CS 194-26: Computer Vision and Computational Photography Lecture Notes

Week 1: Lecture 1 Introduction (8/25)

Brief History of Visual Data
First paintings: animals, line drawings
Middle Ages: Straight forward, simple pictures
Renaissance: Showed depth 3 dimensions, revolution, depth
Toward perfection
Realism: capturing subjective parts of the world to represent what we're trying to say

- Represent what we want to represent

Modern Computer Graphics: graphics too perfect dirty stuff is hard to animate
Realism, manipulation, ease of capture

- Computational photography combines them in
- Computer Vision
- - Trying to understand the visual world
- Measurement vs Understanding
    - Need to understand what a picture depicts
    - How to get computers to interpret the visual world in a way that humans do

Week 2: Lecture 2 Capturing Light… in a human and machine (8/30)

Light gets projected onto film
Sensor Array

- CMOS Sensor

Sampling and Quantization

- Quantize grayscale values into 256

Rolling Shutter

- Cheap cameras, scanning the world 1 row, pixel at a time

Saccadic eye movement

- Eyes moves around several times a second, saccadic eye movement

Human eye

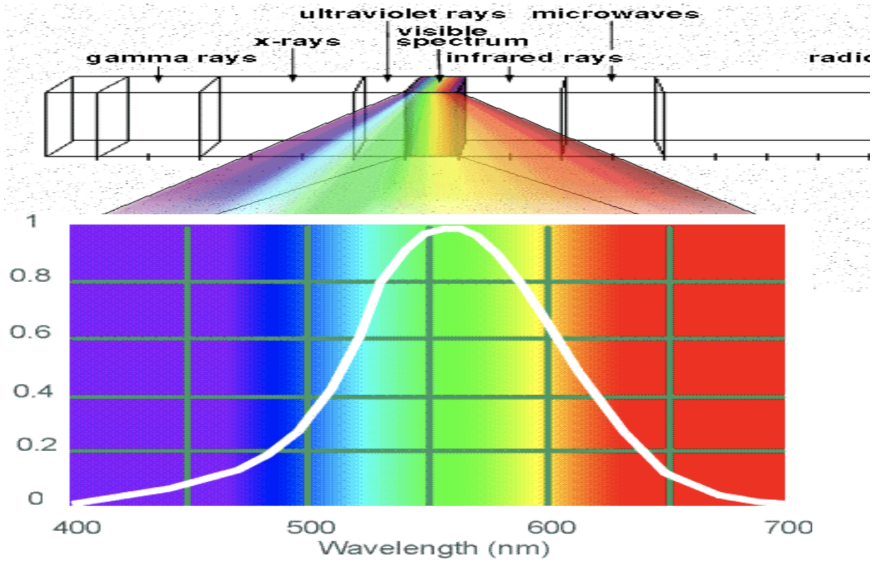
- Eye is a camera
- Isis - colored annulus with radial muscles
- Pupil - hole (aperture) whose size is controlled by the iris
- Film is the sensors on the back of the pupil

TYpe of light sensitive receptors

- Cones
    - Cone shaped, color vision, ghith light
- Rods
    - Gray scale, low light

- Fovea, middle of the eye has much more resolution

Human Luminance Sensitivity Function



Human Luminance Sensitivity Function

-
- Black body radiation emits waves and specific set of frequencies
- That's where the sun emits radiation

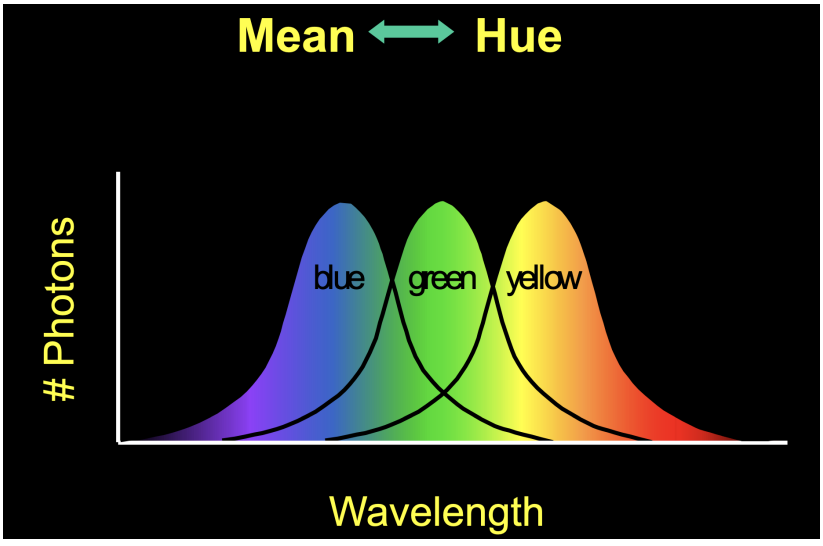
Physics of Light

- Examples of reflectance spectra of surfaces
- Reflected light gets visible

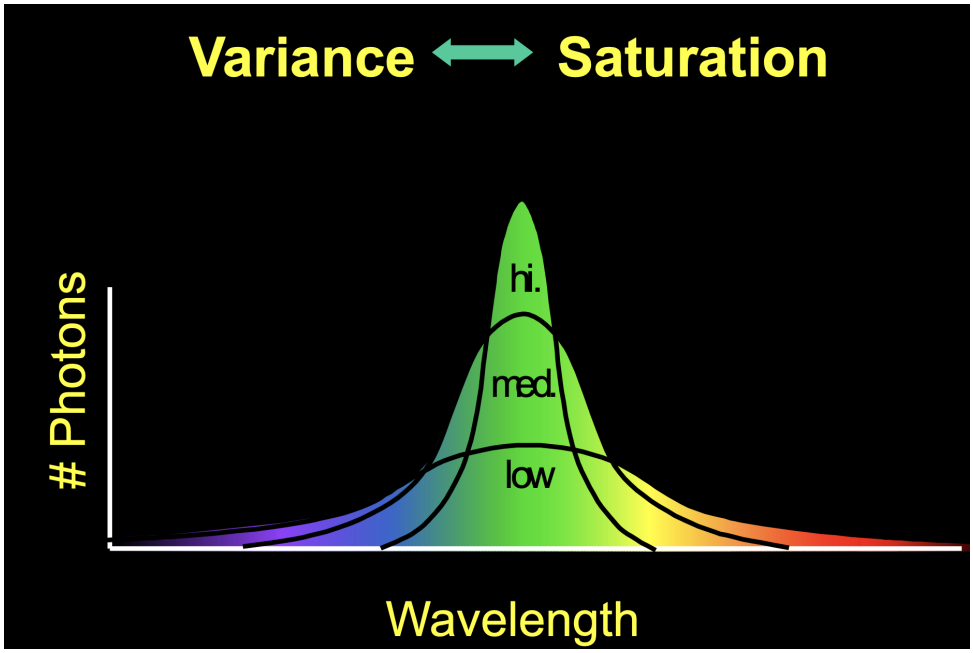
Psychophysical Correspondence

- Consider only physical spectra with normal distributions

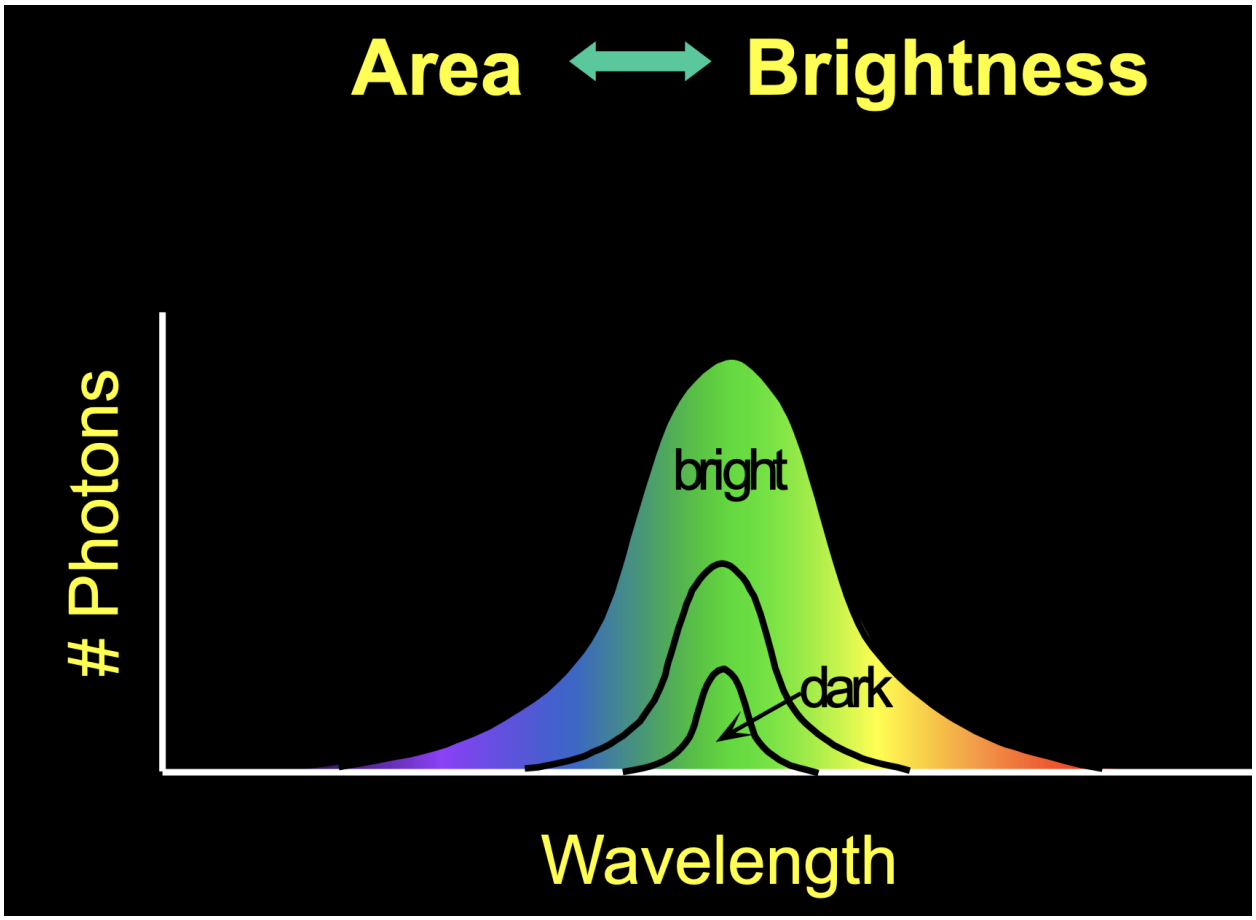
Hue



Saturation

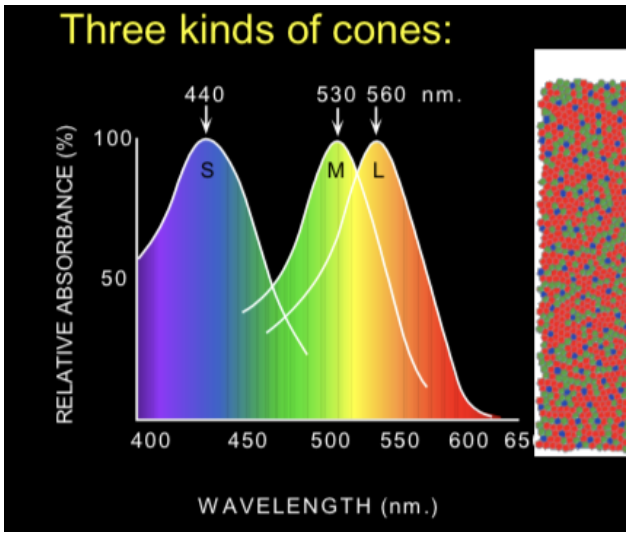


Variance
brightness



Trichromacy: 3 different

- rods and cones act as filters on the spectrum
- Humans can only project onto the 3 that we can see



Photometer metaphor:

- color perception is determined by spectrum of light on each retinal receptor (measured by a photometer)
- Able to sorted out the illuminum and color
- Eye is doing more processing

Do we have constancy over all global color transformations

Color Constancy

- Ability to change perception based on what the color filter is applied on

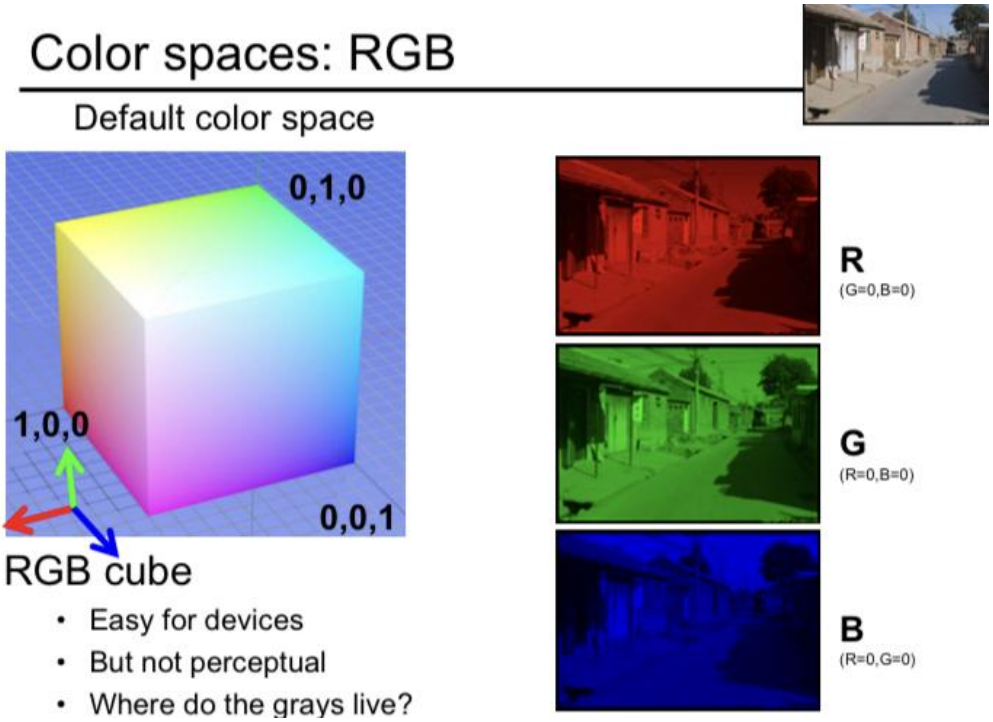
Camera WHite Balancing

- Manual
  - Choose color natural object and normalize
- Automatic
  - Gray world: force avg color of scene to gray
  - White World: force brightest object to white

Color Sensing in Camera (RGB)

- Bayer filter
  - Each has a different filter, less blue than red and green bc we Have less blue cones
- 3 filters
- Sum of squared differences (SSD)
- Normalized correlation (NCC)

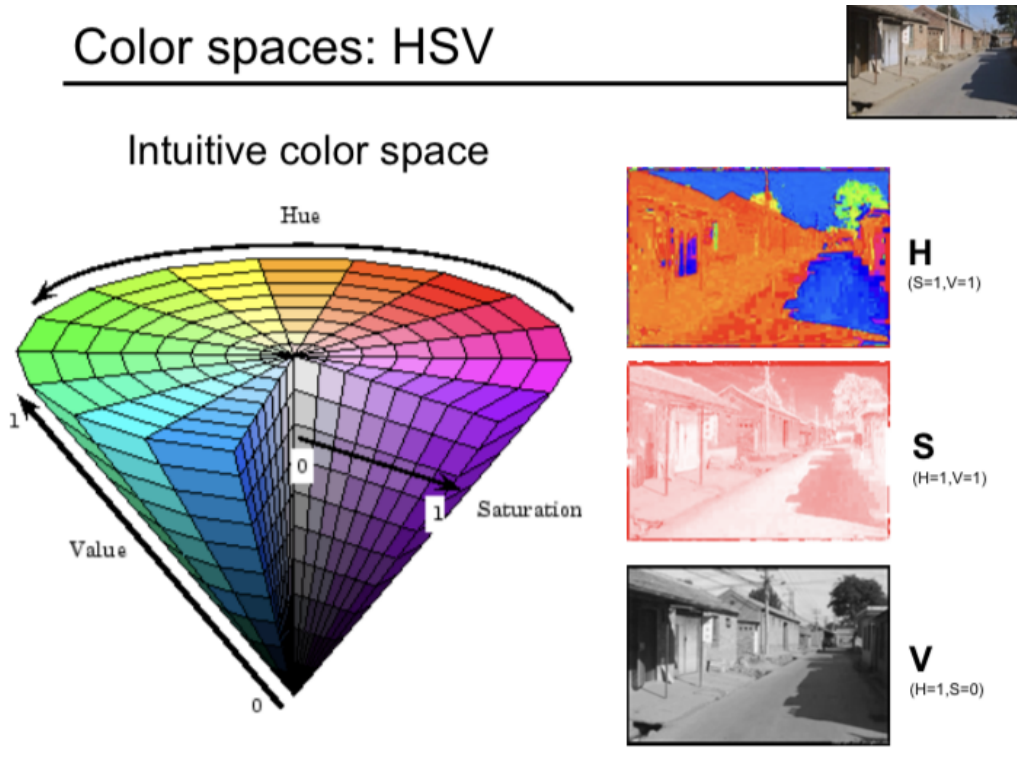
Week 2: Lecture 3 Color Spaces (9/1)



RGB

- easy for devices but not perceptual

Hue Saturation Value



Cube turned on corner

L* a*b

- perceptually uniform color space

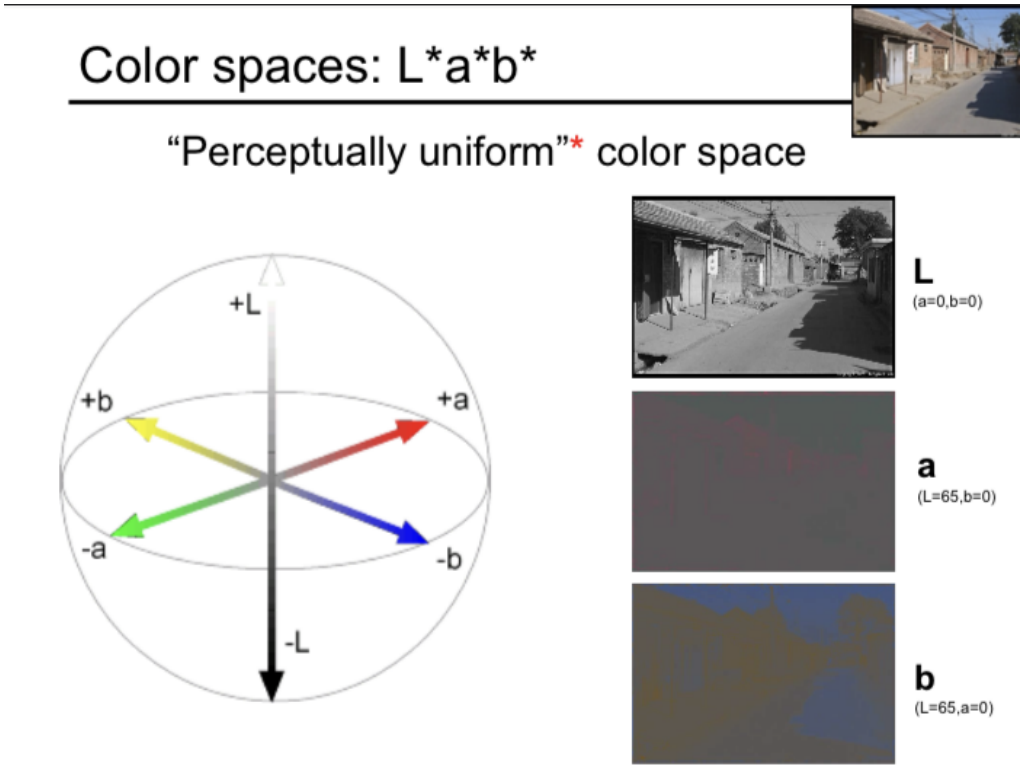


Image FOrmation:

- can think about it in terms of functions
- F(x, y) = reflectance(x, y) * illumination (x, y)

Dynamic Range

- High Dynamic Range

Long Exposure

- has to fit inside the 256, don't trust 0 or 255

Simple POint Processing: enhancement

- Making it look better y adjusting parameters
- Point processing: computing some brightness for every pixel

1. Log,
2. Contrast Stretching
3. Image Histograms
    a. Dynamic range image makes histogram equalization
    b. Given an image make image as uniform as you can
    c. S = T(r) where T is the cumulative histogram, maps to the y axis

Limitations of Point Processing

- Drawback: independent of each pixel

1D Audio

- Digital analog circuit

- Aliasing: signals traveling in disguise, true signal is lowing

Antialiasing

- Sample more often, cannot go on forever
- Make the signal less “wiggly”, will lose information
- Have low pass filter to remove high frequencies

Linear filtering: a key idea

- Transformations on signals:
    - Bass/treble controls on stereo
    - Blurring/sharpening operations in image editing
- Key properties
    - Linearity: filter(f + g) = filter(f) + filter(g)
    - Shift invariance: behavior invariant to shifting the input
- Can be modeled mathematically by convolution

Moving Average

- definite a new function by averaging over a sliding window

Cross-correlation

- F be image, Hwe kernel, G be output image
- G[i, j] = sum h[k, l] f[m + k, n + l]

Image filtering

- smoother version of image, more blurry

Box Filter

- box replaces image with its average

Can use Weighted Moving Average instead

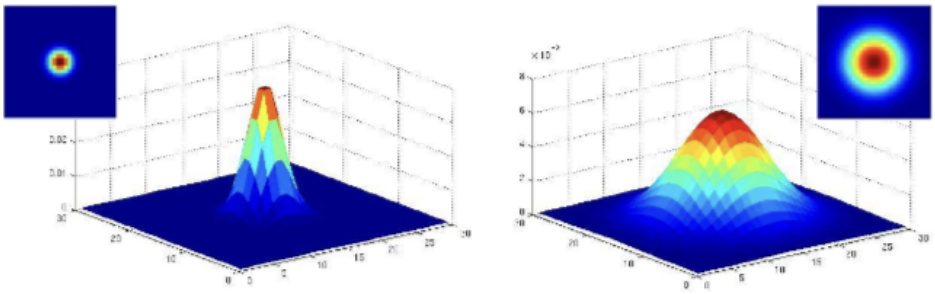
Week 3: Lecture 4 Convolution and Image Derivatives (9/8)

Use a bell curve (Gaussian) filter

- normalized by weights
- Gaussian is a better blur

Gaussian Kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



σ = 2 with 30 x 30 kernel

σ = 5 with 30 x 30 kernel

Standard deviation σ: determines extent of smoothing

-
- Sigma determines the gaussian filter difference
- Want to make sure it is smooth, can't have the box too tight or too big

Rule of thumb: size of window should be 3 sigma

Cross-correlation vs Convolution

cross-correlation: $G = H \otimes F$

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

A **convolution** operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

It is written:

$$G = H \star F$$

Convolution is **commutative** and **associative**

-
- First flips it upside down then left right then does cross correlation

- cross-correlation: flipped, convolution make it proper mathematical operation, can apply filters one after another
- Precompute filter and add it to image once

Gaussian and convolution

- Removed high-frequency components from the image (low-pass filter)
- Convolution with self is another gaussian
- Convolving twice with gaussian

Image Half-sizing

- image subsampling
    - Take every other pixel
    - Aliasing problem
- Gaussian (lowpass) pre-filtering
    - Solution: filter the image, then subsample

Image (Gaussian Pyramid)

- take an image, prefilter, subsample, prefilter, subsample

What are they good for?

- Improve Search
    - Search over translations
        - Coarse to fine strategy
        - Project 1
    - Search over scale

Derivative by convolution

Partial derivatives with convolution

Image is function f(x,y)

Remember: $$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x,y)}{\epsilon}$$

Approximate: $$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$

-1	1

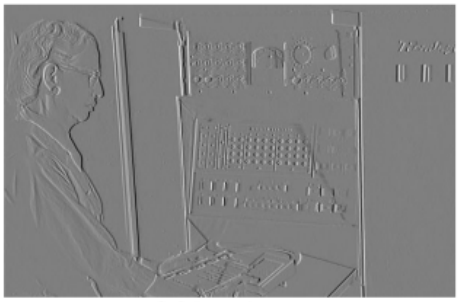
Another one: $$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x-1,y)}{2}$$

-1	0	1

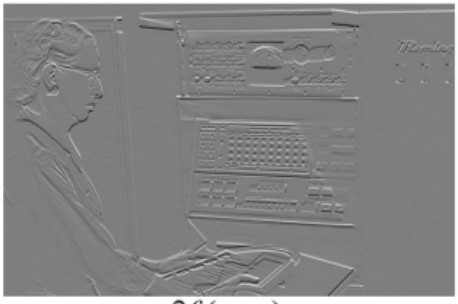
-

Image Gradient

- Points to the highest number (white)
- Edge strengthen is given by the gradient magnitude
- Arctan tells you the direction of the gradient
- Computer the gradient
- Gray images in computer means there is pos and neg and gray becomes 0
- Magnitude tells you the edges



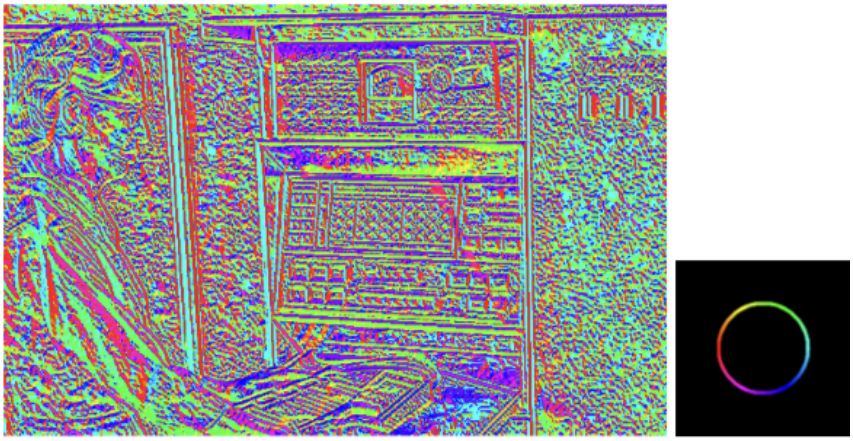
$$\frac{\partial f(x,y)}{\partial x}$$



$$\frac{\partial f(x,y)}{\partial y}$$

Image Gradient

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

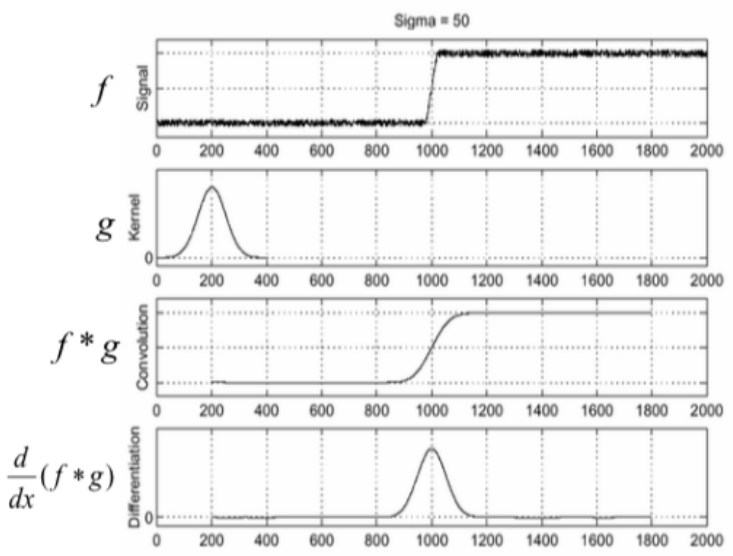


Now I'm showing *all* the gradients

Source: D. Fouhey

Blur to get rid of high frequencies in noise

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S

Derivative theorem of convolution

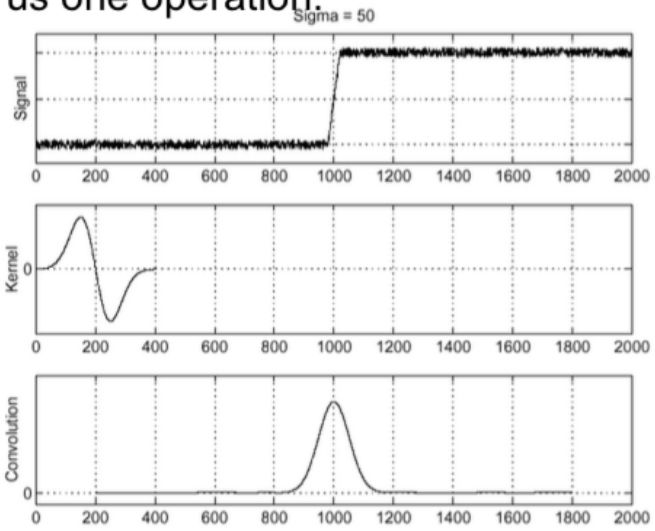
Derivative theorem of convolution

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



Classic derivative filters

- Prewitt:
- Sobel: approximation of derivative of gaussian

0	-1	0
-2	0	2
0	1	0

Filtering: practical matters

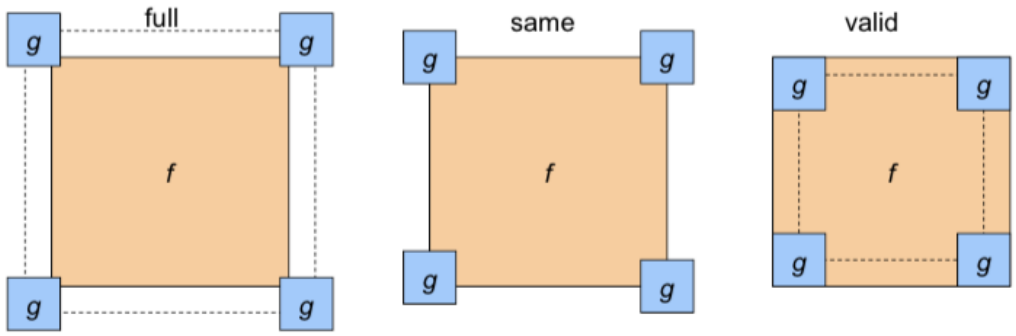
- Shape full, same, valid

Filtering: practical matters

What is the size of the output?

(MATLAB) filter2(g, f, *shape*) or conv2(g,f,shape)

- *shape* = 'full': output size is sum of sizes of f and g
- *shape* = 'same': output size is same as f
- *shape* = 'valid': output size is difference of sizes of f and g



-

Either do valid, same: wrap around, copy edge, reflect across edge

Week 4: Lecture 5 Frequency Domain, without tears (9/13)

Spatial Frequencies and Perception

- Campbell-Robson contrast sensitivity curve

Jean Baptiste Joseph Fourier

- Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies

A Sum of sines

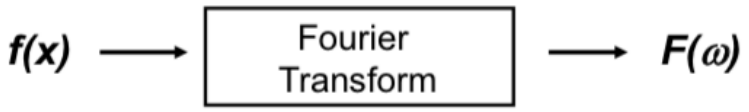
- building block
- Asin(\omega x +\phi)
- Add enough to get any signal f(x) you want
- 3 degrees of freedom
- Which encodes the coarse vs fine structure of the signal

Fourier Transform

- understand the frequency \omega of the signal.
- For every \omega from 0 to inf, F(\omega) holds the amplitude A and phase \phi of the corresponding sin Asin(\omega x + \phi)

Fourier Transform

We want to understand the frequency ω of our signal. So, let's reparametrize the signal by ω instead of *x*:



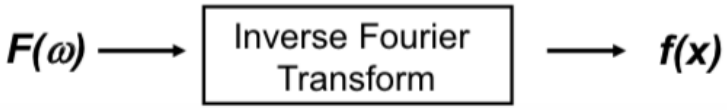
For every ω from 0 to inf, ***F(ω)*** holds the amplitude *A* and phase ϕ of the corresponding sine $A\sin(\omega x + \phi)$

- How can *F* hold both?

$$F(\omega) = R(\omega) + iI(\omega)$$

$$A = \pm\sqrt{R(\omega)^2 + I(\omega)^2} \qquad \phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$$

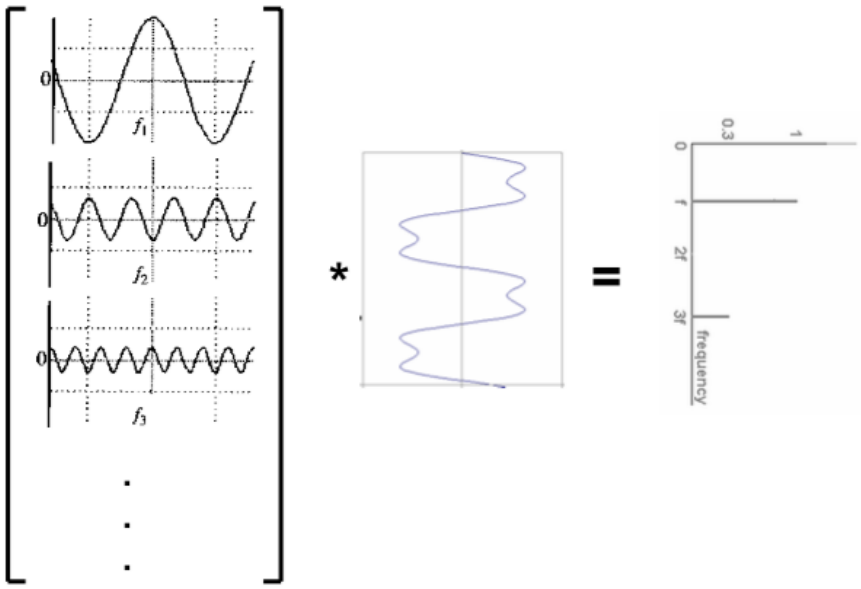
We can always go back:



-

FT: Just a change of basis

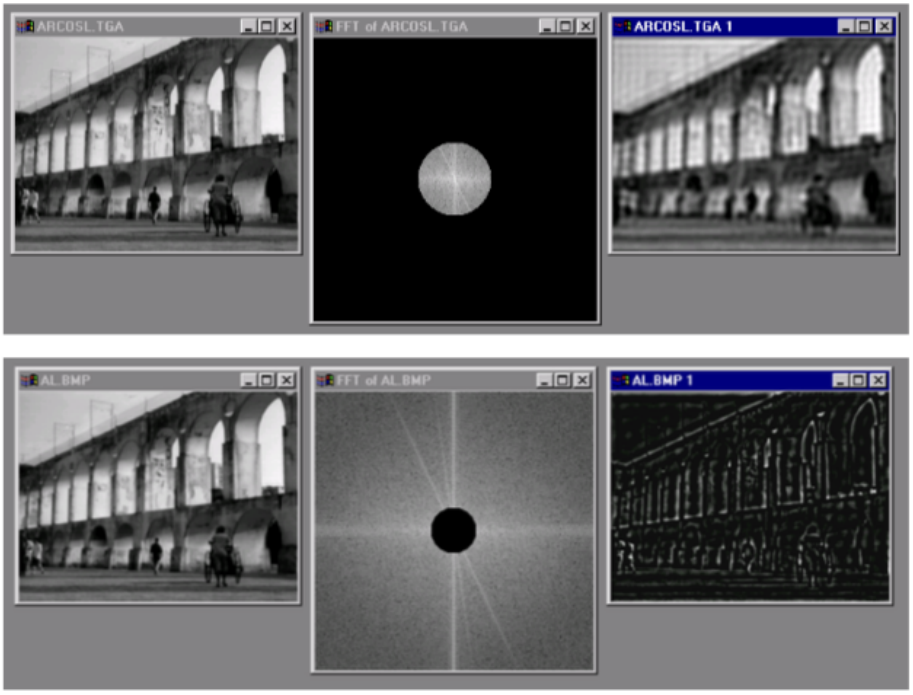
M * *f(x)* = *F(ω)*



-

Man-made scene

- Keep everything up to certain frequency



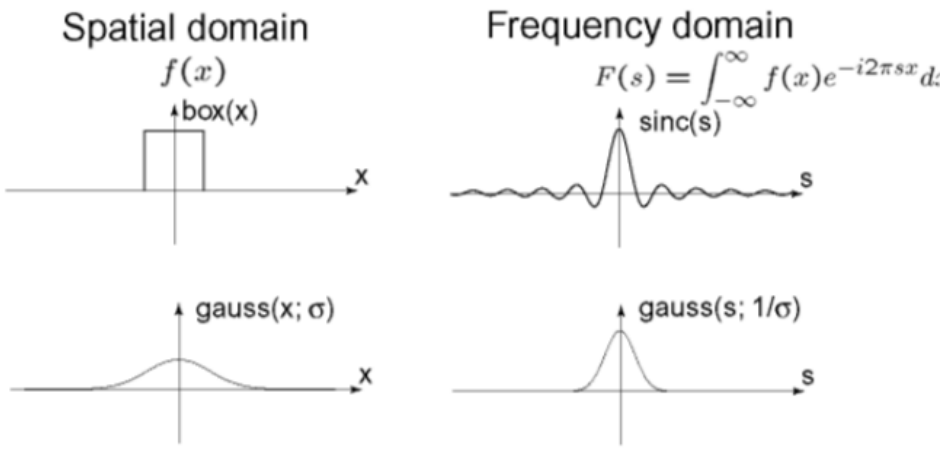
- take a chunk from the frequency graph

Convolution Theorem

- Fourier transform of the convolution of two functions is the product of their Fourier Transforms
    - F[g * h] = F[g] F[h]
- Inverse Fourier transform of the produce of two Fourier transforms it he colvolution of the two inverse Fourier transforms
    - F^-1[gh] = F^-1[g] * F^-1[h]
- Courier domain or spatial domain to do Fourier

Fourier Transform pairs

- box filter Fourier transform is sinc



Gaussian is well behaved in spatial domain and frequency domain

Week 4: Lecture 6 Pyramid Blending, Templates, NL Filters (9/15)

Gaussian is not perfect

FIltering — sharpening

- Image + alpha Details (high frequencies)

-

Unsharp mask filter

- $$f + \alpha(f - f * g) = (1+\alpha)f - \alpha f * g = f * ((1+\alpha)e - \alpha g)$$

  image: f; blurred image: $f * g$; unit impulse (identity): e

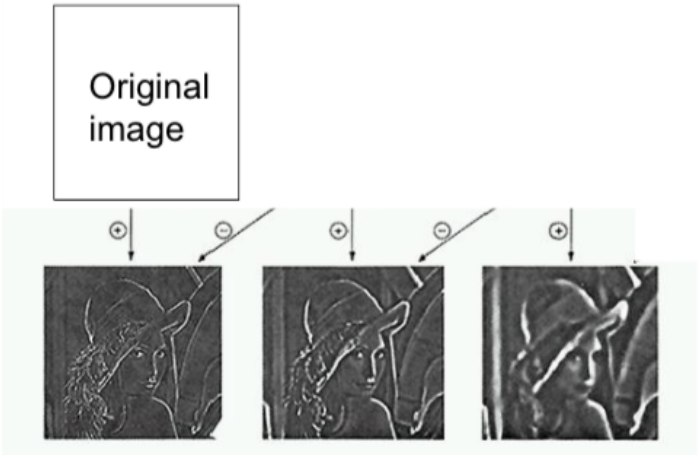
Make edges look sharper

Hybrid Images

- align images

Band pass filtering in spatial domain

- gaussian pyramid (low-pass filter)
- Laplacian pyramid
- Keep subtracting image from gaussian
- layers of bands



Lending ‘

Vehicles

- becoming transparent can seem one through the other
- “Optimal window, to avoid seams,
  - Avoid seams
    - Window = size of largest prominent features
  - to avoid ghosting : two different features lying on top of each other
    - Window <= 2 *size of smallest prominent features
  - Natural to cast this in the Fourier domain
    - Largest frequency <= 2 * size of smallest freq
    - Image frequency content should occupy one octave (power of two)

Select once blending region

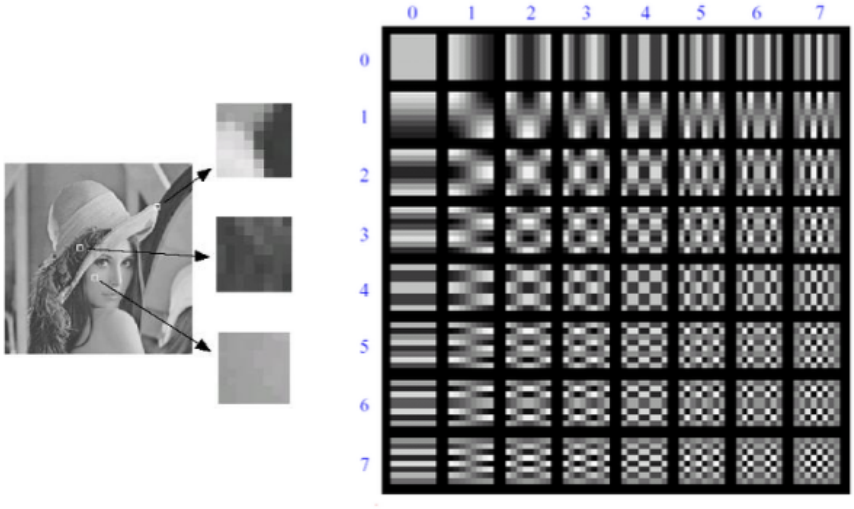
Laplacian Pyramid: Blending

1. Build a laplacian pyramids LA and LB from images A and B
2. Build a gaussian pyramid GR from selected region R

3. Form a combined pyramid LS from LA and LB using nodes of GR as weights
    a. LS(i, j) = GR(l, j) * LA(l, j) + (1 - GR(l, j)) * (LB(l, j)
4. Collapse the LS pyramid to get the final blended image

Side note: image compression

- Store
- Lossless compression: Huffman coding
    - shorter encodings for the colors that are more common
    - Not effective for larger images
- Lossy Image Compression (JPEG)
    - Some information humans can't see
    - Use limits of human perception as way to compress the data
    - Block-based Discrete Cosine Transform (DCT)
    - On each block run a version of Fourier transform

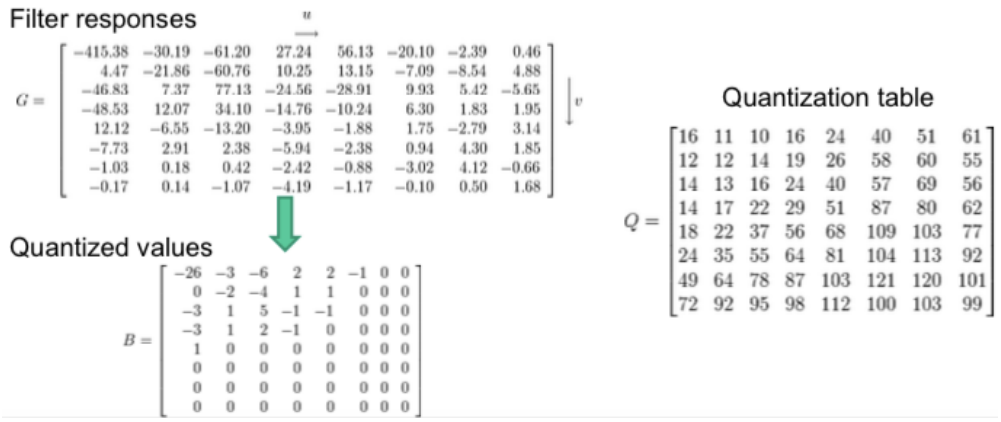


Block-based Discrete Cosine Transform (DCT)

- every block encode in freq representation separating high freq and low freq
- Get rid of high freq

Image compression using DCT

- Quantize
    - More coarse lay for high frequencies (smaller values)
    - Quantized high freq values will be zero
- Encode
    - Can decode with inverse DCT



$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

1. Computer DCT coefficients
2. Coarsely quantized
    a. Many high freq components will become zero
3. Encode (Huffman)

Block size : 8x8

Subsample color by factor of 2

- people have bad resolution for color

Review: Smoothing vs derivative filter

smoothing filters

- Guassian: remove “high frequency” components; “low pass filter”
- Non negative smoothing
- Values sum up to 1

Derivative Filters

- derivatives of Gaussian
- values can be negative
- Values should sum up to 0
- high abs values at point of contrast

Template matching

- good similarity or difference between one or other
- Use derivative filter by subtracting filter
- L2 SSD does not work if the color changes
- Normalized cross-correlation works better
- Zero mean filter: fastest but not a great matc Cher
- SSD: nest fastest, sensitive to over intensity
- NCC: slowest invariant to local average intensity and don’t start s

Week 5: Lecture 7 Image Transformations (9/20)

Image Transformations

- image filtering: change range of image

- G(x) = T(f(x))

image filtering: change ***range*** of image

$g(x) = T(f(x))$

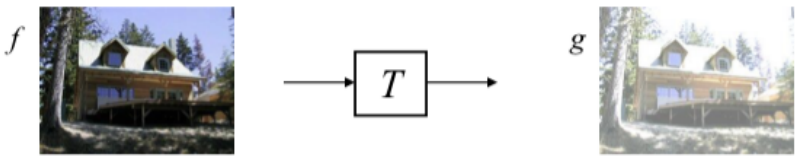
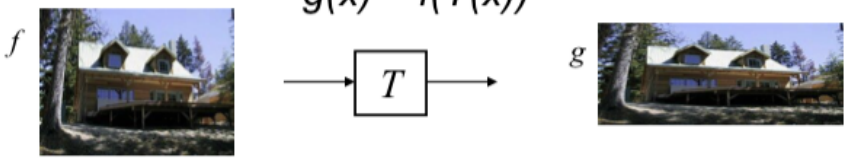


image warping: change ***domain*** of image

$g(x) = f(T(x))$



-

Global Warping

- Transformation T is a coordinate-changing machine
- P' = T(p)
- Represent linear T as a matrix

$$p' = \mathbf{M}p$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scaling

- Scaling a coordinate means multiplying each of its components by a scale
- Uniform scaling: scalar is the same for all components
- Non-uniform scaling: different scalars per component

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Inverse is transpose, found by negative angle theta
Shear

2D Shear?

$$x' = x + sh_x * y \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$y' = sh_y * x + y$$

Mirror around Y axis

2D Mirror about Y axis?

$$x' = -x \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$y' = y$$

2D Mirror over (0,0)?

$$x' = -x \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$y' = -y$$

All 2d linear transformations are produced by 2x2 coefficient matrix
Properties

- origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

Linear Transformation as Change of Bases

- Any linear transformation is a basis
- To get back to Cartesian basis, take inverse of conversion matrix

Homogeneous Coordinates

- How can we represent translation as a 3x3 matrix
- Represent coordinates in 2 dimensions with a 3 vector

$$x' = x + t_x$$
$$y' = y + t_y$$

\: Using the rightmost column:

$$\mathbf{Translation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Add a 1 at the end and can combine transformations and translation

Affine Transformations

- Affine Transformations are combinations of ...
    - Linear transformations and
    - Translation

Properties of affine transformations

- origins does not necessarily map to origin
- lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition
- Models Change of basis
-

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

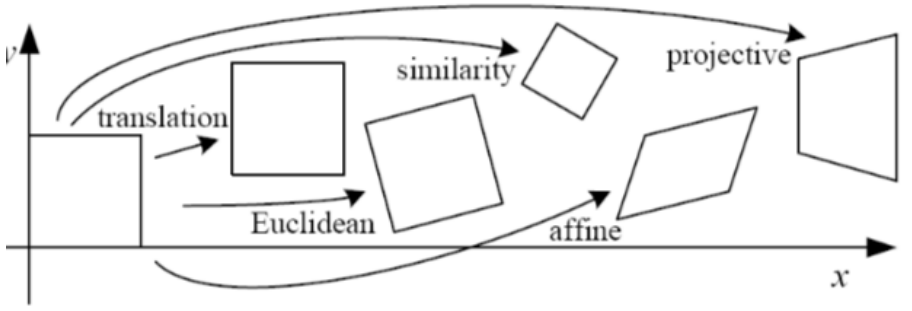
Projective Transformations

- affine transformations and projective warps
- Properties of projective transformations
    - Origins does not necessarily map to origin

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Translation: DoF 2, preserves orientation
Rigid is transformation + rotation 3 degrees of freedom: preserves lengths change
Similarity: 4 DoF, preserves angles
Affine: 6 degrees of freedom, preserves parallelism
Projective: 8 degrees of freedom, preserves straight lines



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$			
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$			
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$			
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$			
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$			

Week 5: Lecture 8 Image Warping and Morphing (9/22)

Recovering Transformations

- What if we know f and g and want to recover the Transformations T
- How many correspondences needed for translation
- How many degrees of freedom

- What is the transformation matrix
- 2 correspondents needed for translation

Recovering Transformations + rotation

- 2 correspondents needed for trial at ions

Recovering Affine: # correspondences

- 3 correspondents, have 6 points

Recovering Projective

- 4 correspondents

Warping Triangles

- affine transformation, 3 correspondents, every triangle related

Convert course to cart Indiana and then to destination

Compute transformation matrix either algebraically or geometrically

- algebraically -> Use matrix and solve linear equations
- Geometrically -> consider liens as vectors and solve for change in basis for new transformation
    - Convert source to Cartesian, then to transformed

Morphing = Object Averaging

- aim to find an average between two objects
- Not an average of two images of objects but an image of the average object
- Make a weight average over time

Averaging points

- draw vector between two points and find the average by getting half of the vector

Extrapolation

- get P influenced by Q but even further long the line
- Interpolate whole images
- Image_halfway = (1 - t) * Image_1 + t * image

First aligned the image then do a cross dissolve

- Morphing = warping + cross-dissolve

2 Stage Morphing Procedure

- for every t
    1. Find the average shape (the "mean" dog)
        a. Warping
    2. Find the average color
        a. Cross-dissolve the warped images

Global warp not always enough

Can tell computer a set of correspondences to align to each other and computer does the rest

Feature matching!

- Nose to nose, tail to tail
- Optialflofiled, have whole correspondance 10,000 dn't ties capjs apt too much

Triangular Mesh

1. Input correspondences at key features points
2. Define a triangular mesh over the points
    a. Smash mesh in both images
    b. Have triangle to triangle correspondence
3. Warp each triangle separately from source to destination

Warping Pixels

- Given a coordinate transformation ((x′, y′) = T(x, y) compute the transformed image
- Pixel lands between two pixels

Forward Warping

- Send each pixel to its corresponding location

Inverse Warping

- Get each pixel g(x′, y′) from its corresponding location
- (X, y) = T^-1(x′, y′) in the first image
- Interpolate color value from neighbors
- Interp2
- Inverse warping is better

Triangulations

- triangulation is partition of convex hull to triangles, points

Quality Triangulation

- can upgrade the triangles by having a notion of quality of triangulation and make it better than T_2 lexicographically
- Delaunay triangulation is the best
    - Maximizes smallest angles
    - Flip any illegal edge until no more exist
    - Cool take a long time to terminate
- Draw the voronoi diagram by connecting each two neighboring sites in the Voronoi diagram
- DT may be constructed in O(blogs) time

1. Create average shape
    a. Create intermediate warp at time t
    b. Use interpolation
2. Create average color
    a. Cross dissolve
3. Compute the average shape at t

Week 6: Lecture 9 Data-driven Methods: Faces (9/27)

Project 3: morphing

1. Define corresponding Points
    a. 30-50 points
2. Define Triangulation on Points

    a. Use state triangulation on both images
3. For each t = 0: step : 1
    a. Compute the average shape at t (weighted average of points)
    b. For each triangle in the average shape
        i. Get the affine projection to the corresponding triangles in each image
        ii. For each pixel in the triangle, find the corresponding points in each image and set value to weighted average (cross-dissolve each triangle)
    c. Save the image as the next frame of the sequence

Foreground:

- make sure the foreground is similar,s are lighting can remove background to white

Other issues

- Beware of folding
    - Probably trying to do something 3d
- Morphing can be generalized into 3D
    - If you have 3D data, that is
- Extrapolation can sometimes produce interesting effects
    - Caricatures

Power of Averaging

- Transient data gets averaged out

8 hour exposure

- longer

Image Composites

- sir Francis Galton
- Multiple averages of people

100 Special moments

- blurry bc not aligned correctly

Computing Means

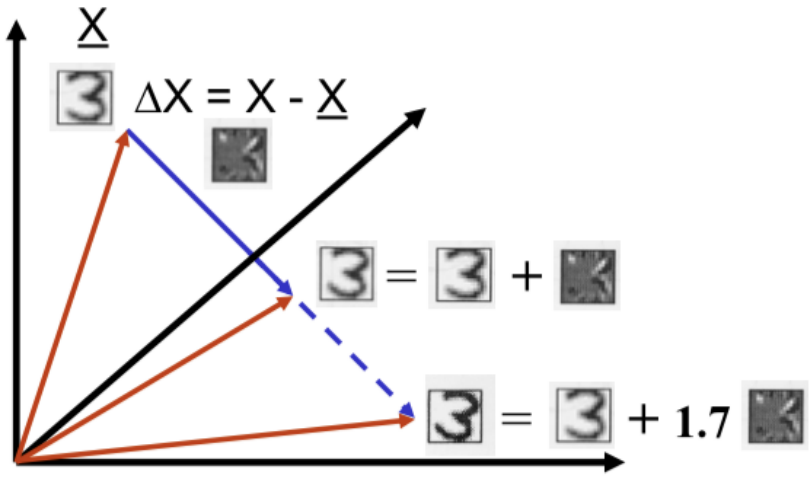
- Two requirements
    - Alignment of objects
    - Objects must span a subspace
- Useful concepts
    - Subpopulation means
    - Deviations from the mean
- Average becomes better looking combing best features / symmetry

Objects must span a subspace

- must belong to subspace
- faces are subspace
    - Glasses are not conformed to subspace
    - Beards
    - Teeth

Deviations from the mean

- Image x - mean x = deviations
- Background subtraction



- Add the delta to make it more exaggerated

Manipulating faces

- Make a face look more email/male, young/old
- Have face and move more toward direction of more smile subset

Still get result about what is in data

- just need subpopulation means and subtraction

Face Modeling

1. Compute average faces
2. Compute deviations between male and female
3. Devore shape and/or color of an input face in the direction of more female
    a. Changing age, face becomes rounder and more textured and grayer

Linear Subspace: convex combinations

- any new image X can be obtained as weighted sum of stored “basis” image s

Any new image X can be obtained as weighted sum of stored “basis” images.

$$X = \sum_{i=1}^{m} a_i X_i$$

-

Morphable face model

- Structure of face is captured in the shape vector
- S = (x_1 , y_1, .. y_n) ^ T
- Containing the (x,y)
- Containing texture vector T containing the color values of the mean -warped face image
- Faces are not linear subspace if just straight morph
- Can only do morph if it is in a linear subspace
- Think of faces in terms of change of basis
- Take 200 Germans. Can represent self as linear combination of Germans

Issues

- How many basis images is enough
- Which ones should they be

Principal Component Analysis (PCA)

- Given a point set in an M dim space, PCA finds a basis st
    - Coeffeicients to he point set in the basis are Nucor related
    - First r < M basis vectors provide an approximate basis that minimizes the Mean squared Error in the approximation

EugeneFaces

- First popular use of PCA on images for modeling and recognition of faces
- Collet a face ensemble normalize for constraint, scale, and orientation

Principal Component Analysis

- choosing subspace dimension r
    - Look at decay of the edge values as a function of r
    - Larger r means lower expected error in the subspace data approximation
    - Keep few top eigenvectors

Week 6: Lecture 10 The Camera (9/29, 10/4)

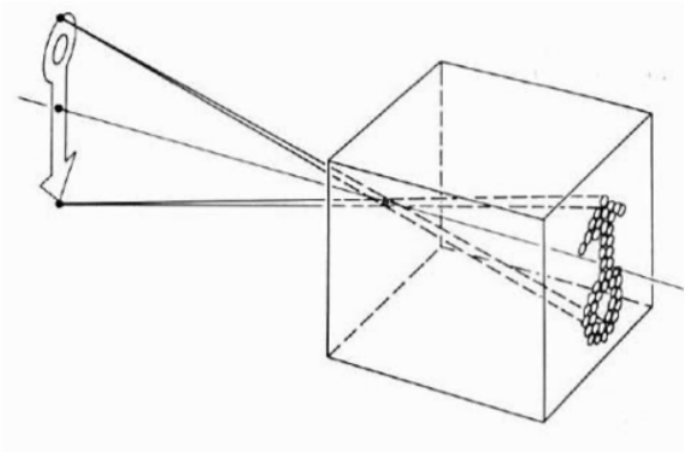
How do we see the world

- add a camera with a barrier and a hole to block off most of the rays
- Known as aperture

Pinhole Camera model

- Pinhole camera model
    - Captures pencils of rays all rays through a single point
    - Point is called center of projections (COP)
    - Image formed on the image plane
    - Effective focal length f is distance from COP to image plane

Pinhole camera model



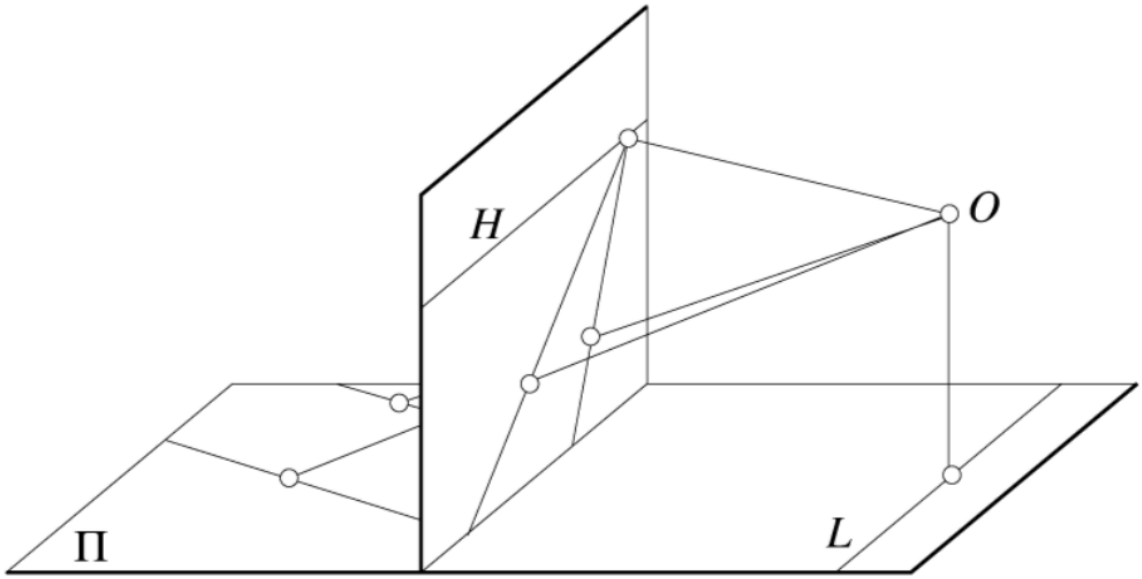
Pinhole model:

- Captures **pencil of rays** – all rays through a single point
- The point is called **Center of Projection (COP)**
- The image is formed on the **Image Plane**
- **Effective focal length *f*** is distance from COP to Image Plane

-

Emission Theory of Vision

- "For every complex problem there is an answer that is clear simple and wrong "
- Send out echolocation from eyes
- Using brain to infer solution,
- Nice to have a passive, long-range sensor
- Can get 3D with stereo or by moving around, plus experience
- What have we lost
    - Angles
    - Distances (lengths)



-

Modeling Projection

- Projection equations
    - Compute intersection with PP of ray from (x, y, z) to COP
    - Derived using similar

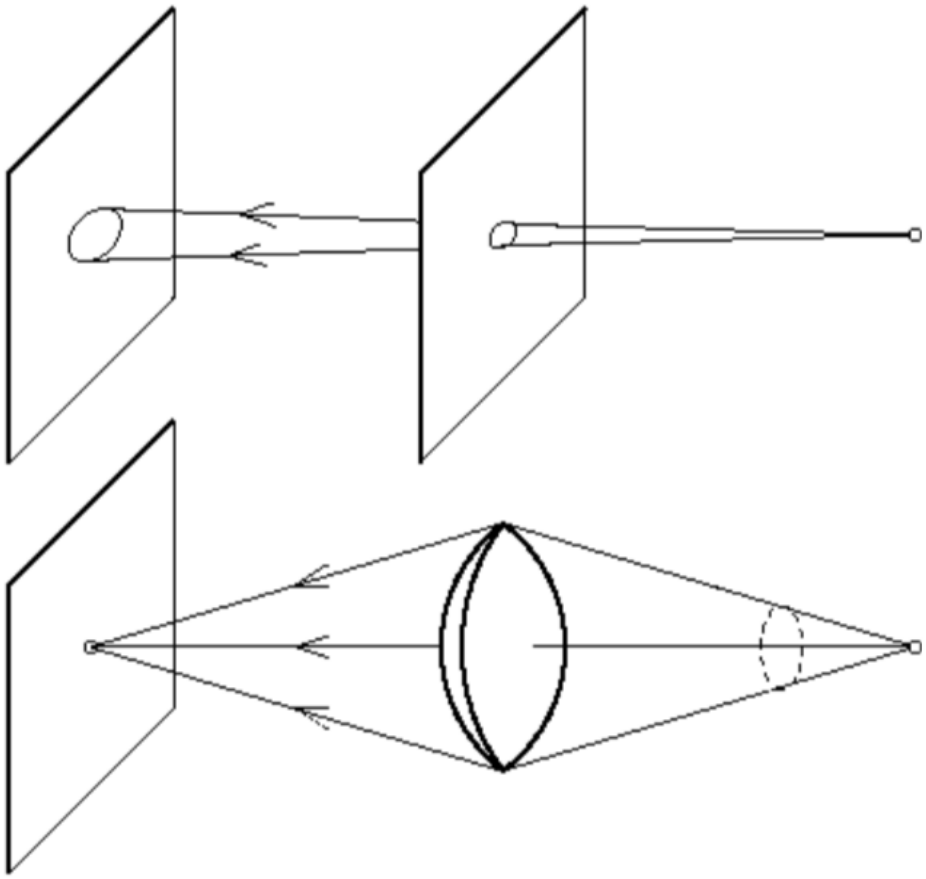
Spherical Projection

- project onto unit sphere by removing d

The Camera

- Camera obscura : use a pinhole to make a picture inside a dark room
- Shrinking the Acura river makes the the image more clear
- When the aperture becomes smaller, because of interference

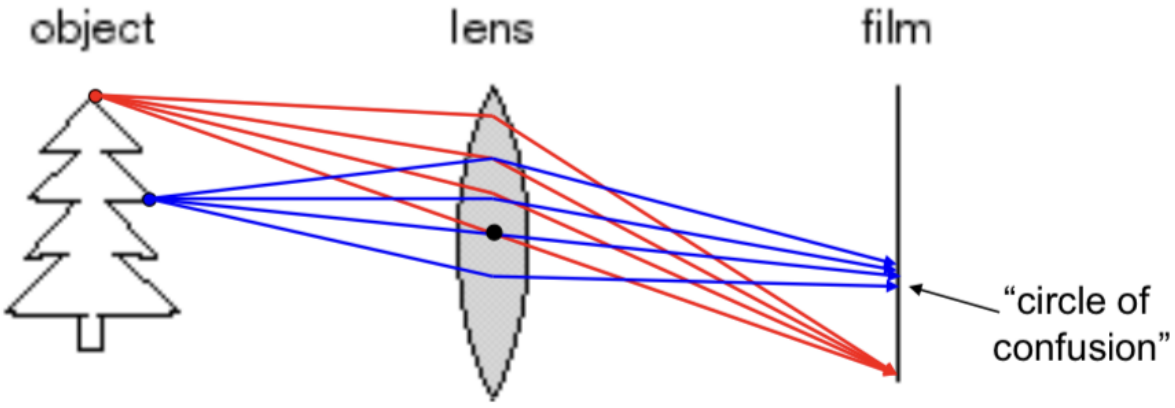
Lenses fix interference,



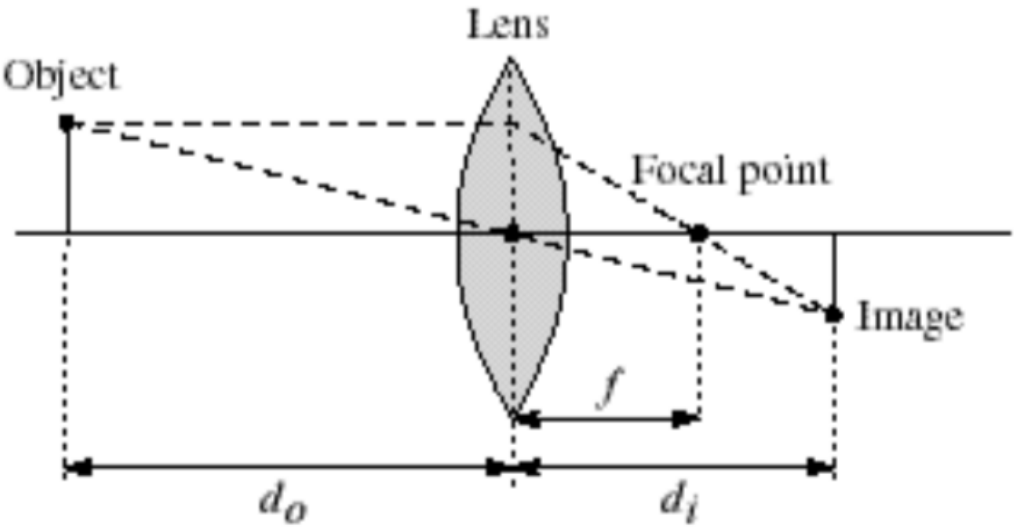
-
- reuse the ones that don't go through the pinhole

Focus and Defocus

- a lens focuses light onto the film
  - There is a specific distance at which objects are "in focus"
    - Other points project to a "circle of confusion" in the image
  - Changing the shape of the lens changes this distance



-



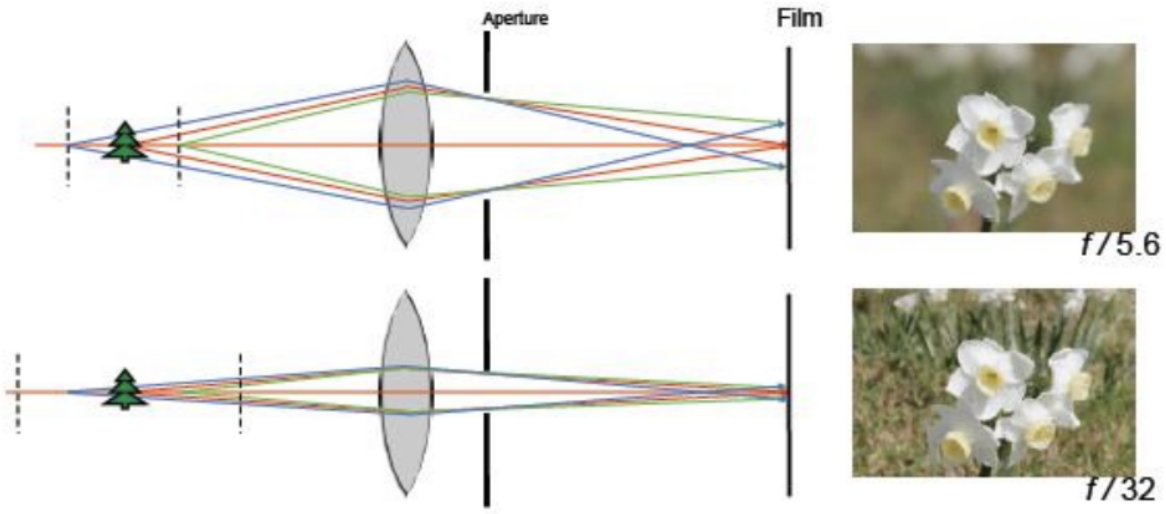
Thin lens equation: $\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f}$

- Any object point satisfying this equation is in focus
- What is the shape of the focus region?

Depth of Field

- Aperture controls depth of field

Aperture controls Depth of Field



Changing the aperture size affects depth of field

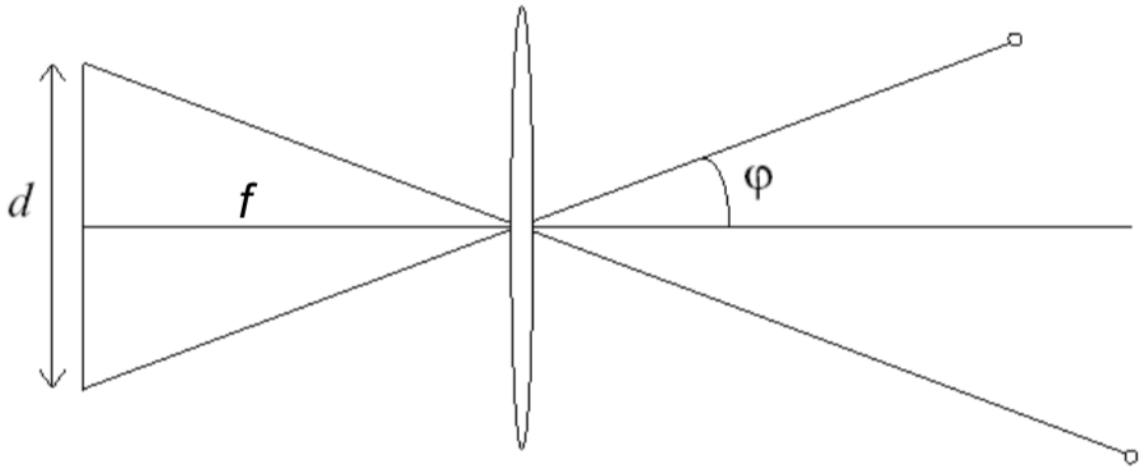
- A smaller aperture increases the range in which the object is approximately in focus
- But small aperture reduces amount of light – need to increase exposure

-

Field of View (Zoom)

- FIxed real estate of ccd array/ film can fill it in with wider view but smaller view or smaller view

FOV depends of Focal Length



Size of field of view governed by size of the camera retina:

$$\varphi = \tan^{-1}(\frac{d}{2f})$$

Smaller FOV = larger Focal Length

-
- Wide angle camera very cheap
- Narrow zoom lenses are very expensive

Field of View/Focal Length

Large FOV / small f
+ Camera close to car

Small FOV / large f
+ Camera far from the car

-
- Change field of view and move, different

Increasing focal length, makes the image less perspective

Focal length / distance in portraiture

Optical Zoom

- Lenses is changing focal length and field of view

Digital Zoom

- Cropping

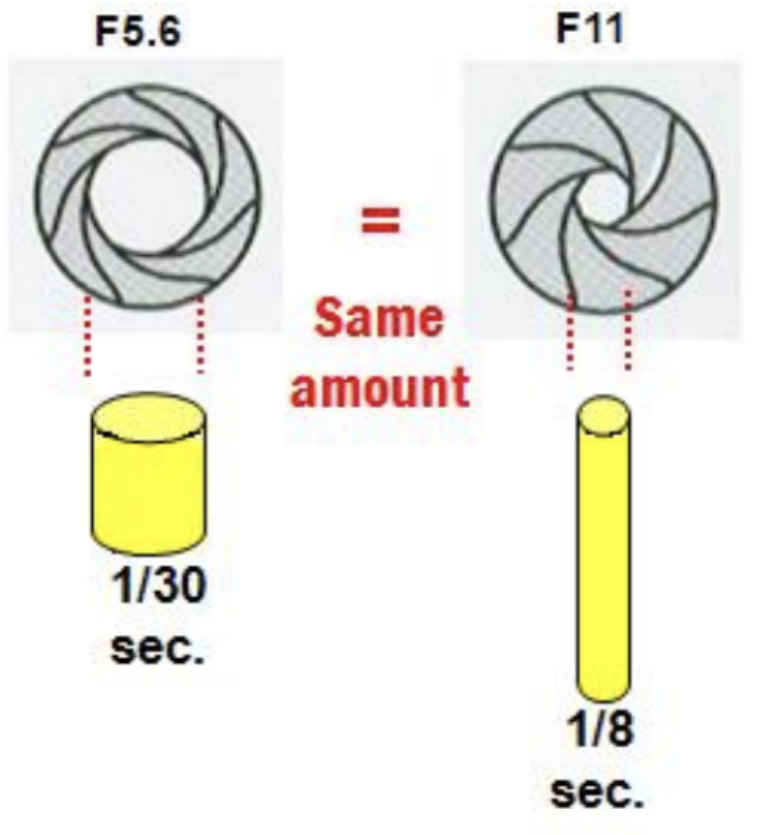
Vertigo Shot

- Change zoom and distance from shot

<u>Exposure</u>

Shutter speed

- How much time the shutter is open for



- averaging things over time by having shutter open longer
-

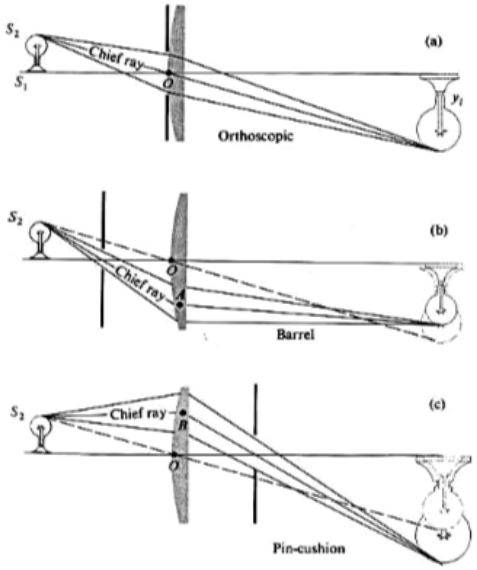
Chromatic Aberration

- gets worse on edges

Radial Distortion ('Barrel' and 'pin-cushion')

- straight lines curve around the image center

Radial Distortion



Week 7: Lecture 12 Homographies and Panoramas (10/6)

Can fake the visual experience

- Philosophy: does the world really exists
- Physics: Slowglass might be possible
- Computer Science: virtual reality, simulate what a person is seeing

Plenoptic Function

- lambda as a function of wavelength

Holographic Movie

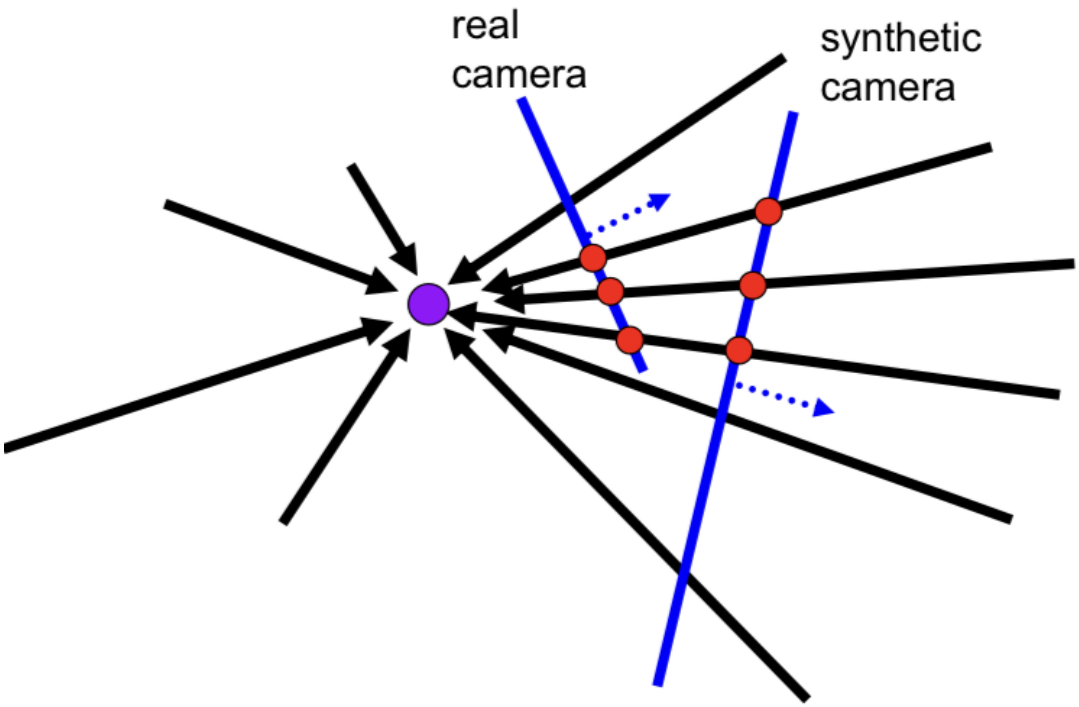
- P(theta, phi, lambda, t, Vx, Vy, Vz)
- Captures every moment from every time from every viewpoint
- Sampling Plenoptic Functions
- QuickTime VR, Google Street View is example of estimated Plenoptic

Spherical panorama

- all light rays through a Point form a panorama
- Captured in 2d array

What is an image

- pencil of arrays through the same plane



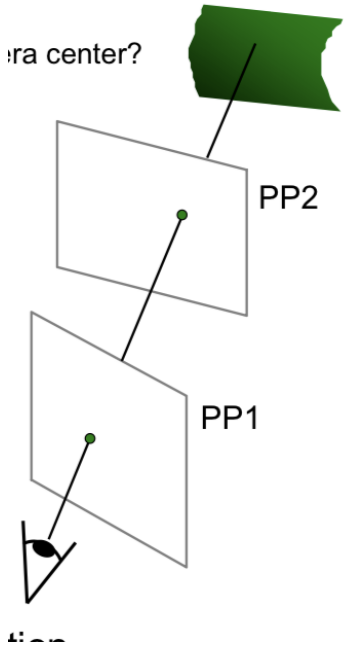
Can generate any synthetic camera view
as long as it has **the same center of projection**!

Image Reprojection

- How to relate two images from same camera center
  -
  -
- Draw the pixel
-

Homography

- projective — mapping between any two PPs with the same center of projection
- Image warping with homographies



- but must preserve straight lines
- same as: unproject, rotate, reproject

called Homography

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\mathbf{p'}$ $\quad$ $\mathbf{H}$ $\quad$ $\mathbf{p}$

To apply a homography **H**

- Compute **p'** = **Hp** (regular matrix multiply)
- Convert **p'** from homogeneous to image coordinates

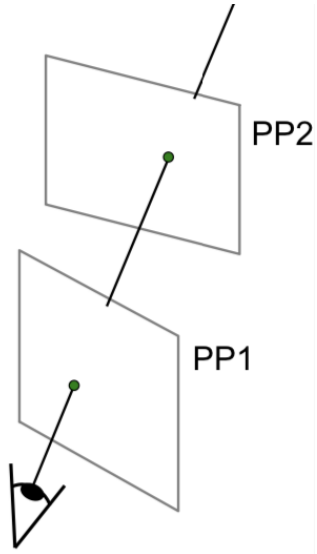
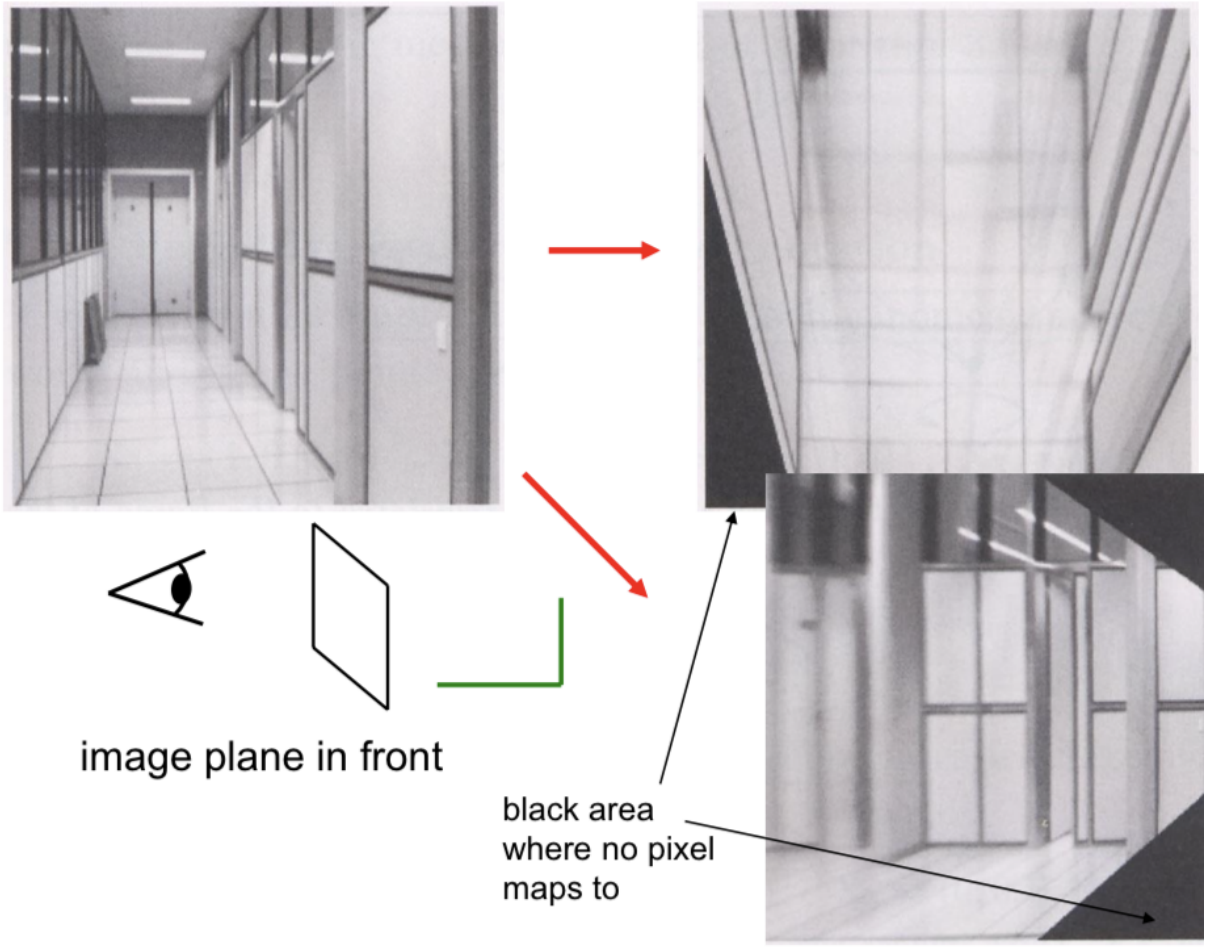


Image warping with homographies



- Projection edges are distorted around the edges

Image rectification

- to unwarp (rectify an image)
    - Find the holography H given a set of p and p' pairs
    - How many correspondences are needed
        - Can solve for it using least squares
        - 4 points, 8 unknowns

Take a picture and rectify it to another plane, homography instead of affine

Least squares - set of data points (p1, p1') , (p2, p2') ...

Want to find x1 x2, need points that fits the best

- data is noisy, can overconstrained
- Min || Ax - b ||^ 2
- X = (A^T A)^-1 A^T b

Solving for homographies

p' = Hp

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Can set scale factor *i*=1. So, there are 8 unkowns.

Set up a system of linear equations:

Ah = b

where vector of unknowns h = $[a,b,c,d,e,f,g,h]^T$

Need at least 8 eqs, but the more the better…

Solve for h. If overconstrained, solve using least-squares:

$$\min \|Ah - b\|^2$$

Can be done in Matlab using "\" command

- see "help lmdivide"

-
- Homogeneous coordinates
- Planar scene, could change the center of projection, big aerial photos work

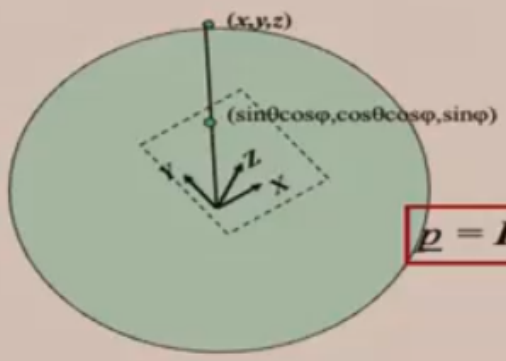
Week 8: Lecture 13 Homographies and Panoramas (10/11)

Can project onto a cylinder or sphere

Map 3D point (X,Y,Z) onto cylinder

$$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X, Y, Z)$$

Rotate image before placing on unrolled sphere



$$\begin{aligned}
\theta &= (x_{sph} - x_c)/f \\
\varphi &= (y_{sph} - y_c)/f \\
\hat{x} &= \sin\theta \cos\varphi \\
\hat{y} &= \sin\varphi \\
\hat{z} &= \cos\theta \cos\varphi \\
x &= f\hat{x}/\hat{z} + x_c \\
y &= f\hat{y}/\hat{z} + y_c
\end{aligned}$$

Polar Projection

Extreme "bending" in ultra-wide fields of view

$$\hat{r}^2 = \hat{x}^2 + \hat{y}^2$$

$$(\cos\theta \sin\phi, \sin\theta \sin\phi, \cos\phi) = s\,(x, y, z)$$

uations become

$$x' = s\phi\cos\theta = s\frac{x}{r}\tan^{-1}\frac{r}{z},$$

$$y' = s\phi\sin\theta = s\frac{y}{r}\tan^{-1}\frac{r}{z},$$

Not realistic

- Search in $O(N^8)$ is problematic
- Not clear how to set starting/stopping value and step

What can we do?

- Use pyramid search to limit starting/stopping/step values

Alternative: gradient decent on the error function

- i.e. how do I tweak my current estimate to make the SSD error go down?
- Can do sub-pixel accuracy
- BIG assumption?
  - Images are already almost aligned (<2 pixels difference!)
  - Can improve with pyramid
- Same tool as in **motion estimation**

Week 8: Lecture 14 More Mosaics (10/13)

Feature - based alignment

1. Feature detection: find a few important features in each image separately
2. Feature matching: match them across two images
3. Compute image transformation: as per Project 5, part 1

Feature detection

- not always going to get the right corners
- How to match the features between the images

Feature Matching

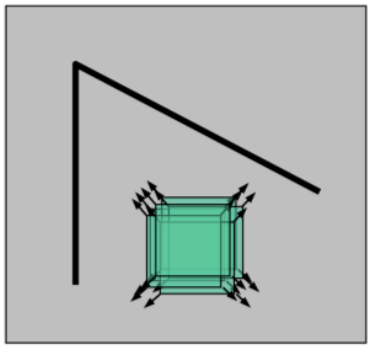
- How do we match the features between the images
- Need a way to describe a region around each features
- Use successful matches to estimate homography
    - Get rid of outliers

Applications

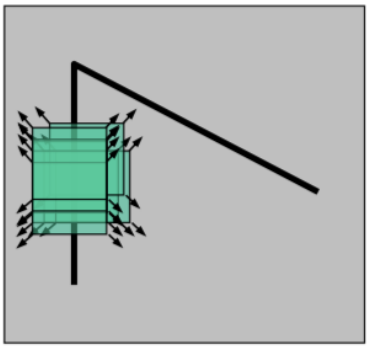
- feature points used for 3d reconstruction, motion tracking, obj recognition

<u>Feature Detector — Harris Corner</u>

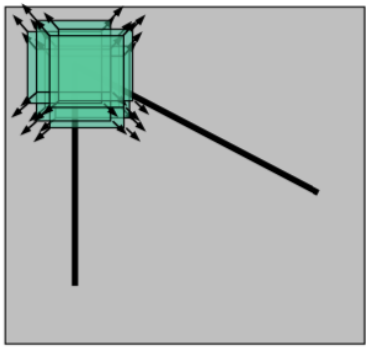
Recognize the point by looking through a small window, shifting in any direction should give us a large change in density



"flat" region: no change in all directions



"edge": no change along the edge direction



"corner": significant change in all directions

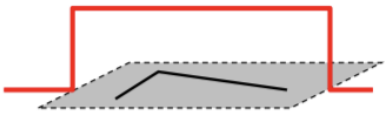
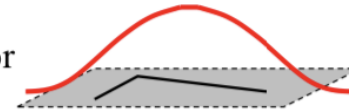
Change of intensity for the shift $[u,v]$:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

Intensity

Window function $w(x,y)$ =  or 

1 in window, 0 outside

Gaussian

For small shifts $[u,v]$ we have a *bilinear* approximation:

$$E(u,v) \cong \begin{bmatrix} u, v \end{bmatrix} \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

where M is a 2×2 matrix computed from image derivatives:

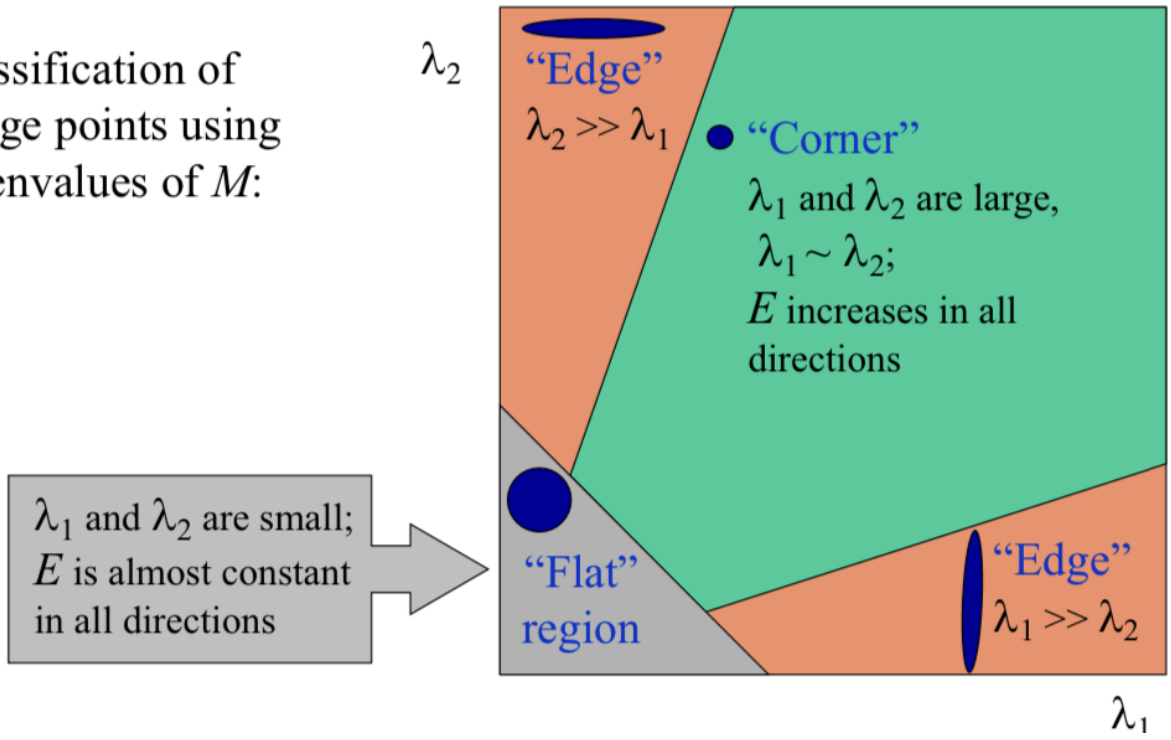
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

Second moment matrix MM

- axis lengths of ellipse are determined by the edges values and ordination is determined by R

Classification of image points using eigenvalues of M:



Measure of corner response R

$$R = \frac{\det M}{\text{Trace}\, M}$$

$$\det M = \lambda_1 \lambda_2$$
$$\text{trace}\, M = \lambda_1 + \lambda_2$$

Harris detector: steps

1. Compute Gaussian derivatives at each pixel
2. Compute second moment matrix in Gaussian window around each pixel
3. Compute corner response function R
4. Threshold R
5. Find local maxima of response function (non-maximum suppression)

Harris Detector: some Properties

- only deriv ages are used -> invariance to intensity shift I-> I + b
- Intensity scale: I -> a I

- Keep the max corner of a couple choose the scale with the best corner
- Distribute points evenly over the image
- Fixed # features per image
- Want evenly distributed spatially
- Sort points by non-maximal suppression radius
- Adaptive Non Maximal Suppression find points across whole image,

Week 9: Lecture 15 Feature Matching (10/18)

Feature Descriptors

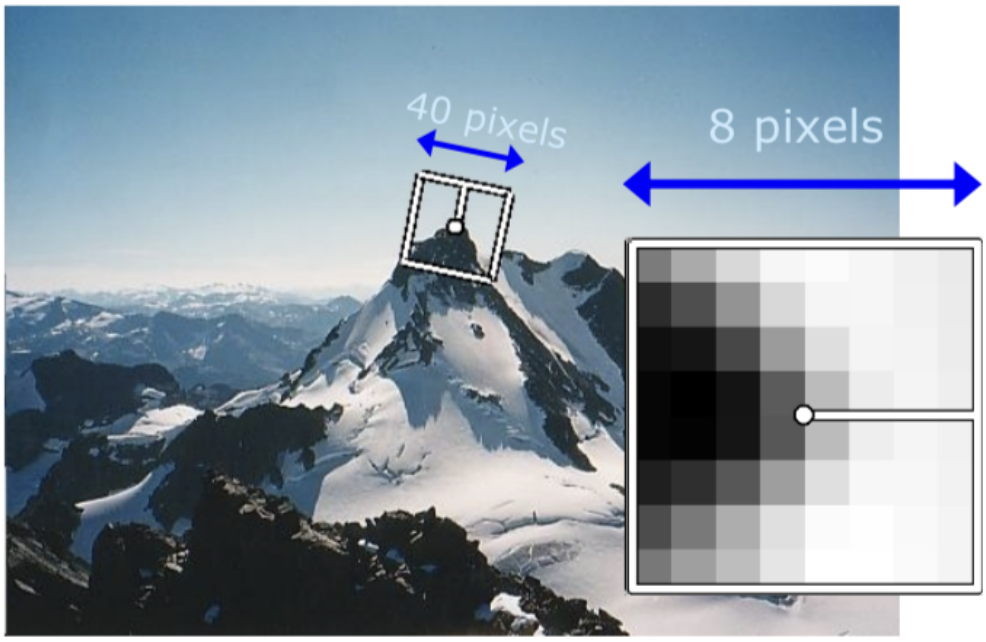
- how to match feature descriptors

Multi Scale oriented patches

- Interest points
    - Multi scale Harris Corners
    - Orientation from blurred gradient
    - Geometrically invariant to rotation
    - Geometrically invariant to rotation
- Descriptor vector
    - Bias/gain normalized sampling of local patch
        - Zero mean standard variance normalized
    - Geometrically invariant to affine changes in intensity

Detect Features, setup frame

- orientation = blurred gradient
- Rotation invariant frame
    - 8x8 oriented patch
    - Subsample each patch, save space



-

- Project: Don't need to implement scale and orientation

Feature matching

- pick the best match !
    - For every patch in image 1, find the most similar patch
    - Called nearest neighbor in machine learning

Feature -space outlier rejection

- don't match all features, but only ones that have similar enough matches
- How can we do it
- SSD(patch1, patch2) < threshold
- Force symmetric only ones that also have the same

Feature space outlier rejection: Lowe's trick

- Only choose when after looking at 2nd it is still much better

Now we have feature space outlier

- still have outliers and can't have outliers for least square regression

Find correspondences that are in consensus

- RASAC
    - Random Sample Consensus
        - Compute the corresponding e for one match, check which ones agree with this one
        - Selection another match, calculate inliners
        - Calculate 4 random homographies until you have multiple homographes and can compare against other correspondences

- RANSAC Loop
    1. Select four feature pairs (at random)
    2. Compute homography H (exact)
    3. Compute in liters where dist(pi', Hpi) < epsilon
    4. Keep largest set of i liters
    5. Re compute least squares H estimate on all of the i liters

Bundle Adjustment

- new images initialized with rotation, focal length of best matching image

Limitation of Alignement

- We need to know the global transform

Optical flow

- will start by estimating motion of each pixel separately, then i'll consider motion of entire image
- Estimate motion
    - Track object behavior
    - Correct for camera jitter
    - Align images
    - 3d Shia reconstruction, spectral effects

Optical flow

- how to estimate pixel motion from image H to image I
- Estimate each pixel motion from image H to image I
- Key assumptions
    - Color constancy: a point in H looks the same in I or brightness constancy
        - H(x, y) - I(x + v, y + v) = 0
    - Small motion: points do not move very far
- The component of the flow in i the gradient direction is determined but the compone tod the flow parallel to an edge is unknown
- Aperture problem, 1 question, 2 unknowns

Lukas-Kanade flow

- more equations than unknowns
- Solve least quests problem

Week 9: Lecture 16 Visual Texture (in human and machine) (10/20)

Lukas-Kanade flow

What is texture

- texture depicts spatially repeating patterns
- Many natural phenomena are textures

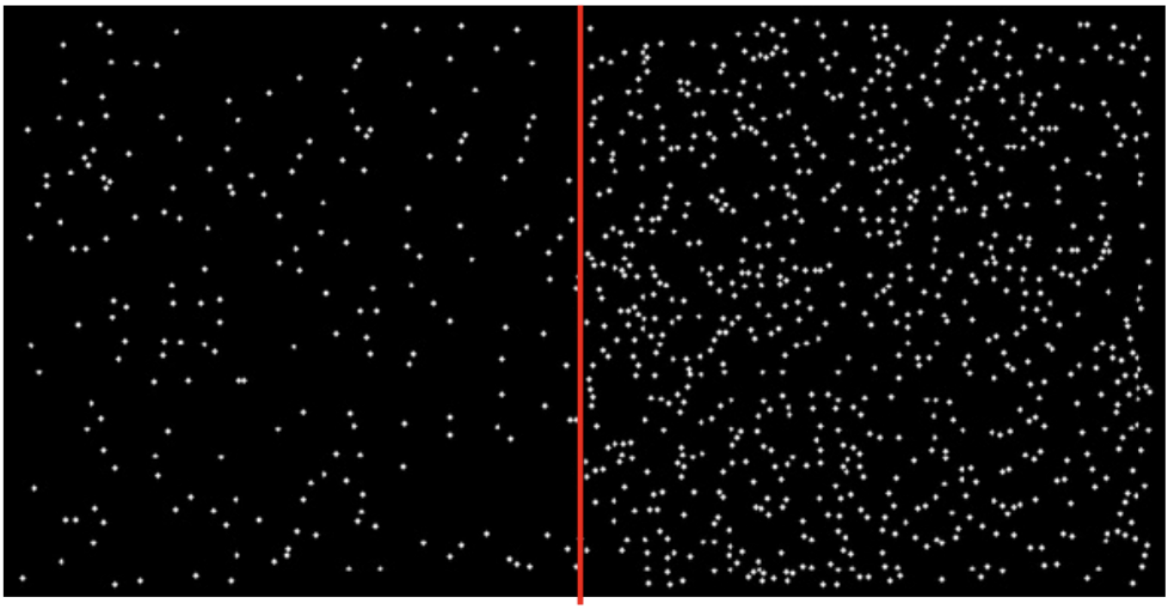
Texture Analysis

- Compare textures and encode if they're made of the same "stuff"

Bela Jules's:
Pre Attentive vs Attentive Vision

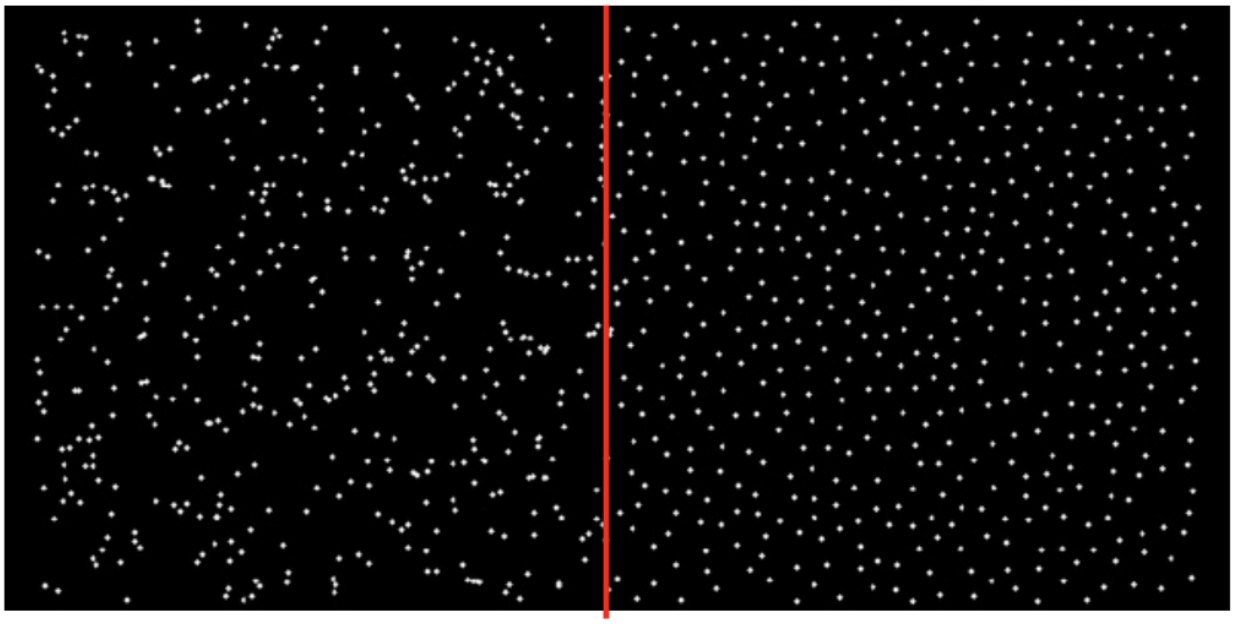
1. Preattentive vision
    a. Parallel, instantaneous, without scrutiny
    b. Independent of the number of patterns,covering. a large visual field
2. Attentive vision
    a. Serial search by focal attention

1st Order Statistics



5% white 20% white

2nd Order Statistics

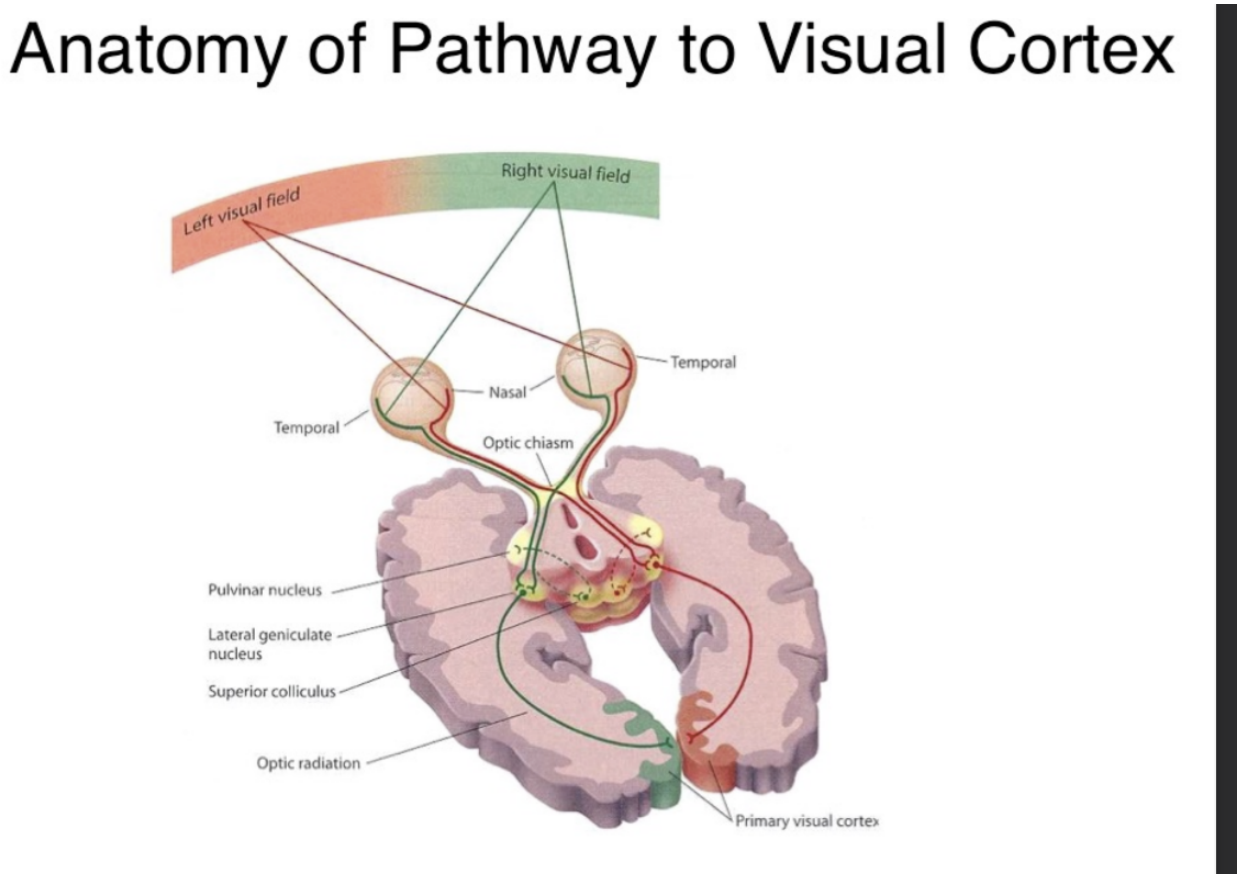


10% white

What's the statistical unit (text on) in real images
High frequency:

- on center off sourcing
- High pass filter

Natural Gaussian blur
Anatomy of pathway to visual cortex



Line detectors in cortical receptive fields
How can we represent texture?

- Can be thought of as an single "orientation"
- Image as frequency of different lines
- Image to thousand vector

Texton histogram matching

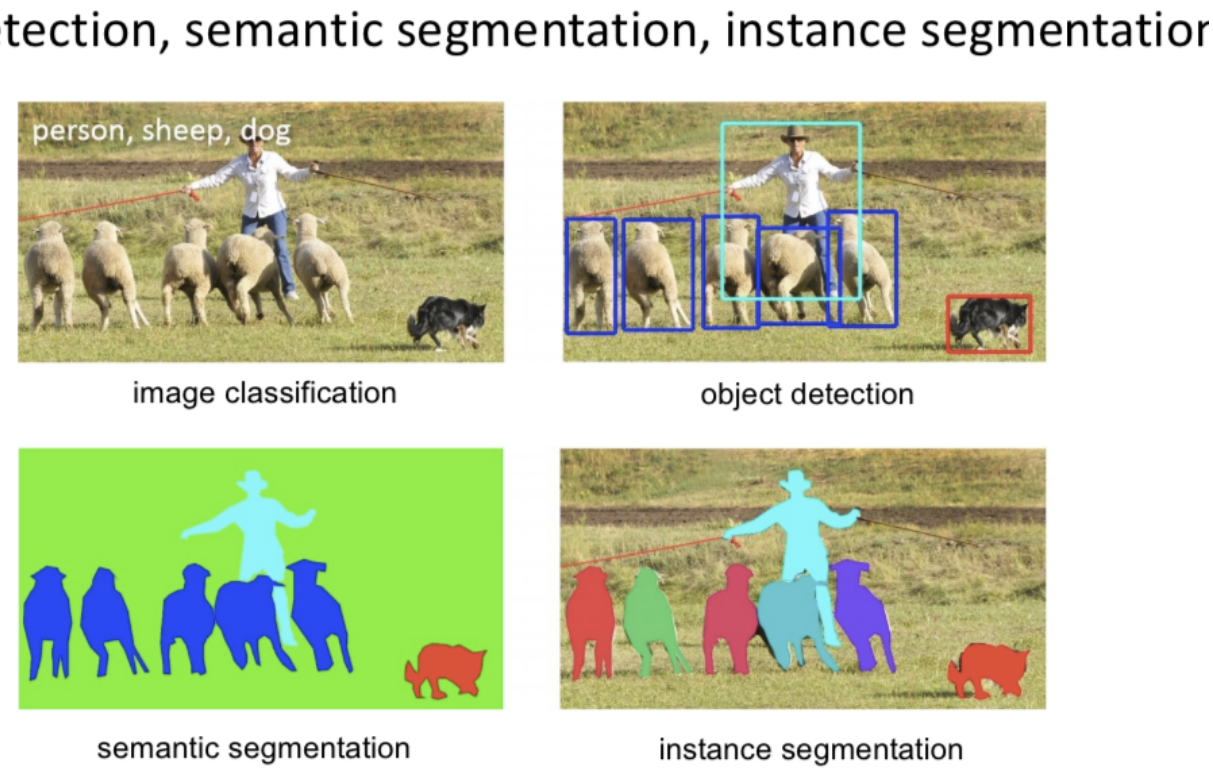
- Nearest neighbors
- Simple algorithm can match human classification at 50ms, given more time humans can classify better
- Wavelet-like receptive fields emerge from a network that learns sparse codes for natural images

Feature detection

Week 10: Lecture 17 Feature Learning with Neural Networks (10/25)

To learn well, we need objective

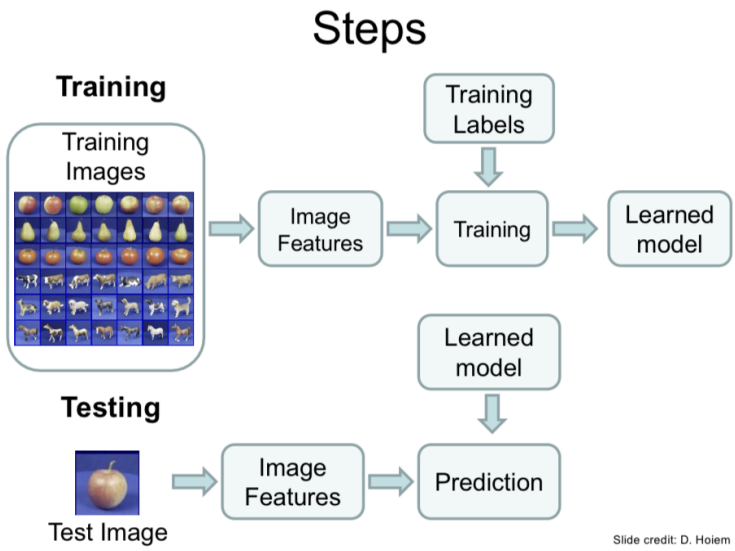
- image classification, object detection
- Key point detection



-

Statistical learning framework

- apply a prediction function to a feature representation of the image to get the desired output
- training set: given a training set of labeled examples
- Test set: apply f to never before seen test example and output predicted value
- Validation set: same as test, held out to tun hyper parameters,



- Classifiers: nearest neighbor

    - No training required
    - Find nearest neighbor and classify according to point
- CLassifiers: binary classification
    - Find a linear function to separate the classes

Linear classifiers: perceptron

- can incorporate bias as component of the weight vector by always including a feature with value set to 1

Objective Loss functions

- find weights w to minimize the prediction loss between true and estimated labels of training examples
- Possible losses for binary problems
    - Quadratic loss l(x, y;w) = f(x) - y)^2
    - Log likelihood : l(x, y; w) = -log Pw(y|x)

Perceptron training algorithm

- Initialize weights **w** randomly
- Cycle through training examples in multiple passes (*epochs*)
- For each training example **x** with label y:
    - Classify with current weights:

$$y' = \text{sgn}(\mathbf{w} \cdot \mathbf{x})$$

    - If classified incorrectly, update weights:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\left(y - y'\right)\mathbf{x}$$

(α is a positive *learning rate* that decays over time)

-
- Linear classifiers vs visual intuition
- Ship or dog
- Approximating With biological neural networks

When linear not enough

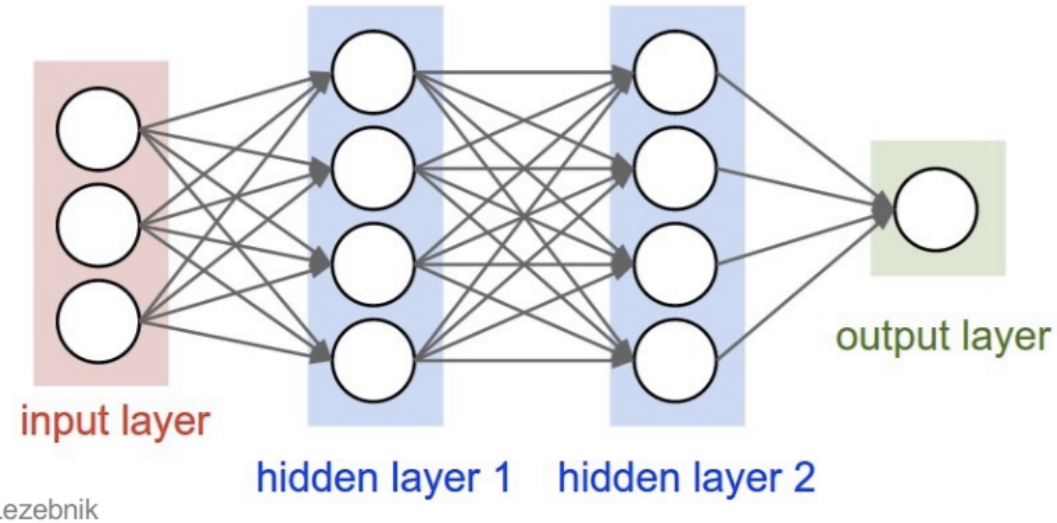
- Feature transformers

“Deep” recognition pipeline

- Learn a *feature hierarchy* from pixels to classifier
- Each layer extracts features from the output of previous layer
- Train all layers jointly

Multi-layer perceptrons

- To make nonlinear classifiers out of perceptrons, build a multi-layer neural network!
  - This requires each perceptron to have a nonlinearity



urtesy of Lana Lezebnik

Training of multi-layer networks

- Find network weights to minimize the prediction loss between true and estimated labels of training examples:

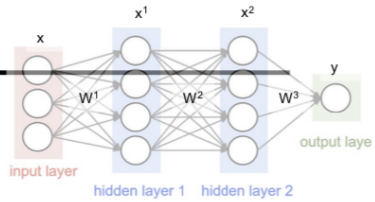
$$E(\mathbf{w}) = \sum_i l(\mathbf{x}_i, y_i; \mathbf{w})$$

- Update weights by **gradient descent:** $\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$

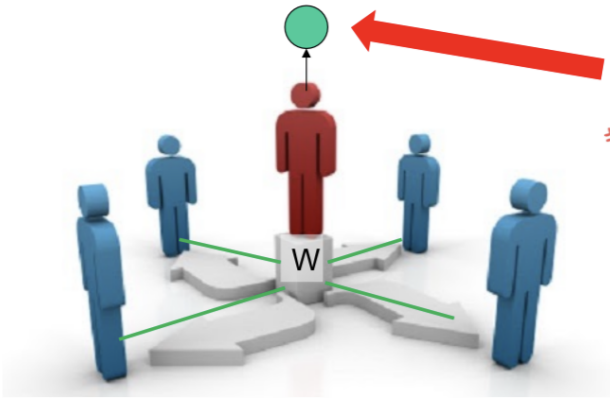
- **Back-propagation:** gradients are computed in the direction from output to input layers and combined using chain rule
- **Stochastic gradient descent:** compute the weight update w.r.t. one training example (or a small batch of examples) at a time, cycle through training examples in random order in multiple epochs

sy of Lana Lezebnik

Back prop intuition



1. Every single layer is like a team of neurons (x), that a manager (W) is listening to decide outputs (y)

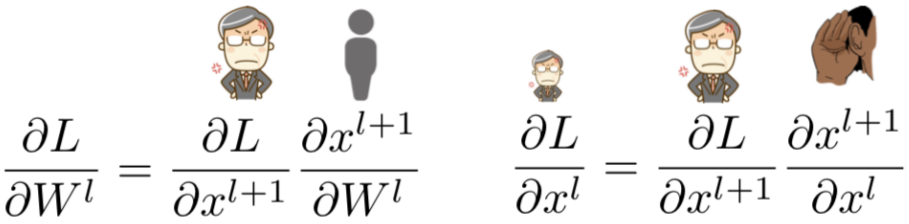


2. Then you get a feedback from higher ups of what you should've done

3. You adjust your confidence in each of your neurons,
4. Then pass on what they should've said

The point is

Each layer needs to adjust how it listened to each neuron (W) and compute gradients wrt its input and pass it down.



$$\frac{\partial L}{\partial W^l} = \frac{\partial L}{\partial x^{l+1}} \frac{\partial x^{l+1}}{\partial W^l}$$

Updates: Higher up orders multiplied by how much the manager listened to each neuron

$$\frac{\partial L}{\partial x^l} = \frac{\partial L}{\partial x^{l+1}} \frac{\partial x^{l+1}}{\partial x^l}$$

Pass down: Higher up orders, weighted by how much the manager listened to each neuron

Convolutional Layer

- share the same parameters across different locations, assuming input is stationary
- Convolutions with learned kernels

Convolutional Layer

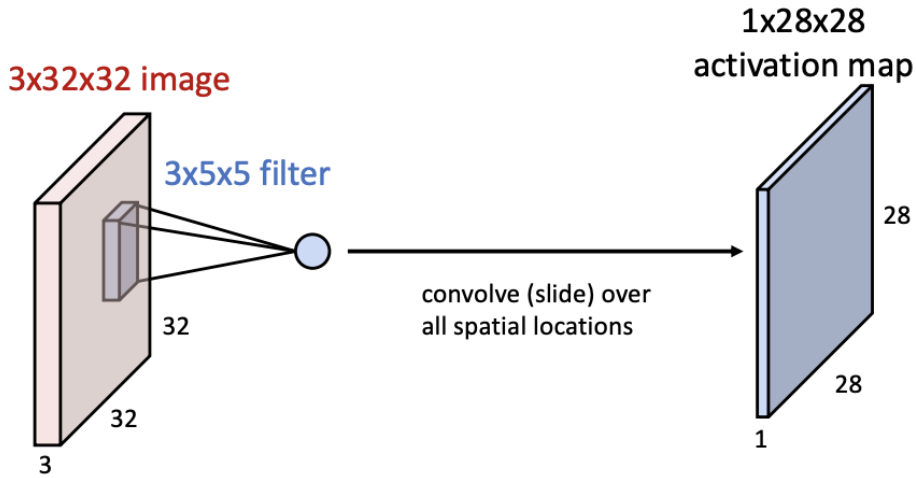
- learn multiple filters
-

Week 10: Lecture 18 Convolutional Neural Nets (10/27)

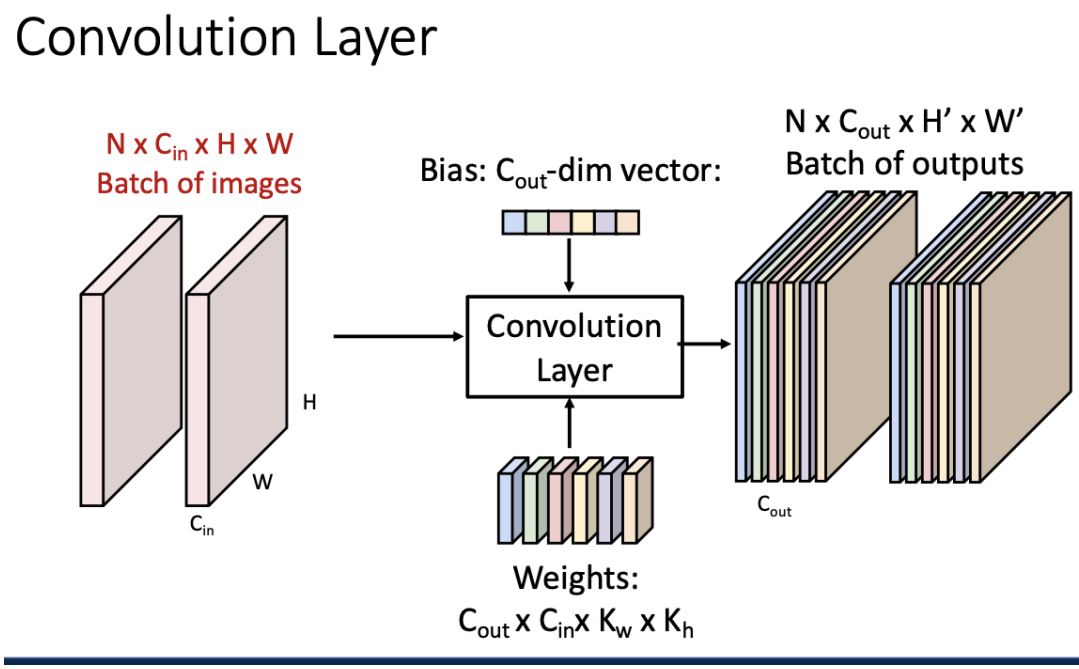
Convolution Layer

Convolution Layer

One neuron, that looks at 5x5 region and outputs a sheet of activation map



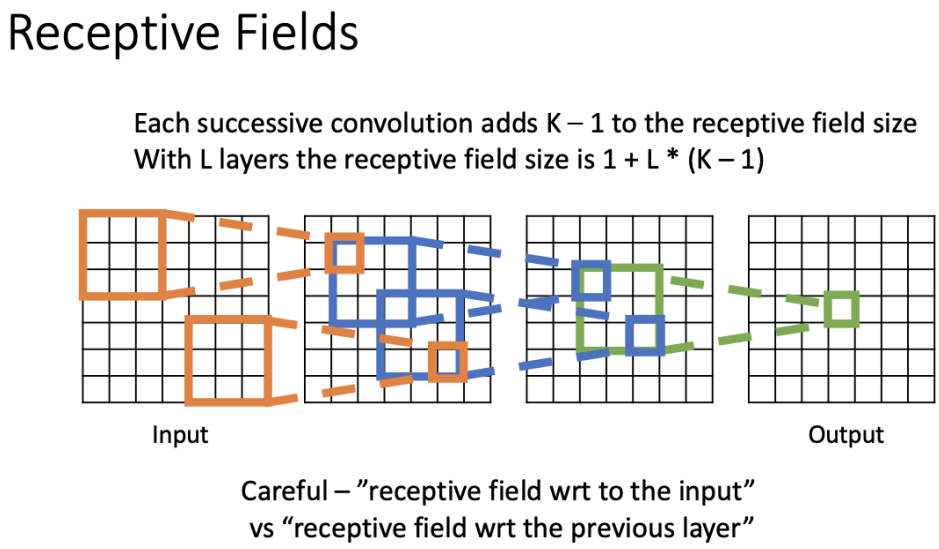
- Add a second green filter



Can do localized & shared matrix multiplication

Receptive Fields

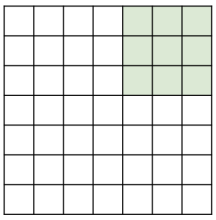
- For convolution with kernel size K, each element in the output depends on a KxK receptive field in the input



Strided Convolution

- Stride: 2 can skip some of them to have a smaller output, downsampling the image by 2

Strided Convolution



Input: 7x7
Filter: 3x3
Stride: 2

Output: 3x3

In general:
Input: W
Filter: K
Padding: P
Stride: S
Output: (W – K + 2P) / S + 1

Need to normalize our data to prevent any different in learning weights

- Sometimes pixel range is 0,255 sometimes 0,1
- Network activations will be completely different

How to normalize

- $\hat{x} = \frac{x - E[x]}{\sqrt{Var[x]}}$

Max pooling/Average pooling

- From a smaller sample, take the average or the max to get a invariance to small spatial shifts

- Q:
- C: corresponding

How to Normalize

- Batch norma: mean and var from each batch
- Layernrom InstantceNorm do not need this

Bottom line: CNN workflow

- Convert HxW image into a F-dimensional vector
    - It is probably a certain class…
    - X,y coordinate of each keypoint

1. Get your dataset
2. Design your CNN architecture
3. Train + update weights with pytorch/tf

Sanity Check

- Make sure you can memorize a single datapoint: (x, y)
- Train your CNN on only 1 image
- Must be able to get a very low training loss
    - If you can't do this, it means something is wrong in your loss function, label, network/pytorch setup

Pooling Summary

Input: C x H x W

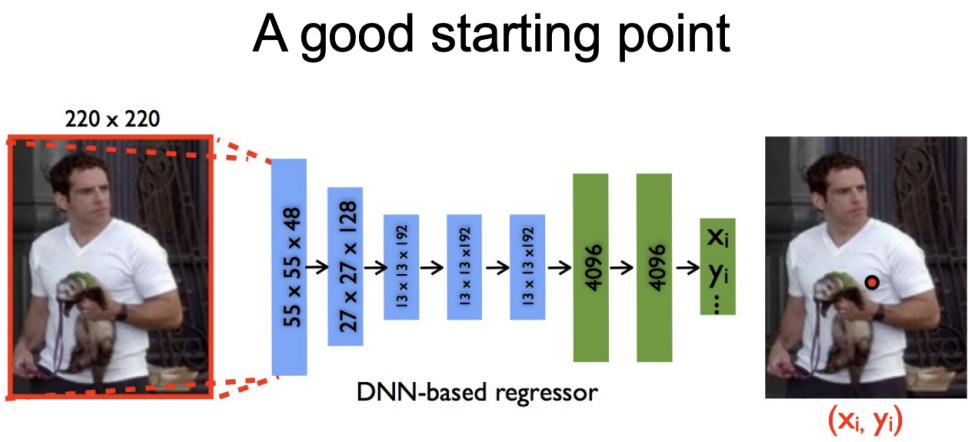
Hyperparameters:
- Kernel size: K
- Stride: S
- Pooling function (max, avg)

Common settings:
max, K = 2, S = 2
max, K = 3, S = 2 (AlexNet)

Output: C x H' x W' where
- H' = (H – K) / S + 1
- W' = (W – K) / S + 1

Learnable parameters: None!

- T should overfit to a single data fairly quickly, if it takes many iterations, that's bad news



DeepPose: Human Pose Estimation via Deep Neural Networks
[Toshev and Szegedy 2014]

Week 10 Discussion: PyTorch demo(10/28)

Pytorch can use

Einops

- can use to reshape data

Import Torch.nn as nn

Parameters are tensors that can be trained

- forward pass

Linear layer nn.Linear

Conv layer: nn.Conv2d

- conv2d in_channels, out_channels, kernel_size, padding
- CrossEntropyLoss, MSELoss, L1 Loss,
- Regression:
- Epoch is run through dataset

Logging W&B

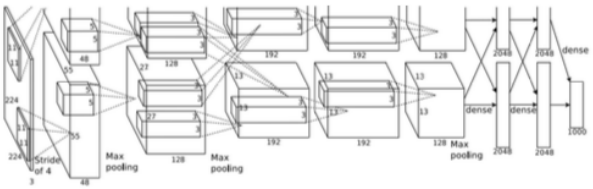
- Import wandb
- How much using memory, performance, website
- Log evaluation metrics and save networks
- Save it directly to wandb
- Tqdm gives you progress bar
- Tqdm.tqdm

Week 11: Lecture 19 Convolutional Neural Nets II (11/1)

Case Study: AlexNet

- Input: 227x277x3 images
- First 11x11 filters applied atstride 4: (CONV1): 9
- Output volume size after conv1 (N - K) / S + 1 = (227-11) / 4 + 1 = 55
    - 27x27x96

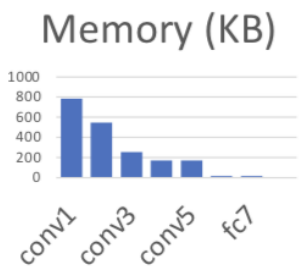
AlexNet



Q: Which part of the network incurs

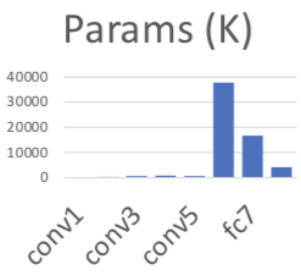
high memory usage?

Most of the **memory usage** is in the early convolution layers



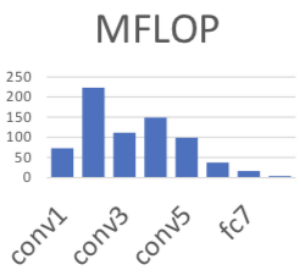
Large # of parameters?

Nearly all **parameters** are in the fully-connected layers



high FLOPs?

Most **floating-point ops** occur in the convolution layers



-

- VGGNet [Simonyan and Zisserman 2015]
    - Input: 224x224x3
    - Simplified design rules
        - Kernel size 3x3
        - Always ReLu
        - All max pool are 2x2, stride 2
        - After pool, double the # of channels

Naively adding more layers

- != better performance

Naively adding more layers != better performance

Q: is this over fitting?

No! (hypothesis) it's a matter of optimization

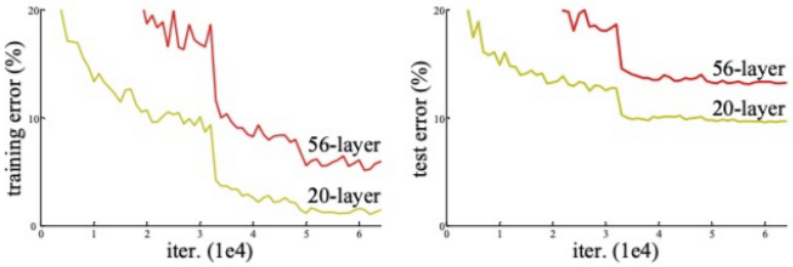


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

CVPR 2016]

-
- Marking extra layers learn the identities so that unit is included

ResNet Solution

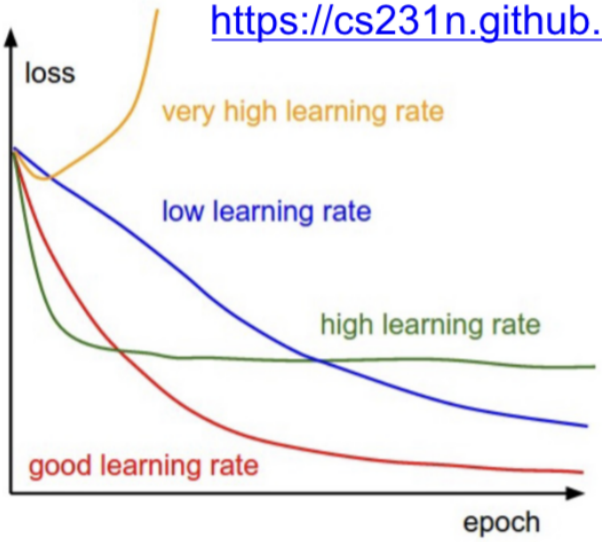
- Input 224, 224x3
- F(x (Wi)) + x
- Bottleneck design
    - Reduce by 1x1, conv by 3x3, restore by 1x1'

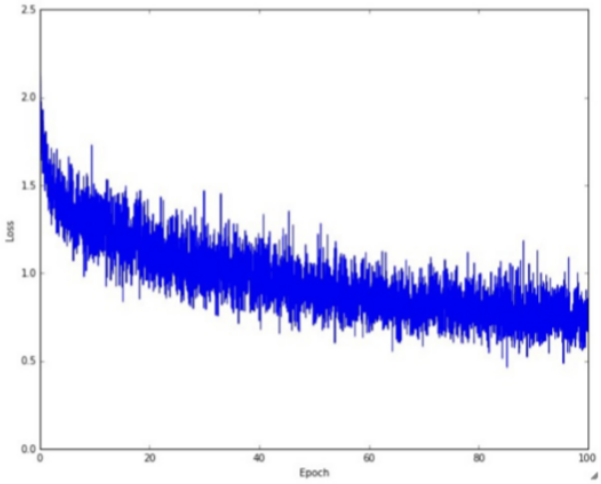
IMAGENETreduced image classification table

- gradual but want to low enough e without overfitting to learn rates,

Training diagnosis

https://cs231n.github.io/neural-networks-3/





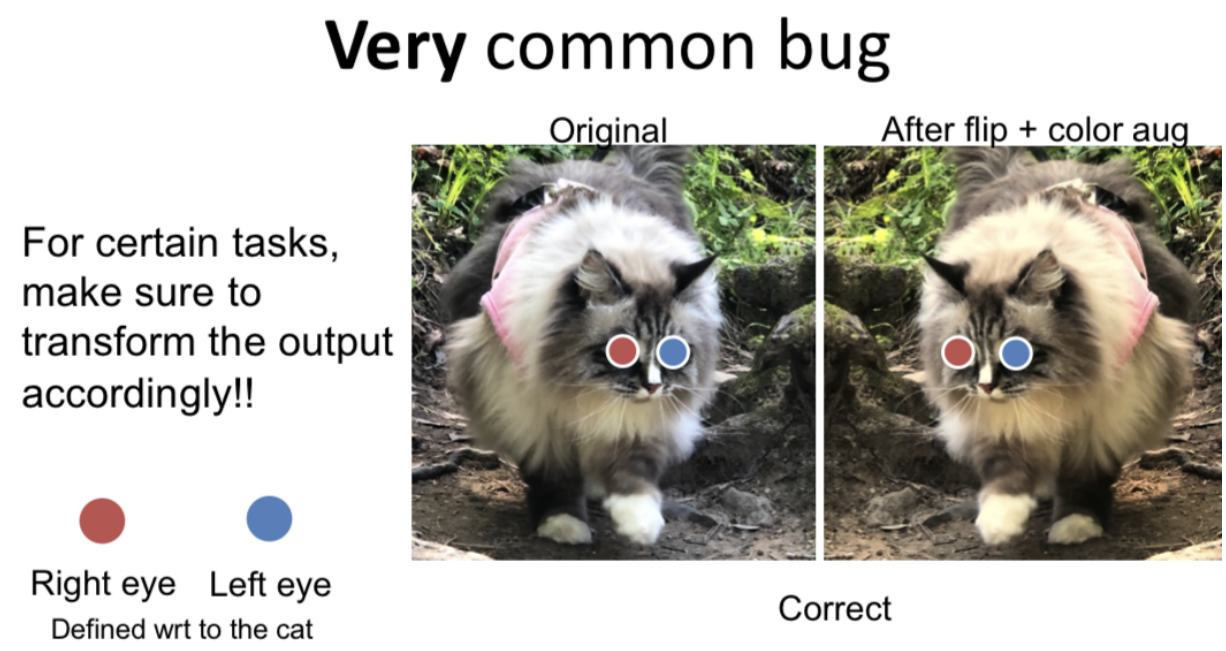
-
-
- What to do if your data is not enough i enough

Data Augmentation

- Apply transformations on the fly toincraing training Eva
- Mix multiple transformations, don
- T change the meaning out output

COmmon bug

- for flipping, right and left eye transformation of output accordingly
- Flip and transformation

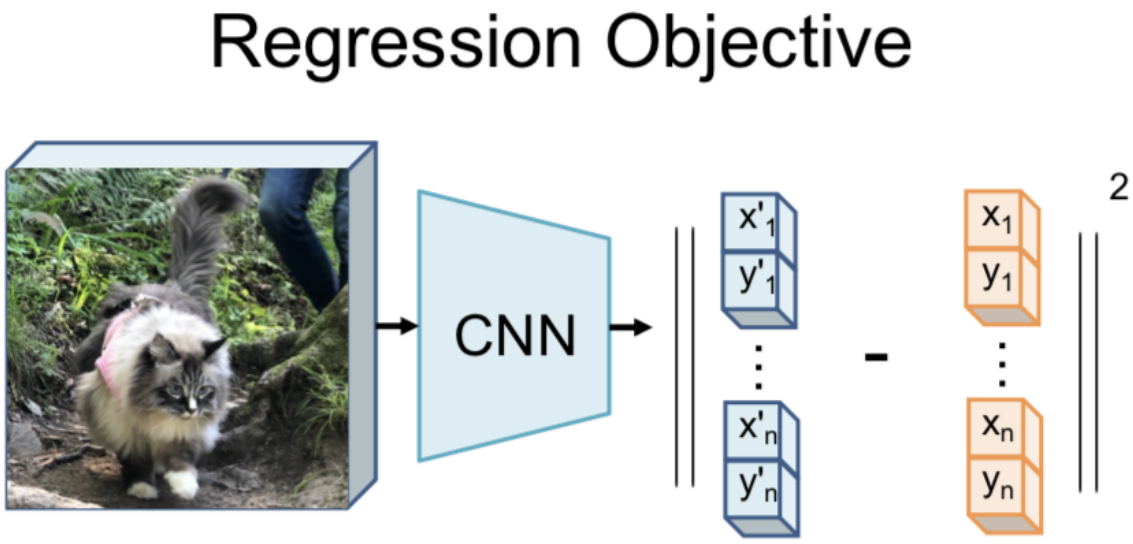


-

Transfer Learning with CNNs

1. Train on Imagenet
2. If small dataset: fix all weights, retrain classifier
3. Median dataset, “fine tune” instead, use a old weights as initialization train only some of the high ear layers

Project 5 starting point

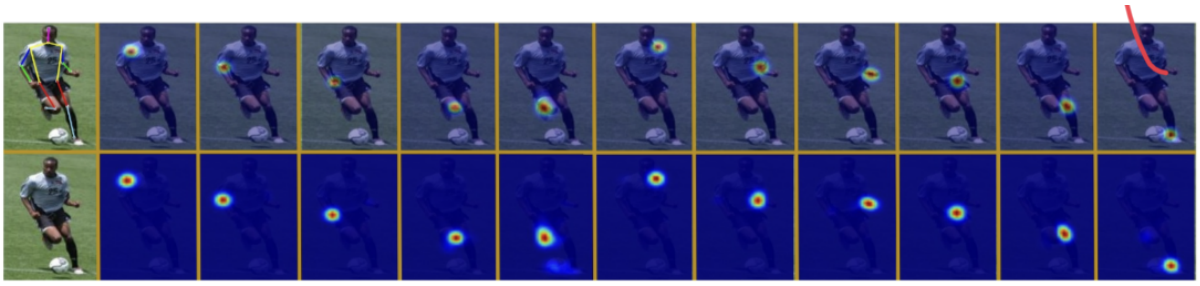


This is what you should do before bells & whistles.
But this is not what the SoTA models do in practice

-

Belief/Confidence Map formulation

- for K key points, train model to predict K many sheets (how) of scores of how likely the pixel is the key point



[Thompson et al. NeurIPS 2014, CVPR 2015, Convolutional Pose Machine, Wei et al. CVPR 2016, (figure credit: Ning et al TMM 2017)…]

For K keypoints, train model to predict K many sheets (h x w) of scores of how likely the pixel is k-th keypoint

-

Idea 1: dense prediction by sliding window

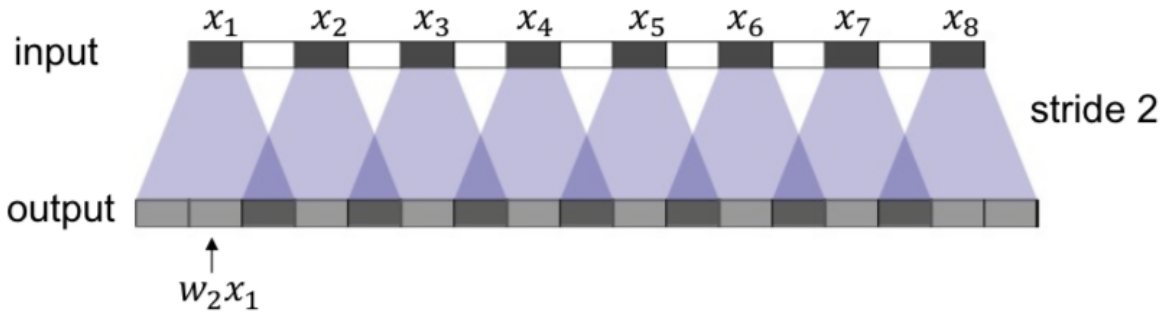
- very inefficient for overlapping patches

Instead, make them hole network convolutional

- last few fully connected network as convolutions with kernels that cover the entire image
- Make predictions for all pixels at once

Fully convolutional networks

- first downsample, then upsample



- can give you a checkerboard like artifacts

Simpler, effective solution

- Upsample, followed by a regular convolution

How to upsample

- Bed nails, nearest Neighbor, bilinear interpolation

Application to pose detection :predict heat maps

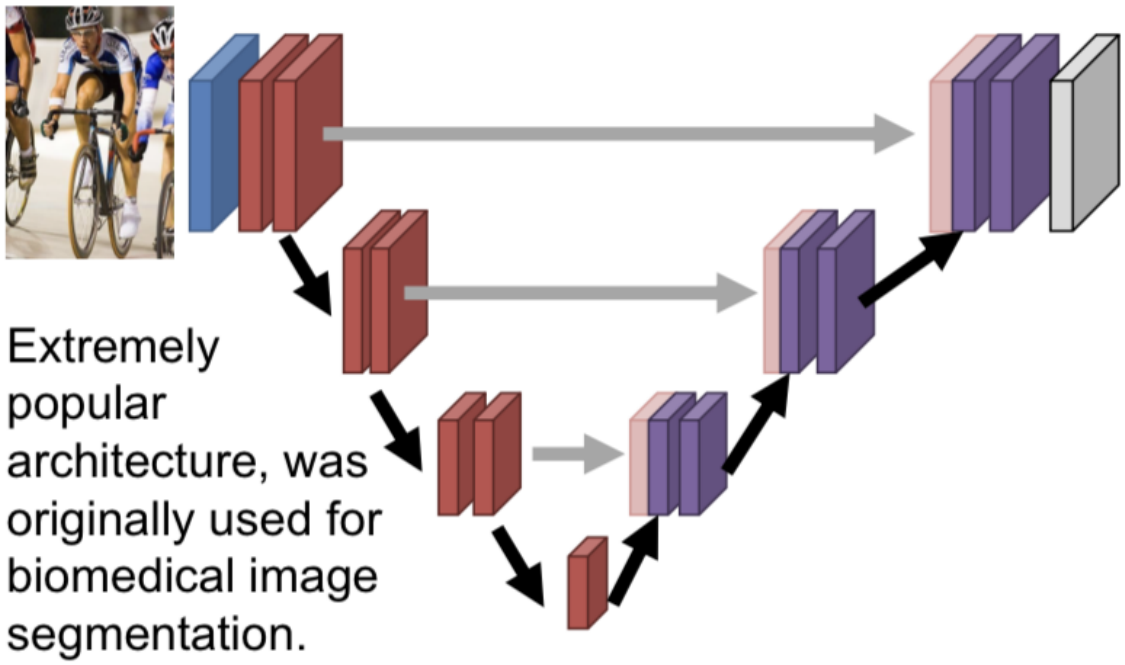
- Target: how likely it is in this specific pixel
- Training loss:
- L2 loss on the target heatmap

OpenPose: convolutional pose machines

Week 11: Lecture 20 Convolutional Neural Nets II (11/3)

A lot of high frequencies got sampled out

U-Net

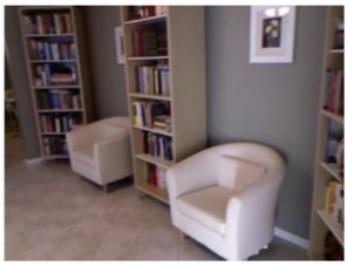


Extremely popular architecture, was originally used for biomedical image segmentation.

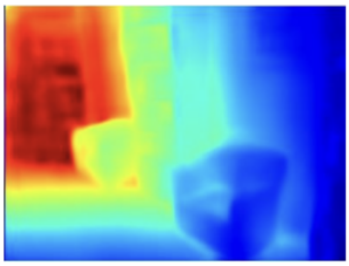
e.g. Depth Prediction

Instead: give label of depthmap, train network to do regression (e.g., $\|z_i - \hat{z}_i\|$ where z_i is the ground-truth and $\hat{z}_i$ the prediction of the network at pixel i).

Input HxWx3 RGB Image



Output HxWx1 Depth Image



True HxWx1 Depth Image

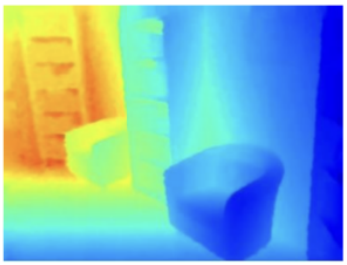


Image Analysis

- drain normal network to minimize ground -truth and prediction at pixel

- Not sure which one, may screw up and take the one that is in between
- Bimodal distribution
- L2 assumes single mode

Better Loss Function

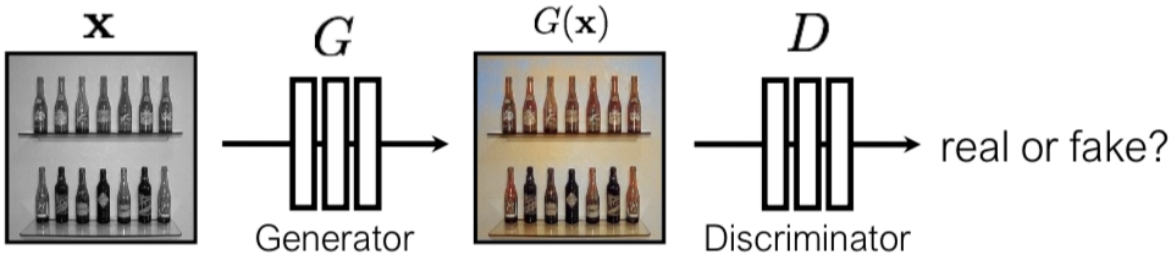
- theta* = arg min l(F(X), Y)
- L2(Y^, Y) = ½ sum || Yh,w - Yh,w|| ^2 L
- Regression becomes classification loss
- Colorful ness is willing to take a chance

Designing loss functions

- have a universal loss

Generative Adversarial Network

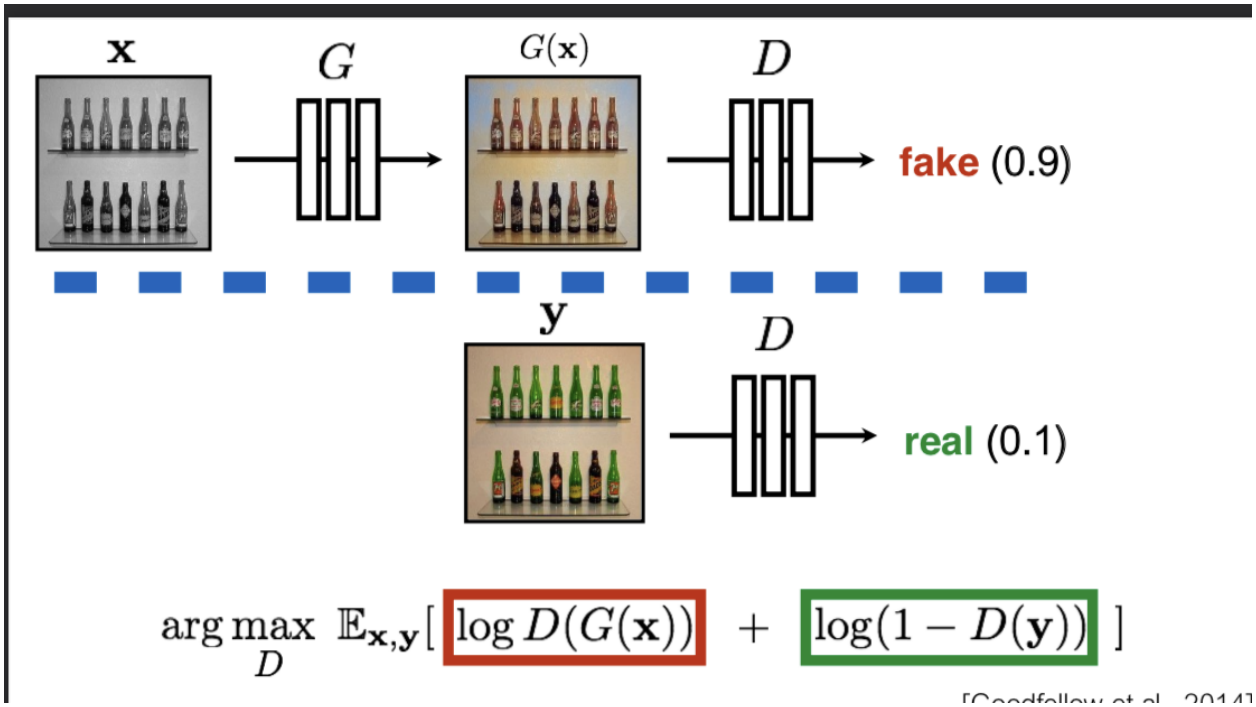
- another classifier to figure out how if it is a real image or a fake image
- Optimize task so that GAN does not know real or fake



G tries to synthesize fake images that fool **D**

D tries to identify the fakes

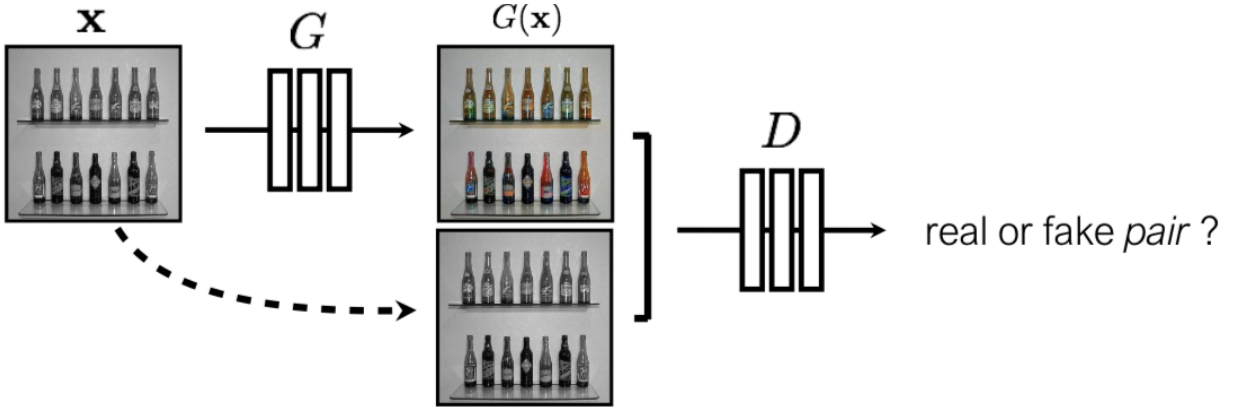
-



G's perspective: D is a loss function

Rather than being hand-designed, it is learned

Given the input and output pair, the GAN finds out

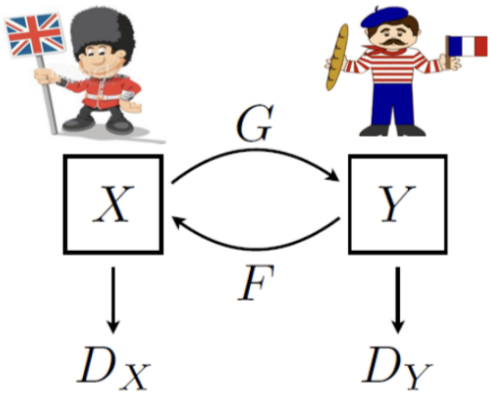


$$\arg\min_G \max_D \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$$

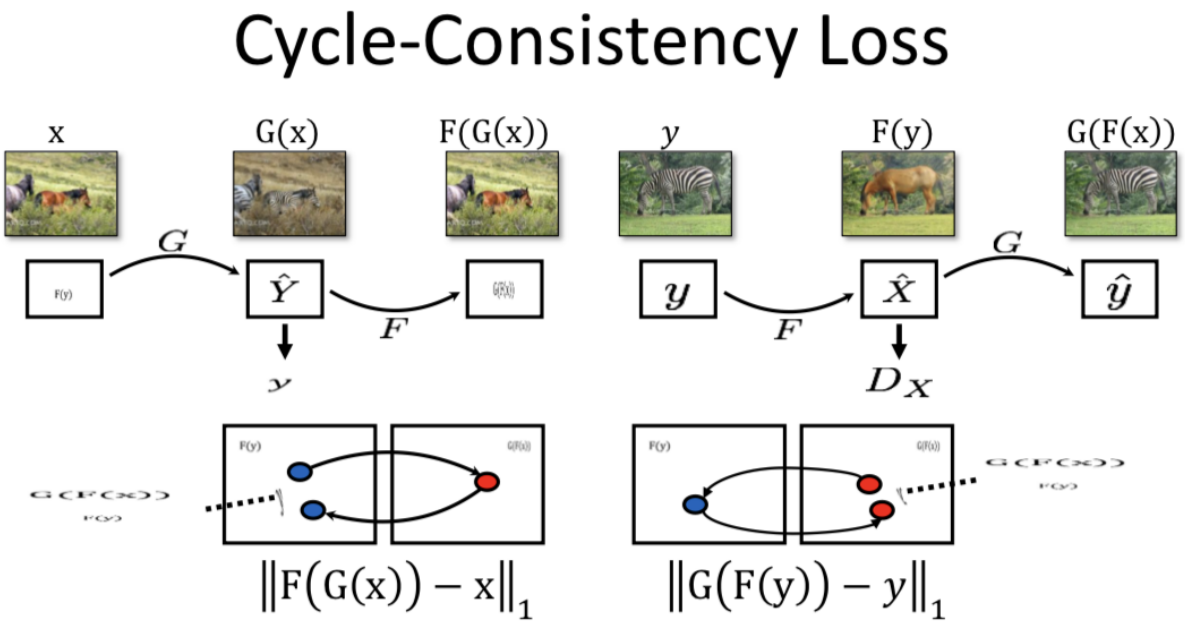
GAN algorithm never converges

CycleGAN

- cycle there and back until you get the same thing you start out and end with when translation g



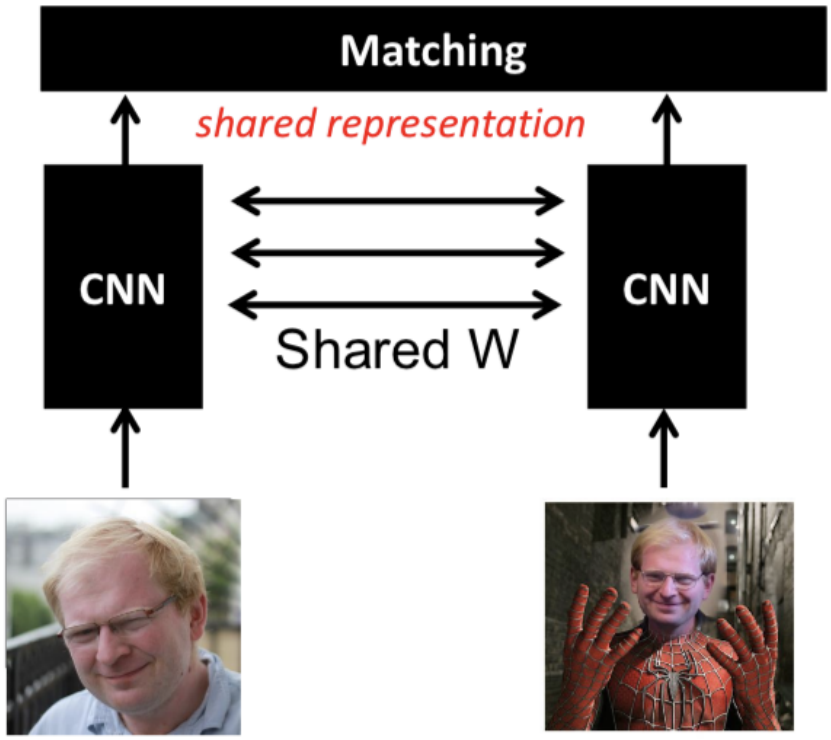
-



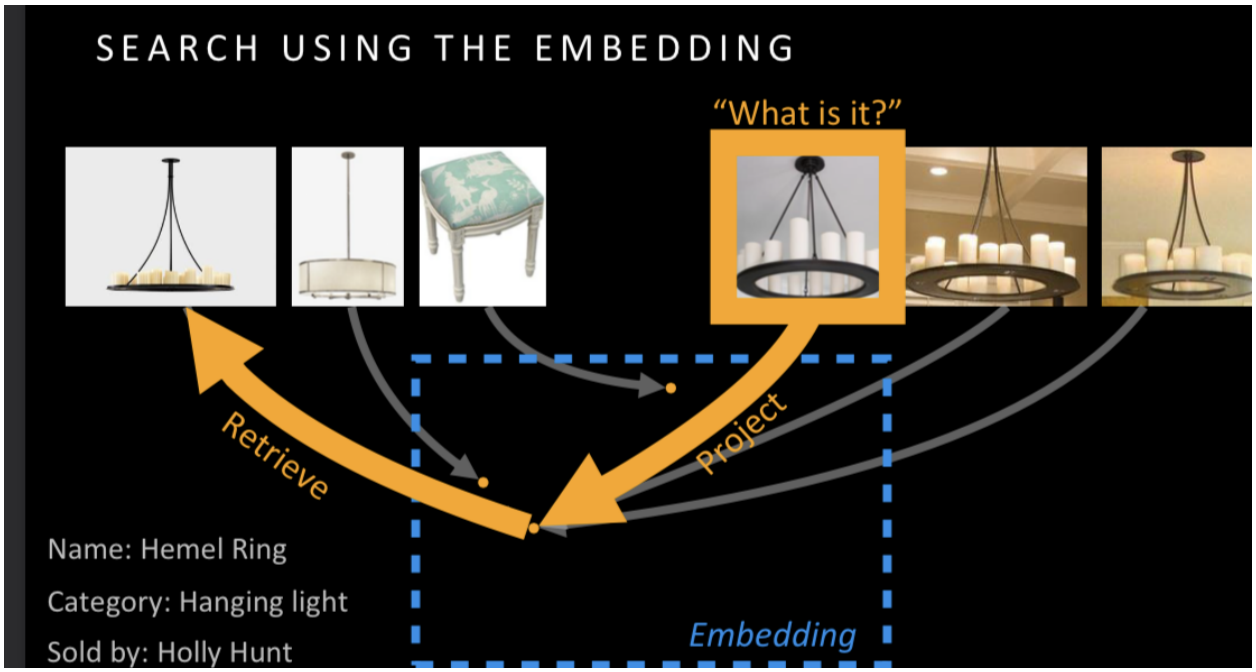
-

Learning an Embedding

- embedding: last layer before the fully connected
- Higher level representation of data
- Encode given image into vector
- Nearby vectors in embedding space



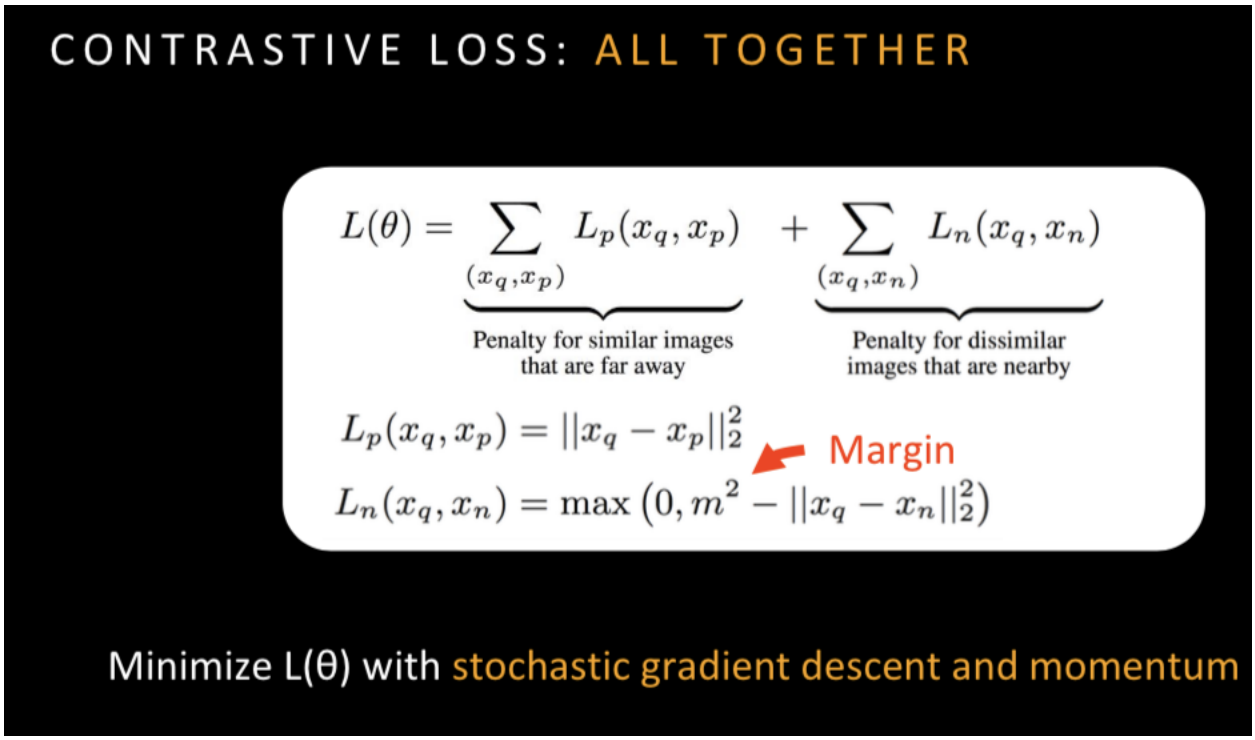
-
- Similar ones need to end up in similar embedding space



-
- Contrastive loss:

- Don't come too close: margin m don't be within this margin

-

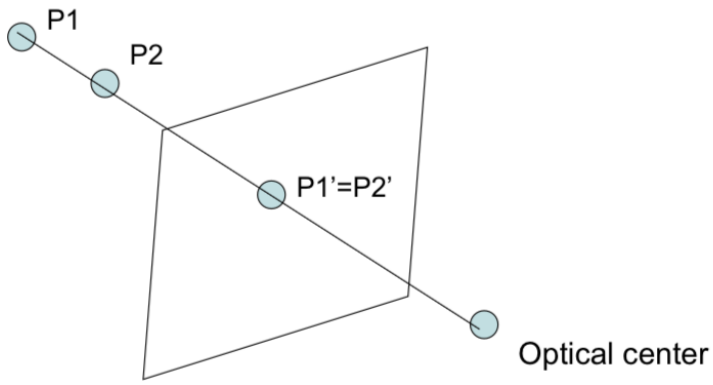


-

Week 12: Lecture 21 3D Computer Vision (11/8)

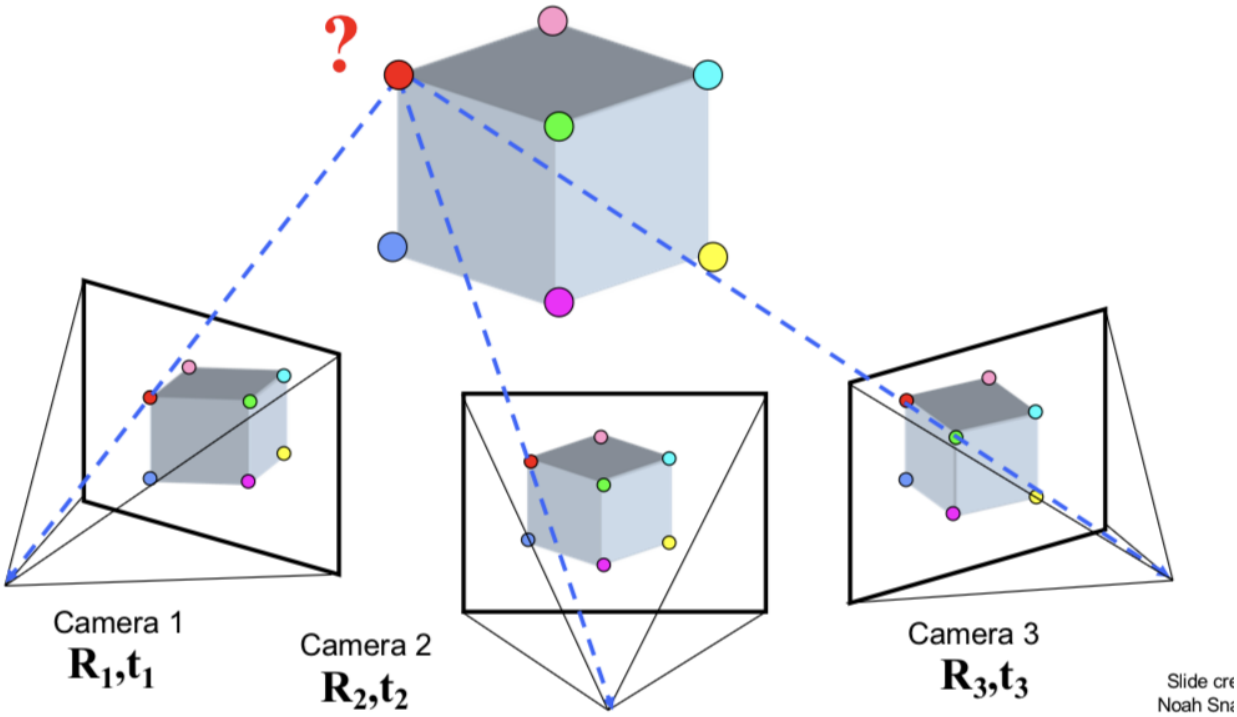
Why multiple views

- structure and depth are inherently ambiguous from single view
  - Structure and depth are inherently ambiguous from single views.



-

Multi view geometry problems

- Given projects in of the same 3D point in two or more images, compute the 3d coordinates of that points
    - **Structure:** Given projections of the same 3D point in two or more images, compute the 3D coordinates of that point
- 

Motion, Structure, or correspondence

Two camera system

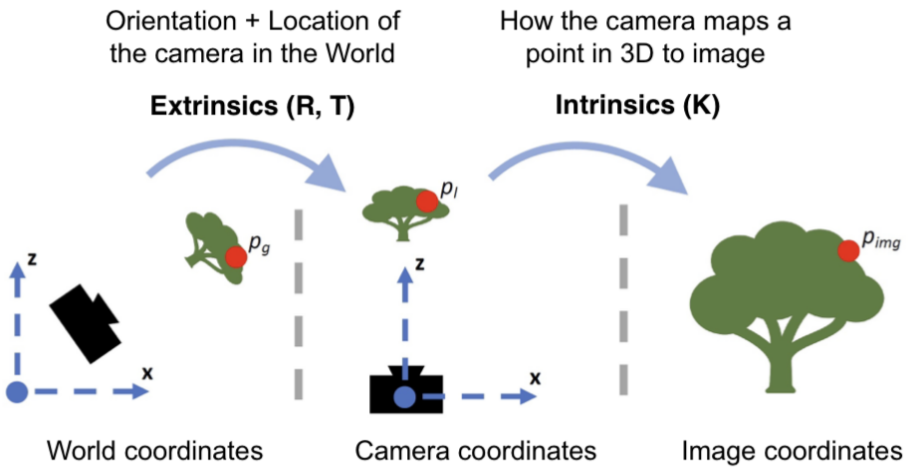
- stereo, calibrating the camera, estimating depth from correspondences

Cameras in the world coordinate frames

- Pixels to 3d location
- Position of the camera with respect to the world (extrinsic)
- How the camera maps a point in the world to the image (intrinsic)

Problem setup: Three important coordinate frames

1. World coordinates
2. Camera coordinates
3. Image coordinates



4.

Camera: Specifics

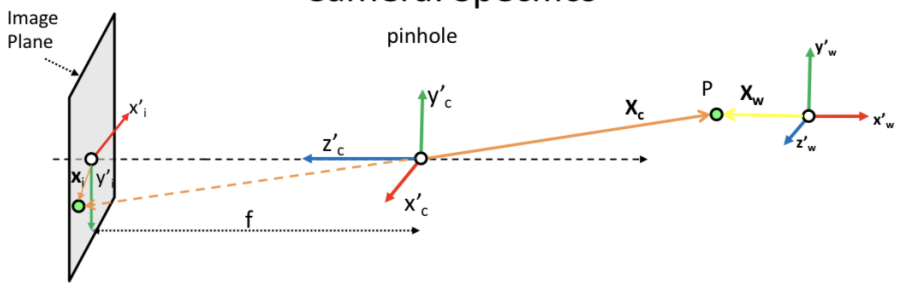
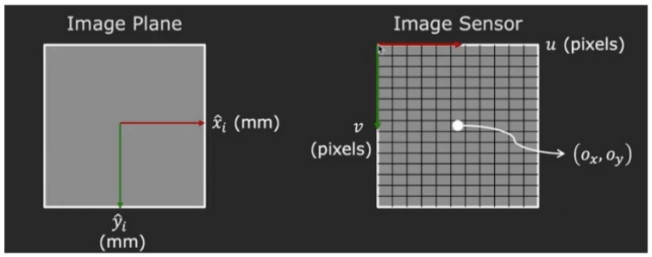


Image Coordinates $\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$ ⟸ Perspective Projection ⟸ Camera Coordinates $\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$ ⟸ Coordinate Transformation ⟸ World Coordinates $\mathbf{X}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$

$$x_i = f\frac{x_c}{z_c}$$

Image Plane to Image Sensor Mapping

Homogeneous Coordinates



1. Account for pixel density (pixel/mm) & aspect ratio by scalars: $[m_x, m_y]$

$$m_x x_i, m_y y_i$$

2. Usually the top left corner of the image is the origin. But in the image plane, where the optical axis pierces the plane is the origin, need to shift by: (o_x, o_y)

$$u_i = \alpha_x x_i + o_x = \alpha_x f\frac{x_c}{z_c} + o_x$$

where $[f_x, f_y] = [m_x f, m_y f]$

Pixel Coordinates:

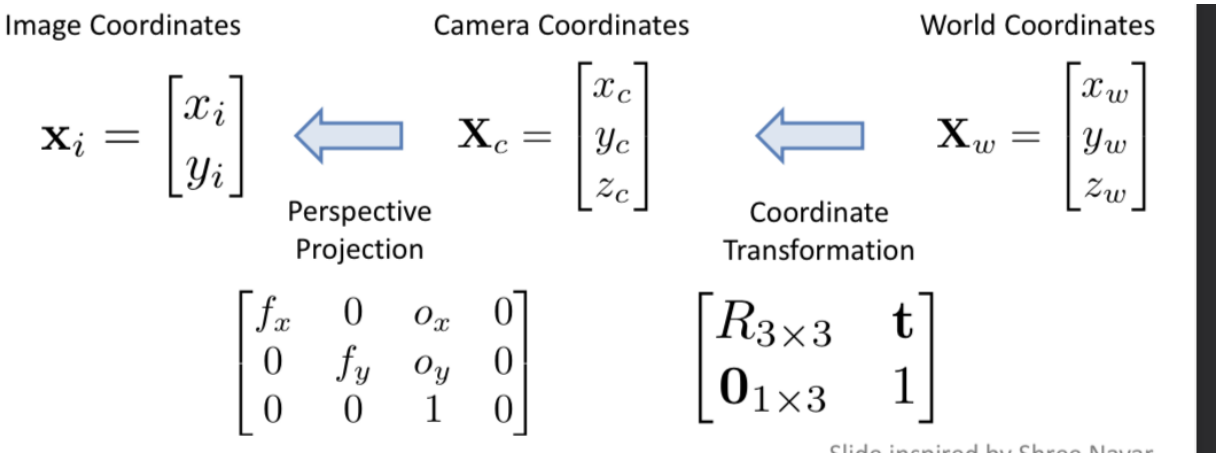
$$u_i = \boxed{f_x}\frac{x_c}{z_c} + \boxed{o_x} \quad v_i = \boxed{f_y}\frac{y_c}{z_c} + \boxed{o_y}$$

Perspective projection + Transformation to Pixel Coordinates:

$$u_i = f_x \frac{x_c}{z_c} + o_x \quad v_i = f_y \frac{y_c}{z_c} + o_y$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Intrinsic Matrix



3 rotation 3, translations

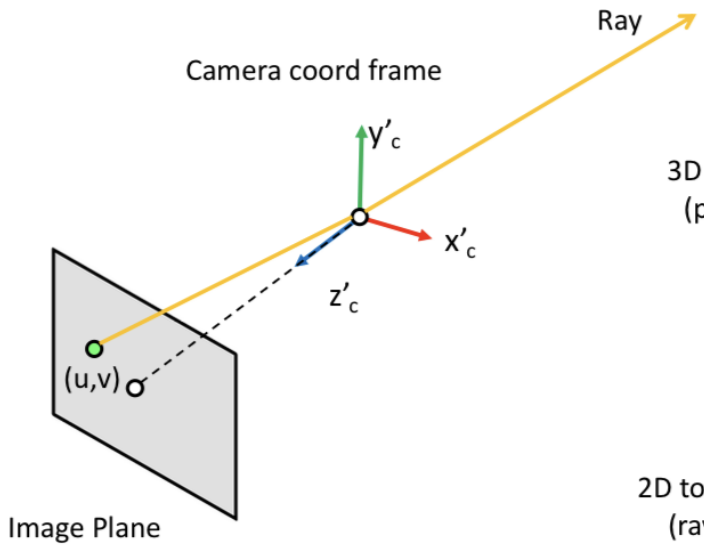
Fundamental Scale Ambiguity

- reconstruction only possible up to a global scale
- Scaling the world & camera doesn't change the projection
- Unless you know something metric about scene

Step 1: with a known 3d object

1. Take a picture of an object with known 3d geometry
2. Identify correspondences

QR Decomp to get K[R | T]



3D to 2D: (point)

$$u = f_x \frac{x_c}{z_c} + o_x$$
$$v = f_y \frac{y_c}{z_c} + o_y$$

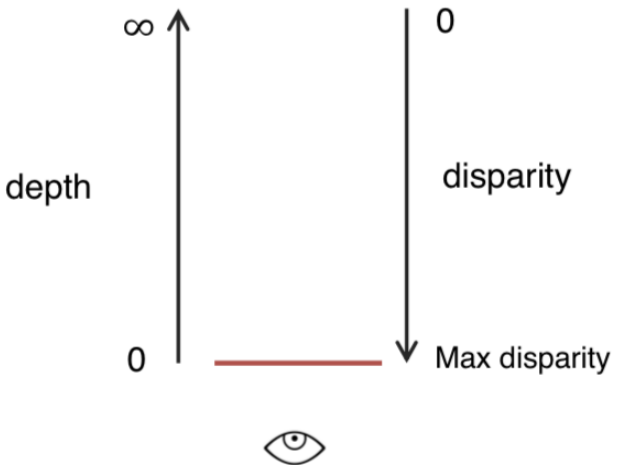
2D to 3D: (ray) Back projection

$$x = \frac{z}{f_x}(u - o_x)$$
$$y = \frac{z}{f_y}(v - o_y)$$
$$z > 0$$

Simple Stereo Setup

- Assume parallel optical axes
- Two cameras are calibrated
- Solving for depth in simple stereo
  - B - (d1 + d2)

Depth is inversely proportional to disparity



$$z = \frac{fB}{u_l - u_r}$$

disparity

$$z \propto \frac{1}{u_l - u_r} = \frac{1}{d}$$

For every epipolar line,

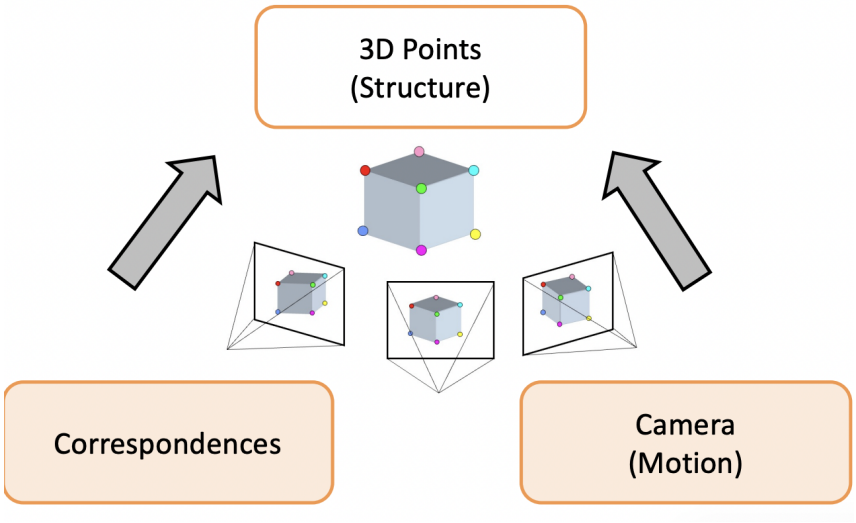
- For each pixel in the left image
  - Compare with every pixel on same epipolar line in right image
  - Pick pixel with i minimum match cost
- Match windows, + clearly lots of matching strategies
- Neighborhood of corresponding points are similar in intensity patterns

Correlation-based window matching

- calculate disparity
- Determine disparity using template matching

Week 12: Lecture 22 3D Computer Vision: Epipolar Geometry (11/10)

Big Picture: 3 key components in 3D

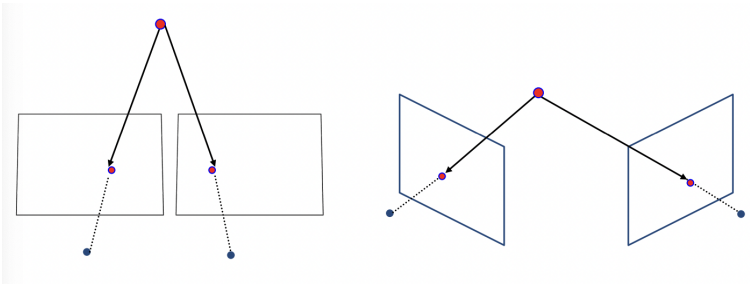


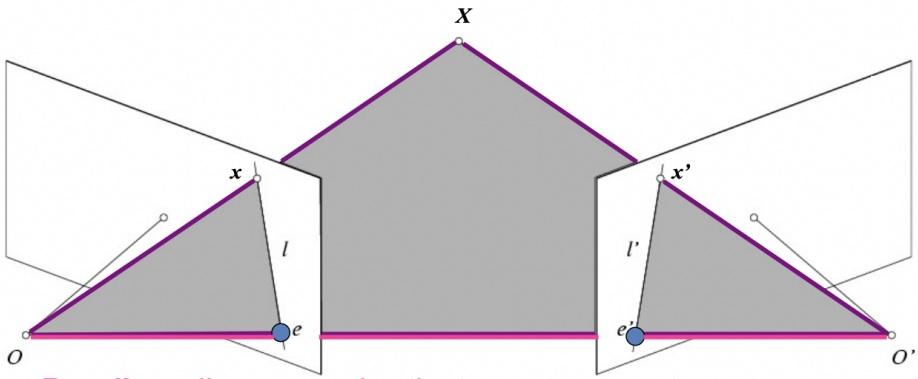
Recap

- "Calibration"
- Solve for instrisics & extrinsics
- Depth is inversely related

General case

- Assume cameras are calibrated
- Can use rectification via homography
    - Reproject image planes onto a common plane





- **Baseline** – line connecting the two camera centers
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipoles**

= intersections of baseline with image planes
= projections of the other camera center
= vanishing points of the motion direction

Epipolar lines coverage on epipolar line

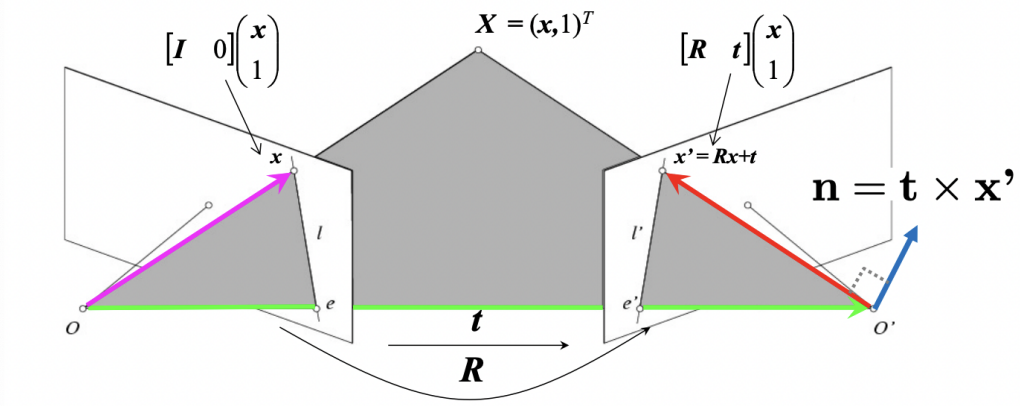
Parallel to Image Plane

- epipoles infinitely far away, epipoles

How to Compute the Epipolar Line

Normalized Image Coordinates

- Know the intrinsics K
- Make it into a 3d point at the image plan [u, v, 1]
- Normalized image coordinates, set of points with K = Identity ;
  - $x_{narm} = K^{-1} x_{pixel} = [I\ 0]\, X$
  - $x_{norm}' = K'^{-1} x'pixel = [R\ t]\, X$
- On the epipolar plane, x, t, x′ are coplanar



The vectors x, t, and x' are coplanar

What can you say about their relationships, given $\mathbf{n} = \mathbf{t} \times \mathbf{x'}$?

$$\mathbf{x'} \cdot (\mathbf{t} \times \mathbf{x'}) = 0$$

$$\mathbf{x'} \cdot (\mathbf{t} \times (R\mathbf{x} + \mathbf{t})) = 0$$

$$\mathbf{x'} \cdot (\mathbf{t} \times R\mathbf{x} + \mathbf{t} \times \mathbf{t})) = 0$$

$\mathbf{t} \times \mathbf{t} = 0$

$$\mathbf{x'} \cdot (\mathbf{t} \times R\mathbf{x}) = 0$$

$$\boldsymbol{x}' \cdot [\boldsymbol{t} \times (\boldsymbol{R}\boldsymbol{x})] = 0 \Rightarrow \boldsymbol{x}'^{T} [\boldsymbol{t}_{\times}] \boldsymbol{R}\boldsymbol{x} = 0 \Rightarrow \boldsymbol{x}'^{T} \boldsymbol{E}\boldsymbol{x} = 0$$

$E = [t_{\times}]R$

Recall: $\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a}_{\times}]\mathbf{b}$

Essential Matrix
(Longuet-Higgins, 1981)

The vectors x, t, and x' are coplanar

Essential Matrix

Ex is the epipolar line associated with x ($I' = Ex$)

$E^T x'$ is the epipolar line associated with $x'(I = E^T x')$

- Recall that we normalized the coordinates

$$x = K^{-1}\hat{x} \quad x' = K'^{-1}\hat{x}' \qquad \hat{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

where $\hat{x}$ is the image coordinates

- But now calibration matrices ***K*** and ***K'*** of the two cameras are unknown!
- We can write the epipolar constraint in terms of *unknown* normalized coordinates:

$$x'^T E x = 0$$

$$(K'^{-1}\hat{x}')'^T E(K^{-1}\hat{x}) = 0$$

$$\hat{x}'^T \underbrace{K'^{-T} E(K^{-1}}\hat{x}) = 0$$

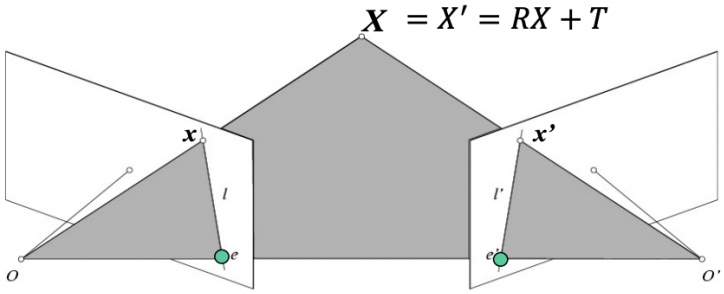
$$\hat{x}'^T F \hat{x} = 0$$

$$F = K'^{-T} E K^{-1}$$

Fundamental Matrix
(Faugeras and Luong, 1992)

Essential matrix; when you know the intrinsic matrix
Fundamental Matrix: when you don't know the intrinsic matrix
Computing Depth by triangulation

We know about the camera, K_1, K_2 and [R t]:



and that these are corresponding points: $x \leftrightarrow x'$

$$x = KX \qquad x' = K'X' = K'(RX + T)$$

How many unknowns + how many equations do we have?

only unknowns!

Solve by least squares

-

Two-View, known Camera

1. Calibrate the camera
2. Find correspondences
    a. Reduce this to 1d search with epipolar geometry
3. Get depth
    a. If simple stereo, disparity is inversely proportional to depth
    b. Triangulate

What if we don't know the camera

- Assume we know correspondences
- $\hat{x'}^T F \hat{x} = 0$
- How many correspondences

Eight point algorithm

- Enforce rank -2 constraint of F and throw out a smallest singular value

Week 13: Lecture 23 Structure-from-Motion and Multi-View Stereo (11/15)

How can we compute the camera parameters

Structure from motion

- Input: images with points in correspondence
    - $p_{i,j} = (u_{i,j}, v_{i,j})$
- Output:

    - structure: 3d location x_i for each point p_i
    - Motion: camera parameters R_j, t_j, K_j

Feature matching
- Using RANSAC

Structure from motion
- Inputs: feature tracks
- Structure from motion, with two creams easy
- Initial model, ady to asd noew
- Increment

Application: Match moving
- Motion tracking, solving for camera trajectory
- Integral for visual effects

Multi-view stereo
- Input: calibrated images from several viewpoints
- Output: 3D model

Multi-view stereo: Basic idea
- Solve for a depth map over the whole reference view
- Combine many depth maps

Volumetric stereo
- Voxel is like a pixel in 3D
- Instead of 2d array, it is 3d array
- Space carving
    - Go from all voxels, solid or air

COLMAP
- General SfM + MVS pipeline

Week 14: Lecture 24 Video & Texture Synthesis (11/22)

Markov Chains

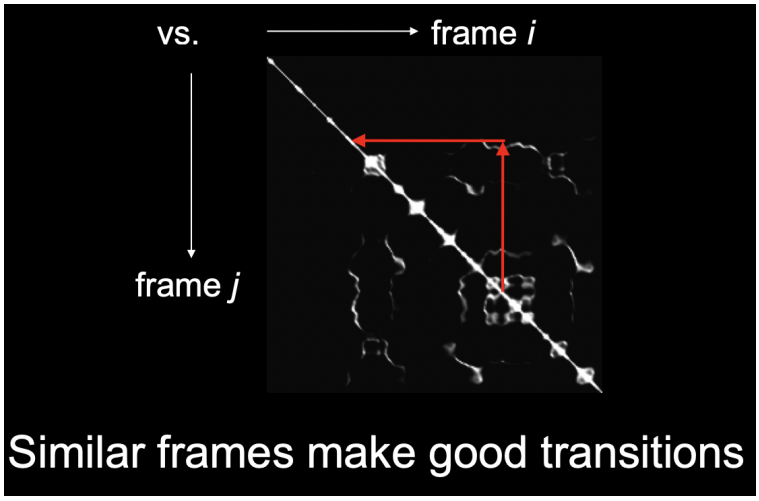
Generalized Markov Chains: multiple days into one

Text Synthesis
- Generate English-looking text using N-gram

Video Textures
- Video clip -> Video textures

Generate markov chain from video

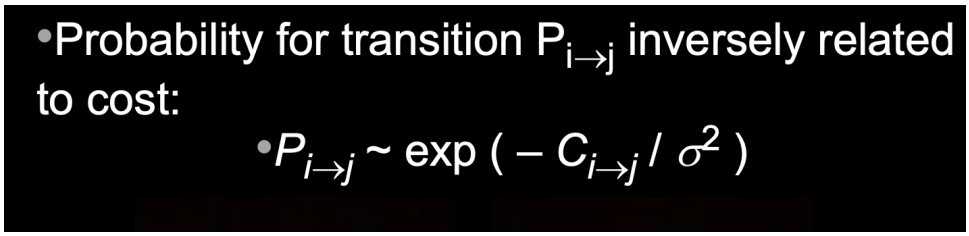
Finding Good Transitions
- Compute L2 distance $D_{i,j}$ between all frames

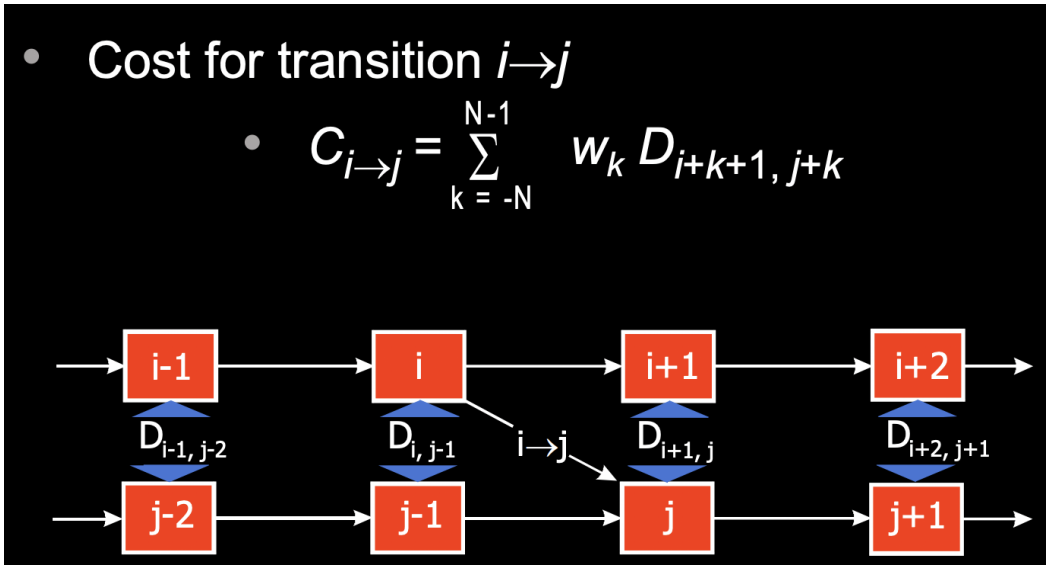


Want to find similar frames on the side of the diagonal

Transition Costs

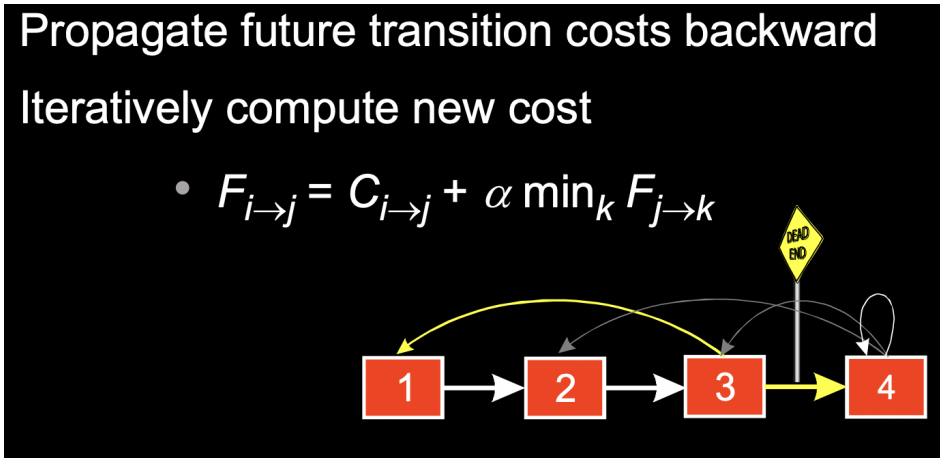
- Transition from i to j if successor of i is similar to j
  - Cost function: $C_{i \to j} = D_{i+1, j}$





Future cost

- Q-learning for dead ends



Region-based analysis

- But too many independently moving things don't work

Assuming Markov property compute $P(p|N(p))$

- Building explicit probability tables infeasible
- We search the input image for all similar neighborhoods
- Sample from this pdf, pick one match at random

Corrupt Professors

- Plagiarize as much as possible and find patch to combine them together

Texture transfer

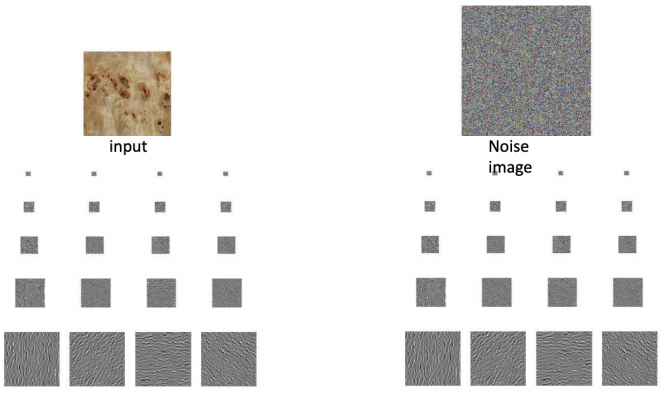
- Take texture from one image and paint it onto another

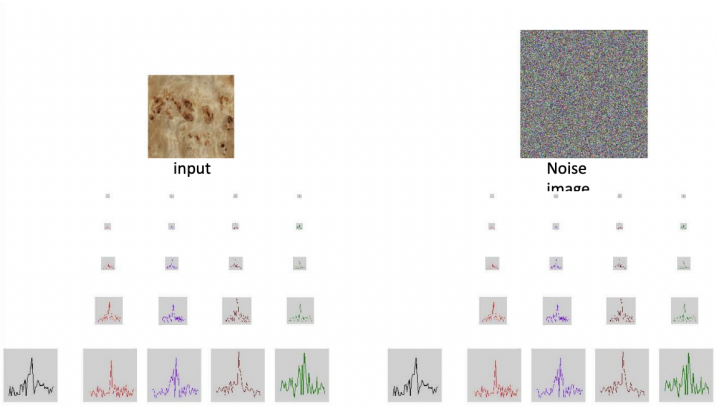
1. Consistency of texture
2. Similarity to the image benign "explained"

SPADE (SPatially Adaptive Denormalization)

- Make sure normalization doesn't kill it by reinputting the image into the layers of the convnet

1. Convolve with filterbank

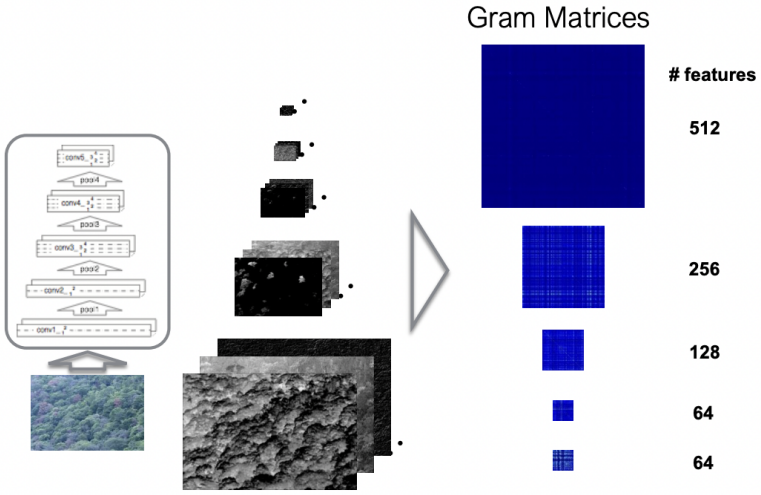




2. Match per channel histograms
3. Collapse pyramid and repeat

Texture image and make features from CNN

- Compute gram matrix, products of outer products



Gram matrix of texture features

Can use for artistic style transfer

GANs as texture synthesis

- Conjecture: GANs might be learning the "right" features to match for natural images

Week 15: Lecture 25 Modeling the Plenoptic Function (11/29)

Plenoptic Function

Today: Modeling the plenoptic function

- Capture a 4D slice of the plenoptic function
    - Combine captured rays for cool effects
- Inference method: neural Radiance Field (NeRF) ECCV 2020
    - From a set of images, model the plenoptic function

Ray

- 5D: 3d position, 2d Direction
- Assume the light is constant along the ray
- 2 parameter parameterization

Lumigraph - organization

- Hold s, t constant
- Let u, v vary -> an image
- Position 3d vacuum, because 2dc

Conventional Versus light fields means

Focu: Where the arrash com9ing Average over values to get true if already in mpa, check that valid username and fille instruct

- Put it into a single camera,
- Moving the observer

Light might change

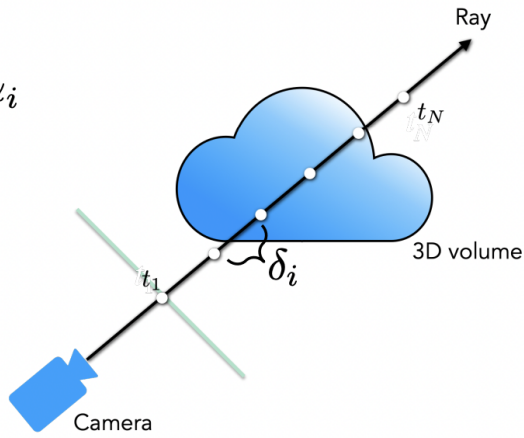
How to model th 5D plenoptic function

- NeRF: representing scenes as enrual radiance fields for view synthesis
- Model with multilayer perceptron
- Network given viewing direction and point gives rgb direction

Every point you have color + density

- At every point you know (c_i, σ_i)
  - Need to integrate values on the rage to render a pixel
- Input channel is continuous value to the algorithm

α_i



How to render a pixel: Volume Rendering

- For a ray : $r(i) = \vec{o} + i\vec{d}$
  - $C(r) \approx \sum_i^N w_i c_i \quad w_i = T_i \alpha_i$
  - How much light a ray segment contributes: $\alpha_i = 1 - exp(-\sigma_i \delta_i)$
  - Transmittance, how much light reaches point i:

$$T_i = \prod_{j=1}^{i-1}(1 - \alpha_j)$$

Optimize NeRF over many calibrated input views

NeRF: Continuous voxel with view dependence

- Voxel -- 3D grid with values that is implicit

You can insert objects with proper occlusion

NeRF limitations

- Expensive / slow to train

- Sensitive to sampling strategy, does not generalize between scenes, sensitive to pose accuracy
- Assumes static scene
- Assumes statlightning

Week 15: Lecture 26 What makes a Great Picture?/ Contemporary Art (12/1)

Framing
- See a subject and put a frame around it
- Frame gives the image depth
- Draw the eye of the viewer of an interest to a particular part of the scene
- Bring context to the shot

Rule of Thirds
- Put it in the third, if multiple opposite thirds corner

Leading Lines
- Like continuity and prediction

Textures and Patterns
- Like to know what to expect

Simplicity
- Obvious what one should be looking at and easy to separate subject from the background

B&W for simplicity
- Colors can be distracting

Clean Backgrounds
- Don't distract

Front Lightning
- Sun is at the back when taking the picture

Side Lighting
- Brings out the 3d structure

Back lightning
- For sun in the background

Go in the shade
- Light is more diffuse

Overcast days are the best
- Just don't put the sky in the frame

Best time of day: sunset & sunrise
- +/- 1 hour nice diffuse light
- After sunset: blue hour

(Sur) Realism
- Bored with everyday, want things that are rare
- Long exposures are also different

Being Comfortable with Modern/Contemporary Art

Some Principal components of art

- Cute ideas/ illusions/ clever tricks
- Direct visceral pleasure
- Familiarity / nostalgia
- Art-Historical "Context"