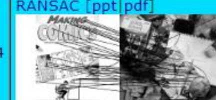
CS5670: Comput

Noah Snavely

Course review

Class	Date	Topic/notes	Readings	Assignments, etc.
0	Jan 24	Introduction and Overview [ppt|pdf] 	Szeliski 1	
1	29	Image filtering [ppt|pdf] 	Szeliski 3.1	
2	31	Image filtering 2 [ppt|pdf] 	Szeliski 3.2	
3	31	Image Resampling[ppt|pdf] 	Szeliski 3.4, 2.3.1	
4	Feb 4	Features Detection [ppt|pdf] 	Szeliski 4.1	
5	4	Features Invariance [ppt|pdf] 	Szeliski 4.1	
6	7	Descriptors [ppt|pdf] 	Szeliski 4.1	
7	12	Image Transformation [ppt|pdf] 	Szeliski 3.6	
8	14	Alignment [ppt|pdf] 	Szeliski 6.1	PA1 due
9	14	RANSAC [ppt|pdf] 	Szeliski 6.1	
10	21	Cameras [ppt|pdf] 	Szeliski 2.1.3-2.1.6	
11	28	Panoramas [ppt|pdf] 	Szeliski 9	

Topics – image processing

- Filtering
- Edge detection
- Image resampling / aliasing / interpolation
- Feature detection
  - Harris corners
  - SIFT
  - Invariant features
- Feature matching

Topics – 2D geometry

- Image transformations
- Image alignment / least squares
- RANSAC
- Panoramas

Topics – 3D geometry

- Cameras
- Perspective projection
- Single-view modeling (points, lines, vanishing points, etc.)
- Stereo
- Two-view geometry (F-matrices, E-matrices)
- Structure from motion
- Multi-view stereo

Topics – geometry, continued

- Light, color, perception
- Lambertian reflectance
- Photometric stereo

Topics – Recognition

- Different kinds of recognition problems
  - Classification, detection, segmentation, etc.
- Machine learning basics
  - Nearest neighbors
  - Linear classifiers
  - Hyperparameters
  - Training, test, validation datasets
- Loss functions for classification

Topics – Recognition, continued

- Regularization
- Neural networks
- Stochastic gradient descent
- Backpropagation
- Convolutional neural networks
  - Architectural components: convolutional layers, pooling layers, fully connected layers
- Generative methods

Questions?

Image Processing

Linear filtering

- One simple function on images: linear filtering (cross-correlation, convolution)
  - Replace each pixel by a linear combination of its neighbors
- The prescription for the linear combination is called the “kernel” (or “mask”, “filter”)

10	5	3
4	6	1
1	1	8

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	8	

Modified image data

Source: L. Zhang

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

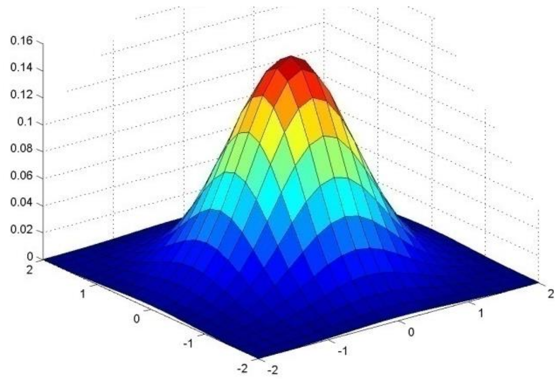
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

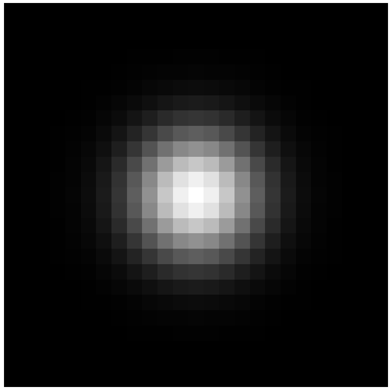
This is called a **convolution** operation:

$$G = H * F$$

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

Gaussian Kernel





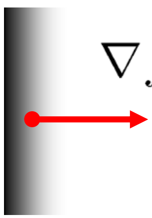
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Source: C. Rasmussen

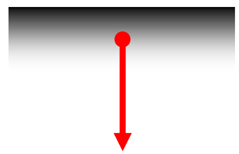
Image gradient

- The *gradient* of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

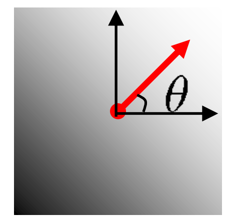
The gradient points in the direction of most rapid increase in intensity



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$



$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$



$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

The *edge strength* is given by the gradient magnitude:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

The gradient direction is given by:

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

- how does this relate to the direction of the edge?

Source: Steve Seitz

Finding edges

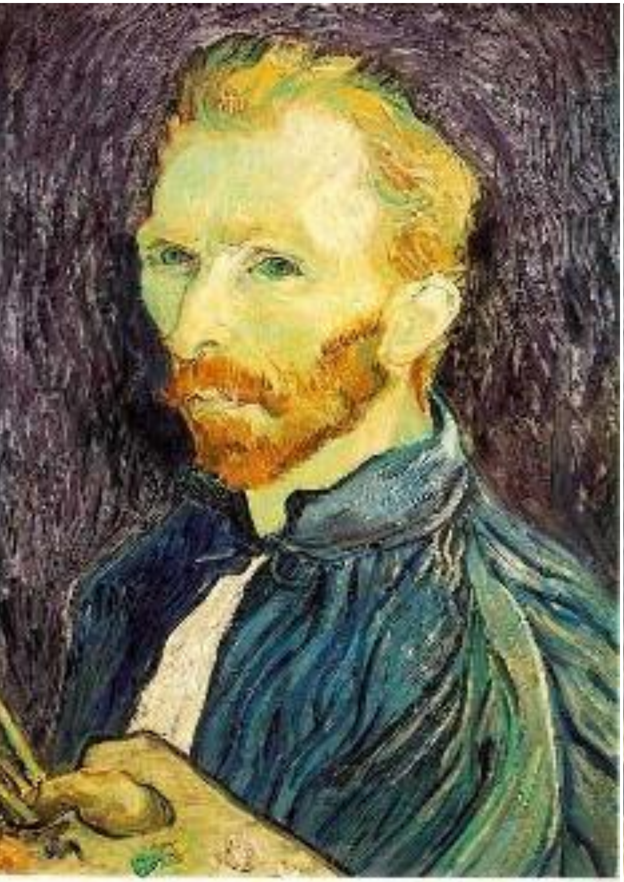
gradient magnitude

Finding edges

thinning

(non-maximum suppression)

Image sub-sampling



1/2

1/4 (2x zoom)

1/8 (4x zoom)

Why does this look so crufty?

Source: S. Seitz

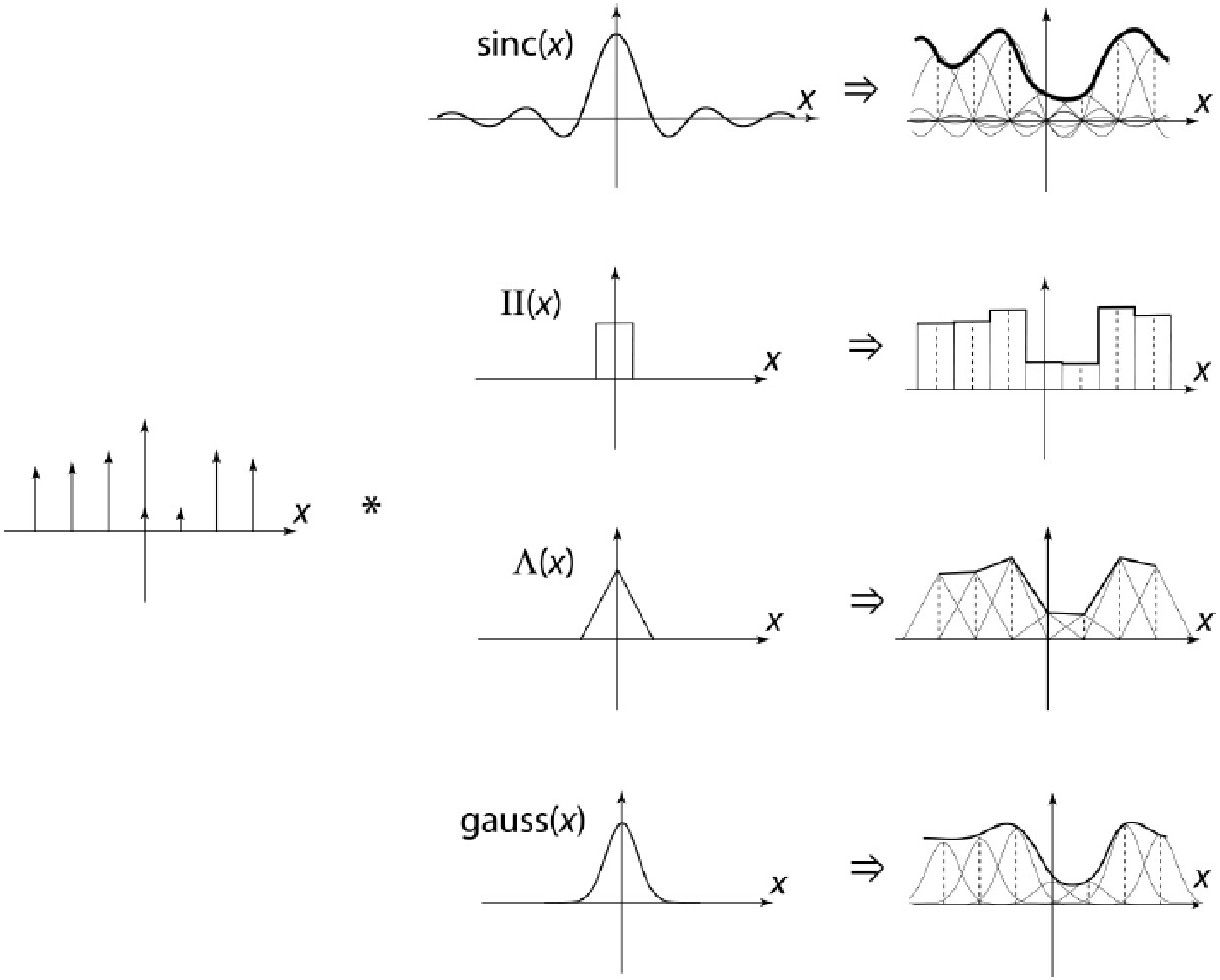
Subsampling with Gaussian pre-filtering

Gaussian 1/2 G 1/4 G 1/8

- Solution: filter the image, *then* subsample

Source: S. Seitz

Image interpolation



"Ideal" reconstruction

Nearest-neighbor interpolation

Linear interpolation

Gaussian reconstruction

Source: B. Curless

Image interpolation

Original image:  x 10

Nearest-neighbor interpolation

Bilinear interpolation

Bicubic interpolation

The second moment matrix

The surface *E*(*u*,*v*) is locally approximated by a quadratic form.

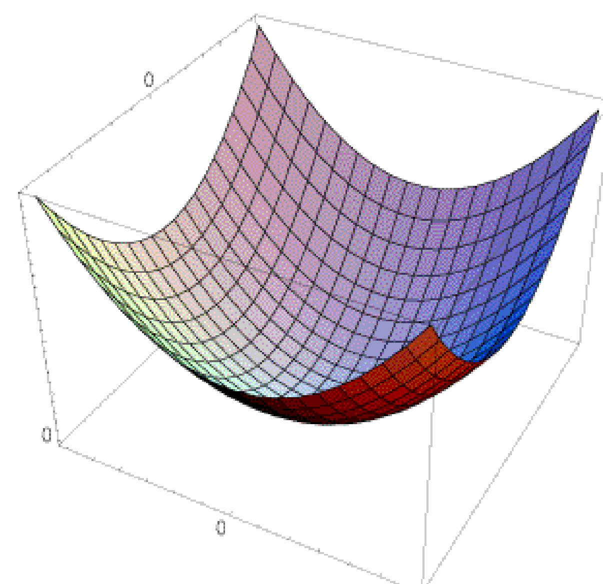
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



The Harris operator

λ_{min} is a variant of the "Harris operator" for feature detection

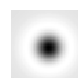
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

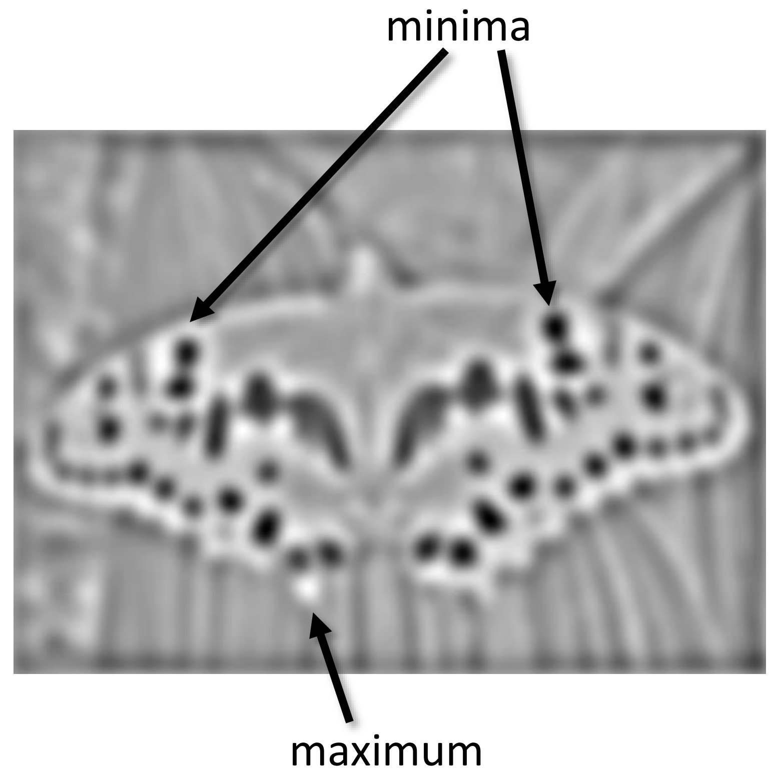
$$= \frac{determinant(H)}{trace(H)}$$

- The *trace* is the sum of the diagonals, i.e., $trace(H) = h_{11} + h_{22}$
- Very similar to λ_{min} but less expensive (no square root)
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

Laplacian of Gaussian

- “Blob” detector

$*$  =



- Find maxima *and minima* of LoG operator in space and scale

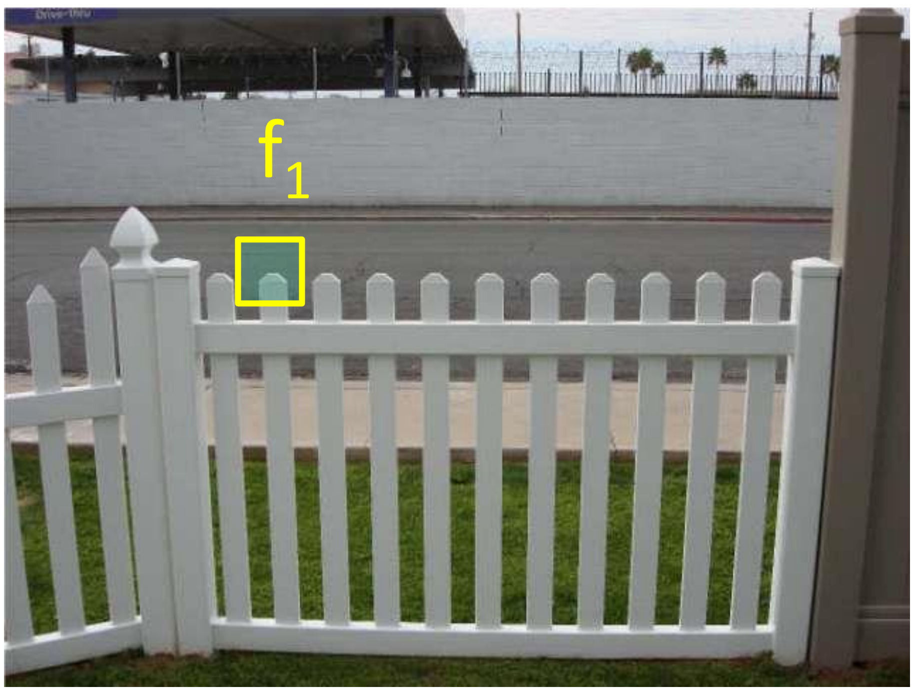
Scale-space blob detector: Example

sigma = 11.9912

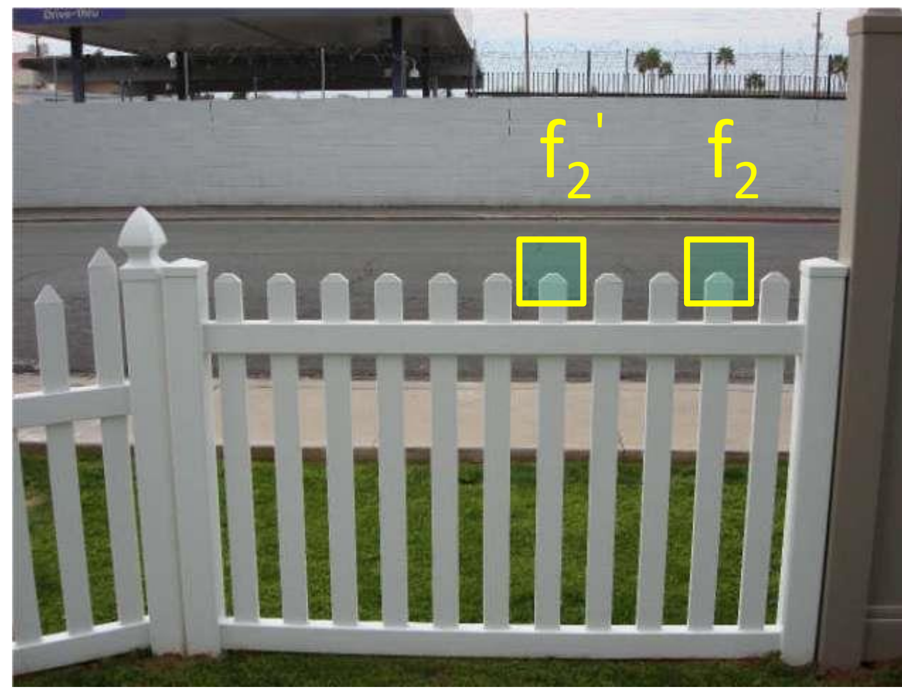
Feature distance

How to define the difference between two features f_1, f_2?

- Better approach: ratio distance = $||f_1 - f_2 || / || f_1 - f_2' ||$
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2[nd] best SSD match to f_1 in I_2
  - gives large values for ambiguous matches



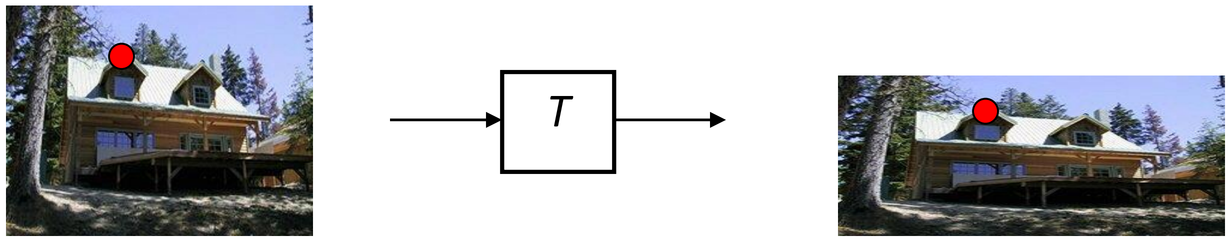
I_1



I_2

2D Geometry

Parametric (global) warping



p = (x,y) **p’** = (x’,y’)

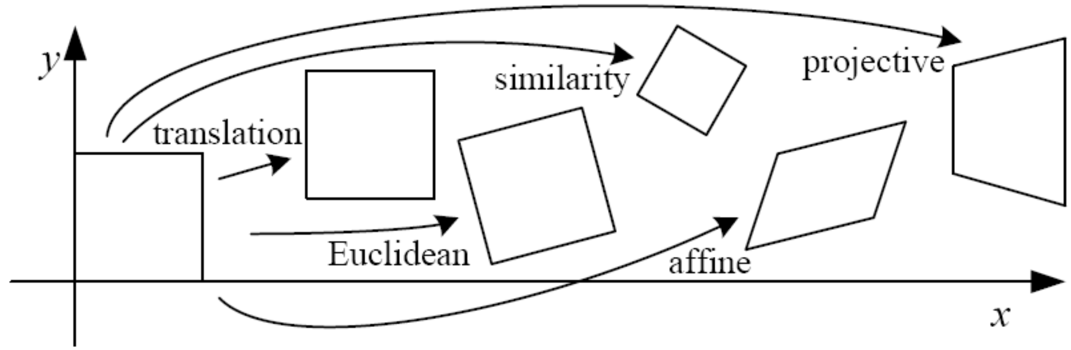
- Transformation T is a coordinate-changing machine:

  p’ = *T*(p)

- What does it mean that *T* is global?
  - Is the same for any point p
  - can be described by just a few numbers (parameters)
- Let’s consider *linear* xforms (can be represented by a 2D matrix):

$$\mathbf{p}' = \mathbf{T}\mathbf{p} \qquad \left[\begin{array}{c} x' \\ y' \end{array}\right] = \mathbf{T}\left[\begin{array}{c} x \\ y \end{array}\right]$$

2D image transformations



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

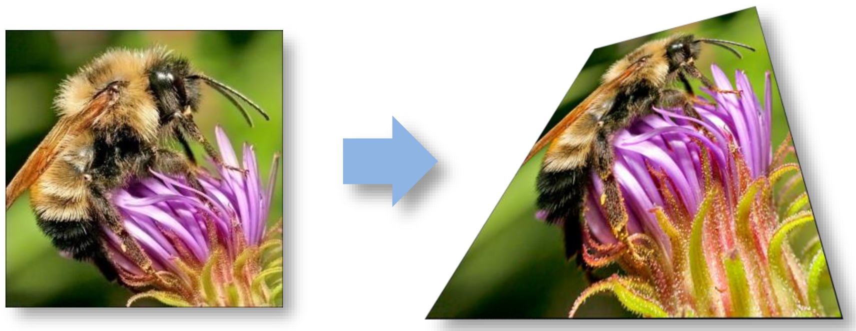
These transformations are a nested set of groups

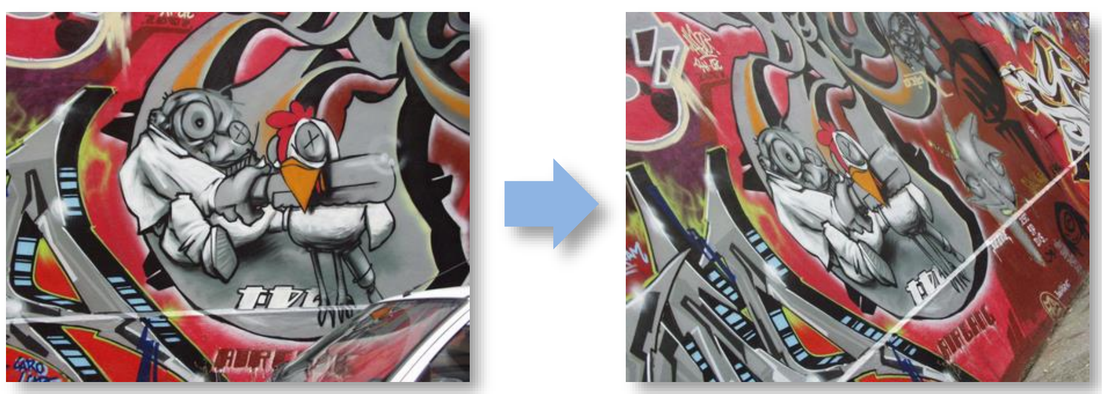
- Closed under composition and inverse is a member

Projective Transformations aka Homographies aka Planar Perspective Maps

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

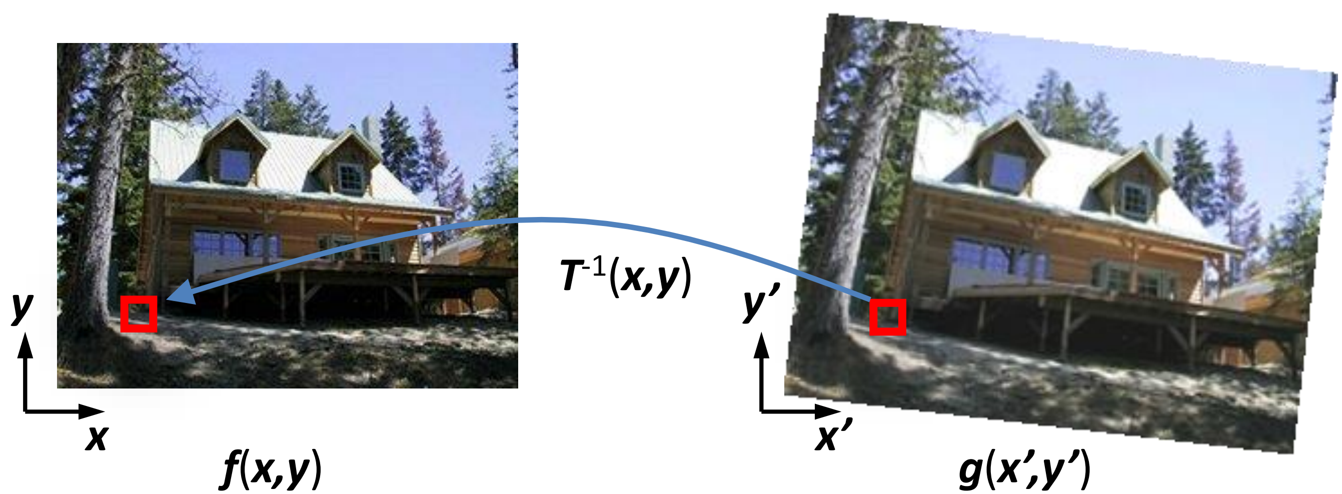
Called a *homography*
(or *planar perspective map*)





Inverse Warping

- Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x,y)$ in $f(x,y)$
  - Requires taking the inverse of the transform



Affine transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

Solving for affine transformations

- Matrix form

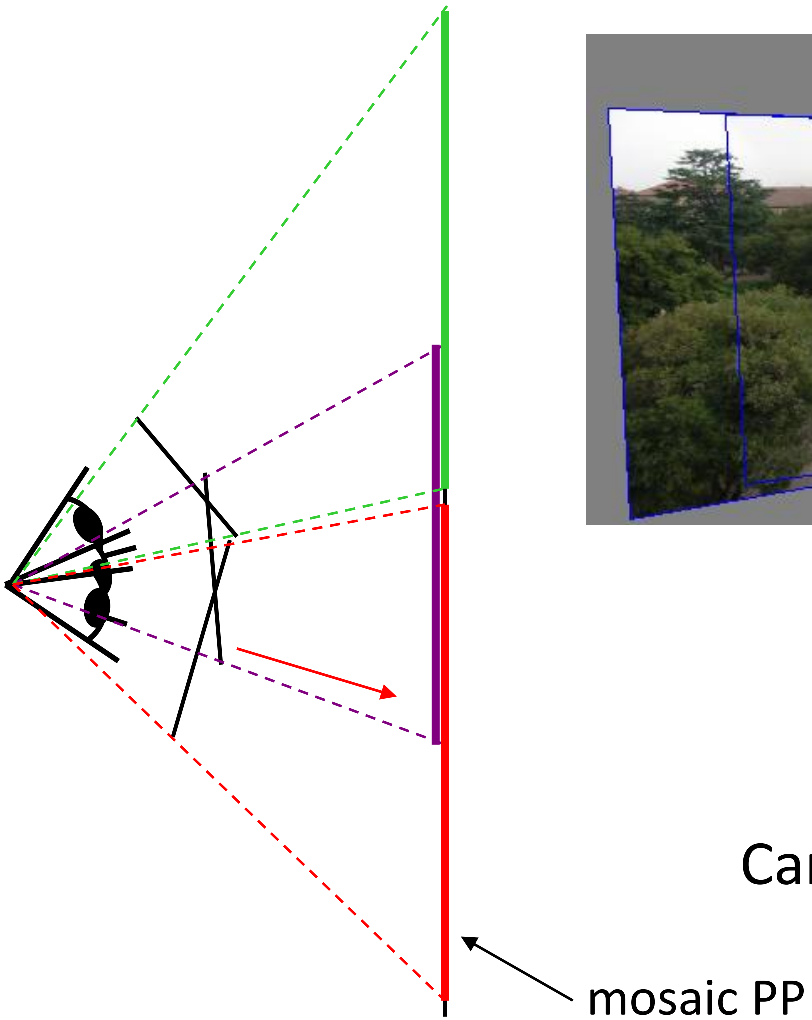
$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & & \vdots & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

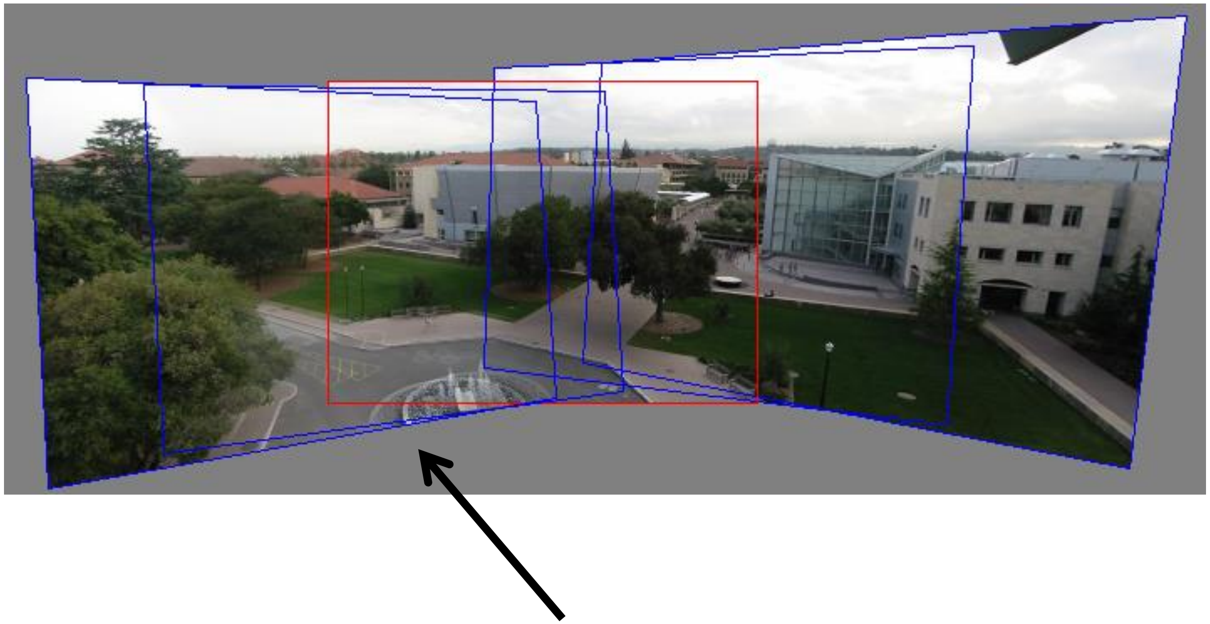
$$\underset{2n \times 6}{\mathbf{A}} \quad \underset{6 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}$$

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times
  5. Choose the model that has the largest set of inliers

Projecting images onto a common plane



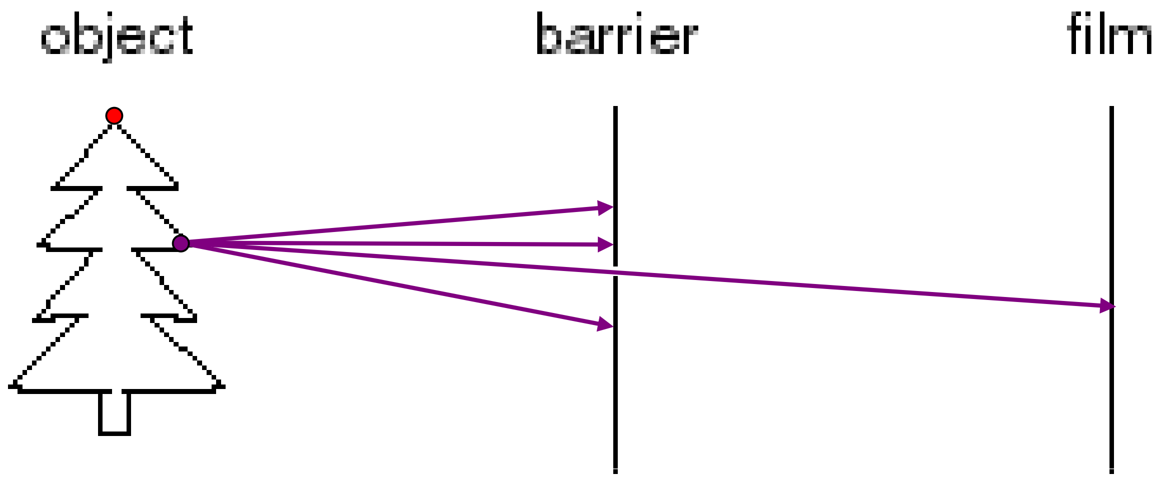


each image is warped with a homography $\mathbf{H}$

Can't create a 360 panorama this way…

3D Geometry

Pinhole camera



- Add a barrier to block off most of the rays
  - This reduces blurring
  - The opening known as the **aperture**
  - How does this transform the image?

Perspective Projection

Projection is a matrix multiply using homogeneous coordinates:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \ \ -d\frac{y}{z})$$

divide by third coordinate

This is known as **perspective projection**

- The matrix is the **projection matrix**

Projection matrix

$$\mathbf{\Pi} = \underbrace{\mathbf{K}}_{\text{intrinsics}} \underbrace{\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 \;\; 0 \;\; 0 & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \;\; 0 \;\; 0 & 1 \end{bmatrix}}_{\text{translation}}}$$

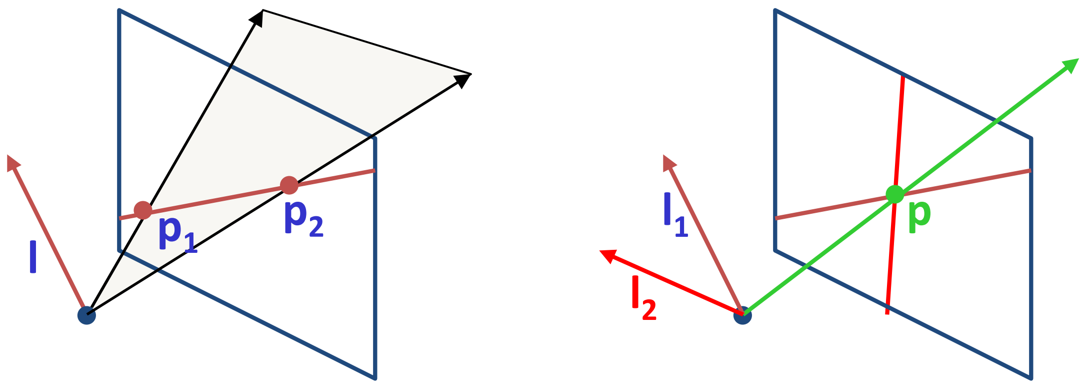
$$\left[\begin{array}{c|c} \mathbf{R} & \underbrace{-\mathbf{Rc}} \end{array}\right]$$

(**t** in book's notation)

$$\mathbf{\Pi} = \mathbf{K}\left[\begin{array}{c|c} \mathbf{R} & -\mathbf{Rc} \end{array}\right]$$

Point and line duality

- A line **l** is a homogeneous 3-vector
- It is $\perp$ to every point (ray) **p** on the line: **l p**=0



What is the line **l** spanned by rays $\mathbf{p}_1$ and $\mathbf{p}_2$?

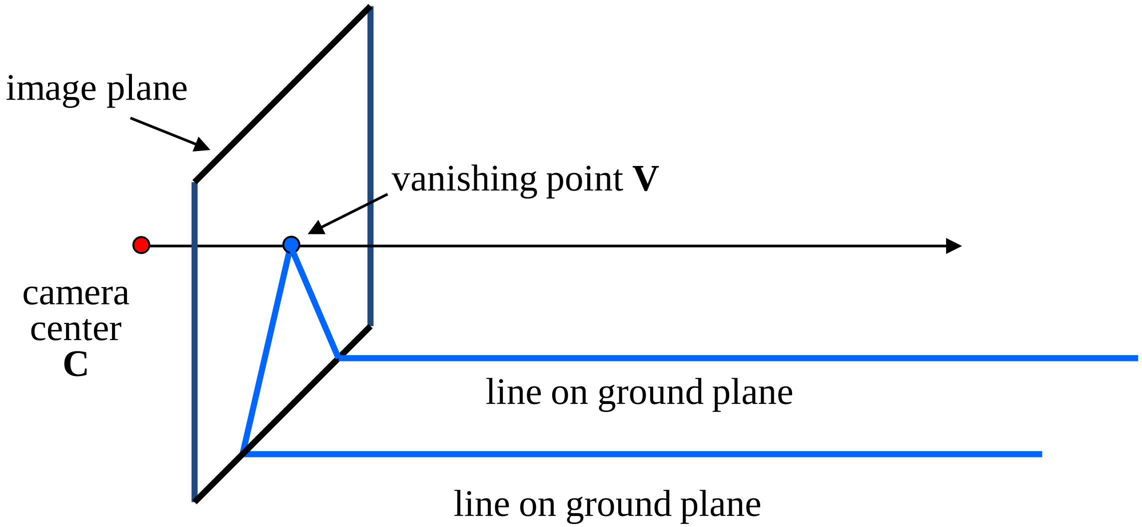
- **l** is $\perp$ to $\mathbf{p}_1$ and $\mathbf{p}_2$ $\Rightarrow$ $\mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2$
- **l** can be interpreted as a *plane normal*

What is the intersection of two lines $\mathbf{l}_1$ and $\mathbf{l}_2$?

- **p** is $\perp$ to $\mathbf{l}_1$ and $\mathbf{l}_2$ $\Rightarrow$ $\mathbf{p} = \mathbf{l}_1 \times \mathbf{l}_2$

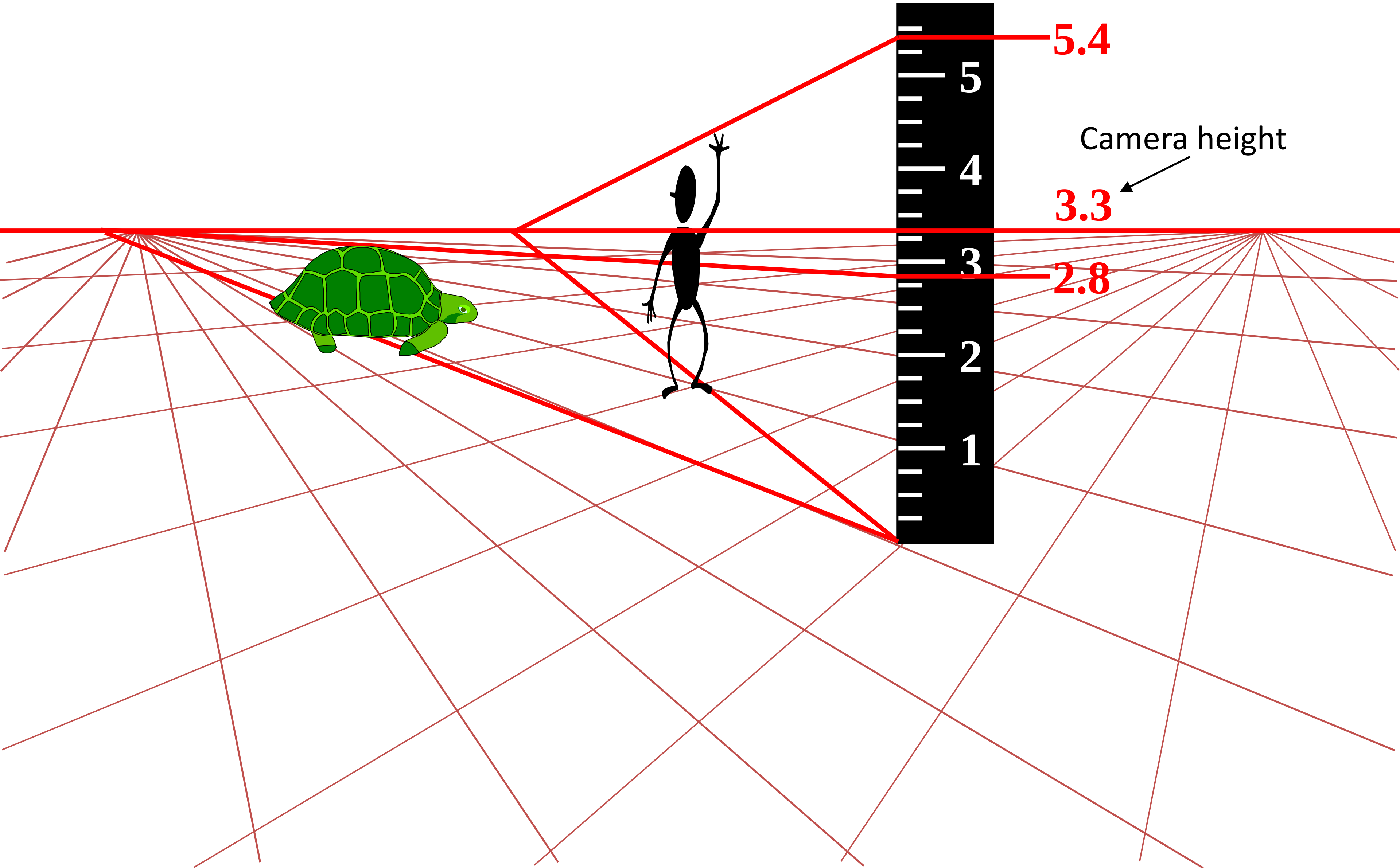
Points and lines are *dual* in projective space

Vanishing points

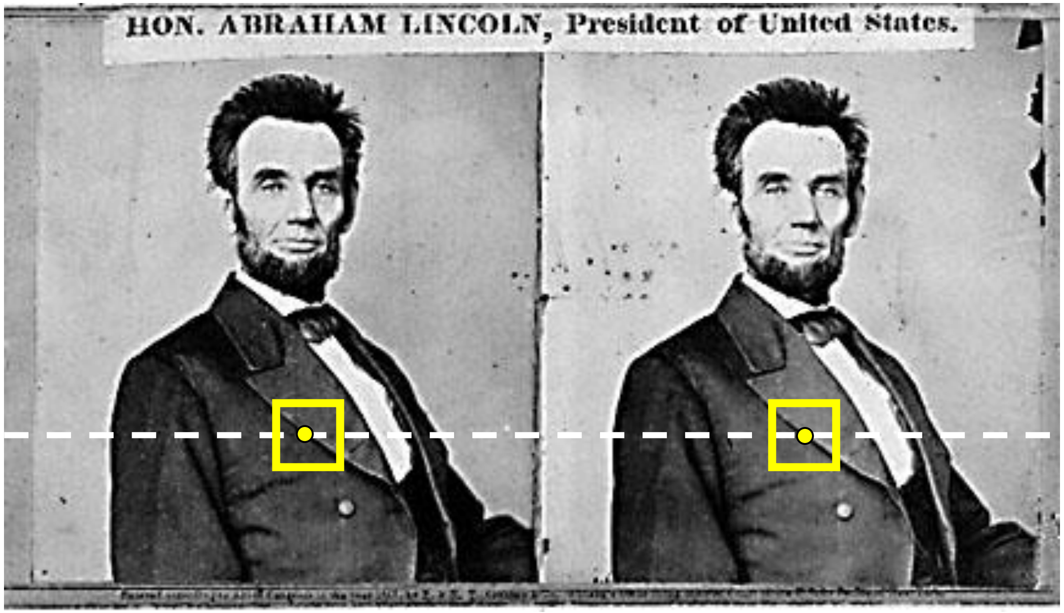


- Properties
  - Any two parallel lines (in 3D) have the same vanishing point **v**
  - The ray from **C** through **v** is parallel to the lines
  - An image may have more than one vanishing point
    - in fact, every image point is a potential vanishing point

Measuring height



Your basic stereo algorithm



For each epipolar line

For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

Stereo as energy minimization

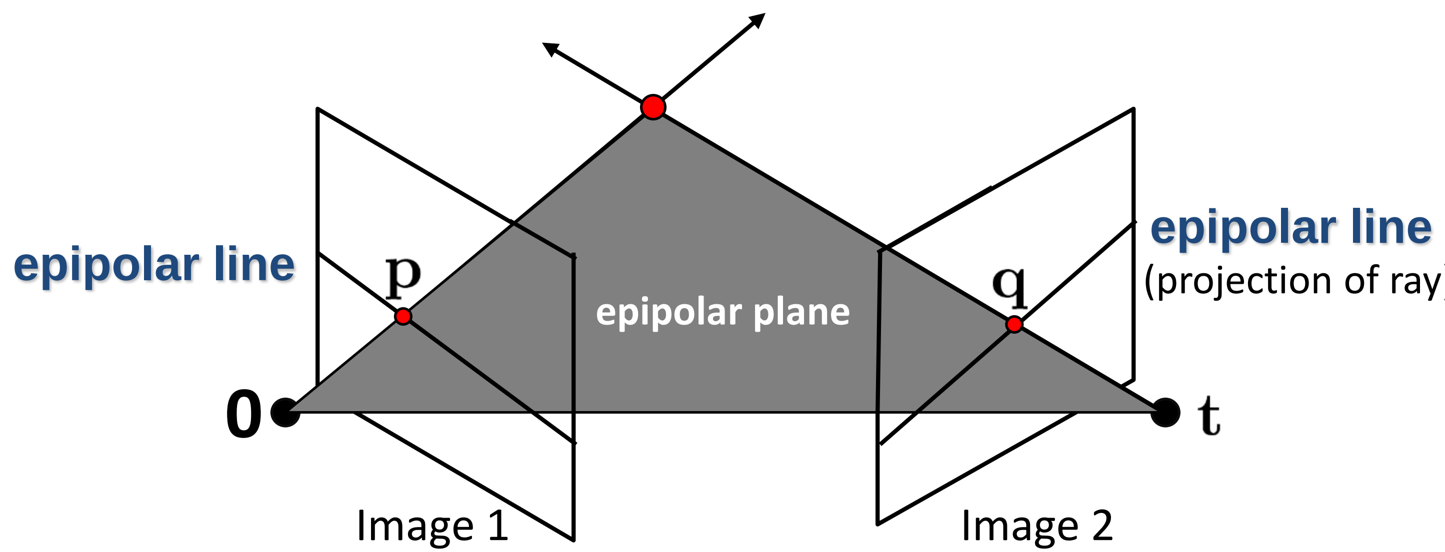
- Better objective function

$$E(d) = \underbrace{E_d(d)}_{\text{match cost}} + \lambda \underbrace{E_s(d)}_{\text{smoothness cost}}$$

match cost: Want each pixel to find a good match in the other image

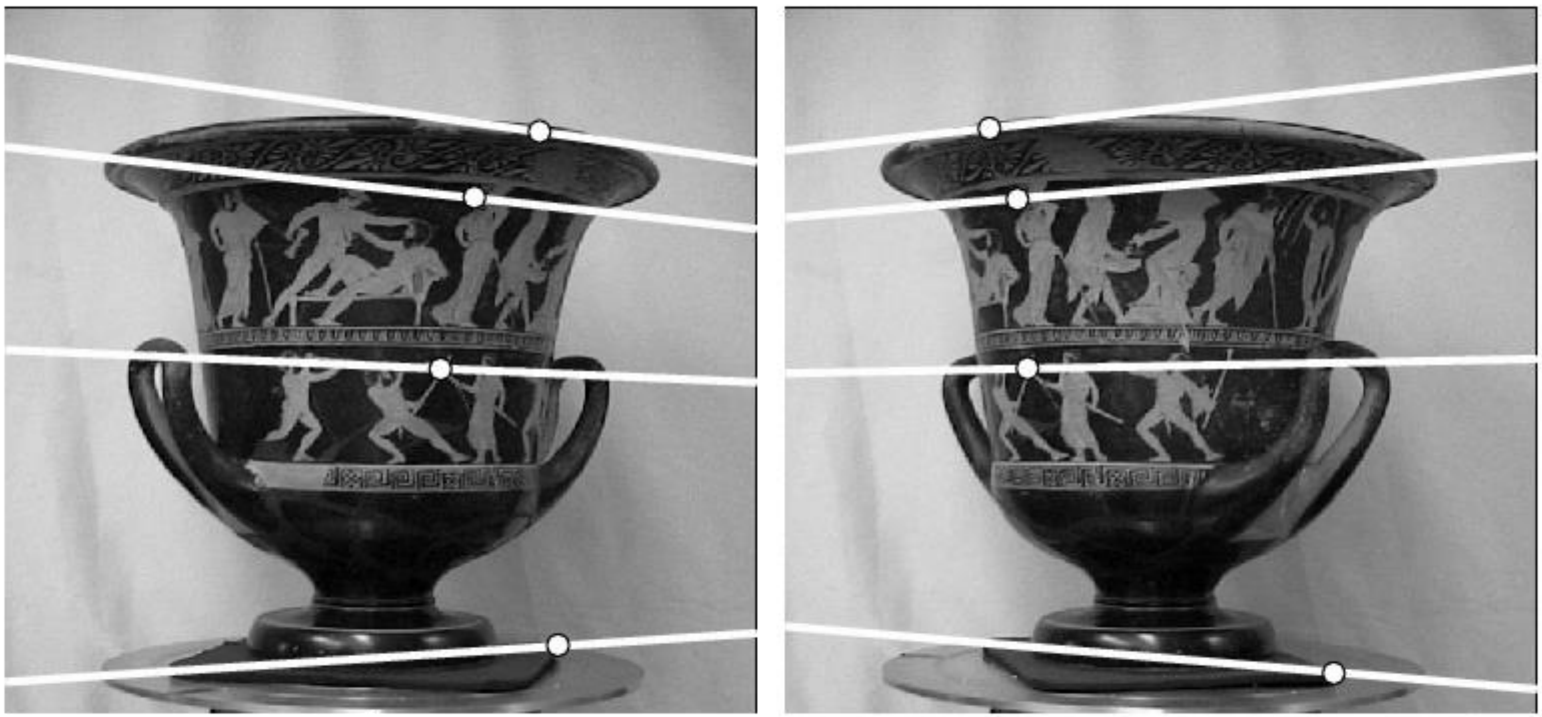
smoothness cost: Adjacent pixels should (usually) move about the same amount

Fundamental matrix



- This *epipolar geometry* of two views is described by a Very Special 3x3 matrix $\mathbf{F}$, called the *F`undamental matrix*
- $\mathbf{F}$ maps (homogeneous) *points* in image 1 to *lines* in image 2!
- The epipolar line (in image 2) of point **p** is: $\mathbf{Fp}$
- *Epipolar constraint* on corresponding points: $\mathbf{q}^T\mathbf{Fp} = 0$

Epipolar geometry demo

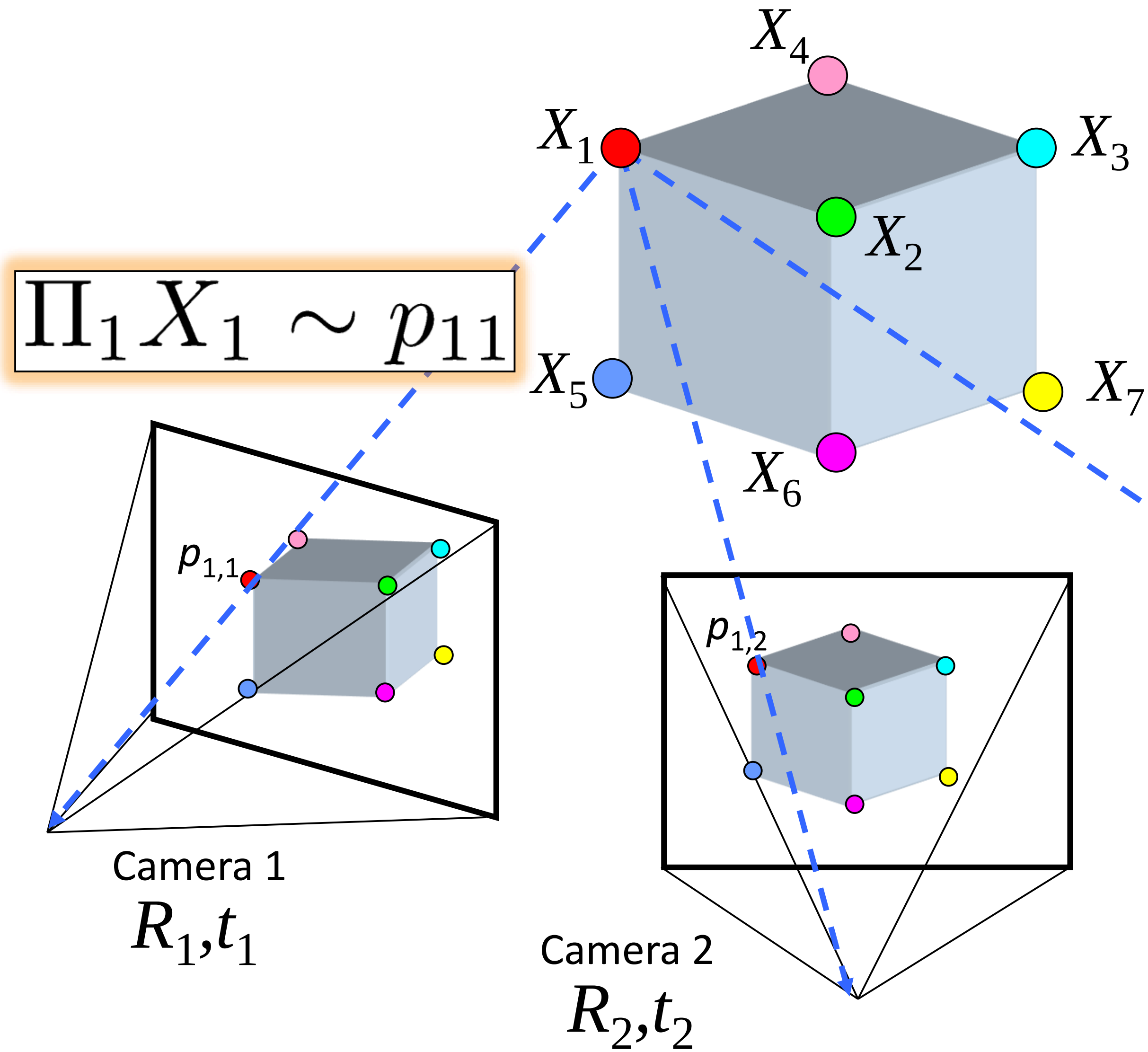


8-point algorithm

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

- In reality, instead of solving $\mathbf{Af} = 0$, we seek $\mathbf{f}$ to minimize $\|\mathbf{Af}\|$, least eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

Structure from motion

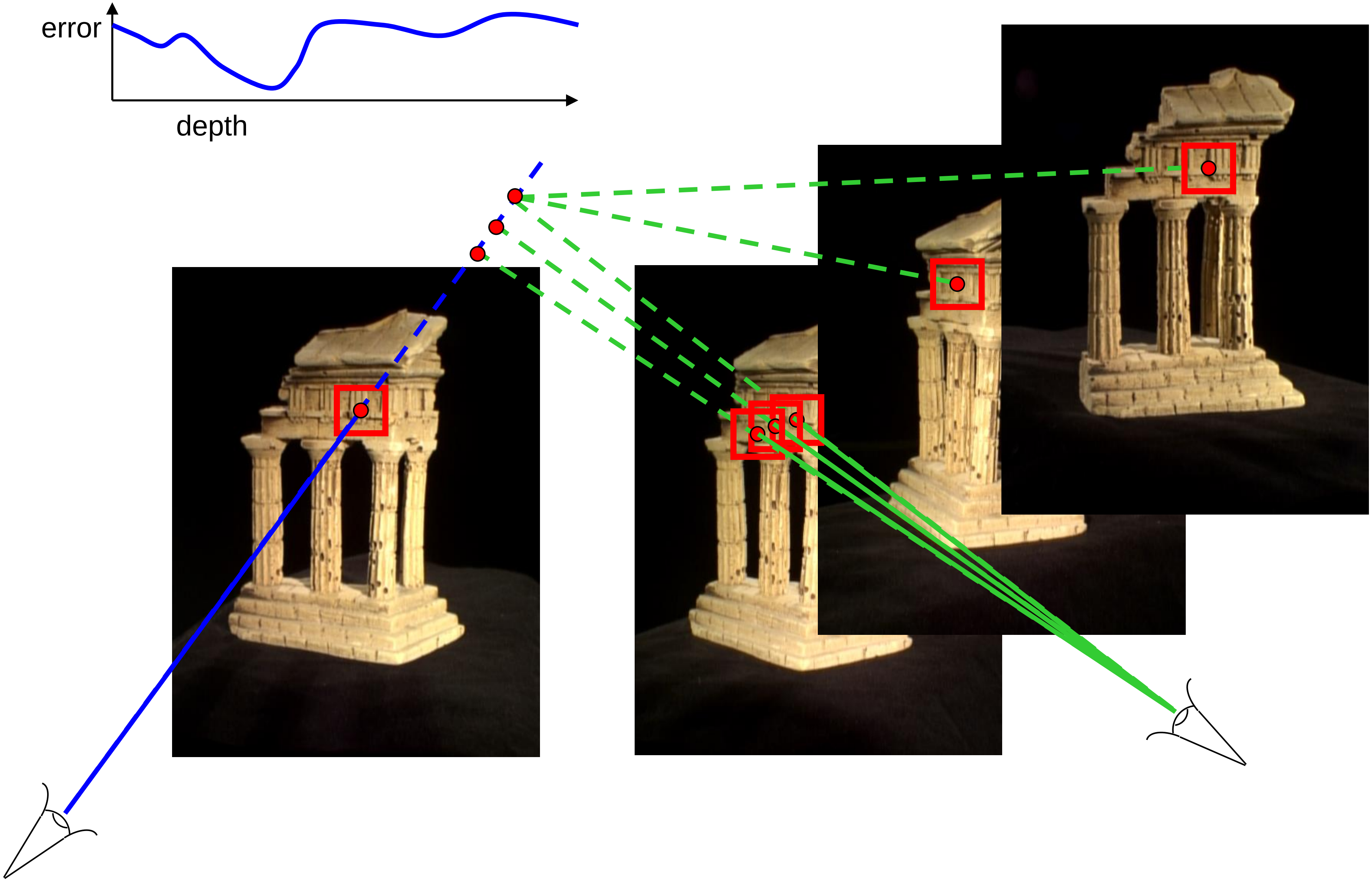


minimize

$$f\,(\mathbf{R}, \mathbf{T}, \mathbf{P})$$

non-linear least squares

Stereo: another view



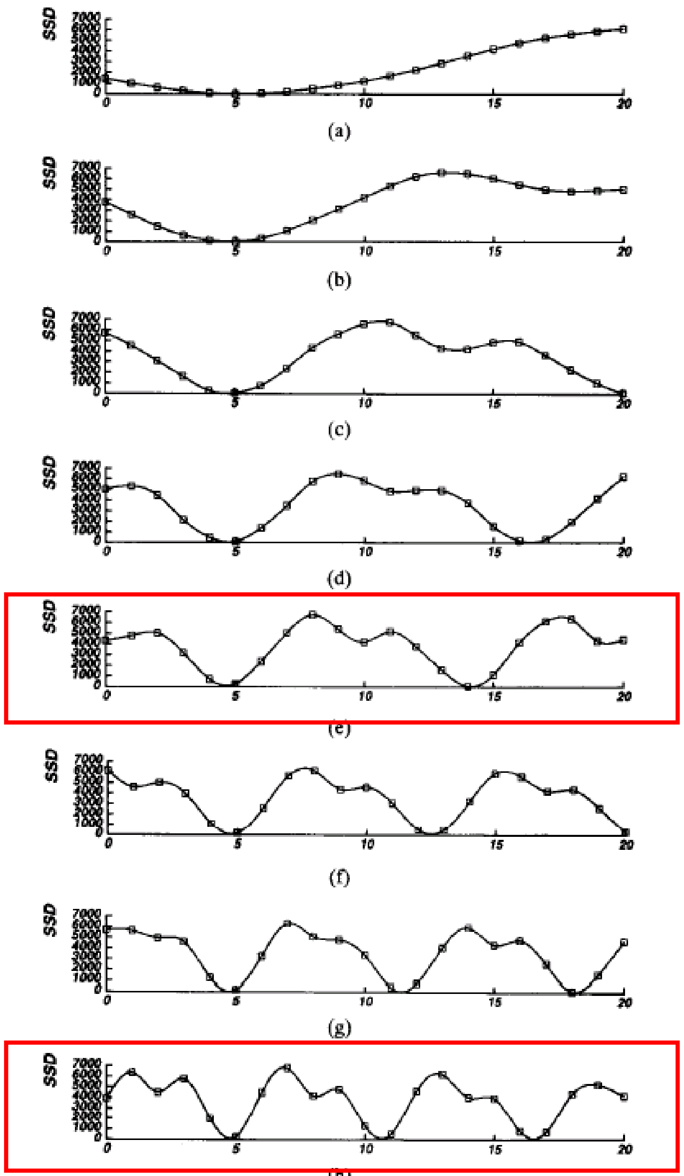


Fig. 5. SSD values versus inverse distance: (a) $B = b$; (b) $B = 2b$; (c) $B = 3b$; (d) $B = 4b$; (e) $B = 5b$; (f) $B = 6b$; (g) $B = 7b$; (h) $B = 8b$. The horizontal axis is normalized such that $8bF = 1$.

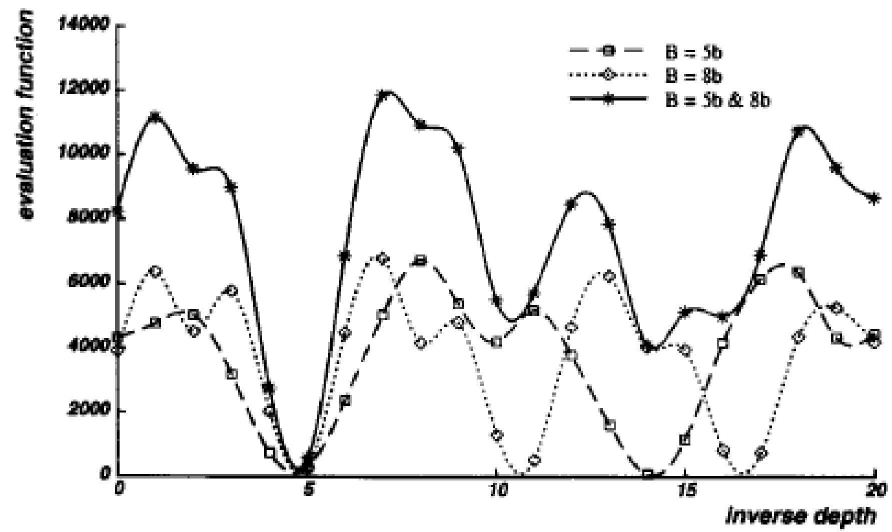


Fig. 6. Combining two stereo pairs with different baselines.

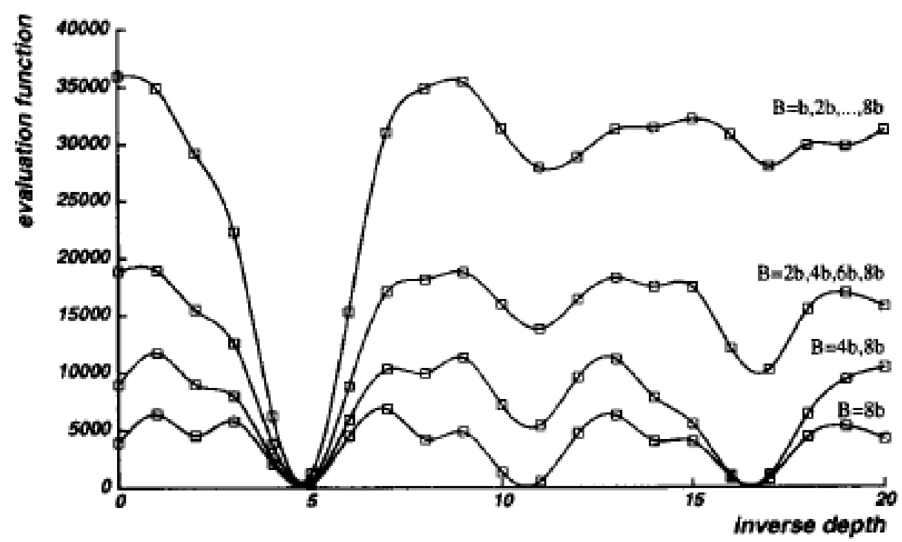
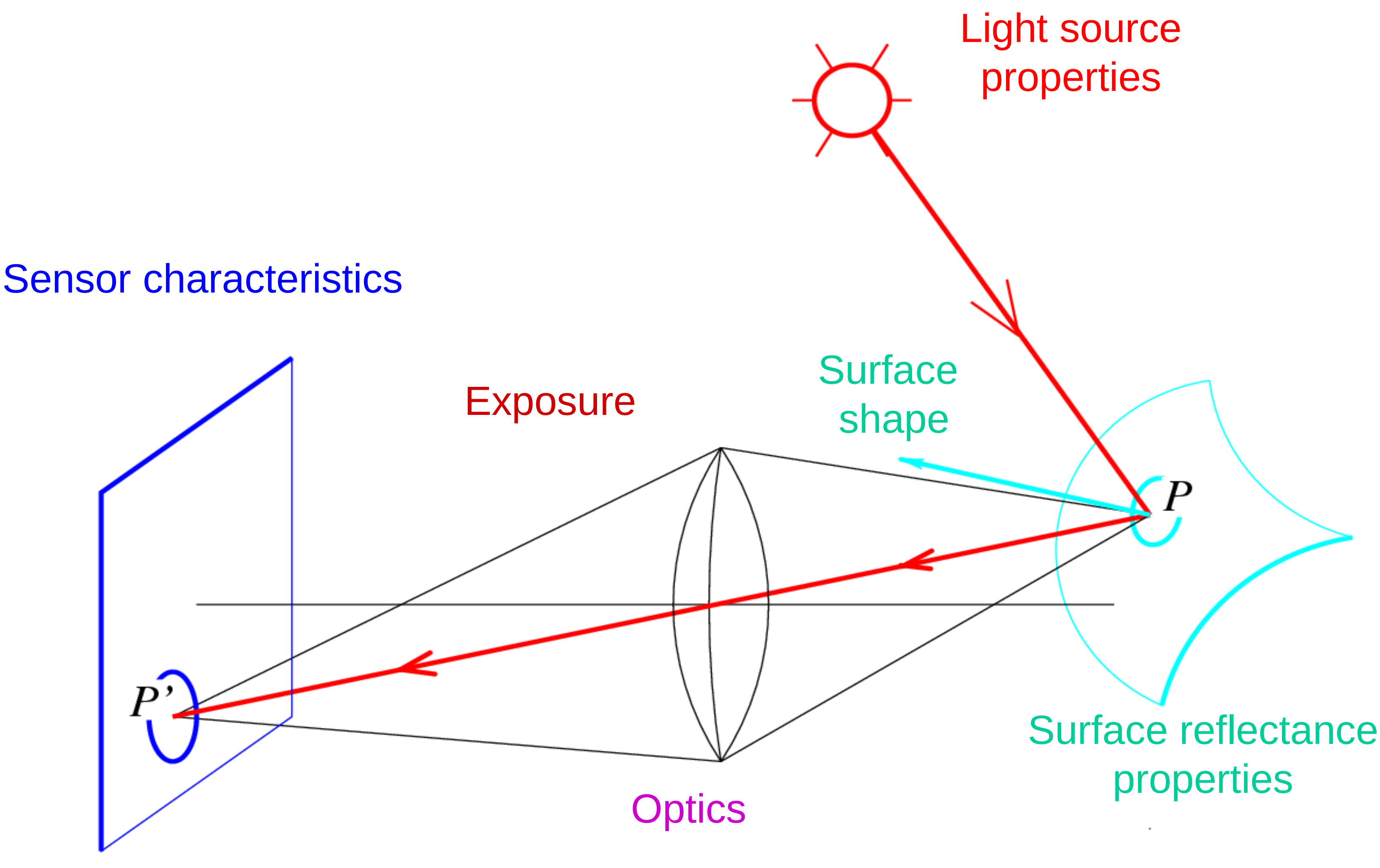


Fig. 7. Combining multiple baseline stereo pairs.

Light, reflectance, cameras

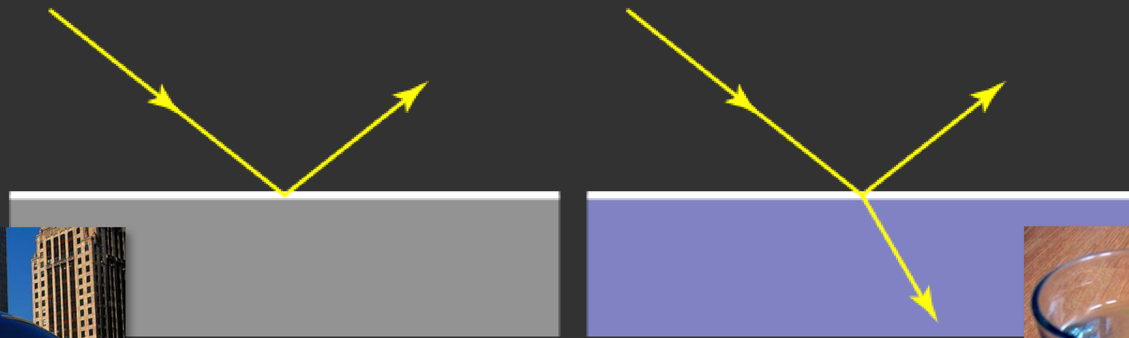
Radiometry

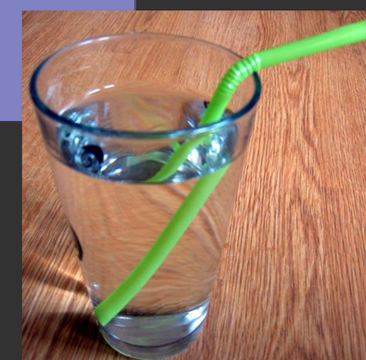
What determines the brightness of an image pixel?



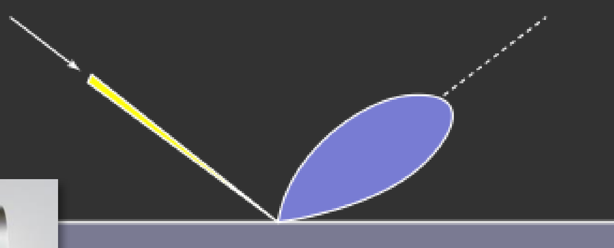
Slide by L. Fei-Fei

Classic reflection behavior

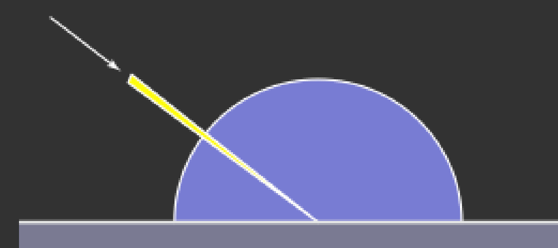


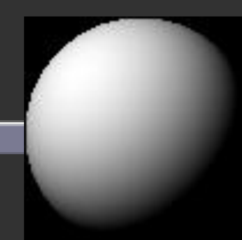


ideal specular



rough specular

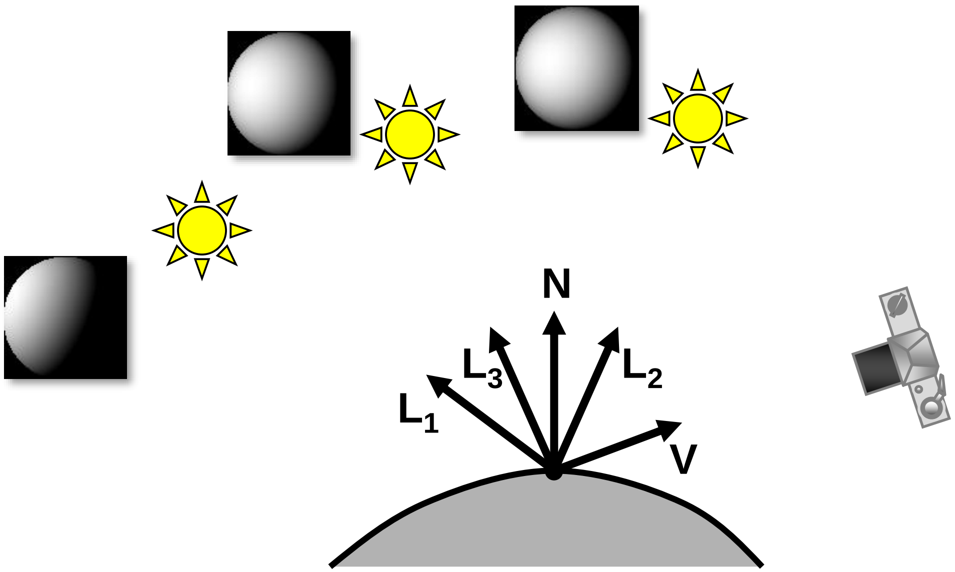




Lambertian

from Steve Marschner

Photometric stereo

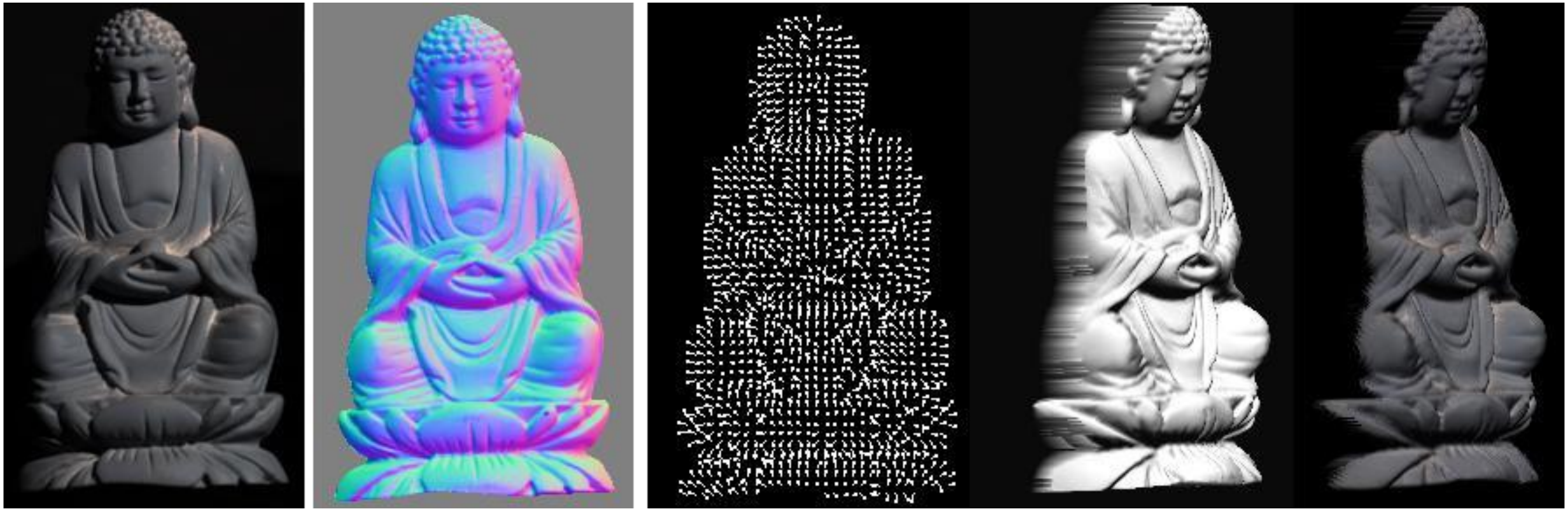


$$I_1 = k_d \mathbf{N} \cdot \mathbf{L_1}$$
$$I_2 = k_d \mathbf{N} \cdot \mathbf{L_2}$$
$$I_3 = k_d \mathbf{N} \cdot \mathbf{L_3}$$

Can write this as a matrix equation:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = k_d \begin{bmatrix} \mathbf{L_1}^T \\ \mathbf{L_2}^T \\ \mathbf{L_3}^T \end{bmatrix} \mathbf{N}$$

Example



Recognition

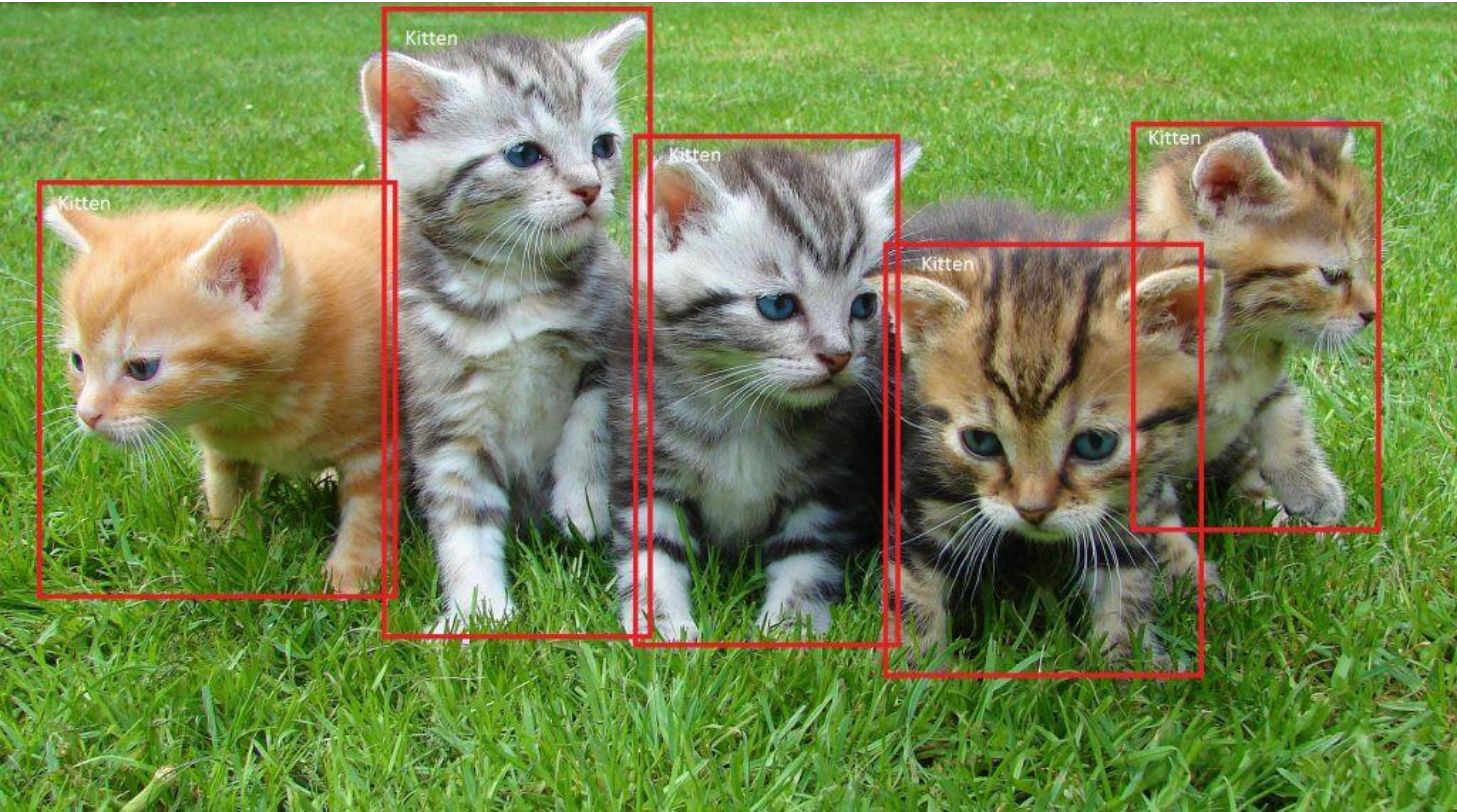
Image Classification

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

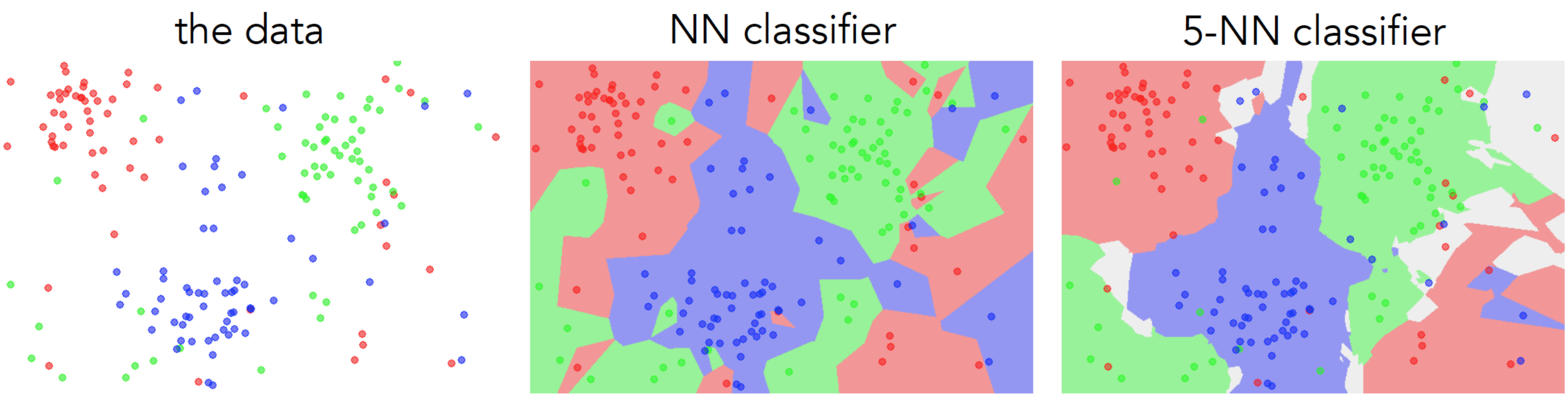
Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Object detection



k-nearest neighbor

- Find the k closest points from training data
- Take **majority vote** from K closest points



Hyperparameters

- What is the **best distance** to use?
- What is the **best value of k** to use?

- These are **hyperparameters**: choices about the algorithm that we set rather than learn

- How do we set them?
  - One option: try them all and see what works best

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train	test

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

train	validation	test

Parametric approach: Linear classifier

$$f(x, W) = Wx \quad (+b)$$

3072x1

10x1

10x1

10x3072

[32x32x3]
array of numbers 0...1

10 numbers, indicating class scores

parameters, or "weights"

Loss function, cost/objective function

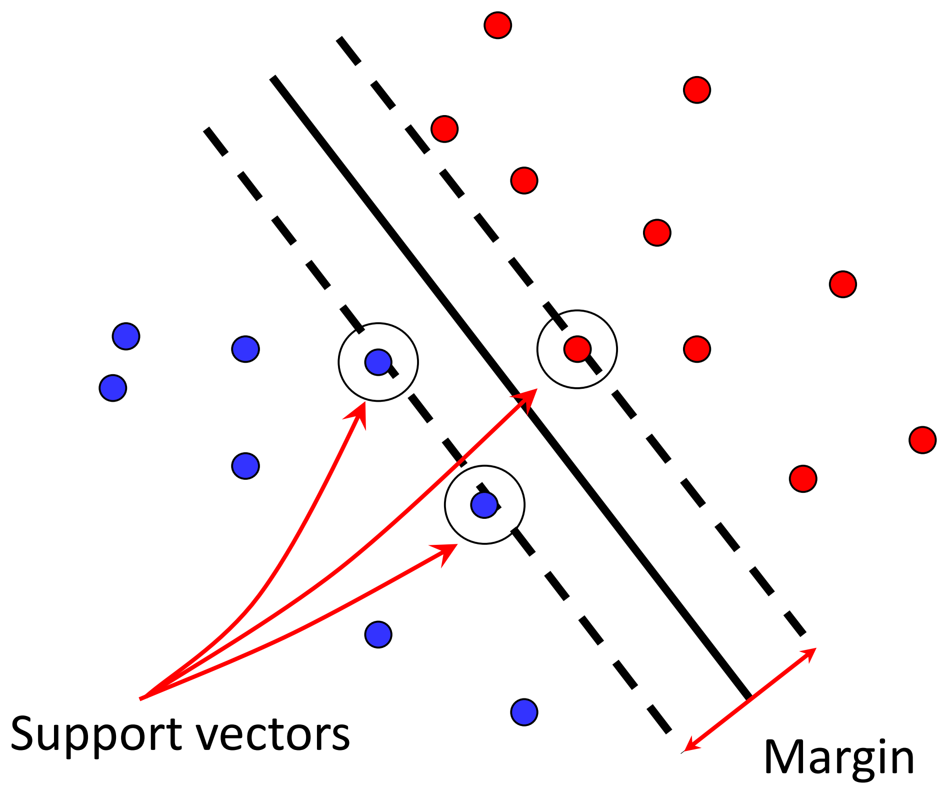
- Given ground truth labels (y_i), scores $f(x_i, \mathbf{W})$
  - how unhappy are we with the scores?

- Loss function or objective/cost function measures unhappiness

- During training, **want to find the parameters W that minimizes the loss function**

Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{||\mathbf{w}||}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

Multi-class SVM loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Softmax classifier

$$f(x_i, W) = Wx_i$$

score function is the same

$$\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

softmax function

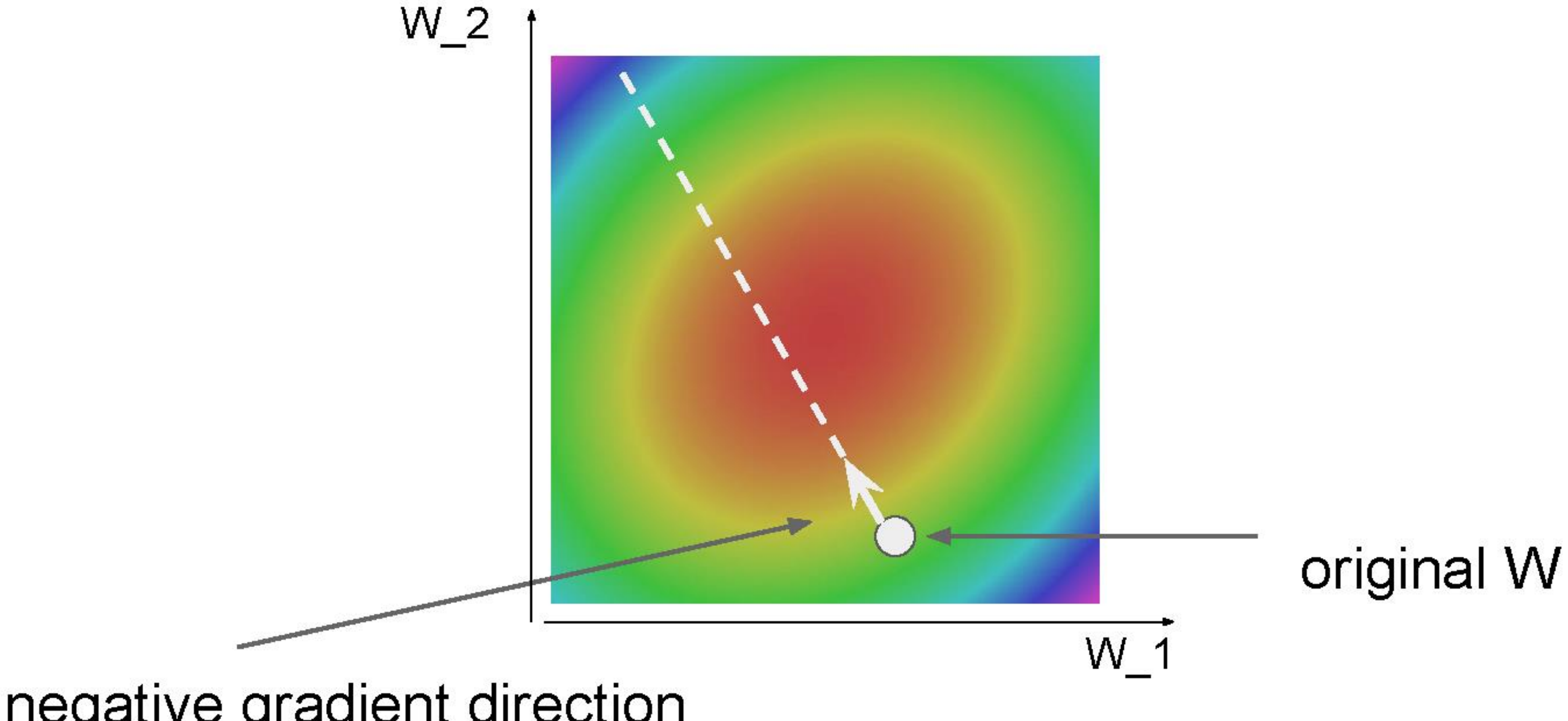
$$[1, -2, 0] \to [e^1, e^{-2}, e^0] = [2.71, 0.14, 1] \to [0.7, 0.04, 0.26]$$

Interpretation: squashes values into range 0 to 1

$$P(y_i \mid x_i; W)$$

Optimizing weights to minimize loss

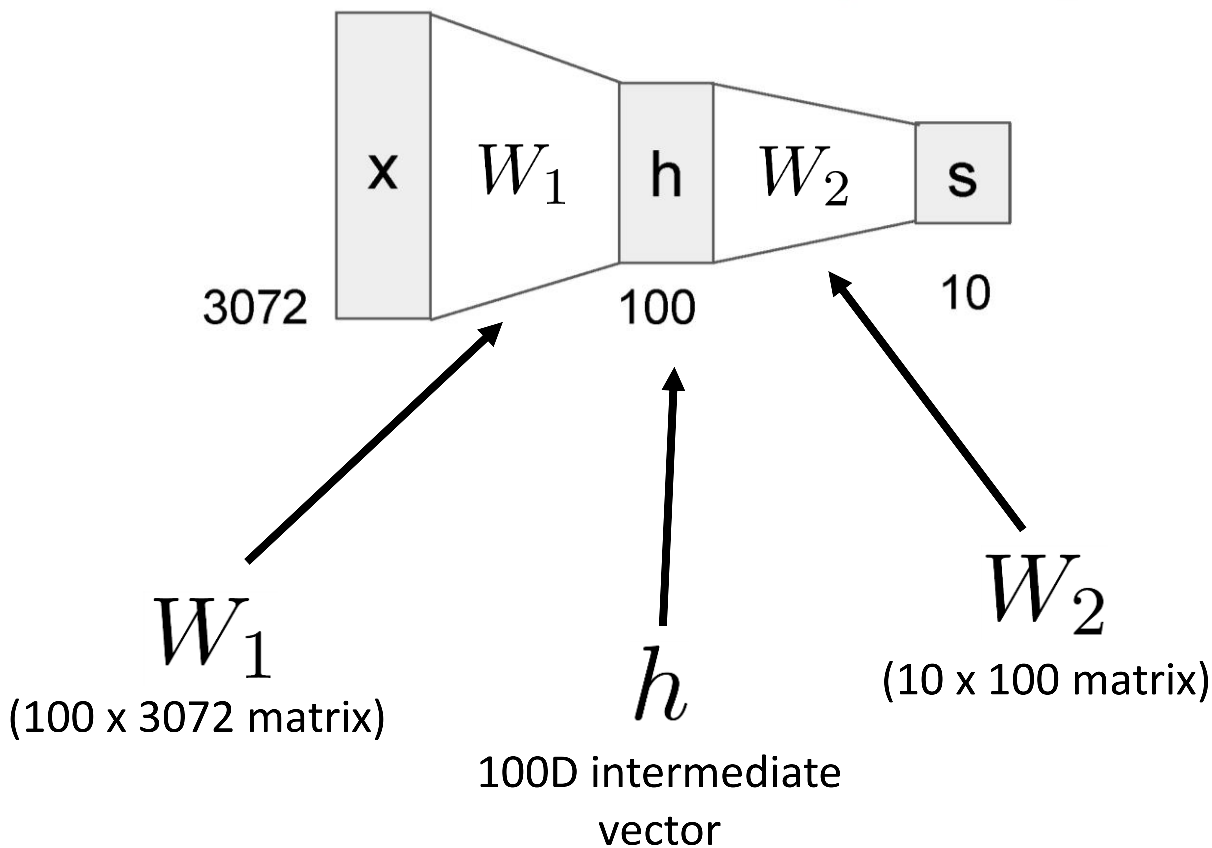
- Stochastic gradient descent



Neural networks

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$



Convolutional neural networks

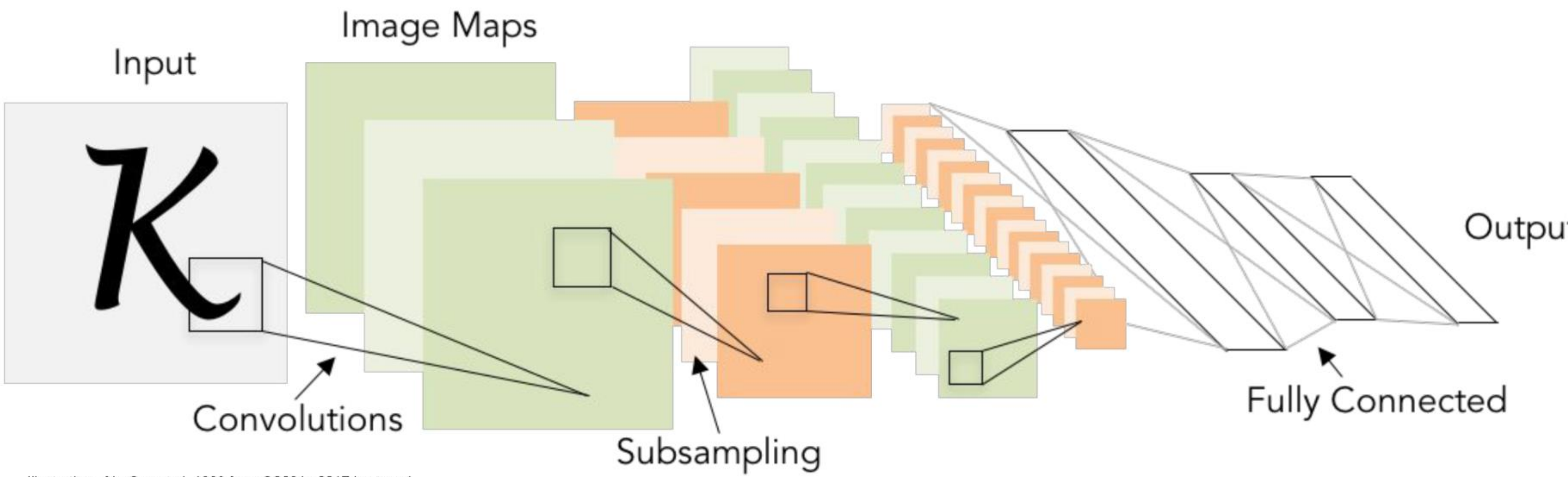


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Backpropagation: a simple example

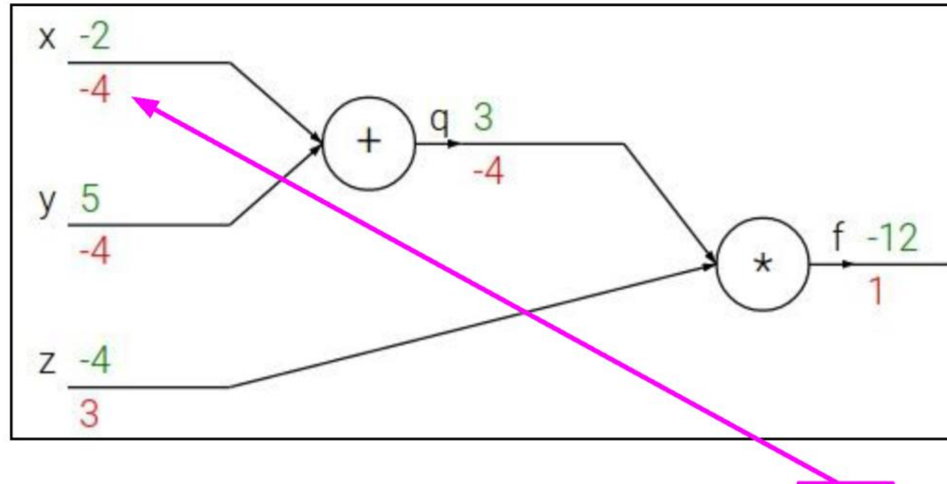
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



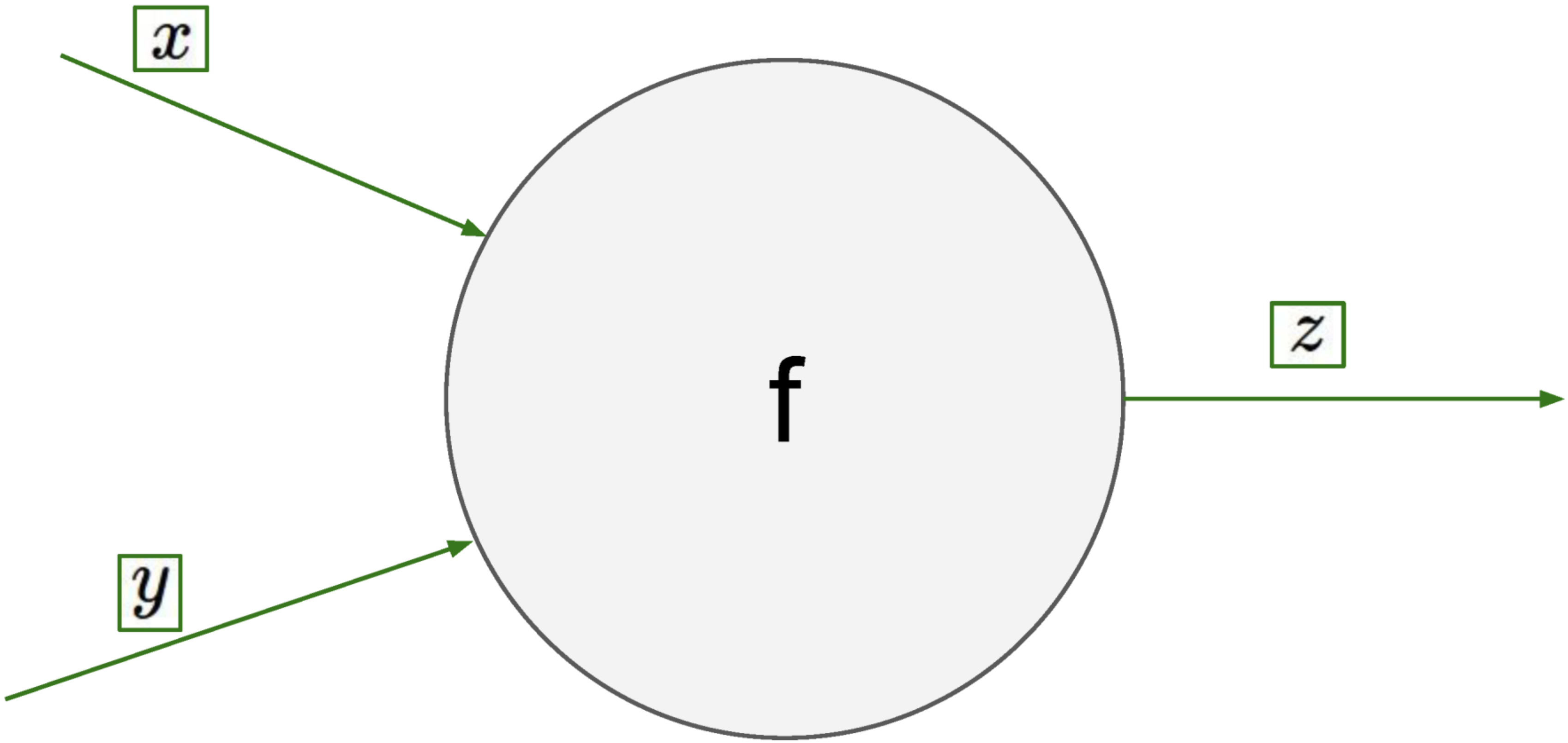
$$\frac{\partial f}{\partial x}$$

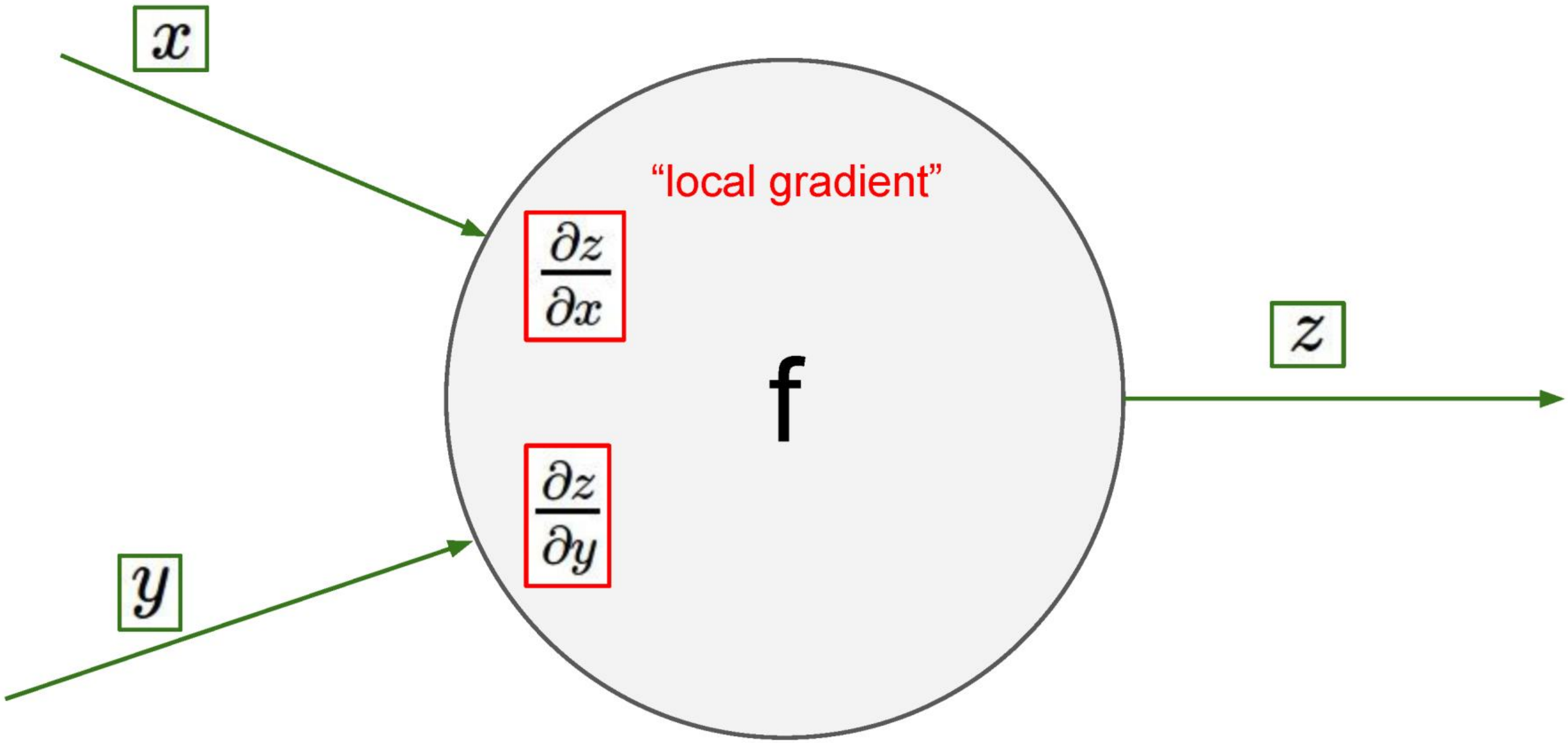
Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

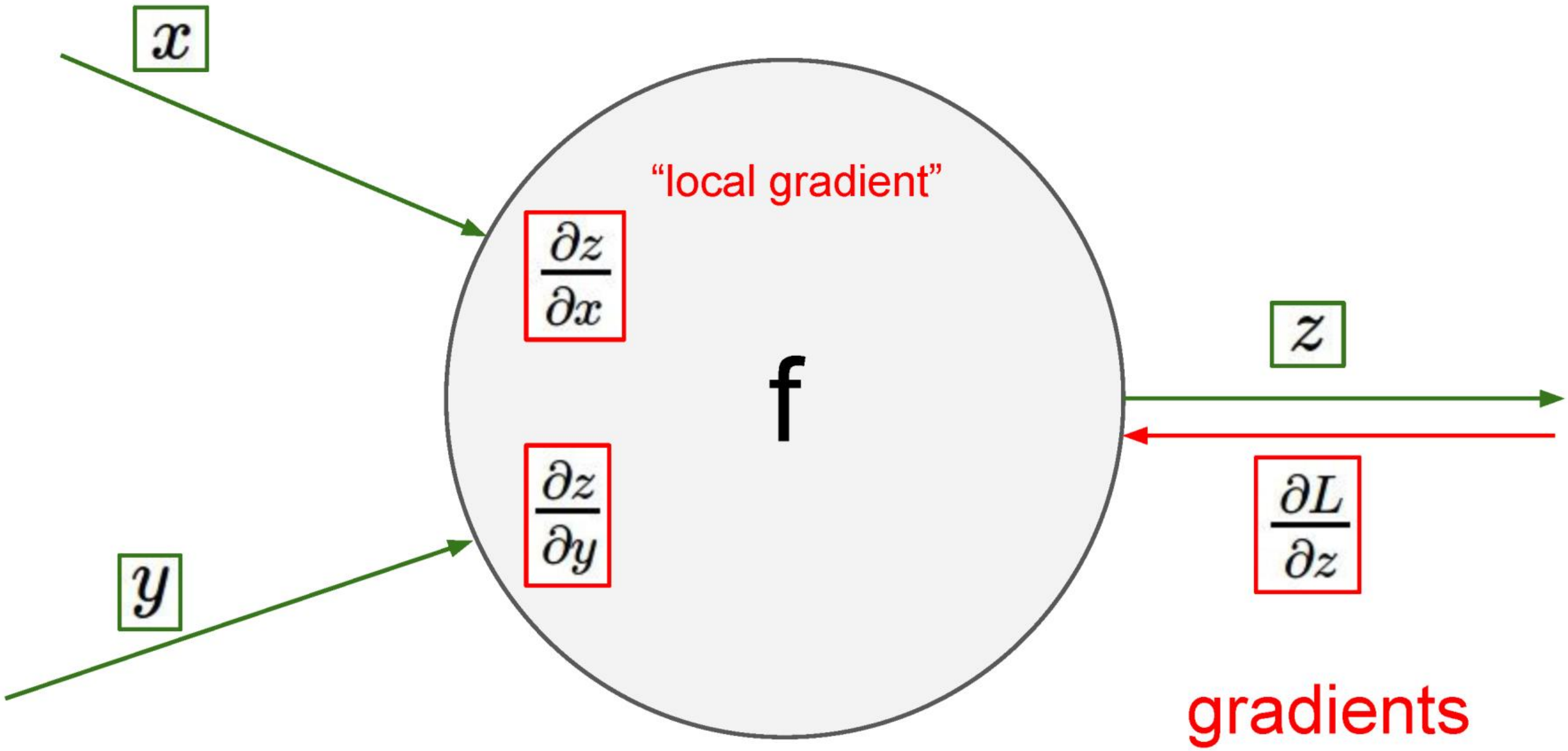
Upstream gradient

Local gradient

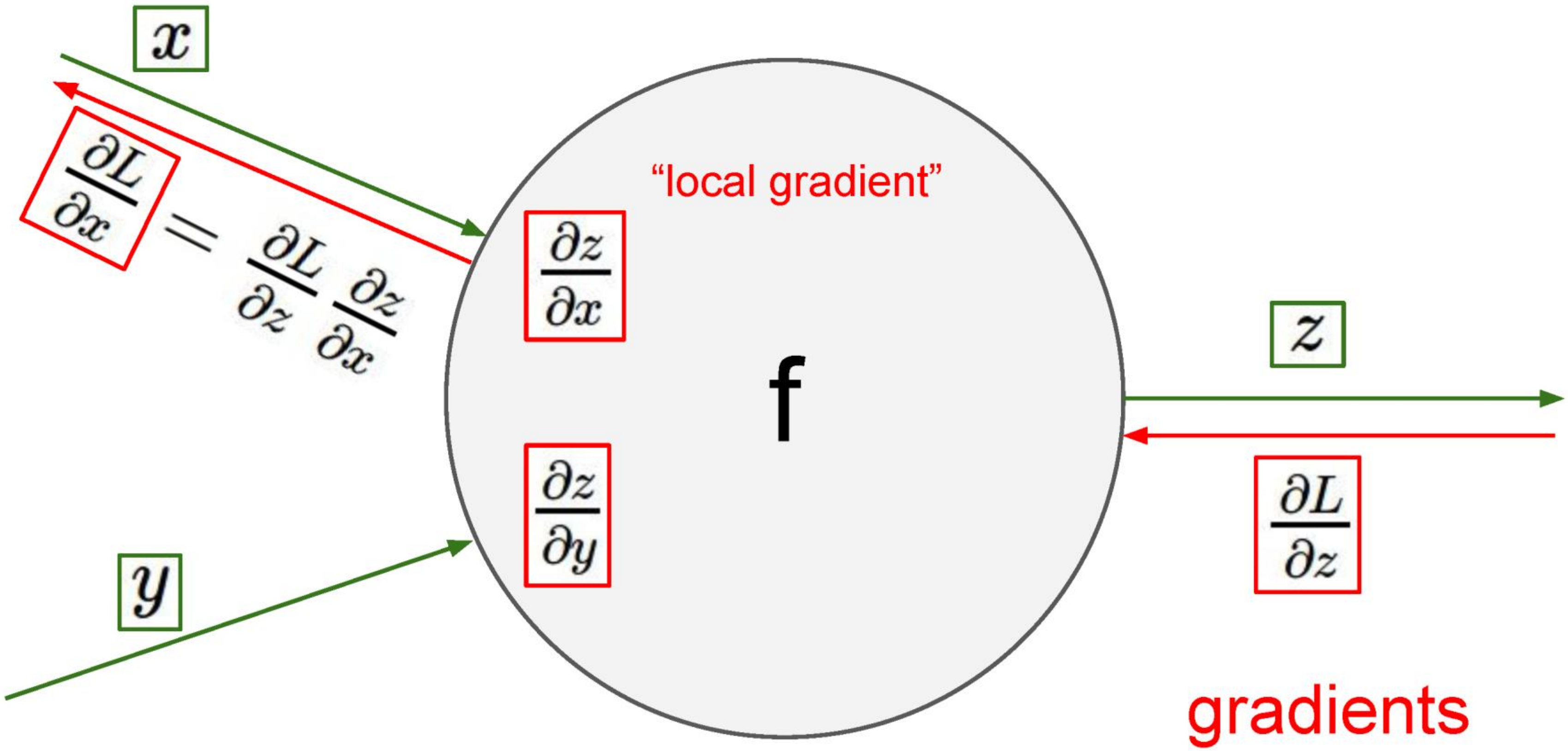

x
y
f
z



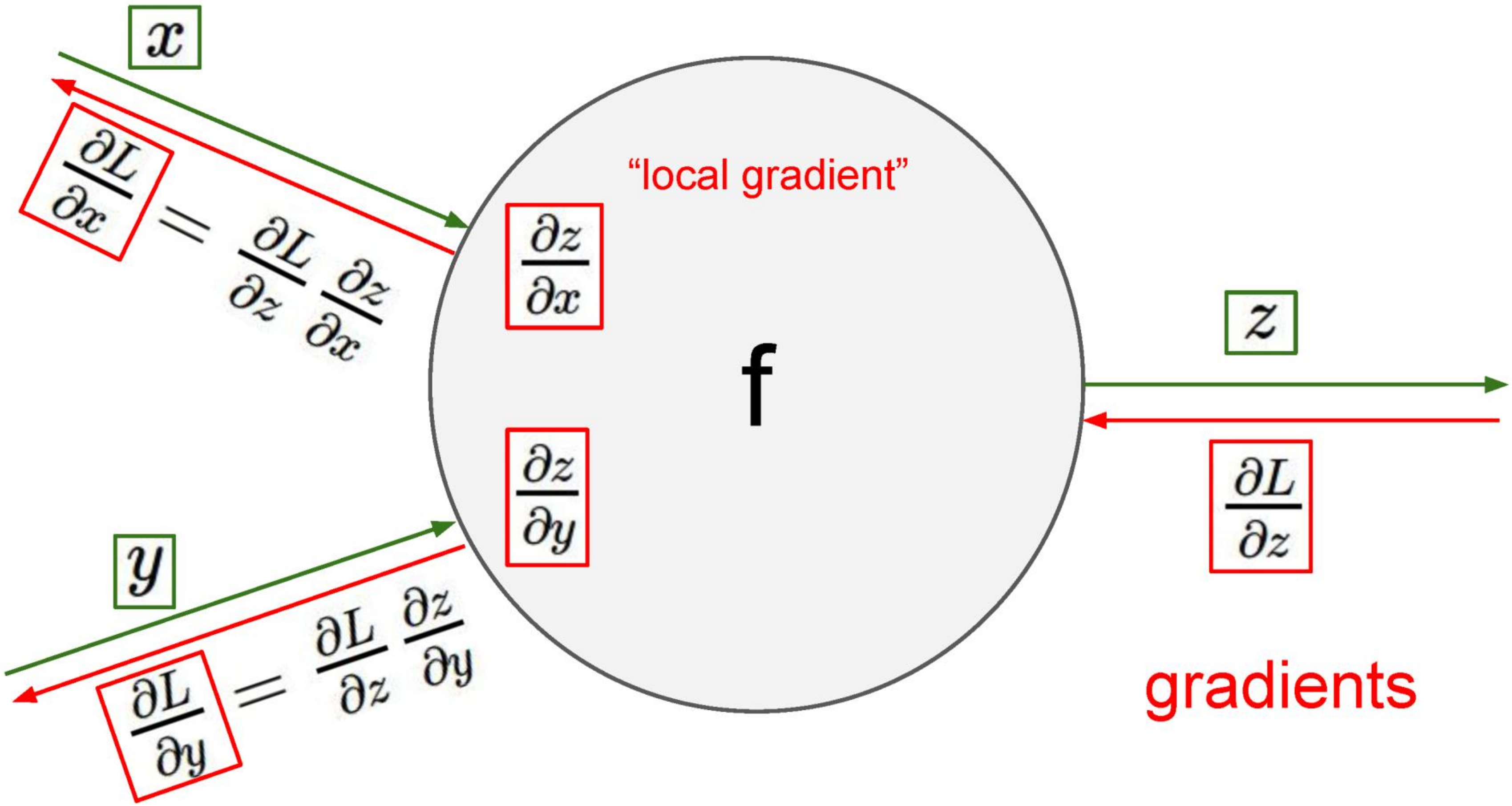
x
"local gradient"
$\frac{\partial z}{\partial x}$
f
z
$\frac{\partial z}{\partial y}$
y



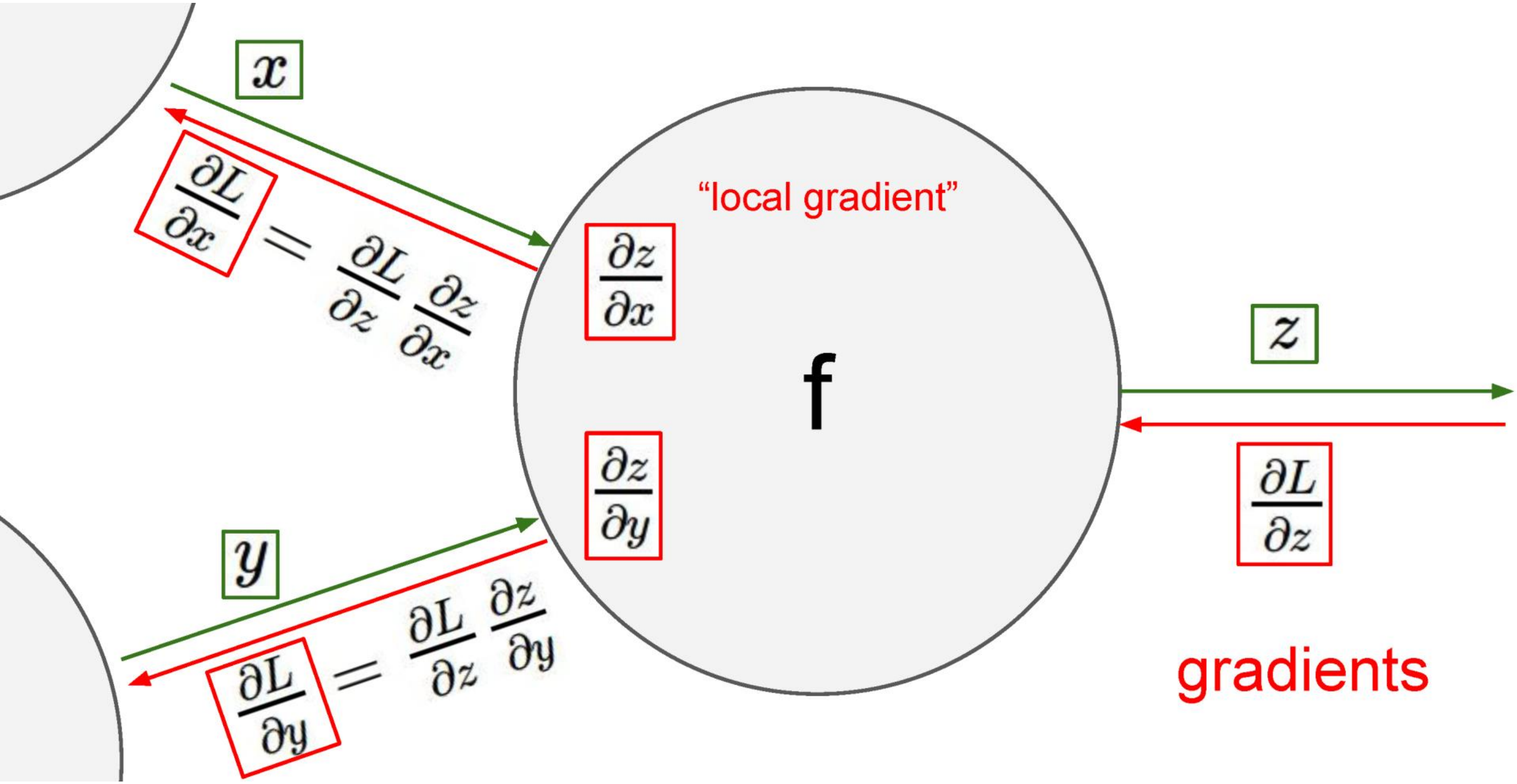
x
"local gradient"
$\frac{\partial z}{\partial x}$
f
z
$\frac{\partial z}{\partial y}$
$\frac{\partial L}{\partial z}$
y
gradients



x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
z
$\frac{\partial z}{\partial y}$
$\frac{\partial L}{\partial z}$
y
gradients



x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients



x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial y}$
gradients

Best practices for training networks

- Preprocess the data (subtract mean, sub-crops)
- Initialize weights carefully
- Use Dropout
- Use SGD + Momentum
- Fine-tune from ImageNet
- Babysit the network as it trains

Transfer learning

Takeaway for your projects and beyond:
Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big ConvNet there
2. Transfer learn to your dataset

Deep learning frameworks provide a "Model Zoo" of pretrained models so you don't need to train your own

Caffe: https://github.com/BVLC/caffe/wiki/Model-Zoo
TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision

Questions?

Good luck!