor!
“Brightness” adjusts black level (offset)

Adjusting Contrast

- Compute mean luminance L for all pixels
  - o luminance = 0.30*r + 0.59*g + 0.11*b
- Scale deviation from L for each pixel component
  - o Must clamp to range (e.g., 0 to 1)

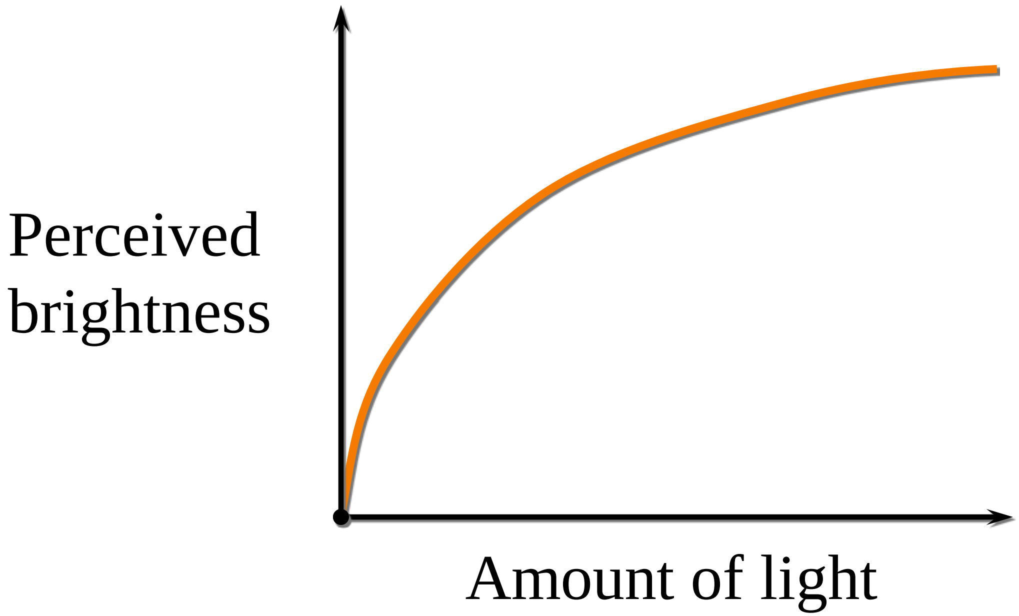
Original

More Contrast



Digression: Perception of Intensity

- Perception of intensity is nonlinear

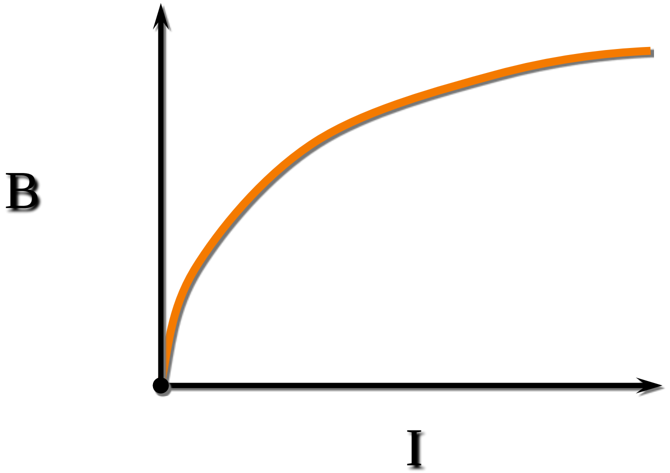


Modeling Nonlinear Intensity Response

- Brightness (*B*) usually modeled as a logarithm or power law of intensity (*I*)

$$B = k \log I$$

$$B = I^{1/3}$$



- Exact curve varies with ambient light, adaptation of eye

Cameras

- Original cameras based on Vidicon obey power law for Voltage (V) vs. Intensity (I):

$$V = I^{\gamma}$$

$$\gamma \approx 0.45$$

CRT Response

- Power law for Intensity (*I*) vs. applied voltage (*V*)

$$I = V^{\gamma}$$

$$\gamma \approx 2.5$$

- Vidicon + CRT = almost linear!
- Other displays (e.g. LCDs) contain electronics to emulate this law

CCD Cameras

- Camera gamma codified in NTSC standard
- CCDs have linear response to incident light
- Electronics to apply required power law
- So, pictures from most cameras (including digital still cameras) will have $\gamma = 0.45$
  - sRGB standard: partly-linear, partly power-law curve well approximated by $\gamma = 1 / 2.2$

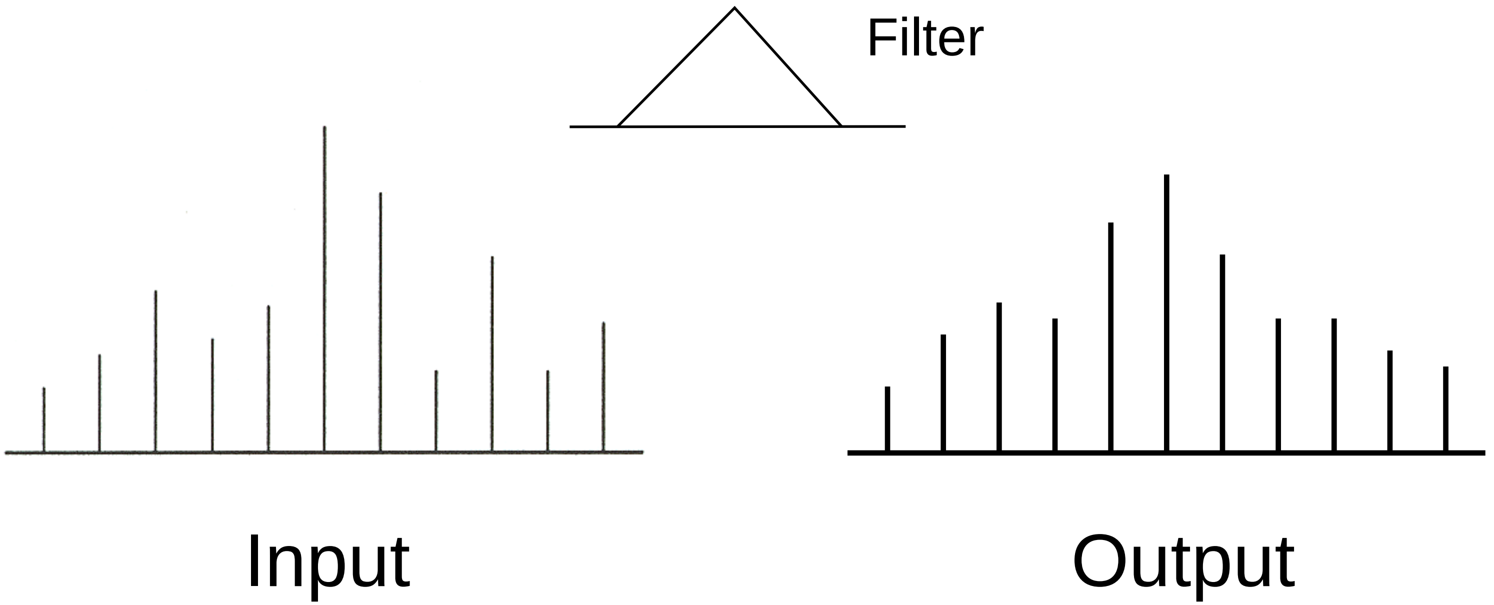
Digital Image Processing

- Changing intensity/color
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Add random noise
- Filtering over neighborhoods
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering

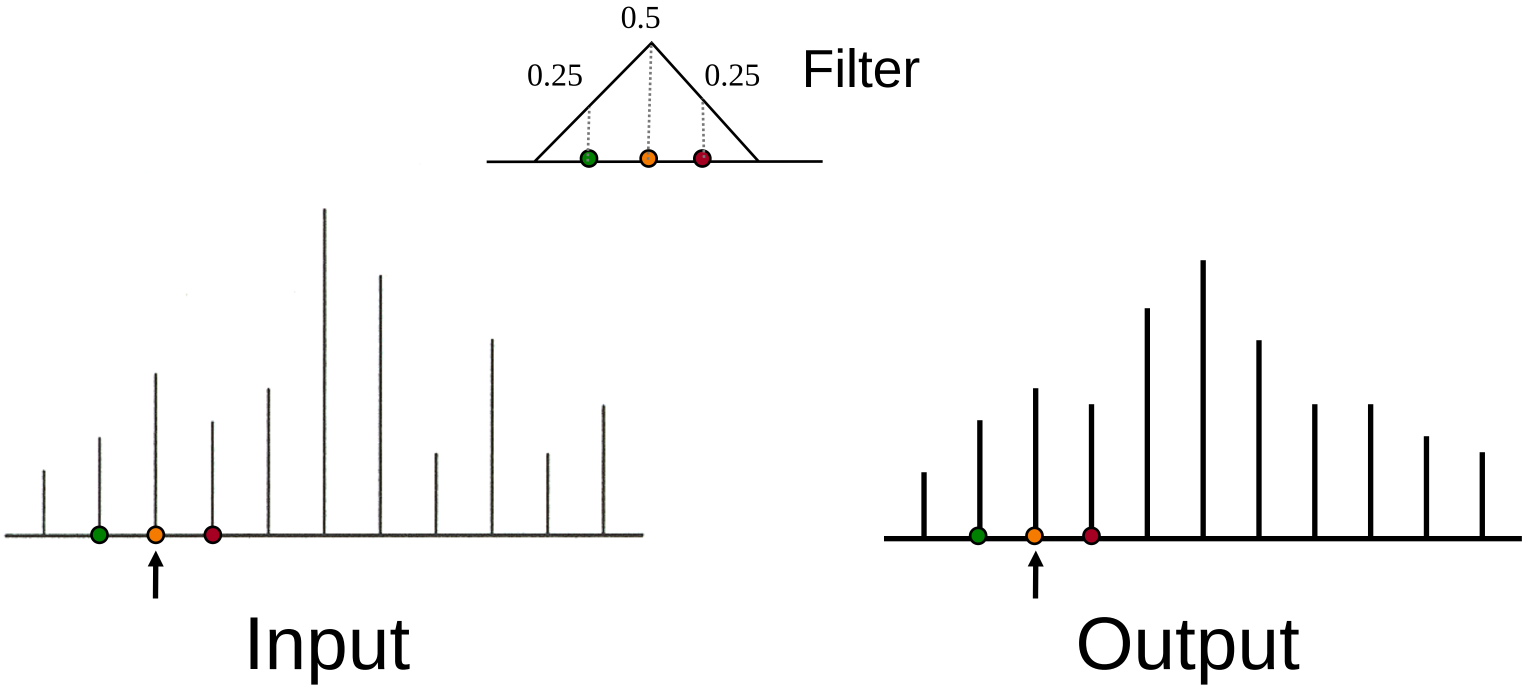
Basic Operation: Convolution

Output value is weighted sum of values in neighborhood of input image

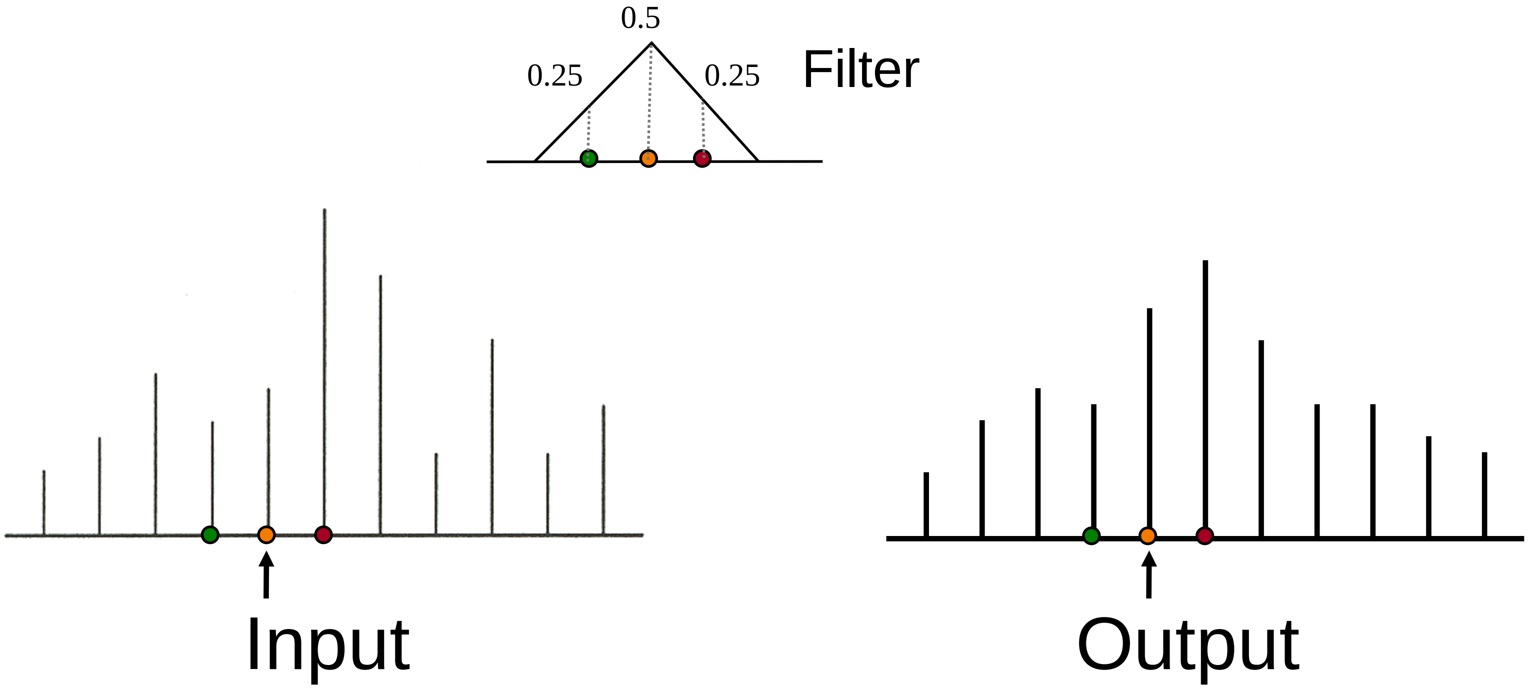
- Pattern of weights is the “filter” or “kernel”



Convolution with a Triangle Filter

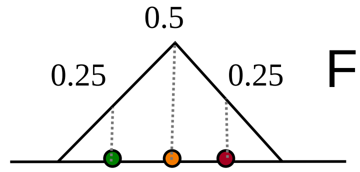


Convolution with a Triangle Filter

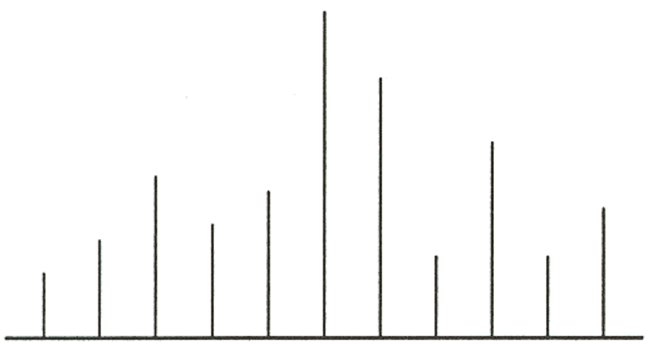


Convolution with a Triangle Filter

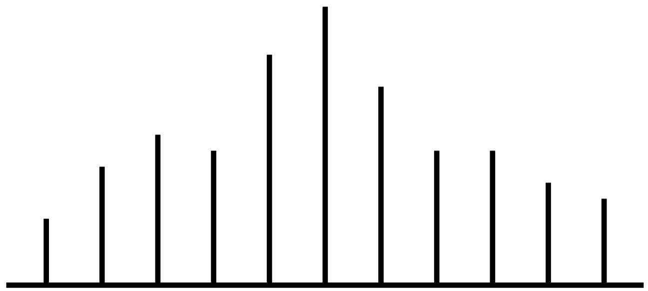
What if the filter runs off the end?



Filter



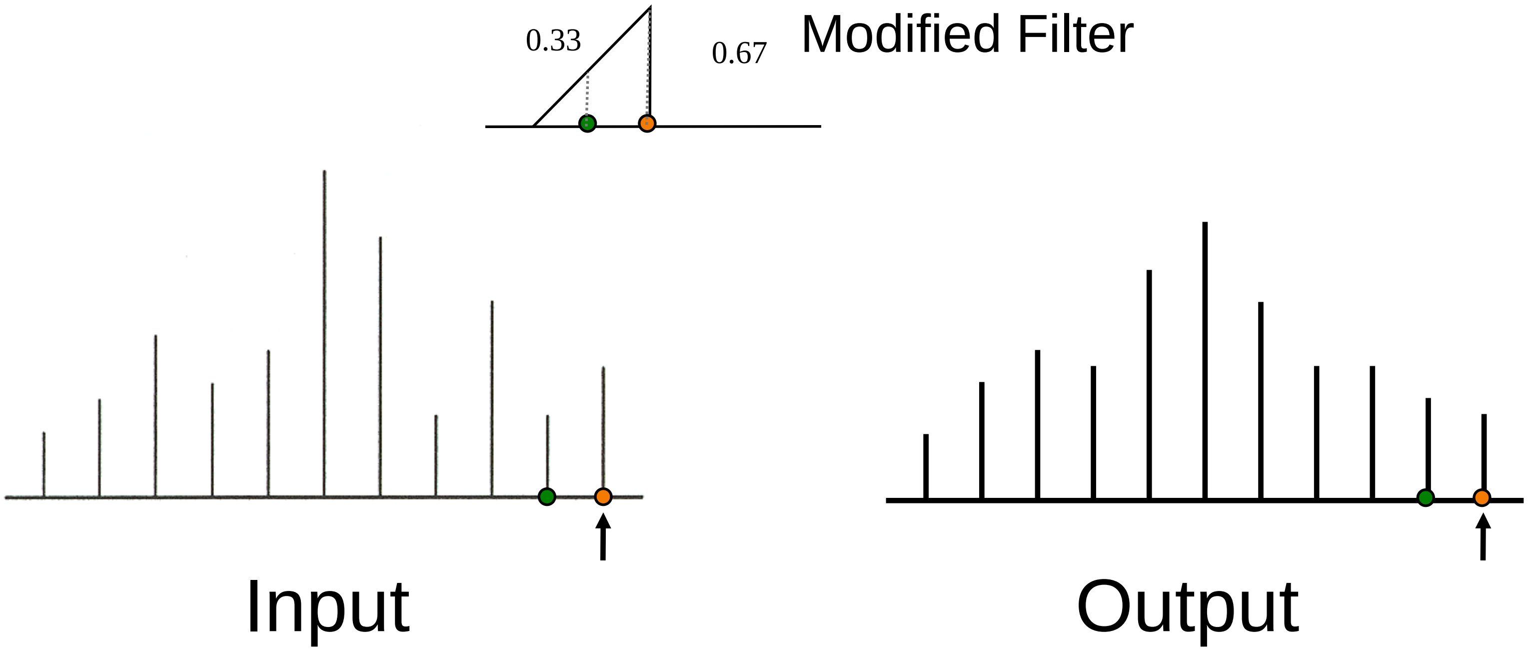
Input



Output

Convolution with a Triangle Filter

Common option: normalize the filter



Convolution with a Gaussian Filter

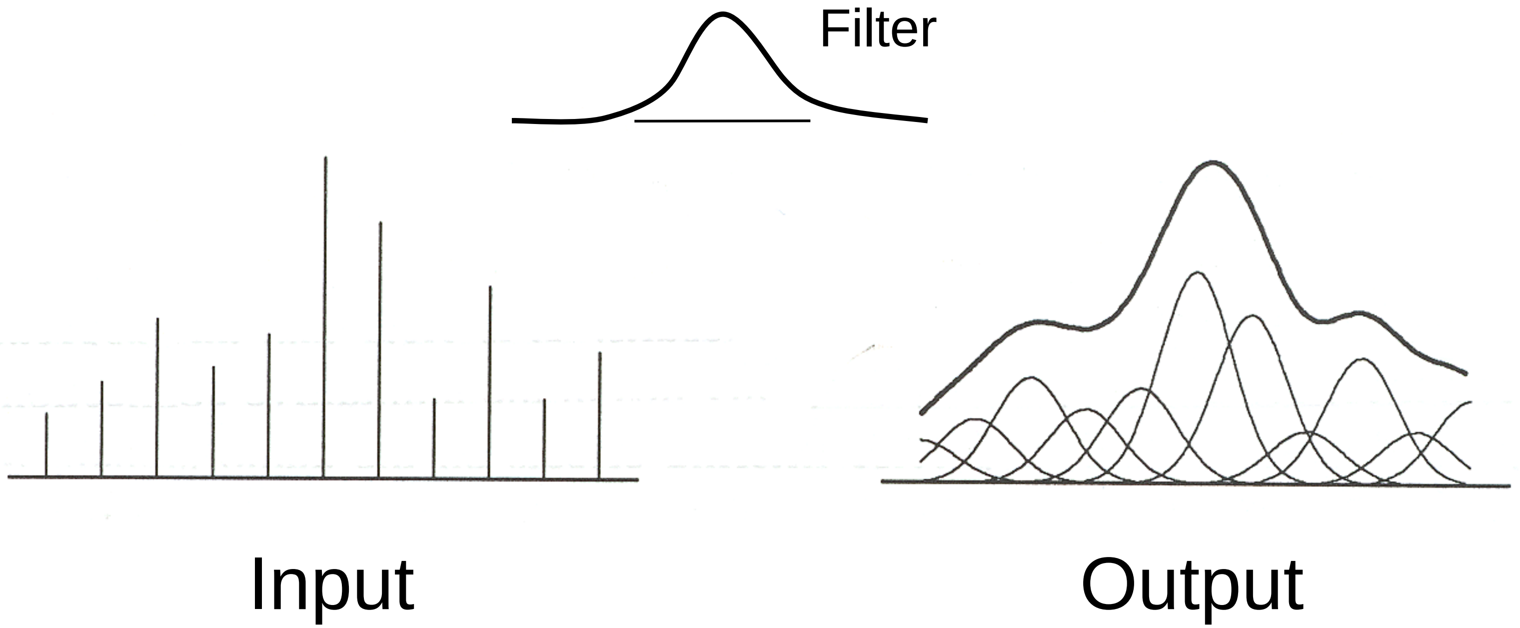
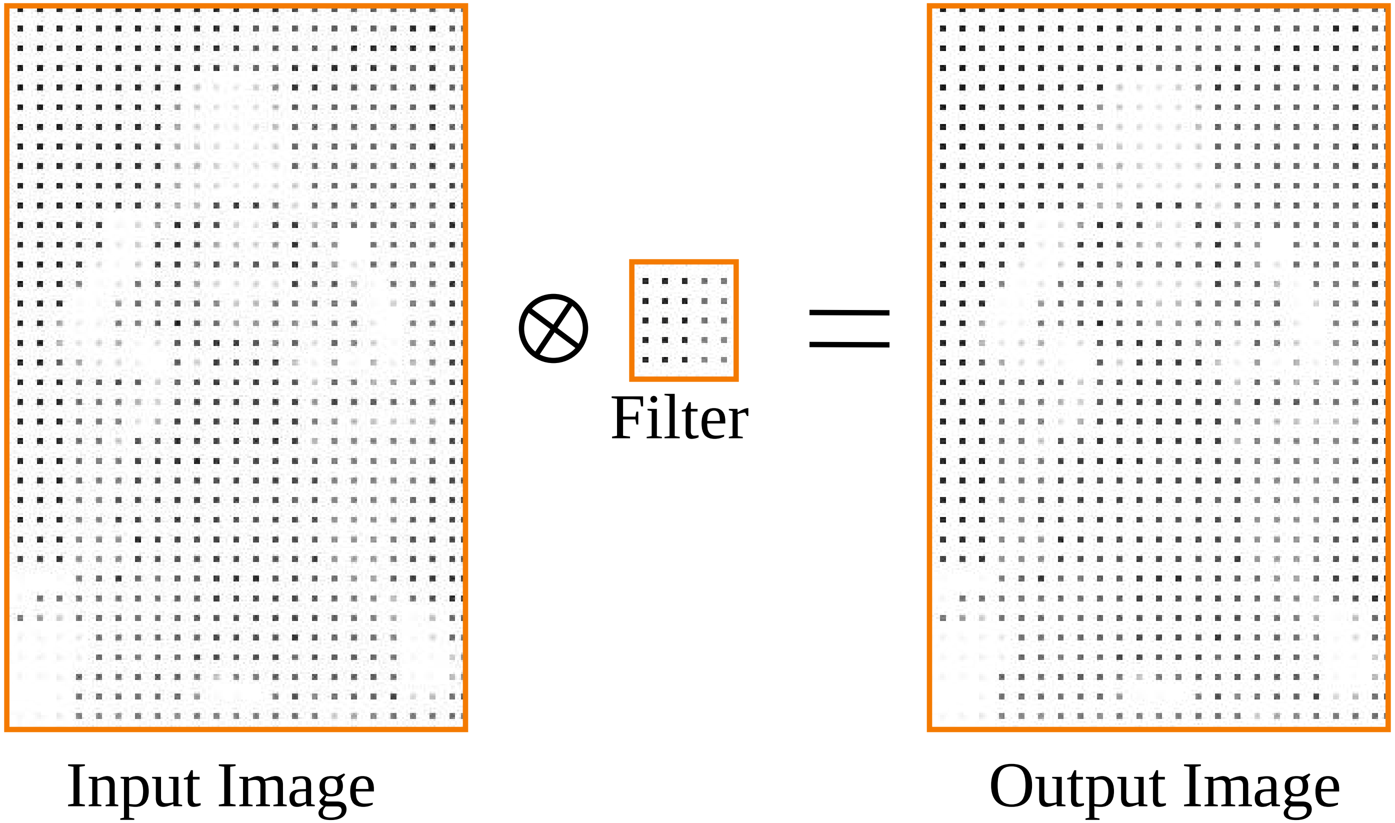


Figure 2.4 Wolberg

Linear Filtering

2D Convolution

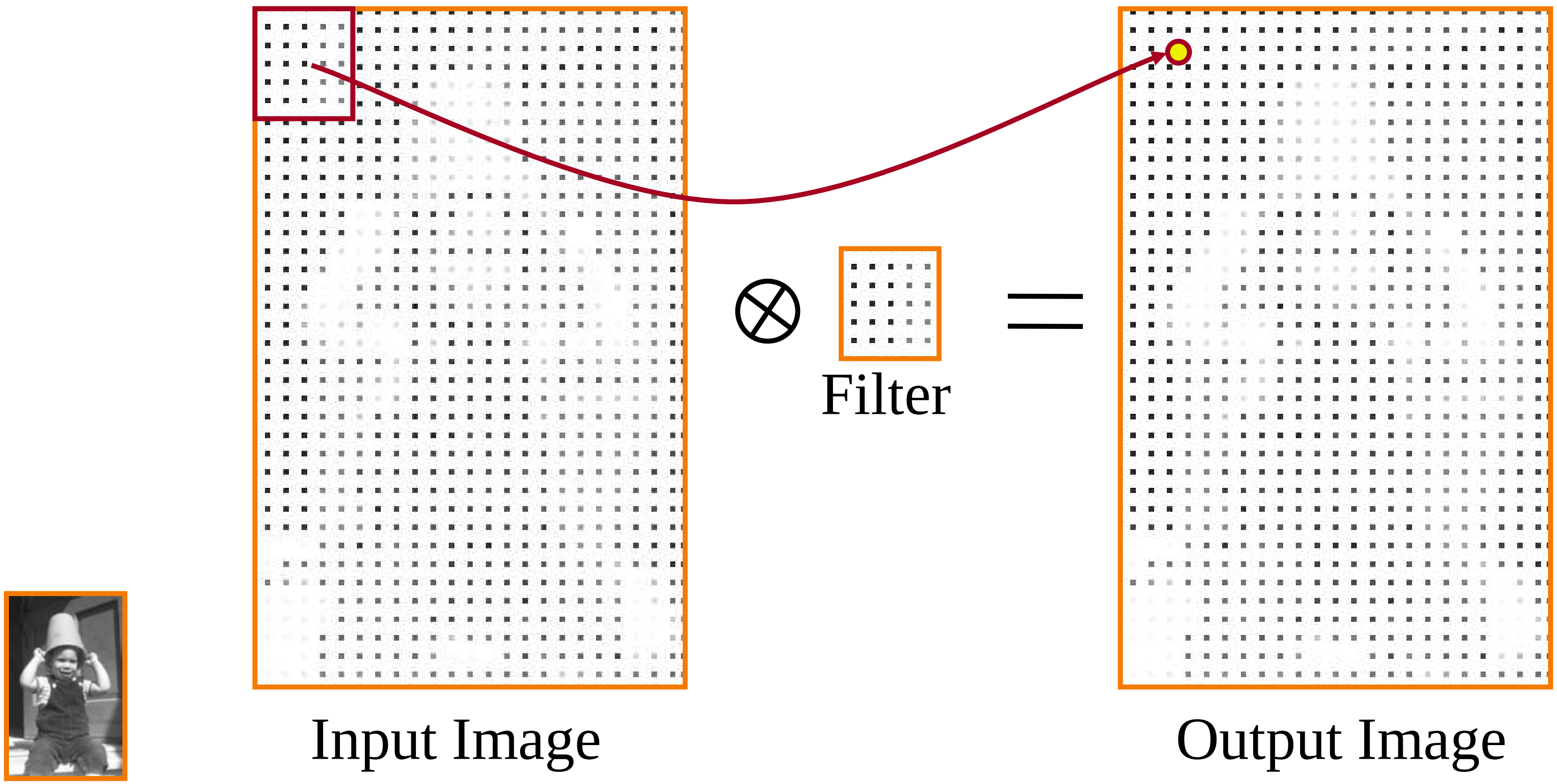
- Each output pixel is a linear combination of input pixels in neighborhood with weights prescribed by a filter



Linear Filtering

2D Convolution

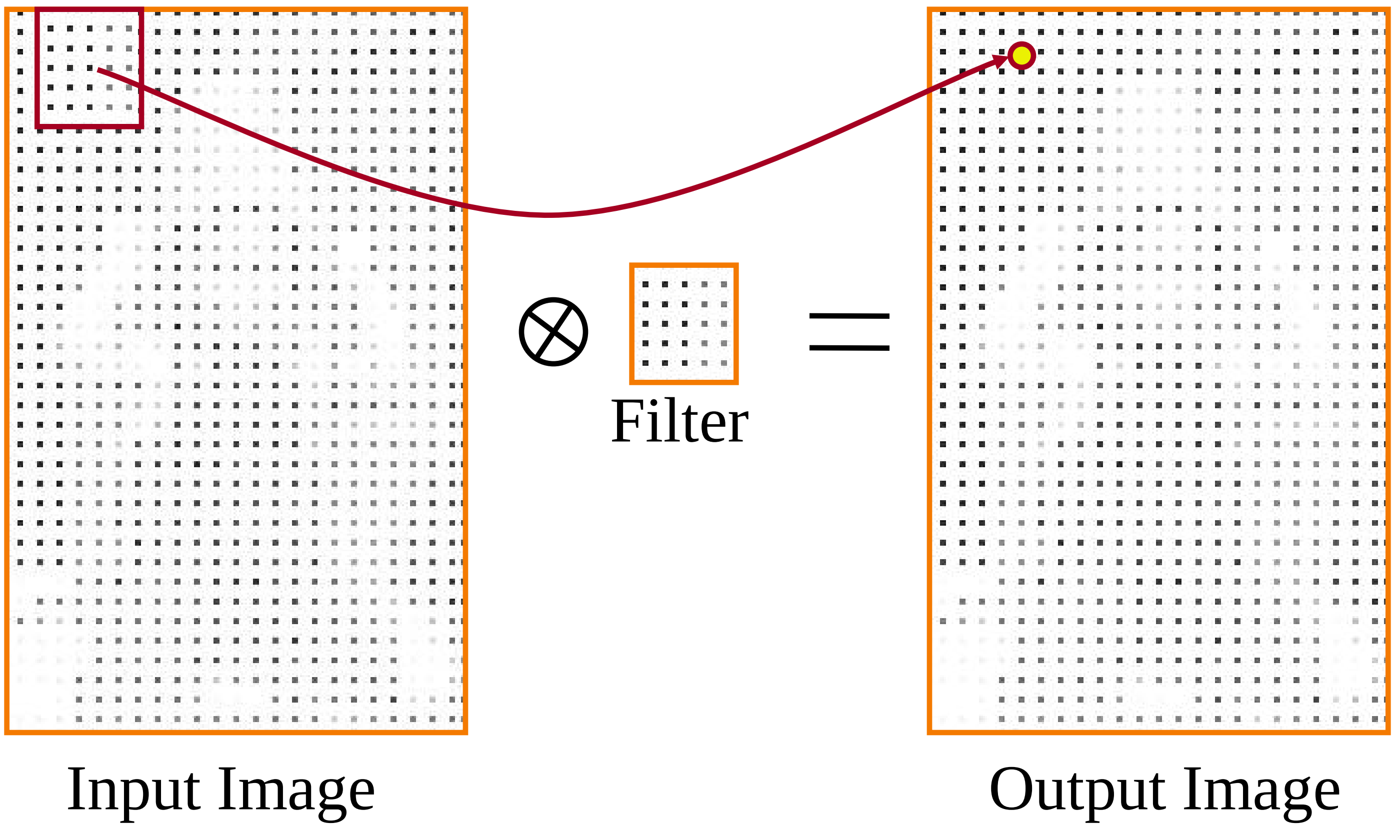
- Each output pixel is a linear combination of input pixels in neighborhood with weights prescribed by a filter



Linear Filtering

2D Convolution

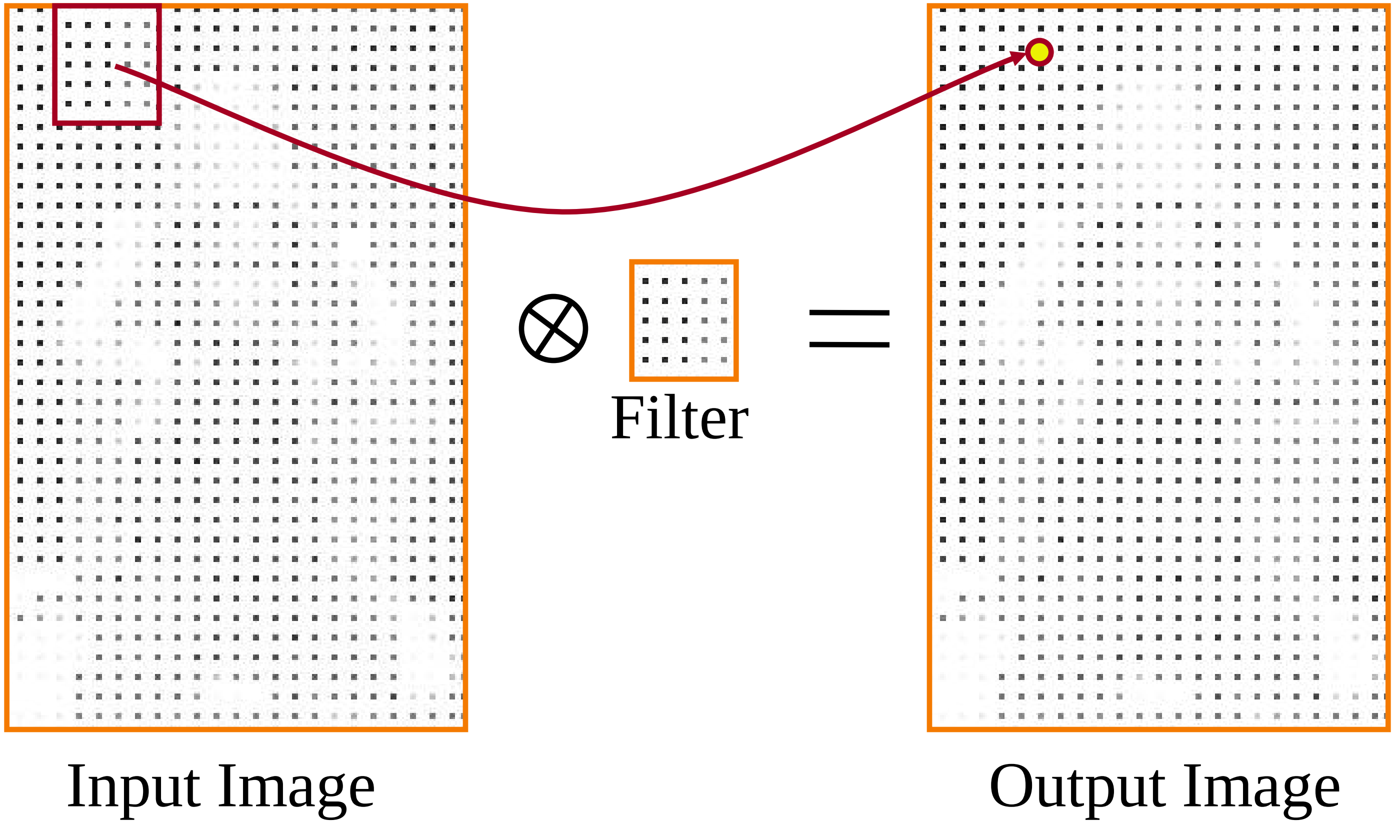
- Each output pixel is a linear combination of input pixels in neighborhood with weights prescribed by a filter



Linear Filtering

2D Convolution

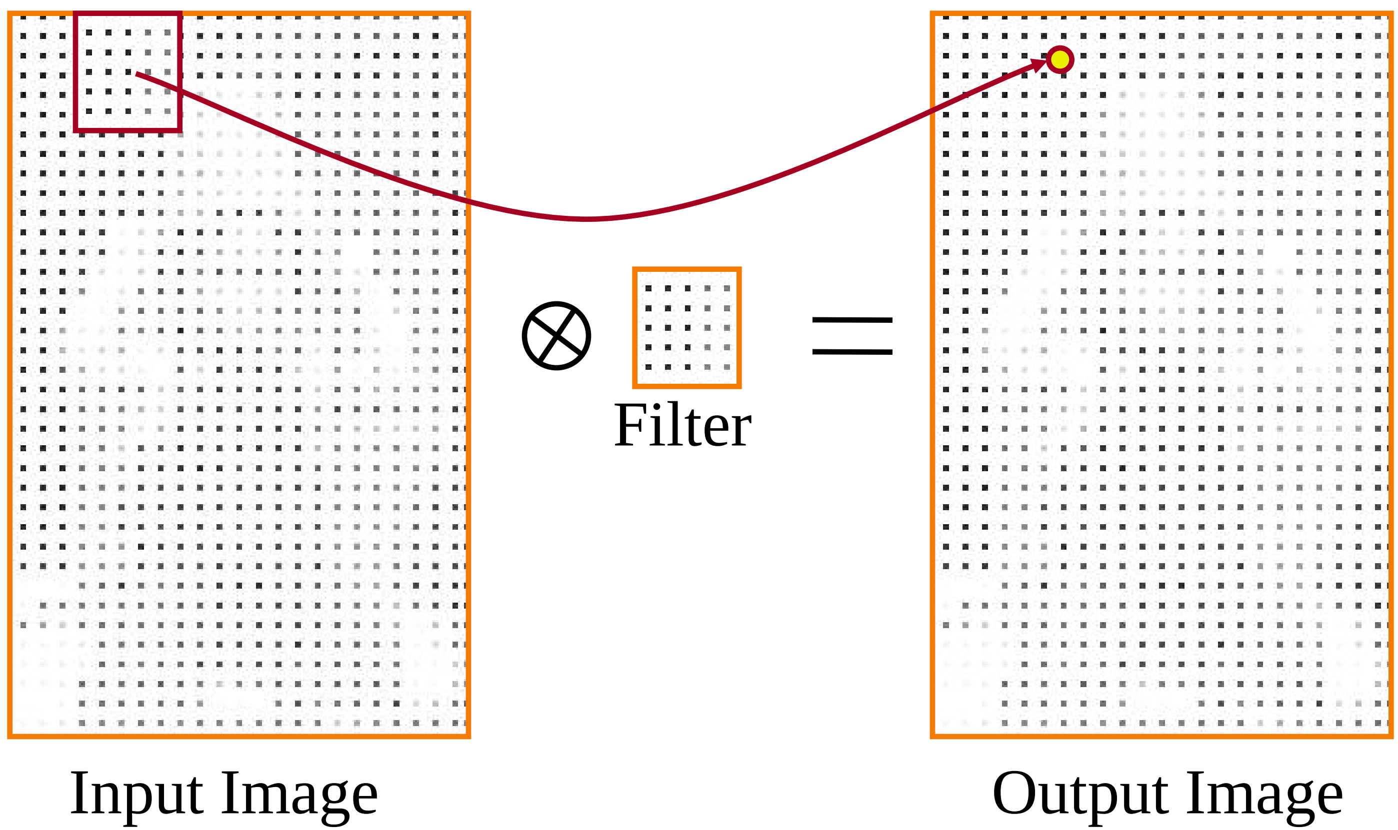
- Each output pixel is a linear combination of input pixels in neighborhood with weights prescribed by a filter



Linear Filtering

2D Convolution

- Each output pixel is a linear combination of input pixels in neighborhood with weights prescribed by a filter



Blur

Convolve with a filter whose entries sum to one

- Each pixel becomes a weighted average of its neighbors

Original

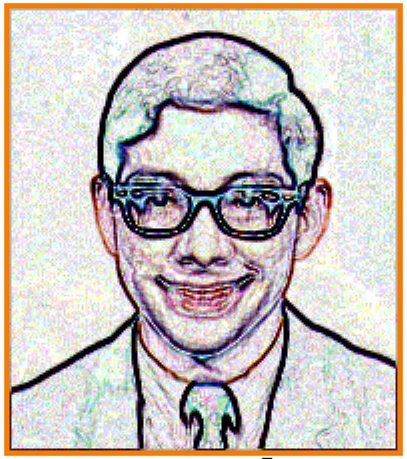
Blur

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

Edge Detection

Convolve with a filter that finds differences between neighbor pixels

Original



Detect edges

$$\text{Filter} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Sharpen

Sum detected edges with original image

Original

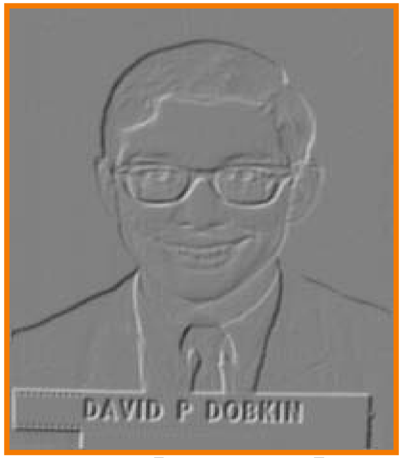
Sharpened

$$\text{Filter} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Emboss

Convolve with a filter that highlights gradients in particular directions

Original



Embossed

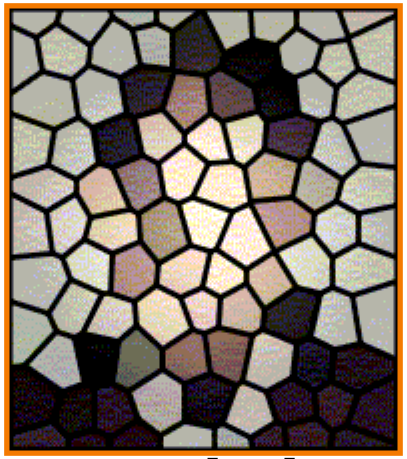
$$\text{Filter} = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Non-Linear Filtering

Each output pixel is a non-linear function of input pixels in neighborhood (filter depends on input)

Original

Oil



Stained Glass

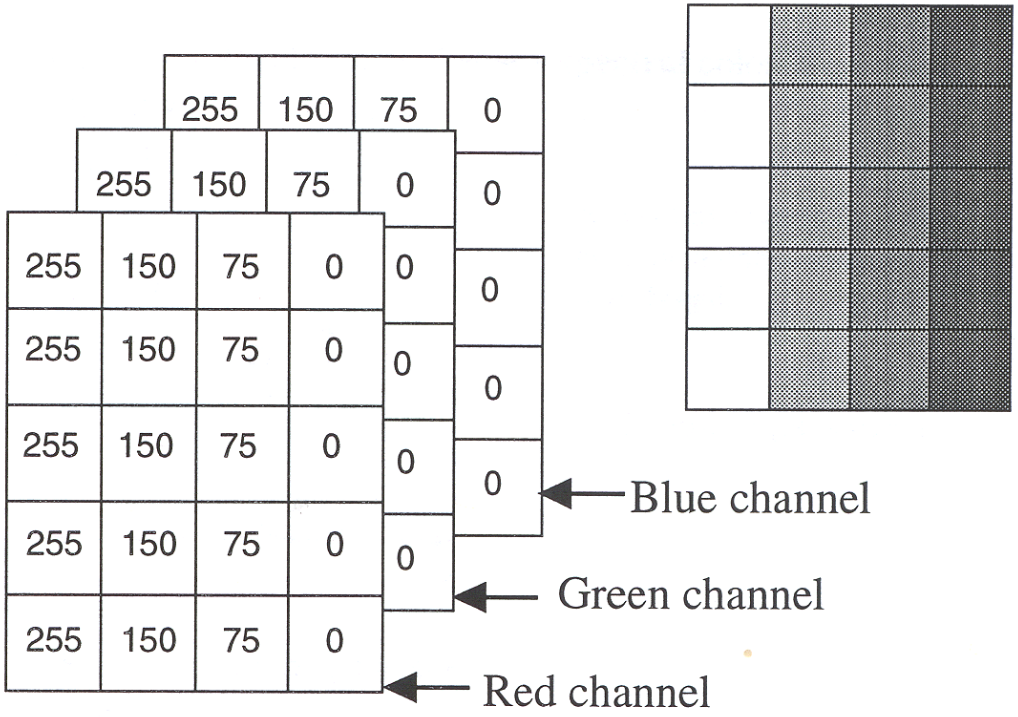
Digital Image Processing

- Changing intensity/color
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Add random noise
- Filtering over neighborhoods
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering

Quantization

Reduce intensity resolution

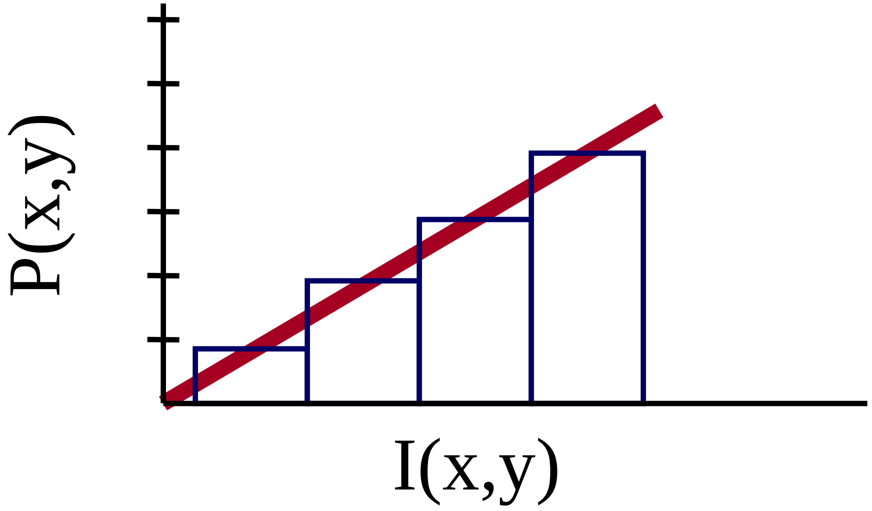
- Frame buffers have limited number of bits per pixel
- Physical devices have limited dynamic range



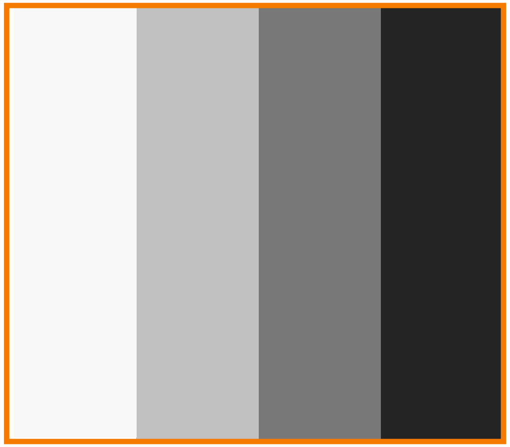
Uniform Quantization

$P(x, y) = \text{round}(\ I(x, y)\ )$

where round() chooses nearest value that can be represented.



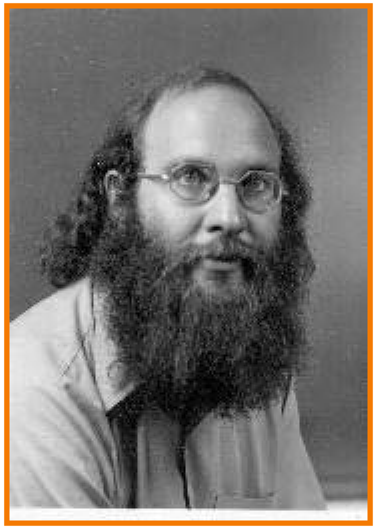
I(x,y)



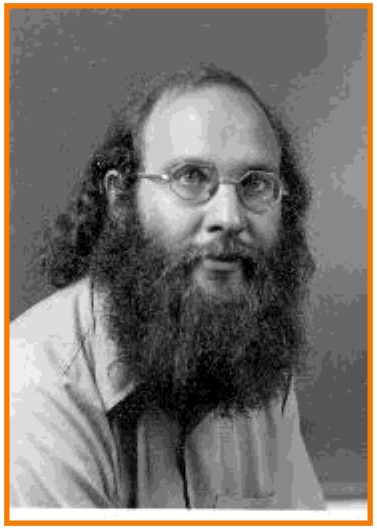
P(x,y)
(2 bits per pixel)

Uniform Quantization

Images with decreasing bits per pixel:



8 bits



4 bits

2 bits

1 bit

Notice contouring.

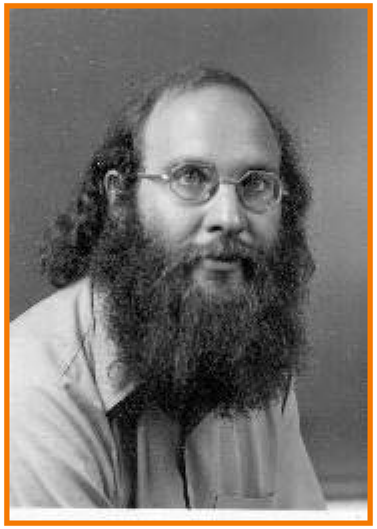
Reducing Effects of Quantization

- Intensity resolution / spatial resolution tradeoff
- Dithering
  - Random dither
  - Ordered dither
  - Error diffusion dither
- Halftoning
  - Classical halftoning

Dithering

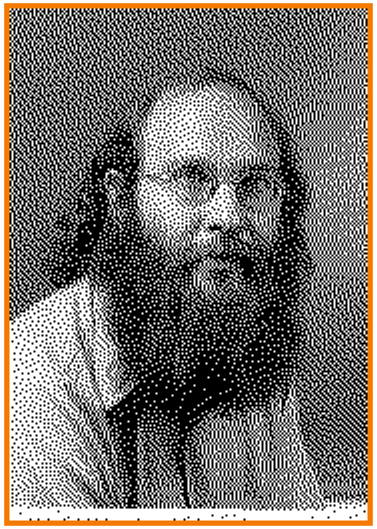
Distribute errors among pixels

- Exploit spatial integration in our eye
- Display greater range of perceptible intensities



Original
(8 bits)

Uniform
Quantization
(1 bit)

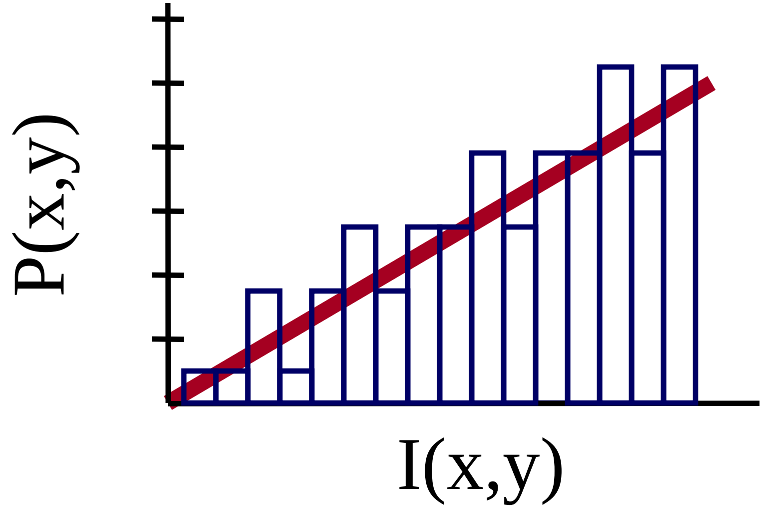


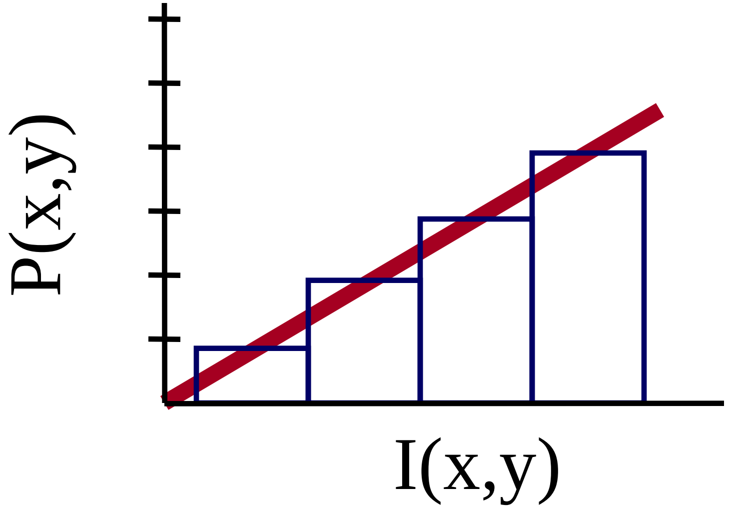
Floyd-Steinberg
Dither
(1 bit)

Random Dither

Randomize quantization errors

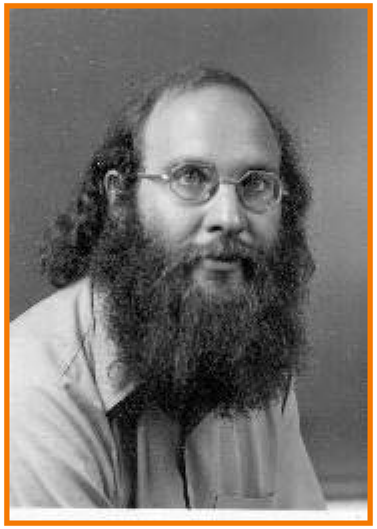
- Errors appear as noise





$$P(x, y) = \text{round}\big(I(x, y) + \text{noise}(x,y)\big)$$

Random Dither



Original
(8 bits)

Uniform
Quantization
(1 bit)

Random
Dither
(1 bit)

Ordered Dither

Pseudo-random quantization errors

- Matrix stores pattern of threshholds

$$D_2 = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$$

i = x mod n

j = y mod n

e = I(x,y) - trunc(I(x,y))

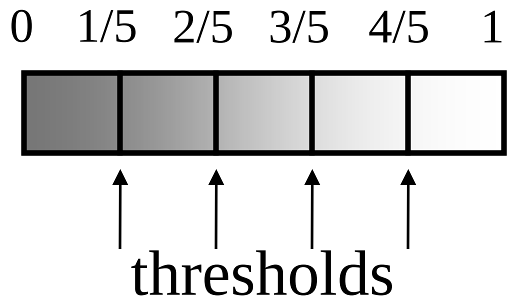
threshold = $(D(i,j)+1)/(n^2+1)$

if (e > threshold)

    P(x,y) = ceil(I(x, y))

else

    P(x,y) = floor(I(x,y))



Ordered Dither

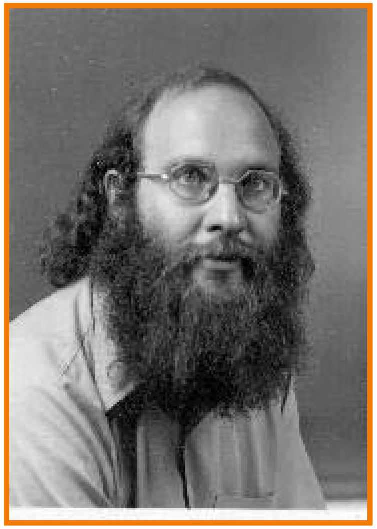
Bayer's ordered dither matrices

$$D_n = \begin{bmatrix} 4D_{n/2} + D_2(1,1)U_{n/2} & 4D_{n/2} + D_2(1,2)U_{n/2} \\ 4D_{n/2} + D_2(2,1)U_{n/2} & 4D_{n/2} + D_2(2,2)U_{n/2} \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$$

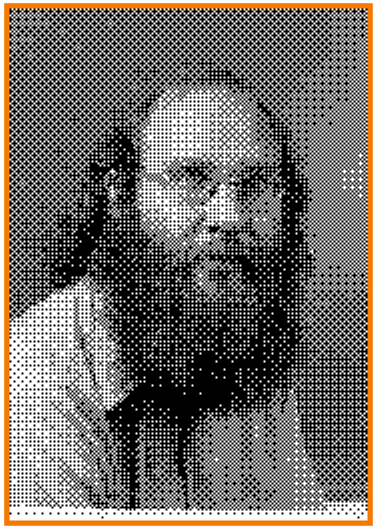
$$D_4 = \begin{bmatrix} 15 & 7 & 13 & 5 \\ 3 & 11 & 1 & 9 \\ 12 & 4 & 14 & 6 \\ 0 & 8 & 2 & 10 \end{bmatrix}$$

Ordered Dither



Original
(8 bits)

Random
Dither
(1 bit)

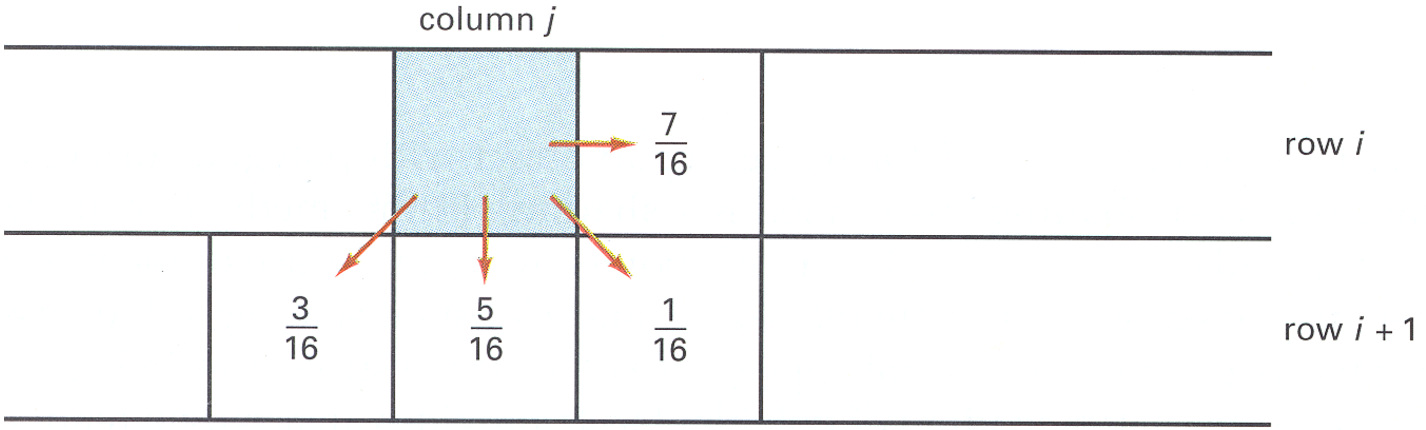


Ordered
Dither
(1 bit)

Error Diffusion Dither

Spread quantization error over neighbor pixels

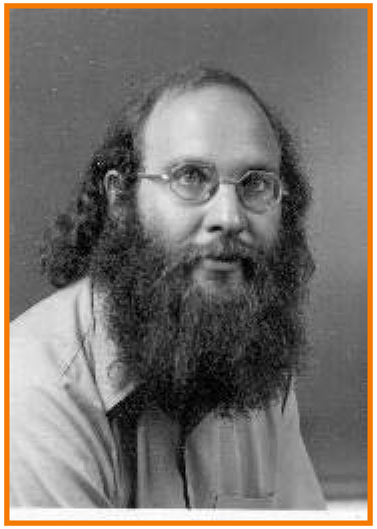
- Error dispersed to pixels right and below
- Floyd-Steinberg weights:



$$3/16 + 5/16 + 1/16 + 7/16 = 1.0$$

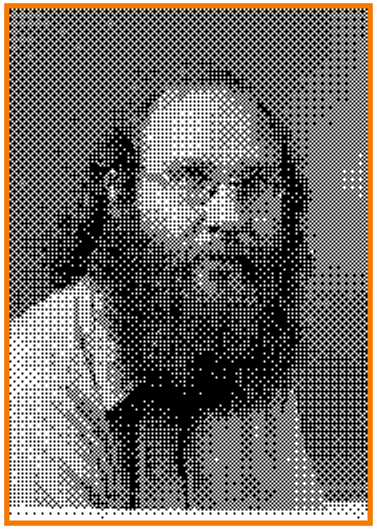
Figure 14.42 from H&B

Error Diffusion Dither

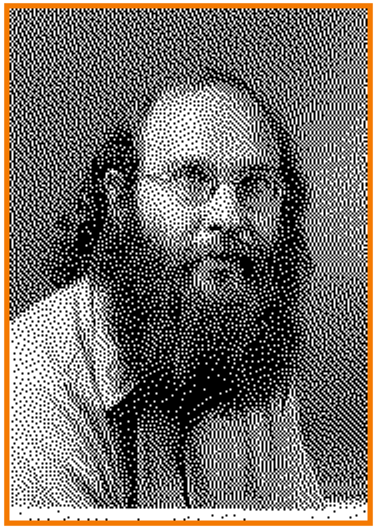


Original
(8 bits)

Random
Dither
(1 bit)



Ordered
Dither
(1 bit)



Floyd-Steinberg
Dither
(1 bit)

Reducing Effects of Quantization

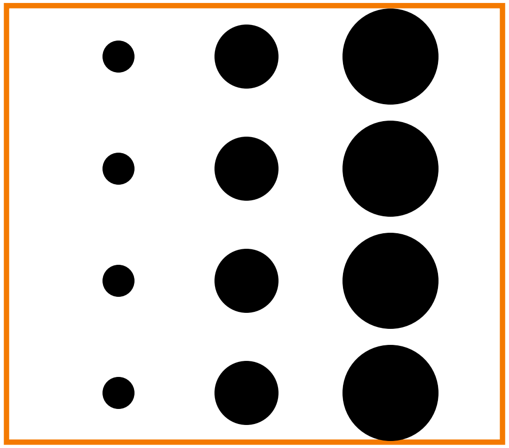
- Dithering
  - Random dither
  - Ordered dither
  - Error diffusion dither
- Halftoning
  - Classical halftoning

Classical Halftoning

Use dots of varying size to represent intensities

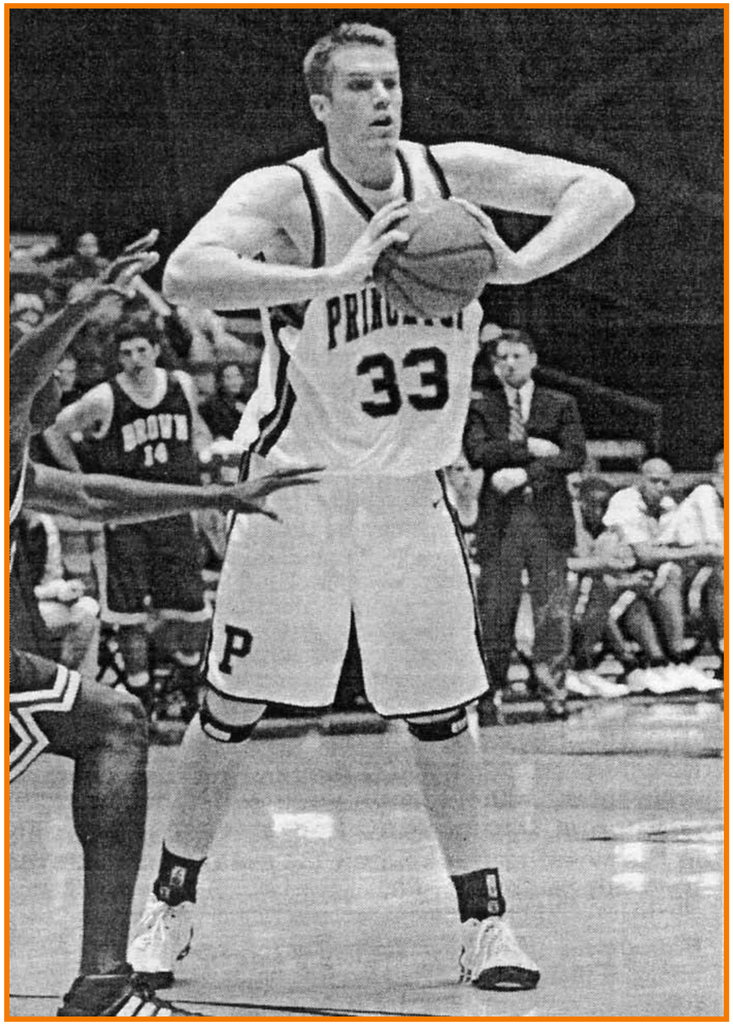
- Area of dots proportional to intensity in image

I(x,y)

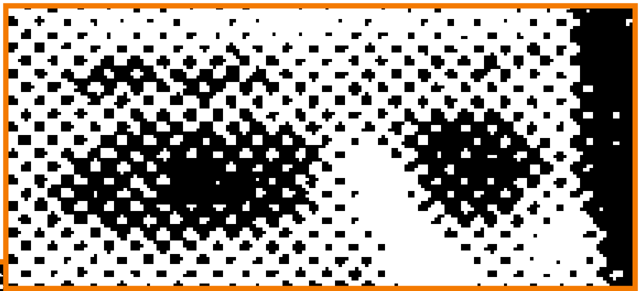


P(x,y)

Classical Halftoning

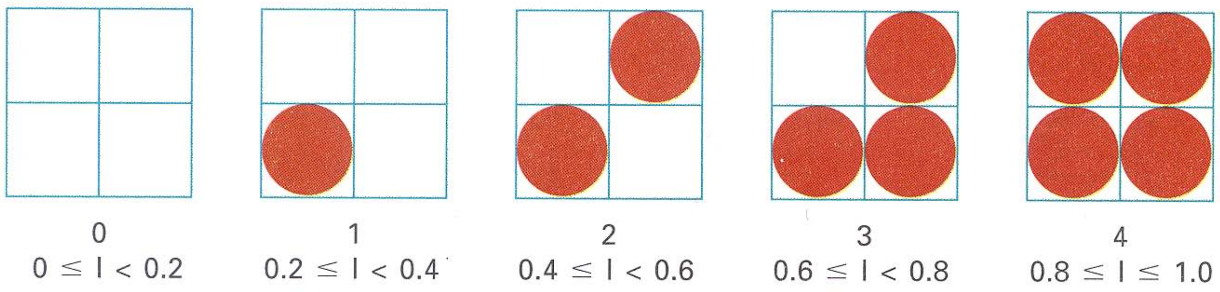


From *Town Topics*, Princeton



Digital Halftone Patterns

Use cluster of pixels to represent intensity



Q: In this case, would we use four “halftoned” pixels in place of one original pixel?

Figure 14.37 from H&B

Summary

- Image filtering
  - o Compute new values for image pixels based on function of old values
- Halftoning and dithering
  - o Reduce visual artifacts due to quantization
  - o Distribute errors among pixels
    - » Exploit spatial integration in our eye

Next Time…

- Changing intensity/color
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Add random noise
- Filtering over neighborhoods
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering