Intro to Deep Generative Models - DIRP Spring 2025

– Meeting 1: Feb 8, 2025 –
Introduction and Computer Vision Basics

Introductions

Topic	Resources
Computer Vision Basics - Image formation and representation - Basic image processing (noise, filtering, denoising) - Convolutions - Edge detection and feature extraction - Introduction to Python libraries (Numpy, OpenCV)	- Image Representation: Image Representation - Linear Filtering: Linear Filtering - Edge Detection: Edge Detection - ⭐ Canny Edge Detector Paper: A Computational Approach To Edge Detection - Feature detection: Feature Detectors : SIFT and Variants - 📖 Szeliski Chapter 3.1-3.3
Deep Learning for Vision - Basics of a neural network - Backpropagation - Activation functions - Loss functions and optimization	First three videos of this playlist (3B1B) Neural networks - YouTube
Convolutional Neural Networks (CNNs) - CNN Architecture - Convolution and pooling - Popular CNN architectures - Batch normalization - Data augmentation	- What is a convolution? But what is a convolution? - 📖 Deep Learning Book Chapter 9: https://www.deeplearningbook.org/contents/convnets.html
Recurrent Neural Networks (RNNs) - RNN architecture - LSTM and GRU	- 📖 Deep Learning Book Chapter 10: https://www.deeplearningbook.org/contents/rnn.html

Convolutional Neural Networks (CNNs) - CNN Architecture - Convolution and pooling - Popular CNN architectures - Batch normalization - Data augmentation	- What is a convolution? But what is a convolution? - 📖 Deep Learning Book Chapter 9: https://www.deeplearningbook.org/contents/convnets.html
Recurrent Neural Networks (RNNs) - RNN architecture - LSTM and GRU	- 📖 Deep Learning Book Chapter 10: https://www.deeplearningbook.org/contents/rnn.html

Autoencoders - Autoencoder architectures - Latent space/dimensionality reduction - Feature learning - Variational Autoencoders (VAEs) - VAEGAN, VQ-VAE	- Variational Autoencoders → Highly recommend this video - An Introduction to VAE-GANs - ⭐ Autoencoding beyond pixels paper: https://arxiv.org/abs/1512.09300
Generative Adversarial Networks (GANs) - GAN architecture - DCGANs, Conditional GANs	- Friendly Introduction to GANs: A Friendly Introduction to Generative Ad... - GAN implementation: Build a GAN From Scratch
Advanced Generative Models - Score based and energy based models - Diffusion Models	- Energy-Based Models Stanford CS236: Deep Generative Models I 2023 ... Stanford CS236: Deep Generative Models I 2023 ... - Score-Based Models Stanford CS236: Deep Generative Models I 2023 ...

Intro to Deep Generative Models

Search Intro to Deep Generative Models

Introduction to Deep Generative Models

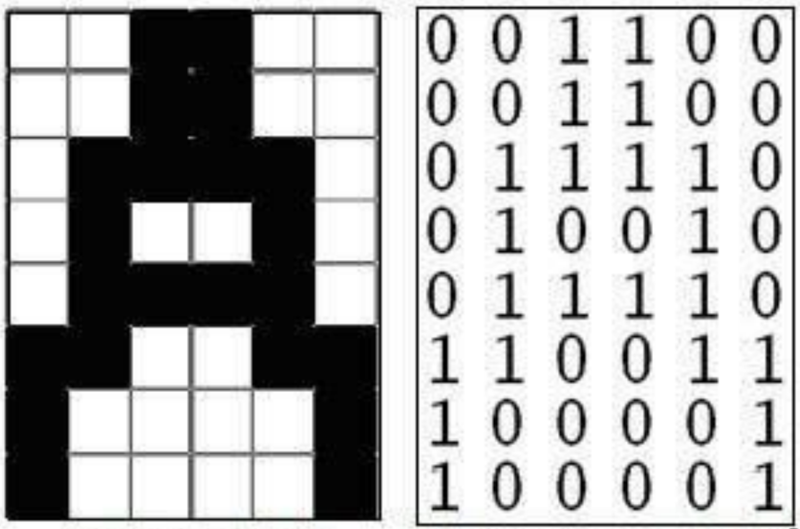
This is a reading group under the Directed Reading Program (DiRP) at UT Austin. This group will cover concepts starting from basic computer vision concepts such as filtering, convolutions, edge/object detection, etc. and will slowly build up to more complex models including CNNs, RNNs, Variational Autoencoders (VAEs, VAE-GANs, VQ-VAEs), GANs, and Diffusion Models. This website is intended to be a central hub for notes, papers, and many other resources that will be helpful to anyone researching this topic.

Website for course notes + other resources!

What is an image? How do we represent it?

- Images are 2D representations of a 3D scene
- Three major ways of representing images:
  - Matrix form
  - Function representation
  - Digital images

Matrix Representation



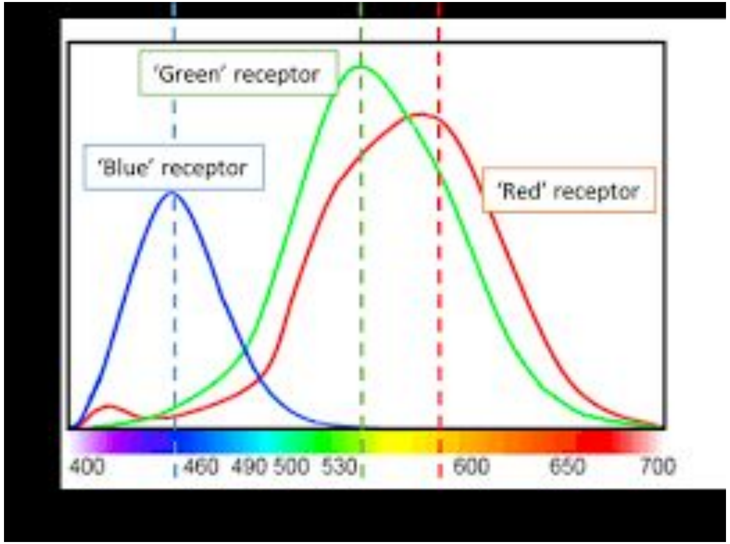
Digitized "A" Matrix Form of "A"

- Each pixel in the image corresponds to some value (i, j) in the matrix
- Values in the matrix represent intensity ⇒ ranges from 0 to 255 (normalized: 0 to 1)
  - 0 means black (no intensity)
  - 1 means white (maximum intensity)

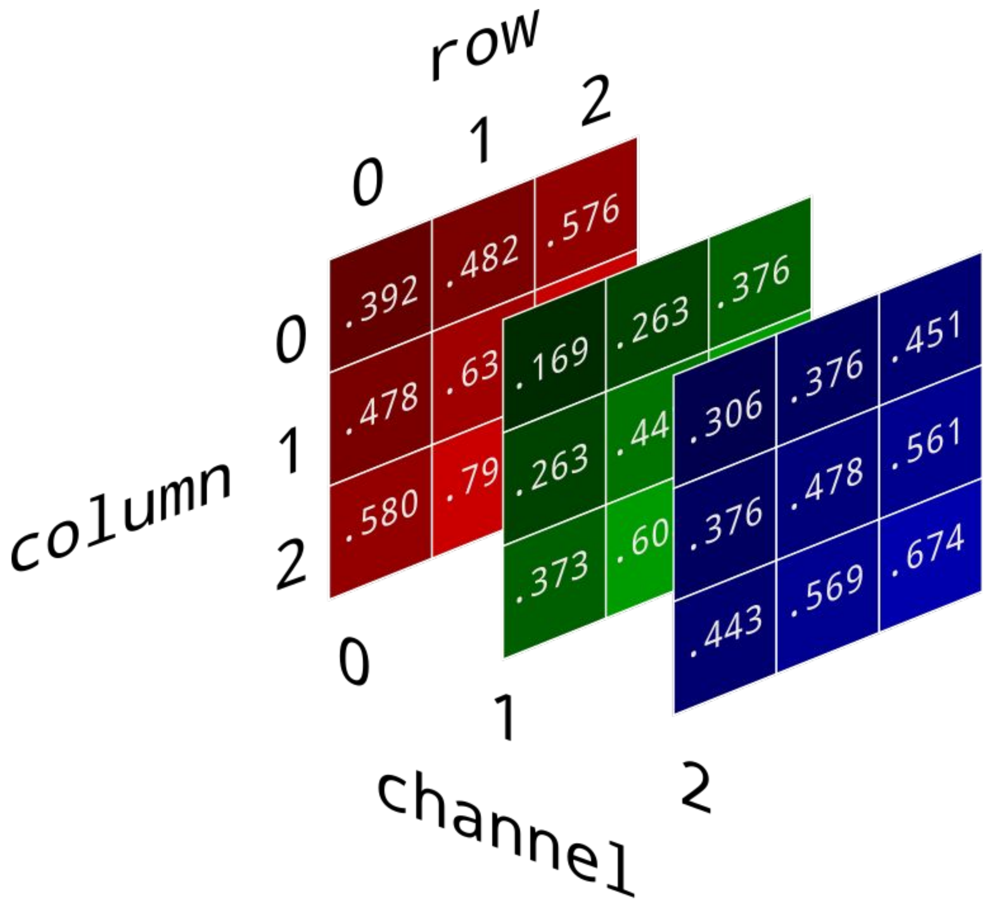
Color vs. Grayscale Images

- Representing grayscale images is pretty easy!
  - Requires one **channel** to represent the image
  - Number of channels tells us how many numbers are required to specify each pixel's color/intensity
- Color images:
  - Requires 3 channels: Red, Green, and Blue (RGB)

Color Images

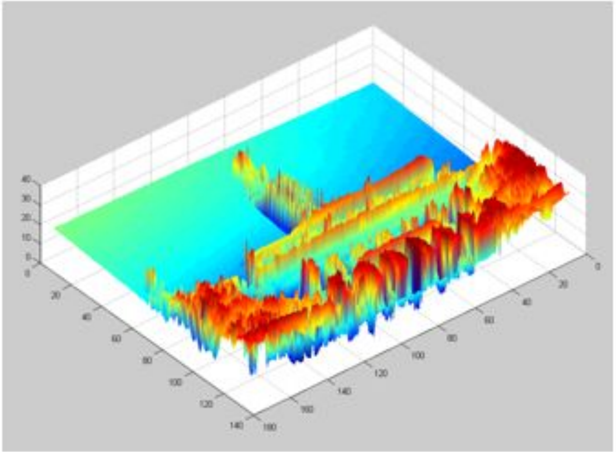


- Cones in the human eye (receptors) perceive light roughly in terms of Red, Green, Blue
- When representing color images in matrix form, we actually require three matrices for R, G, B intensities
  - Stacked together to create a single matrix representing the image where (i, j) in the matrix a tuple containing intensities for red, green, and blue



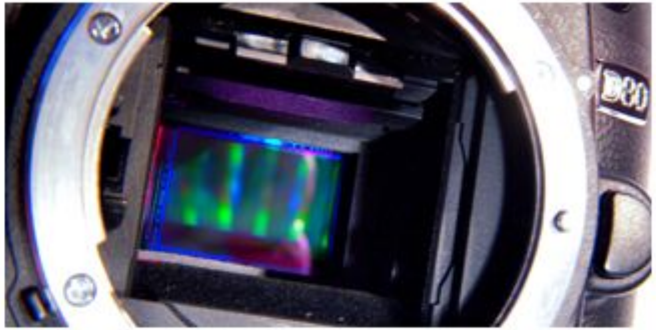
row
0
1
2
column
0
1
2
channel
0
1
2
.392
.482
.576
.478
.63
.580
.79
.169
.263
.376
.263
.44
.373
.60
.306
.376
.451
.376
.478
.561
.443
.569
.674

Function representation



- Images can be represented as continuous functions, where f: (i, j) → I (intensity)
- Function representation is **continuous**, as opposed to the matrix representation, which is **discrete**
  - This makes function representations useful for certain mathematical operations, such as calculating gradients
- For grayscale images, $f(x, y) = i$
- For color images, function f is **vector-valued** $f(x, y) = (f_r(x, y), f_g(x, y), f_b(x, y))$

Digital Images



- Use of sensor arrays
- Digital images are a discrete, 2D, digital representation of the function representation
  - Sampled and quantized

Image Transformations

What is a transformation? What can you with transformations?

Image Transformations

What is a transformation? What can you with transformations?

- Dimming/Brightening
- Rotating
- Mirroring
- much, much more!

⇒ Image Filtering

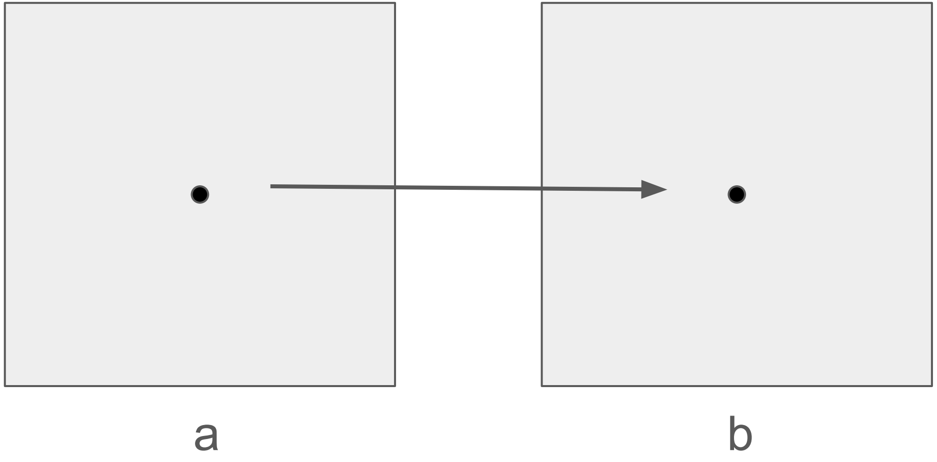
Image Transformation Operations

- Point
- Local
- Global

Point

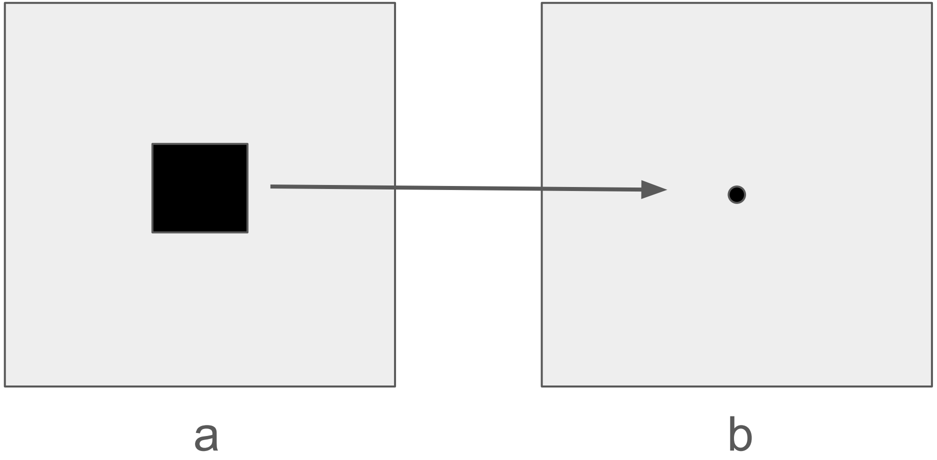
Output value (m_i, n_j) is dependent upon the input value at the same coordinate.

- Pixel to pixel



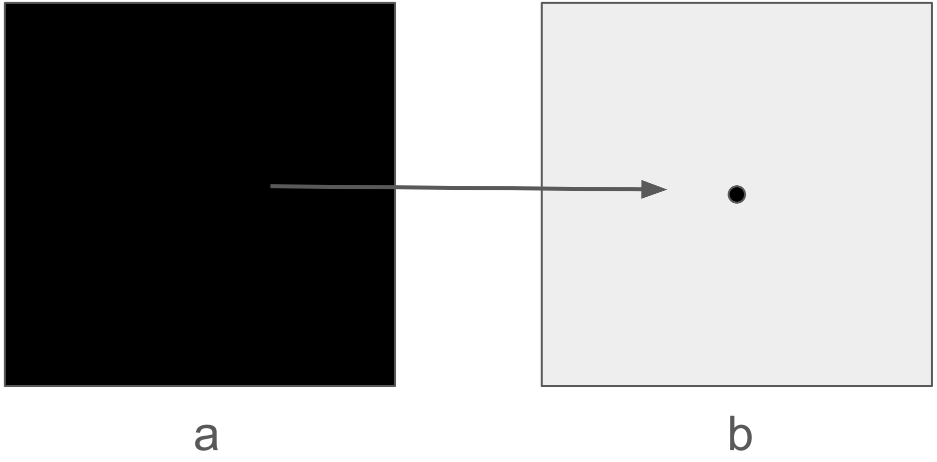
Local

Output value (m_i, n_j) is dependent upon the input values of pixels in a pxp neighborhood of the corresponding pixel in the input image.



Global

Output value (m_i, n_j) is dependent upon all values in the input image.



Point Operations: Applications

- Dimming/brightening
- Mirroring
- Increasing/decreasing contrast
- etc.

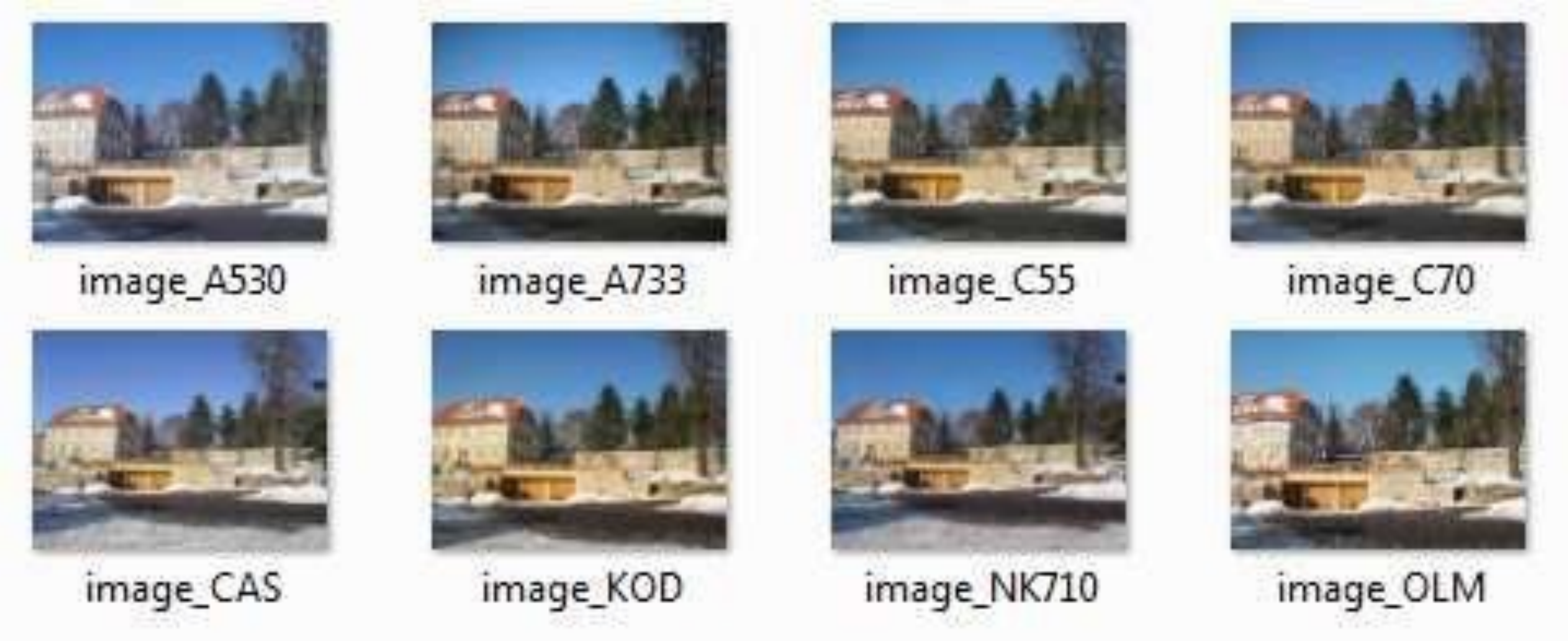
Local Operations: Applications

- Reducing noise
- Filtering
- Convolutions
- etc.

Global Operations: Applications

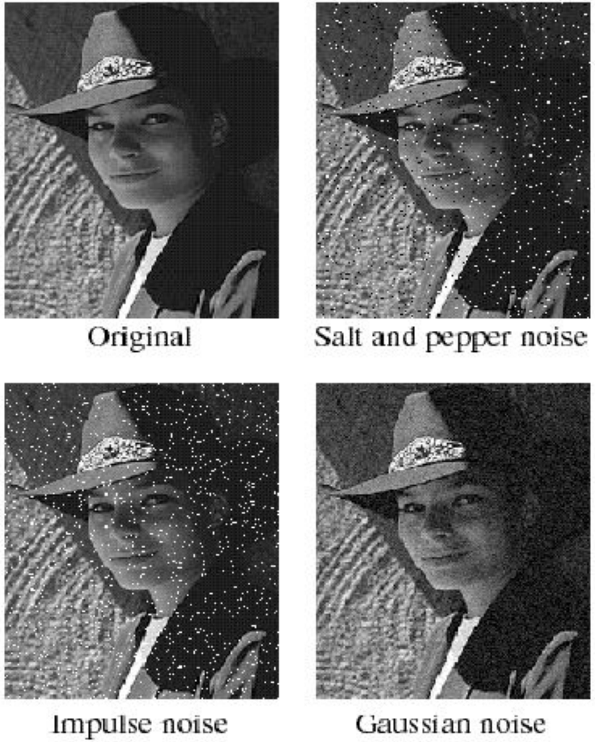
- Fourier transforms
  - The Fourier transform breaks down an image into a sum of complex exponentials with different frequencies, phases, and magnitudes.
  - The output of the Fourier transform is a representation of the image in the frequency domain

Reducing Noise: What is noise?



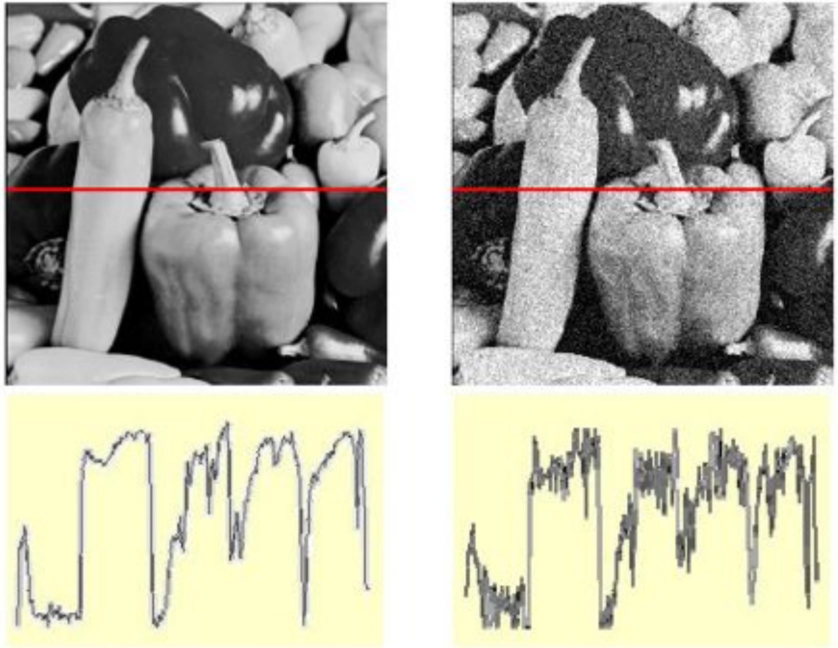
Types of noise

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution



Source: S. Seitz

Gaussian noise



$$f(x,y) = \overbrace{\hat{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

```
>> noise = randn(size(im)).*sigma;
>> output = im + noise;
```

Moving Average

- Take an nxn window and simply take the average of all the pixels in that window
- This becomes the value of the center pixel

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0								

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10							

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20						

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30					

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30					

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30				

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Image Filtering

- A linear filter is where an output pixel’s value is a weighted sum of pixel values within a small neighborhood N
- The weights that you multiply each pixel by are called the kernel/filter/mask

Correlation filtering

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\substack{\text{Attribute uniform} \\ \text{weight to each pixel}}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]}_{\substack{\text{Loop over all pixels in neighborhood} \\ \text{around image pixel } F[i,j]}}$$

Cross-Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

denoted $G = H \otimes F$

What can you do with correlation/cross-correlation filtering?

- Blur images (using a Gaussian filter)
- Shift images
- Dim/Brighten images
- Increase/Decrease contrast
- Edge detection
- etc.

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

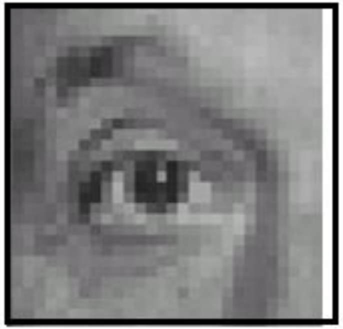
Original

0	0	0
0	0	1
0	0	0

?

Original

0	0	0
0	0	1
0	0	0



Shifted left
by 1 pixel
with
correlation

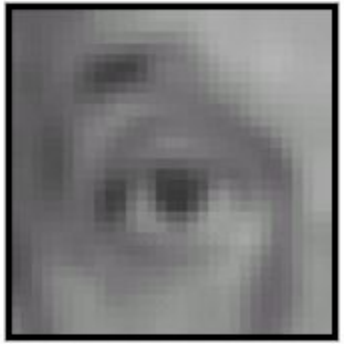
Original

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

?

Original

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



Blur (with a box filter)

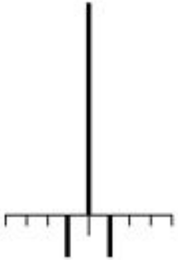
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

Original

$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$



Sharpening filter:
accentuates differences
with local average