

# CS5670: Computer Vision

## Course review

Class	Date	Topic/notes	Readings	Assignments, etc.
February				
0	8	Introduction and Overview [ppt pdf] 	Szeliski 1	
1	10	Image filtering [ppt pdf] 	Szeliski 3.1	
2	15	Image filtering and edge detection [ppt pdf] 	Szeliski 3.2	PA1 Released
3	17	Image Resampling [ppt pdf] 	Szeliski 3.4, 2.3.1	
4	22	Feature Detection [ppt pdf] 	Szeliski 4.1	
5	24	Feature Invariance [ppt pdf] 	Szeliski 4.1	PA1 due on Thursday
March				
6	1	Feature Descriptors [ppt pdf] Feature Matching [ppt pdf] 	Szeliski 4.1	PA2 Released
7	3	Image Transformations [ppt pdf] 	Szeliski 3.6	
8	8	Image Alignment [ppt pdf] 	Szeliski 6.1	PA2 due on Friday
9	15	RANSAC [ppt pdf] 	Szeliski 6.1	Take-home midterm exam release;
10	17	Cameras [ppt pdf] 	Szeliski 2.1.3- 2.1.6	Take-home midterm exam due on Friday;
11	22	Panoramas [ppt pdf] 	Szeliski 9	PA3 Released

# 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

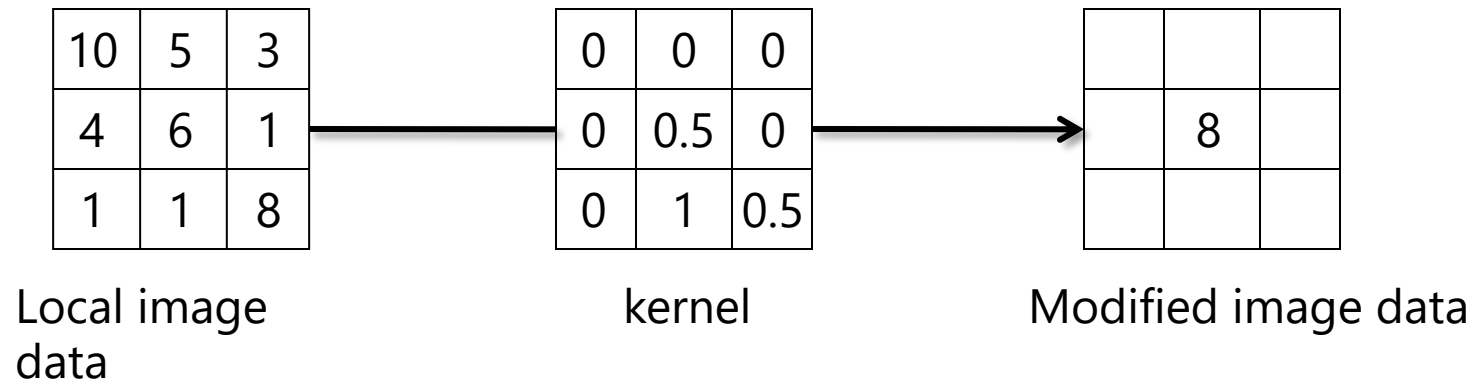
- Neural networks
- Convolutional neural networks
  - Architectural components: convolutional layers, pooling layers, fully connected layers
  - Training CNNs
- Ethical considerations in computer vision
- Neural Rendering (NeRF, positional encoding, etc)
- Generative methods: GANS and diffusion

# 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")



# Convolution

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

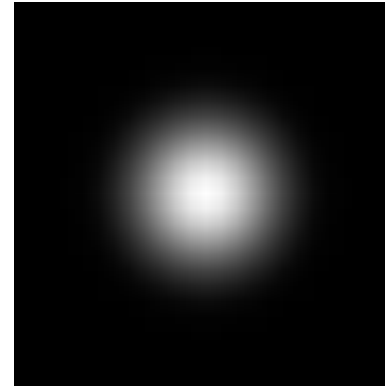
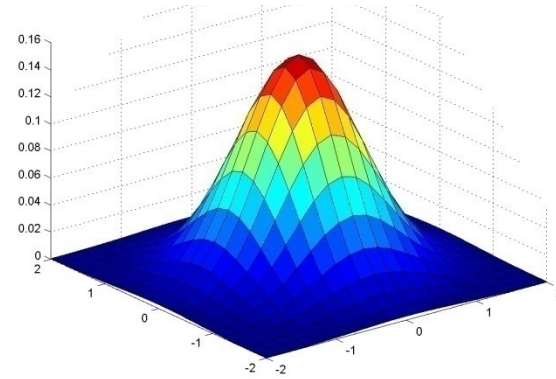
$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i - u, j - v]$$

This is called a **convolution** operation:

$$G = H * F$$

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

# Gaussian Kernel

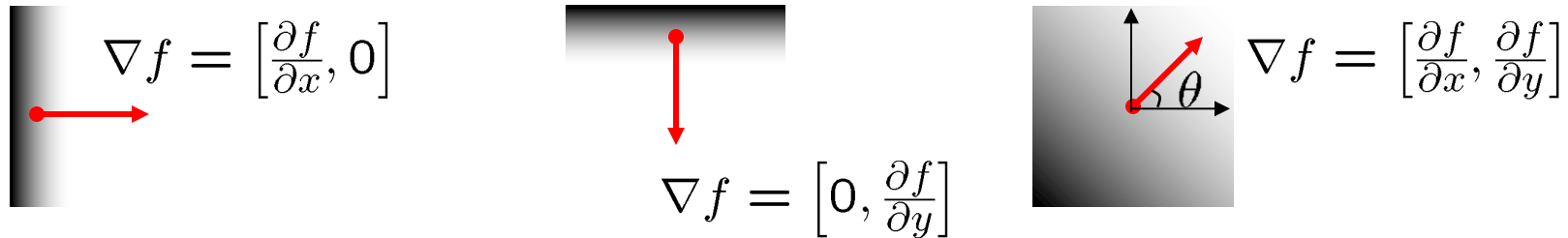


$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

# 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



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?

# Finding edges



gradient magnitude

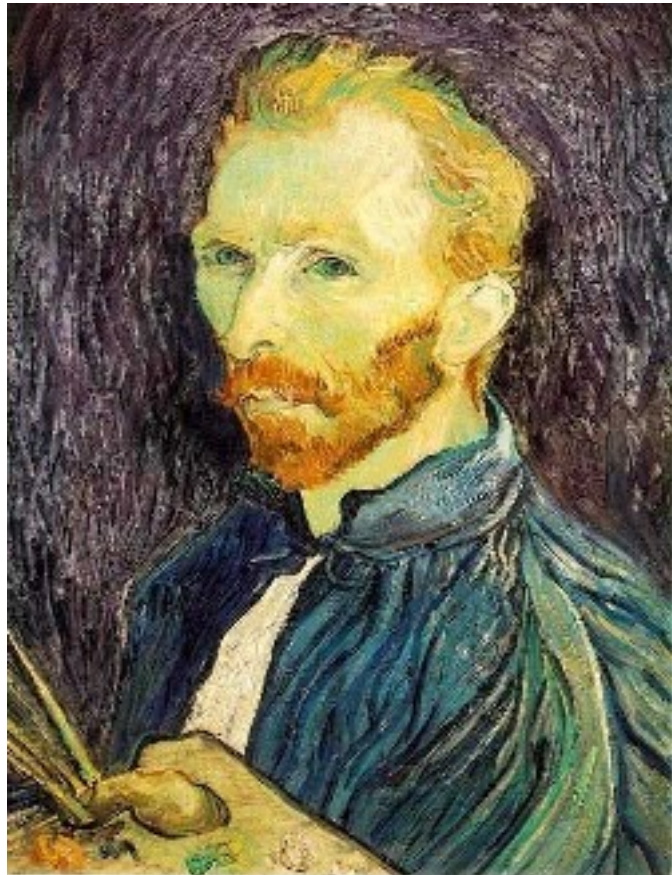
# Finding edges



thinning

(non-maximum suppression)

# Image sub-sampling



1/2



1/4 (2x zoom)

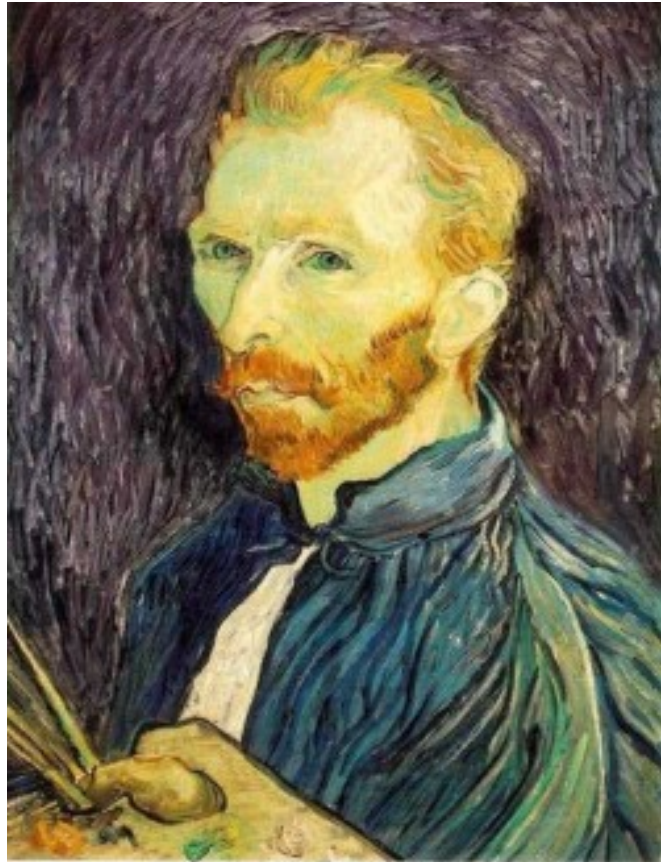


1/8 (4x zoom)

Why does this look so crufty?

Source: S. Seitz

# Subsampling with Gaussian pre-filtering



Gaussian 1/2



G 1/4

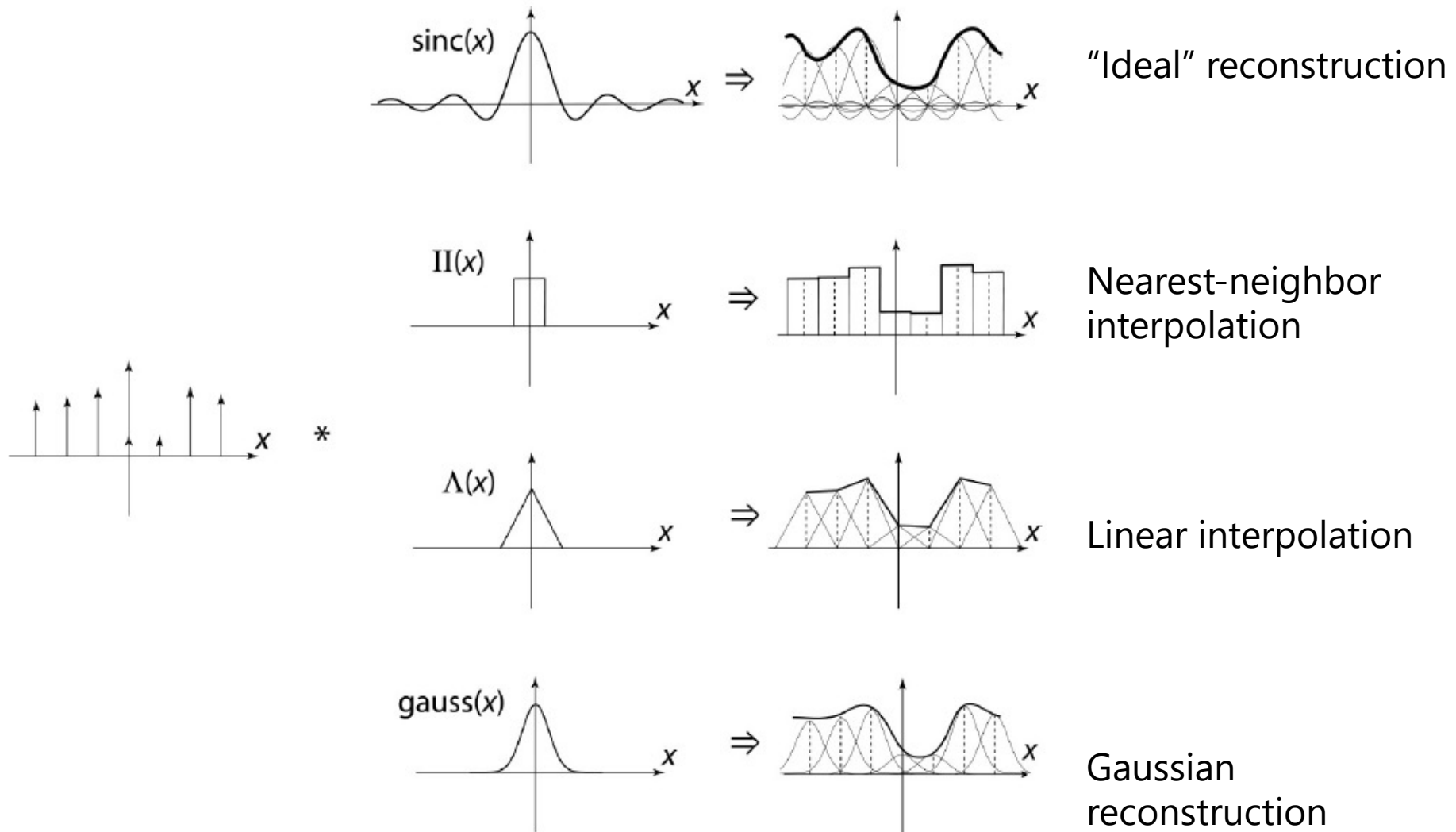


G 1/8

- Solution: filter the image, *then* subsample



# Image interpolation



# Image interpolation

Original image  x  
10



Nearest-neighbor interpolation



Bilinear interpolation



Bicubic interpolation

# The second moment matrix

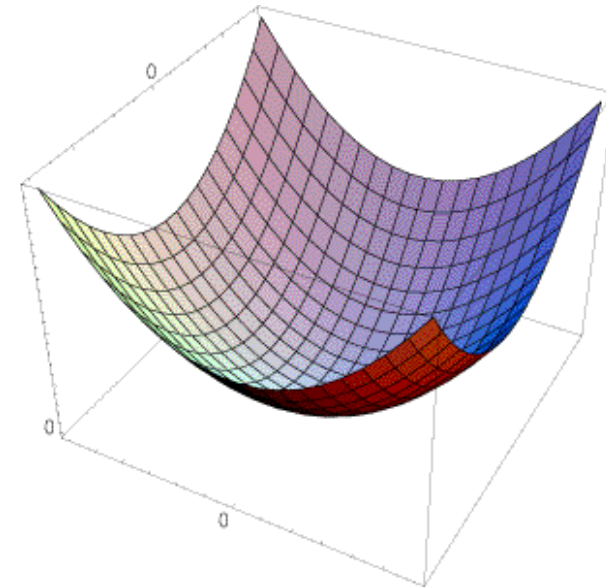
The surface  $E(u,v)$  is locally approximated by a quadratic form.

$$E(u, v) \approx Au^2 + 2Buv + Cv^2$$
$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

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



# The Harris operator

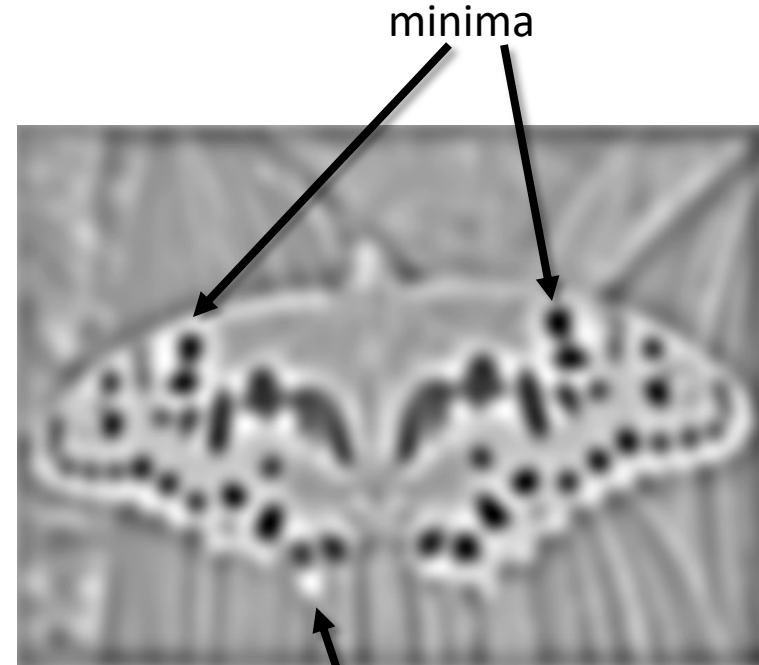
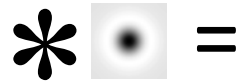
$\lambda_{\min}$  is a variant of the “Harris operator” for feature detection

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$
$$= \frac{\mathit{determinant}(H)}{\mathit{trace}(H)}$$

- The *trace* is the sum of the diagonals, i.e.,  $\mathit{trace}(H) = h_{11} + h_{22}$
- Very similar to  $\lambda_{\min}$  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

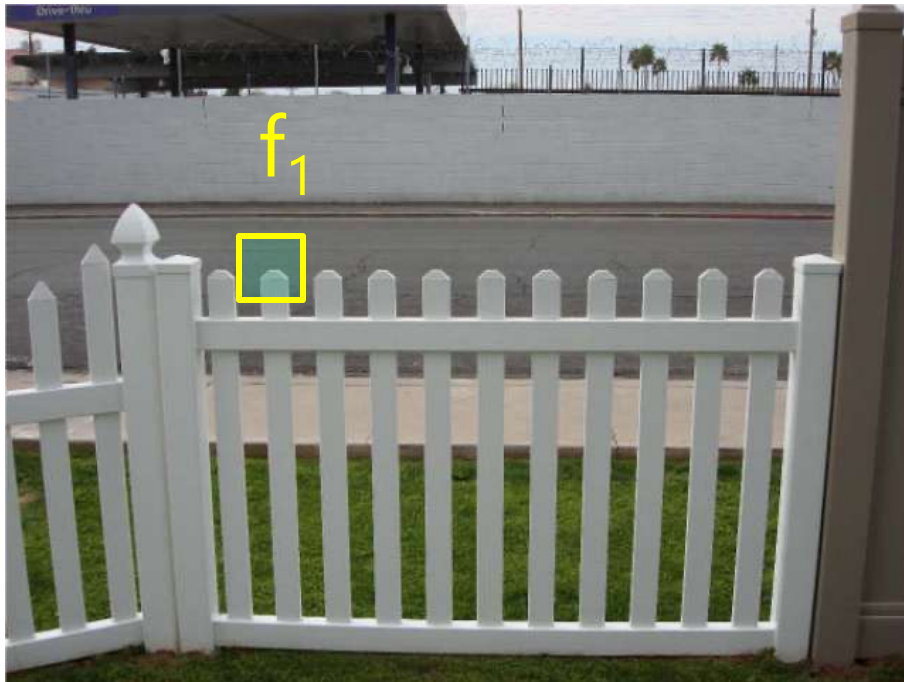


sigma = 11.9912

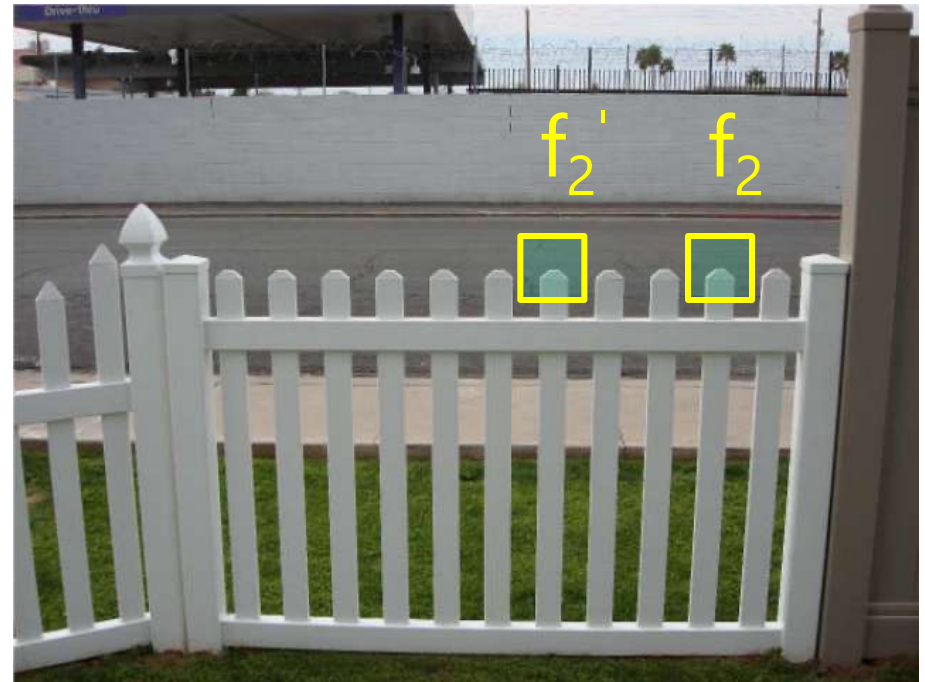
# Feature distance

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<sup>nd</sup> best SSD match to  $f_1$  in  $I_2$
  - gives large values for ambiguous matches



$I_1$



$I_2$

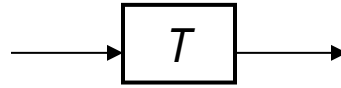
# 2D Geometry



# Parametric (global) warping



$$\mathbf{p} = (x, y)$$



$$\mathbf{p}' = (x', y')$$

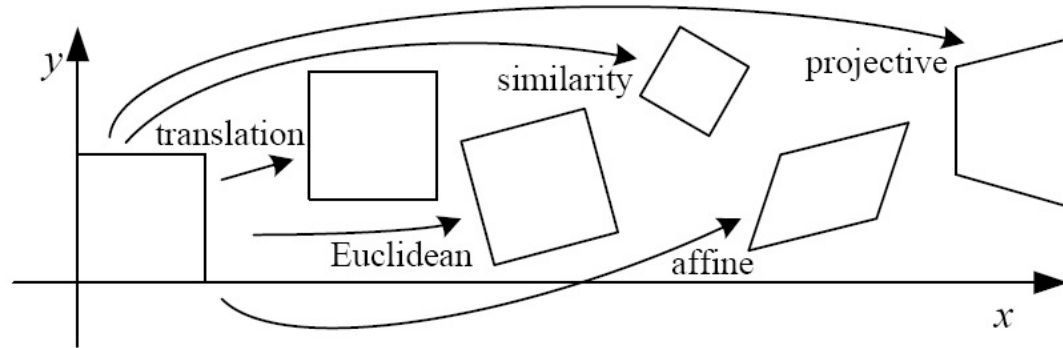
- Transformation  $T$  is a coordinate-changing machine:


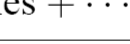
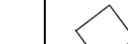


$$\mathbf{p}' = T(\mathbf{p})$$

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

$$\mathbf{p}' = \mathbf{T}\mathbf{p} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

# 2D image transformations



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

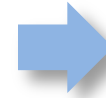
These transformations are a nested set of groups

- Closed under composition and inverse is a member

# Projective Transformations aka Homographies aka Planar Perspective Maps

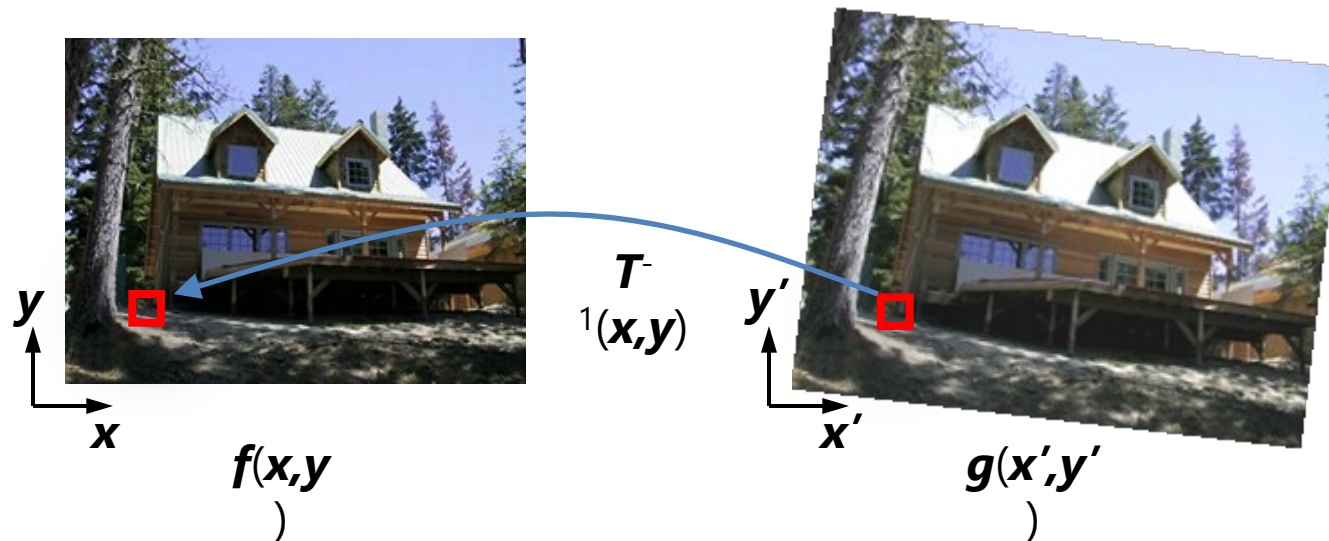
$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

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

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

**A**  
 $2n \times 6$

**t**  
 $6 \times 1$

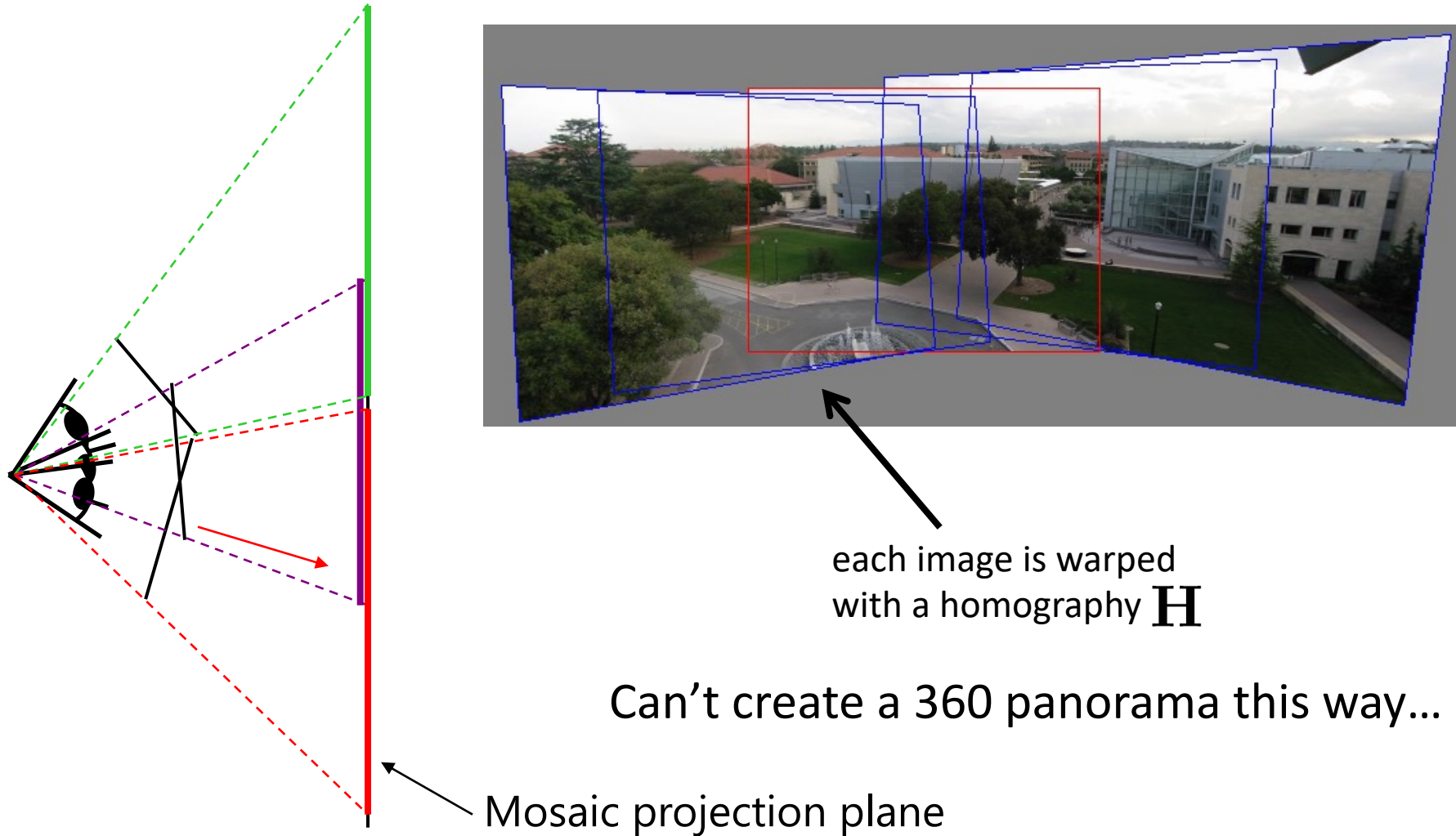
**=**

**b**  
 $2n \times 1$

# 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

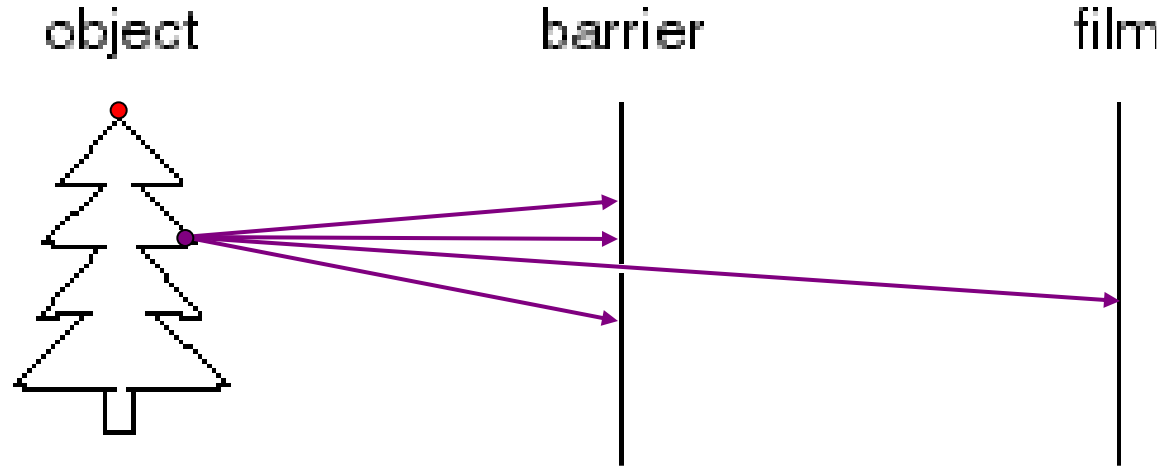
# Projecting images onto a common plane





# 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:

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

divide by third coordinate

This is known as **perspective projection**

- The matrix is the **projection matrix**

# Projection matrix

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

$$\left[ \mathbf{R} \mid \underbrace{-\mathbf{R}\mathbf{c}} \right]$$

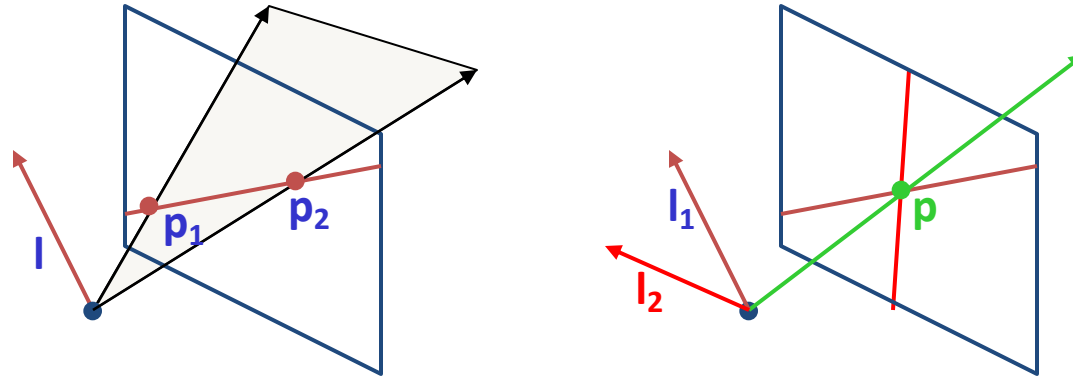
(t in book's notation)



$$\mathbf{\Pi} = \mathbf{K} \left[ \mathbf{R} \mid -\mathbf{R}\mathbf{c} \right]$$

# Point and line duality

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



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

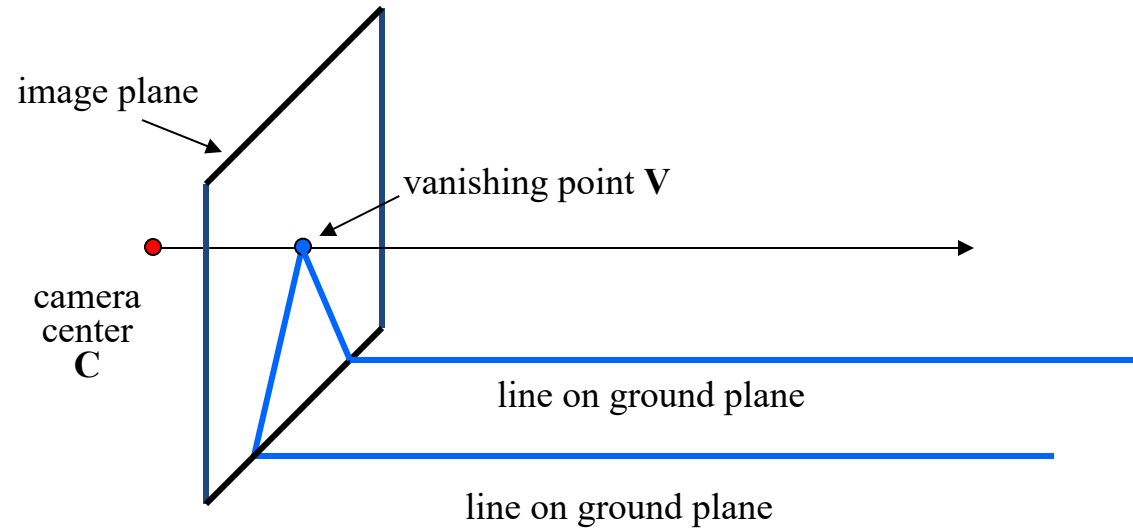
- $\mathbf{l}$  is  $\perp$  to  $\mathbf{p}_1$  and  $\mathbf{p}_2 \Rightarrow \mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2$
- $\mathbf{l}$  can be interpreted as a *plane normal*

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

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

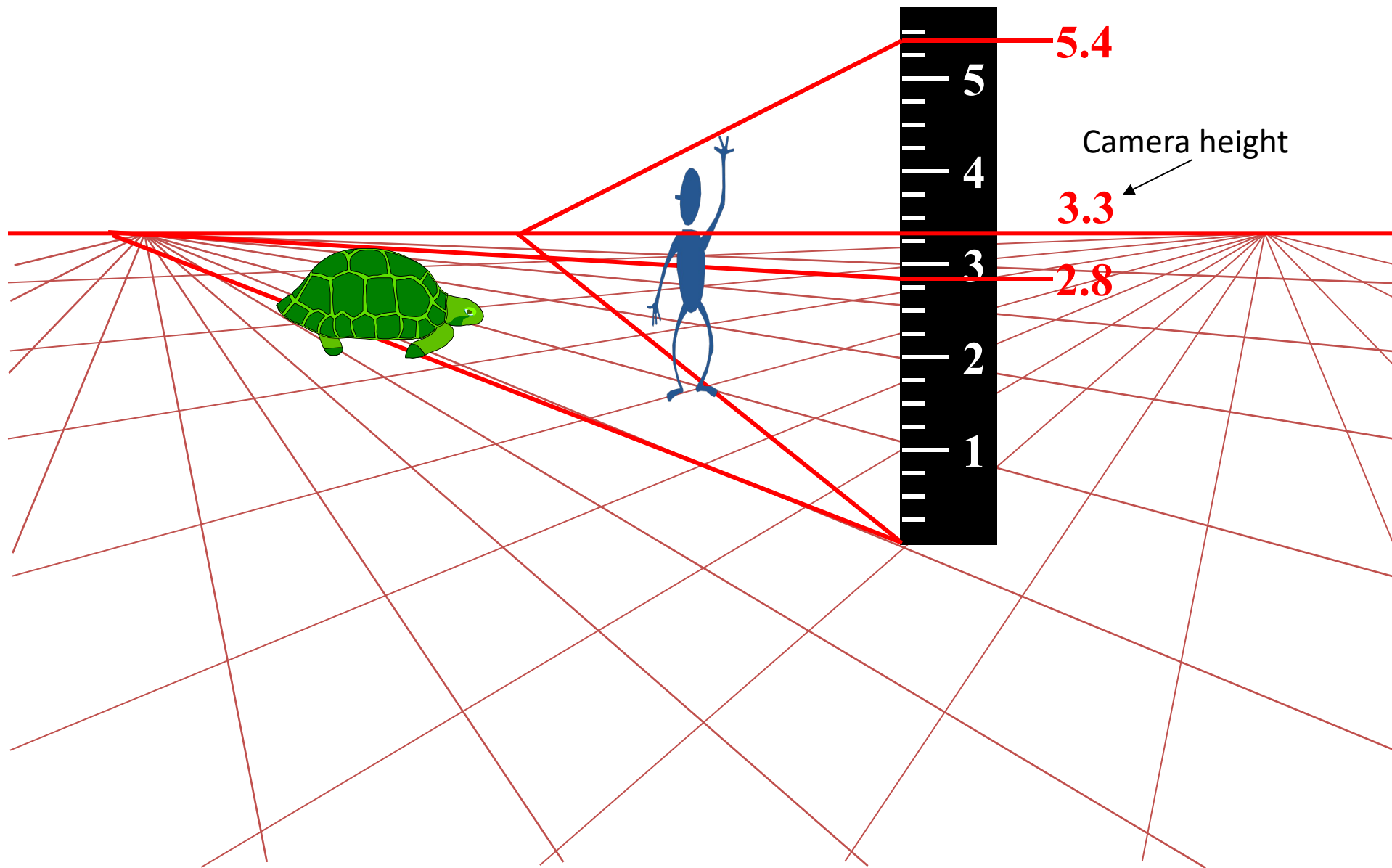
Points and lines are *dual* in projective space

# Vanishing points

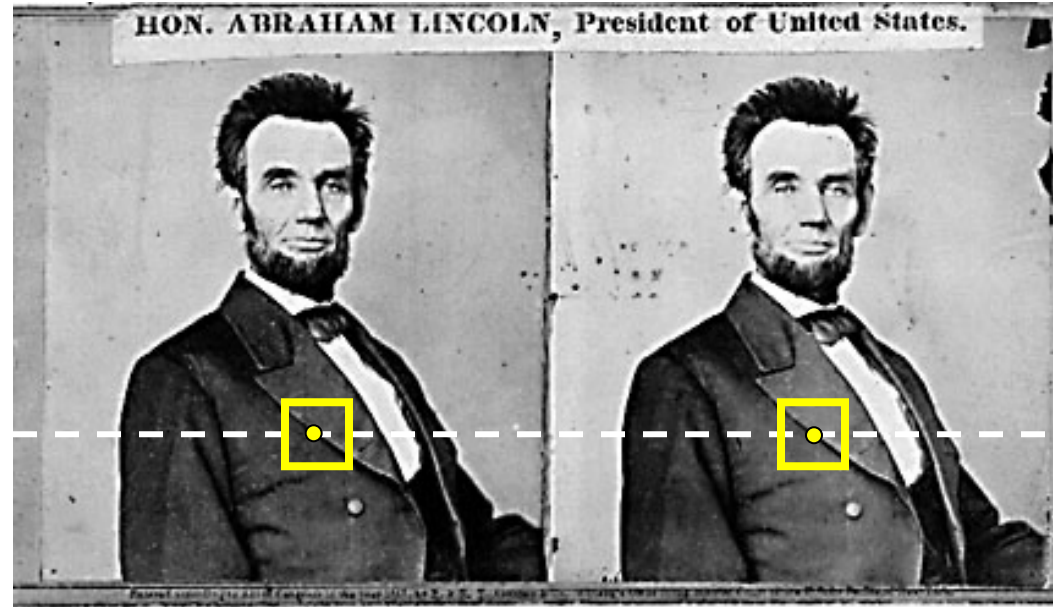


- Properties
  - Any two parallel lines (in 3D) have the same vanishing point  $\mathbf{v}$
  - The ray from  $\mathbf{C}$  through  $\mathbf{v}$  is parallel to the lines
  - An image may have more than one vanishing point
    - in fact, every image point is a potential vanishing point

# 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

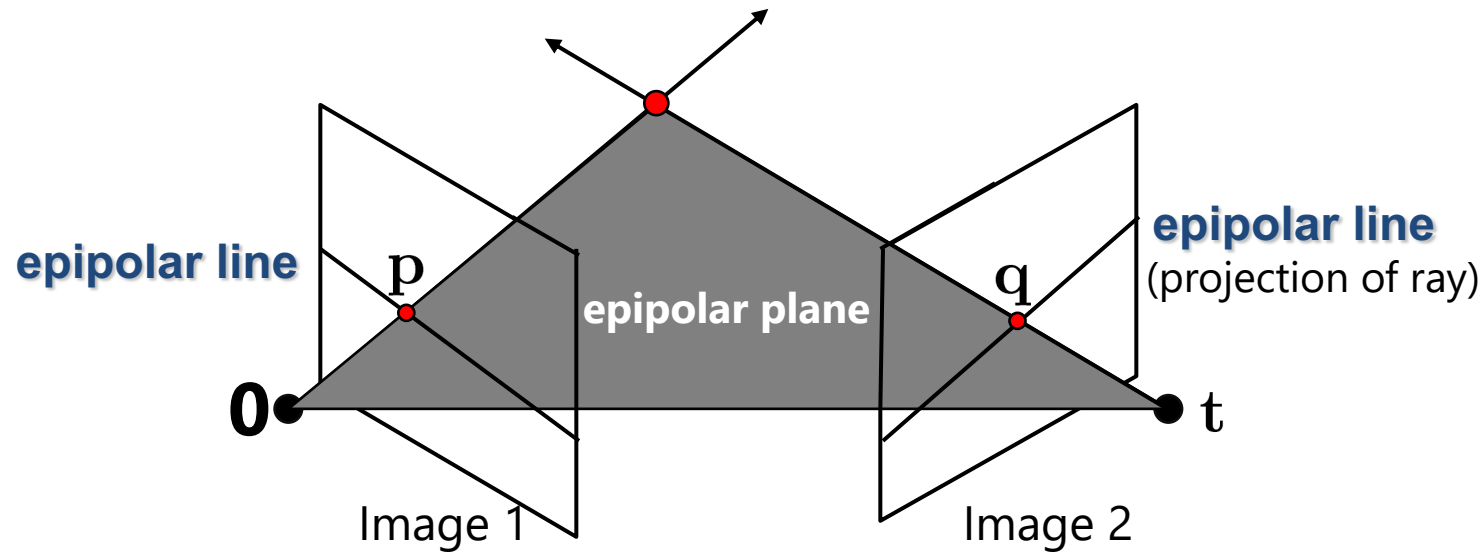
- Better objective function

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

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

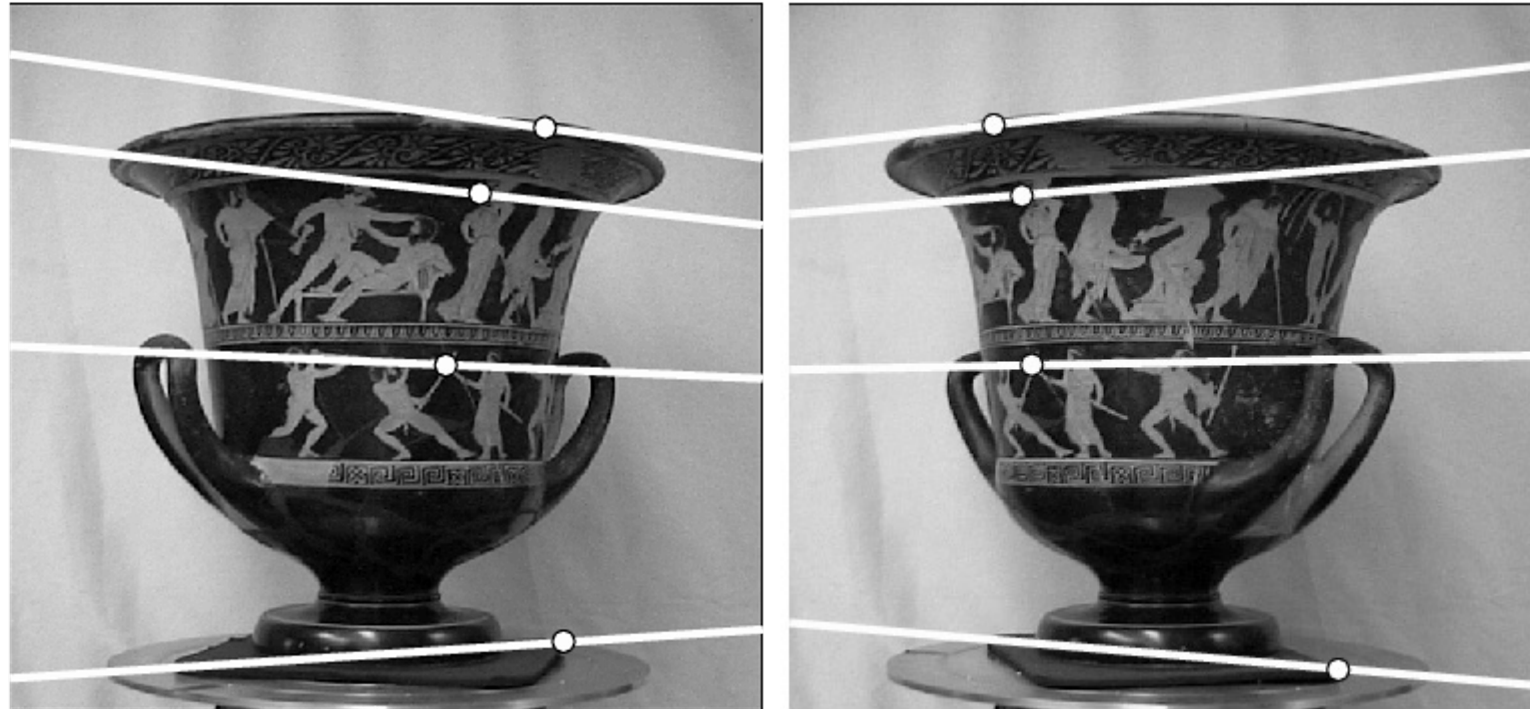
Adjacent pixels should (usually) move about the same amount

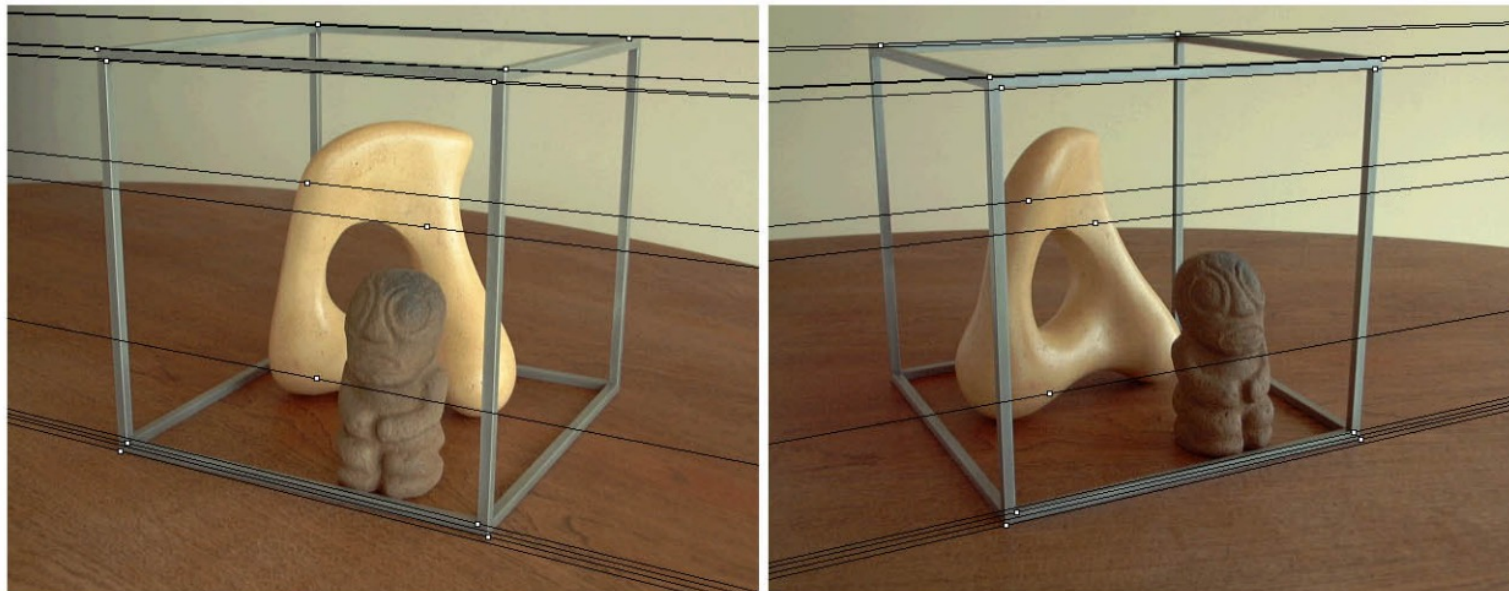
# Fundamental matrix



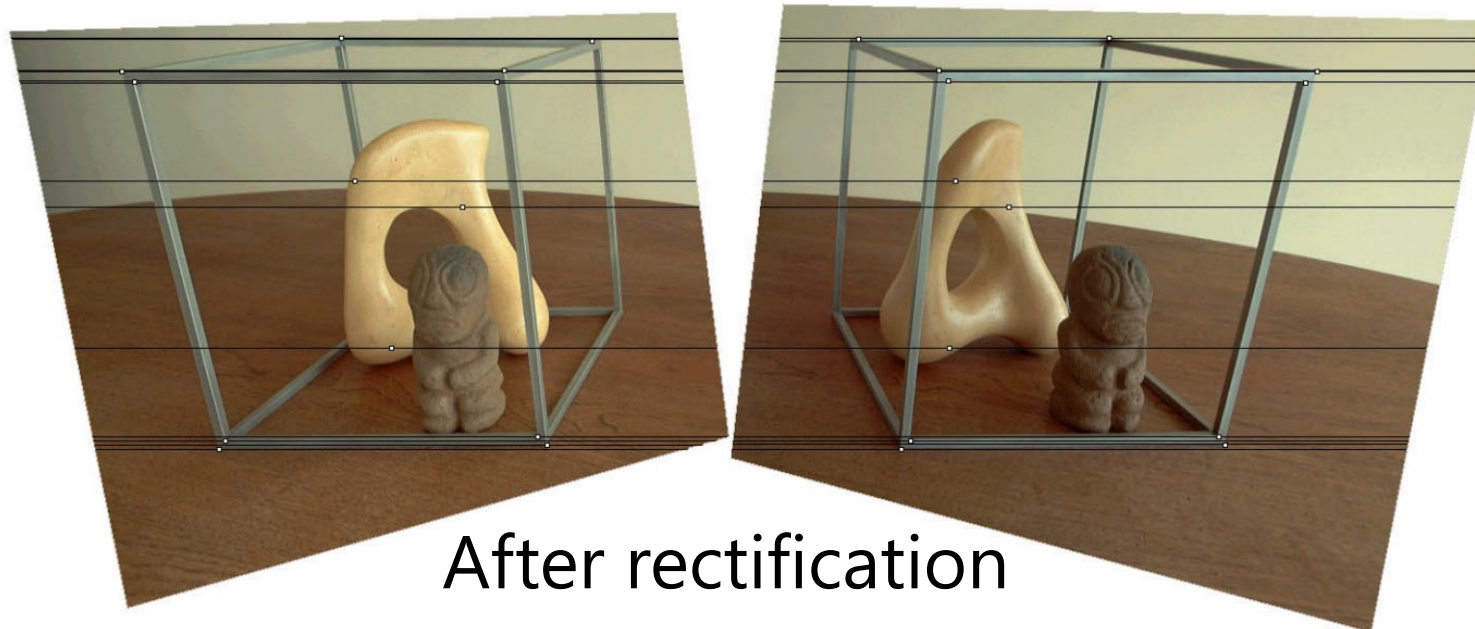
- This *epipolar geometry* of two views is described by a Very Special  $3 \times 3$  matrix  $\mathbf{F}$ , called the *Fundamental matrix*
- $\mathbf{F}$  maps (homogeneous) *points* in image 1 to *lines* in image 2!
- The epipolar line (in image 2) of point  $\mathbf{p}$  is  $\mathbf{F}\mathbf{p}$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

# Epipolar geometry example





Original stereo pair



After rectification

# Estimating F – 8-point algorithm

- The fundamental matrix F is defined by

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

for any pair of matches  $\mathbf{x}$  and  $\mathbf{x}'$  in two images.

- Let  $\mathbf{x} = (u, v, 1)^T$  and  $\mathbf{x}' = (u', v', 1)^T$ ,  $\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$   
each match gives a linear equation

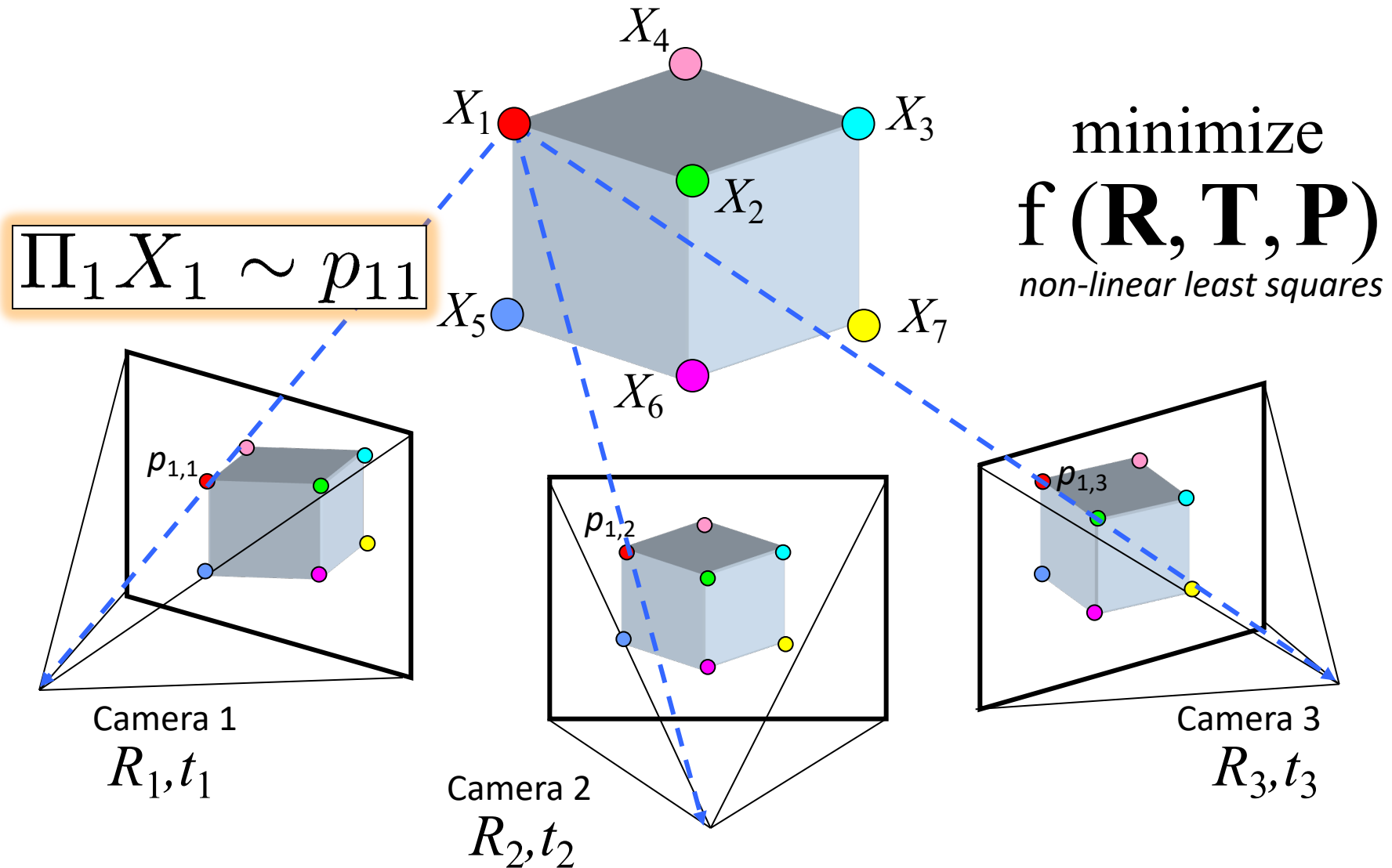
$$uu' f_{11} + vu' f_{12} + u' f_{13} + uv' f_{21} + vv' f_{22} + v' f_{23} + uf_{31} + vf_{32} + f_{33} = 0$$

# 8-point algorithm

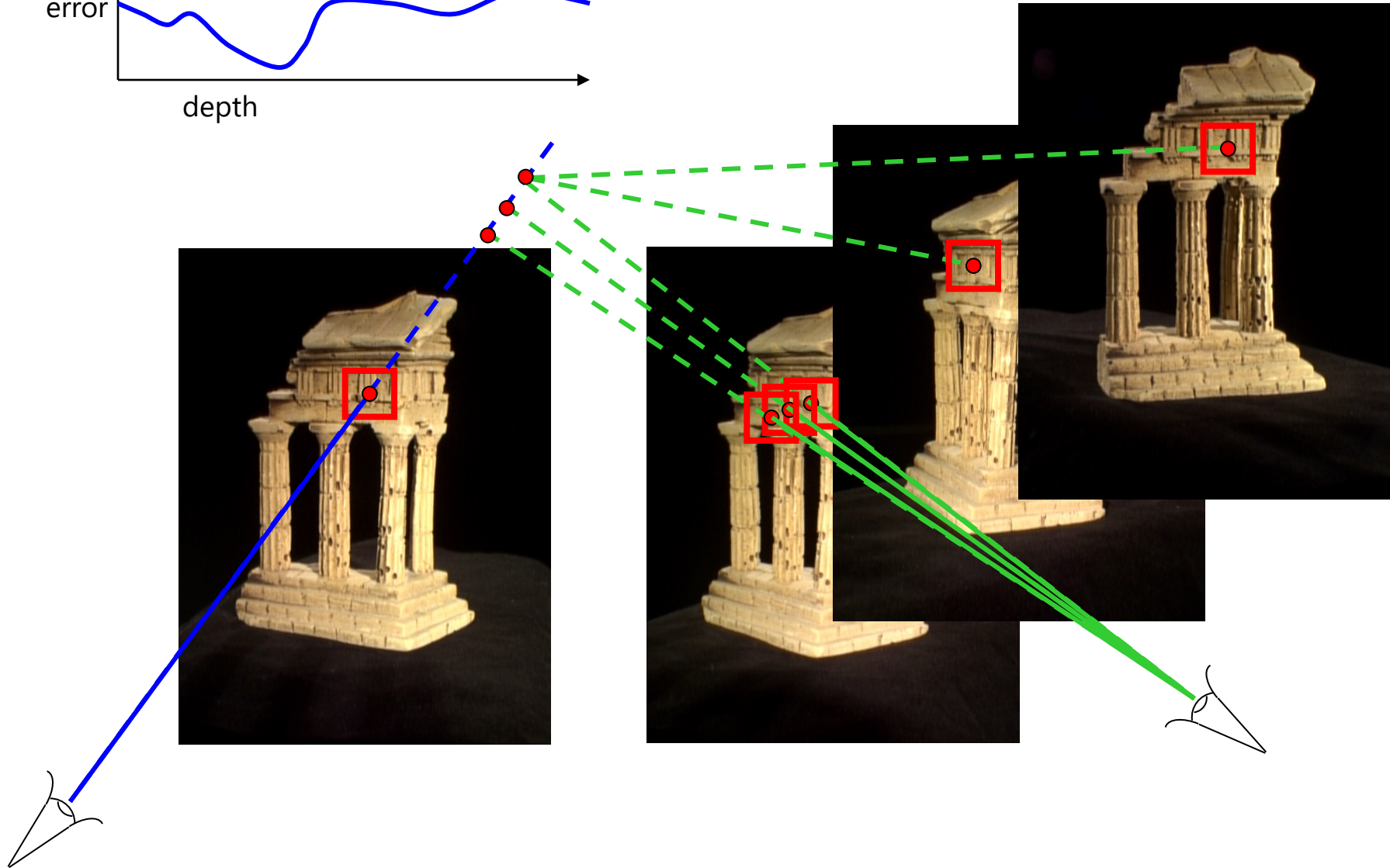
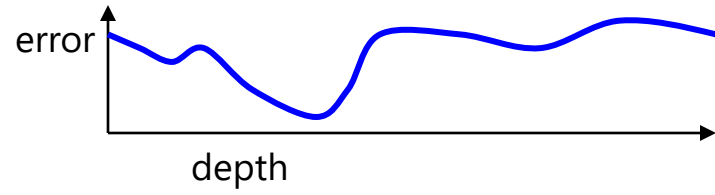
$$\begin{bmatrix}
 u_1 u_1' & v_1 u_1' & u_1' & u_1 v_1' & v_1 v_1' & v_1' & u_1 & v_1 & 1 \\
 u_2 u_2' & v_2 u_2' & u_2' & u_2 v_2' & v_2 v_2' & v_2' & u_2 & v_2 & 1 \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 u_n u_n' & v_n u_n' & u_n' & u_n v_n' & v_n v_n' & v_n' & u_n & v_n & 1
 \end{bmatrix}
 \begin{bmatrix}
 f_{11} \\
 f_{12} \\
 f_{13} \\
 f_{21} \\
 f_{22} \\
 f_{23} \\
 f_{31} \\
 f_{32} \\
 f_{33}
 \end{bmatrix}
 = 0$$

- As with homographies, instead of solving  $Af = 0$ , we seek unit length  $f$  to minimize  $\|Af\|$ : least eigenvector of  $A^T A$ .

# Structure from motion



# Multi-view stereo





# Multiple-baseline stereo

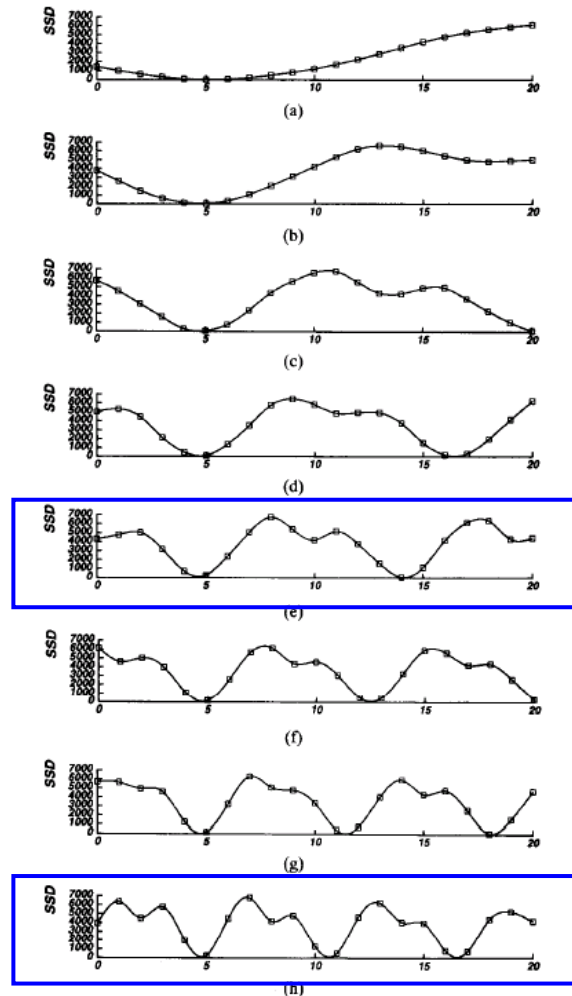


Fig. 5. SSD values versus inverse distance: (a)  $B = b$ ; (b)  $B = 2b$ ; (c)  $B = 3b$ ; (d)  $B = 4b$ ; (e)  $B = 5b$ ; (f)  $B = 6b$ ; (g)  $B = 7b$ ; (h)  $B = 8b$ . The horizontal axis is normalized such that  $8bF = 1$ .

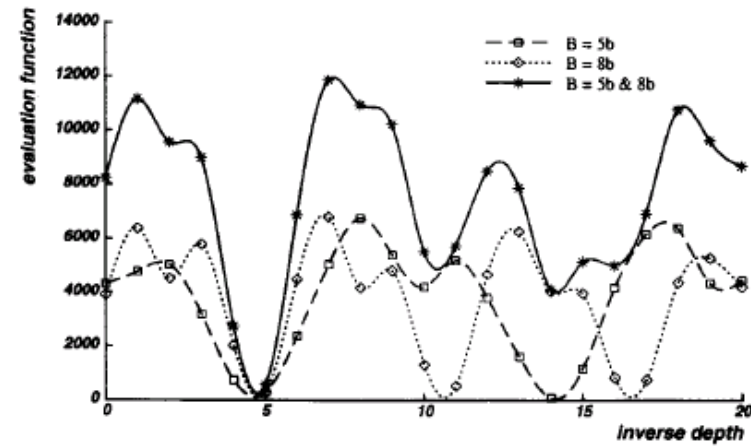


Fig. 6. Combining two stereo pairs with different baselines.

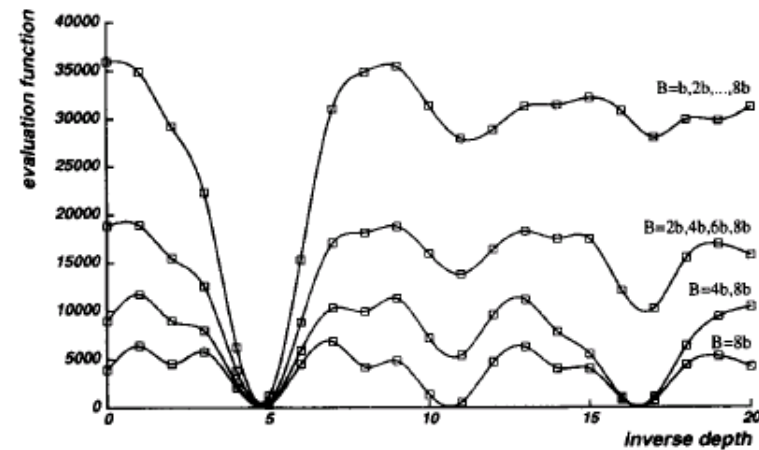
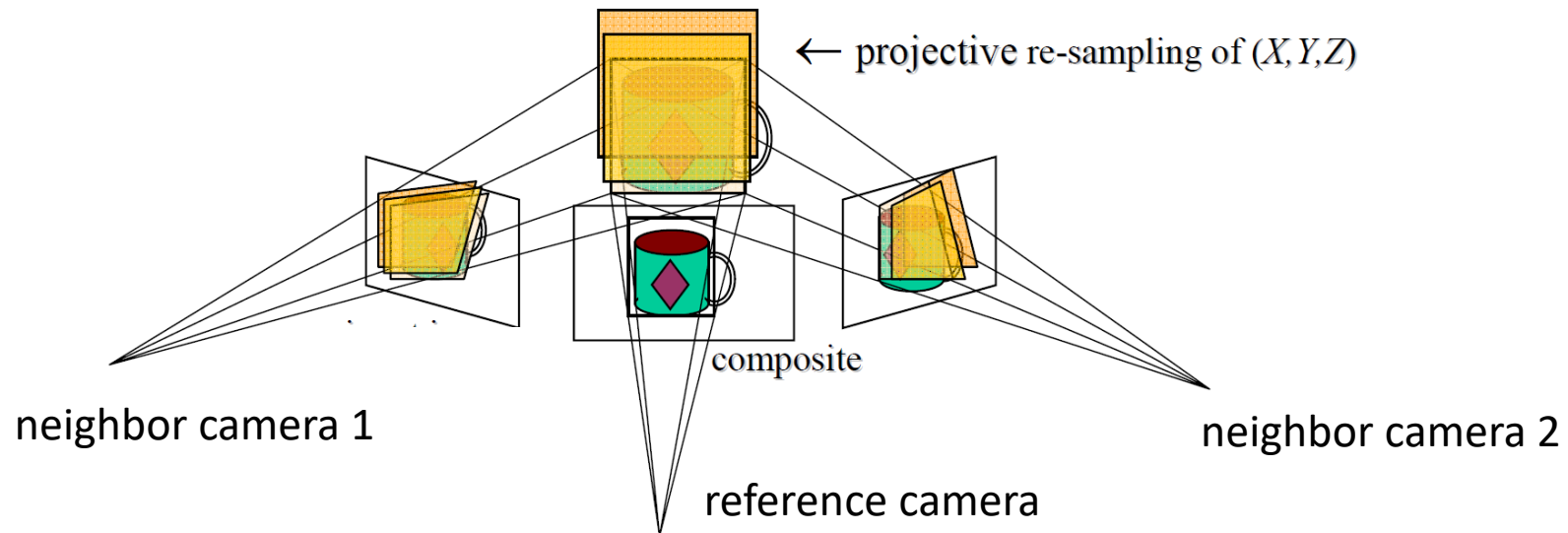


Fig. 7. Combining multiple baseline stereo pairs.

# Plane-Sweep Stereo

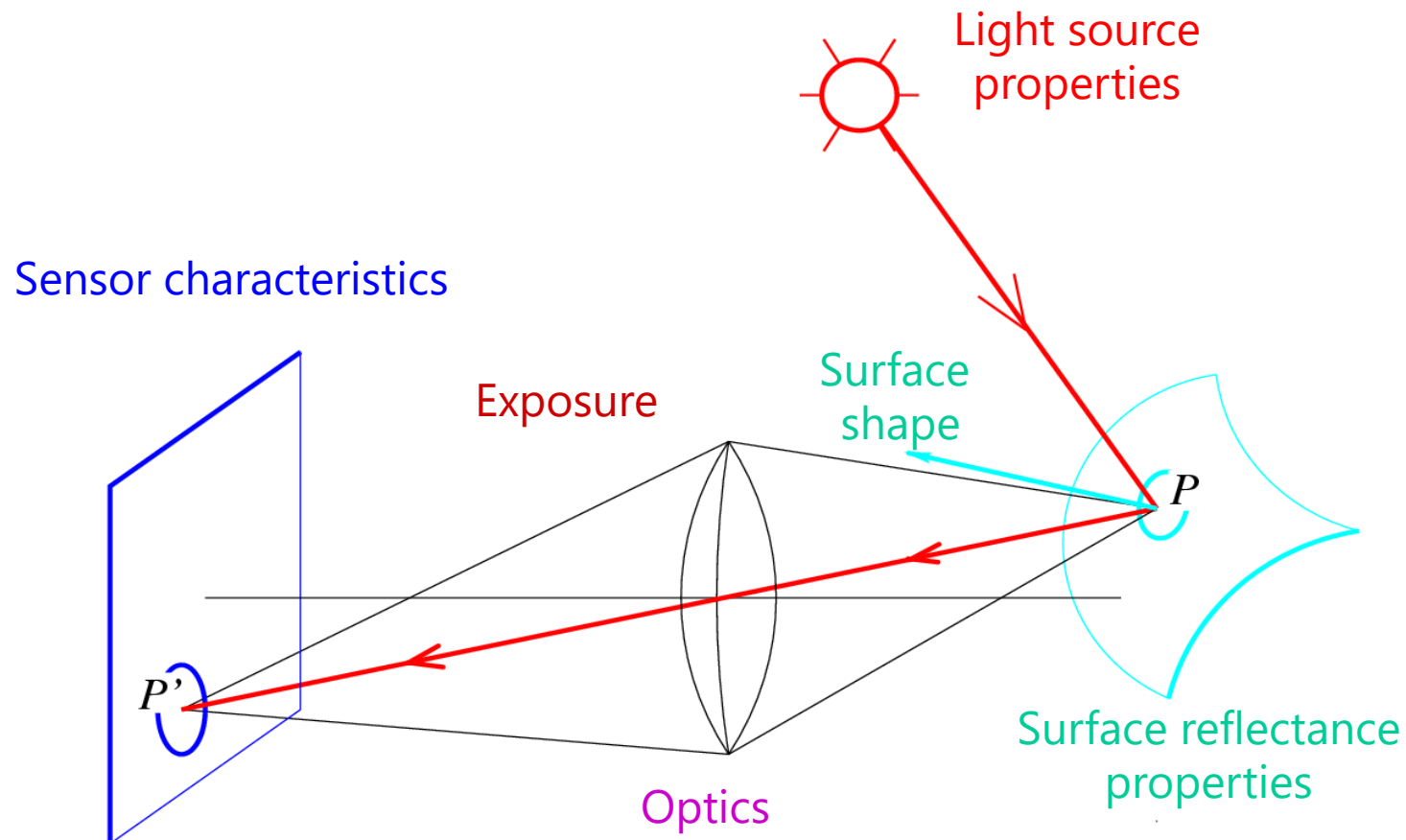
- Sweep family of planes parallel to the reference camera image plane
- Reproject neighbors onto each plane (via homography) and compare reprojections



**Light, reflectance, cameras**

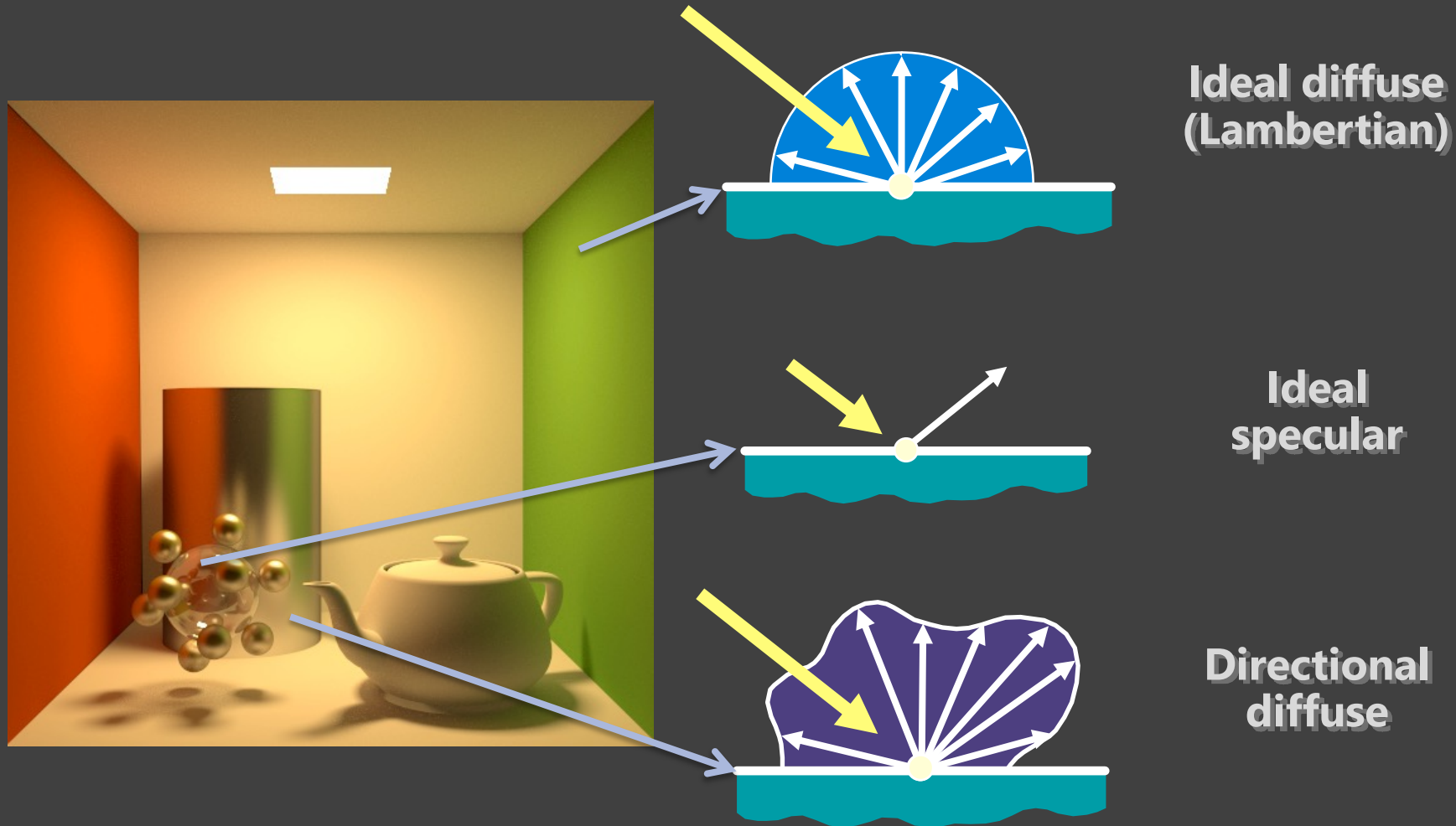
# Radiometry

- What determines the brightness of an image pixel?

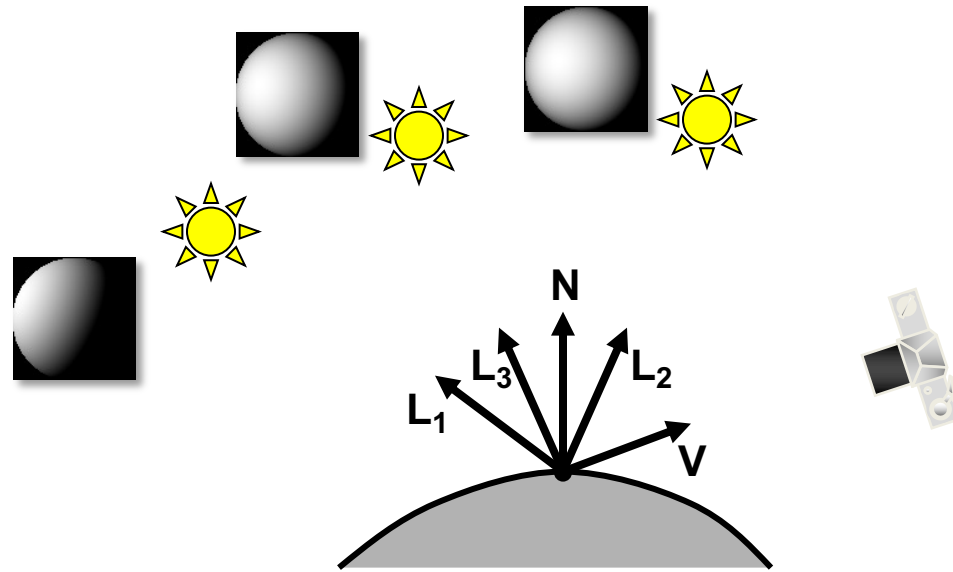


# Materials - Three Forms

In computer vision, we like Lambertian materials



# Photometric stereo



$$I_1 = k_d \mathbf{N} \cdot \mathbf{L}_1$$

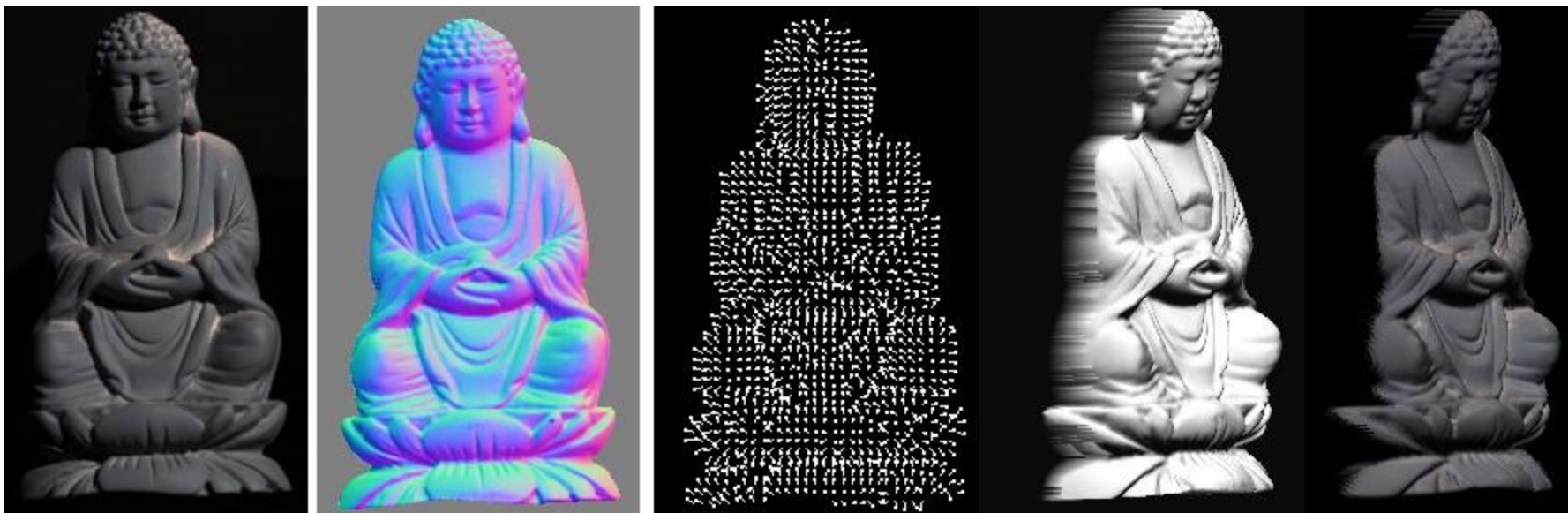
$$I_2 = k_d \mathbf{N} \cdot \mathbf{L}_2$$

$$I_3 = k_d \mathbf{N} \cdot \mathbf{L}_3$$

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 / Deep Learning**



# Image Classification

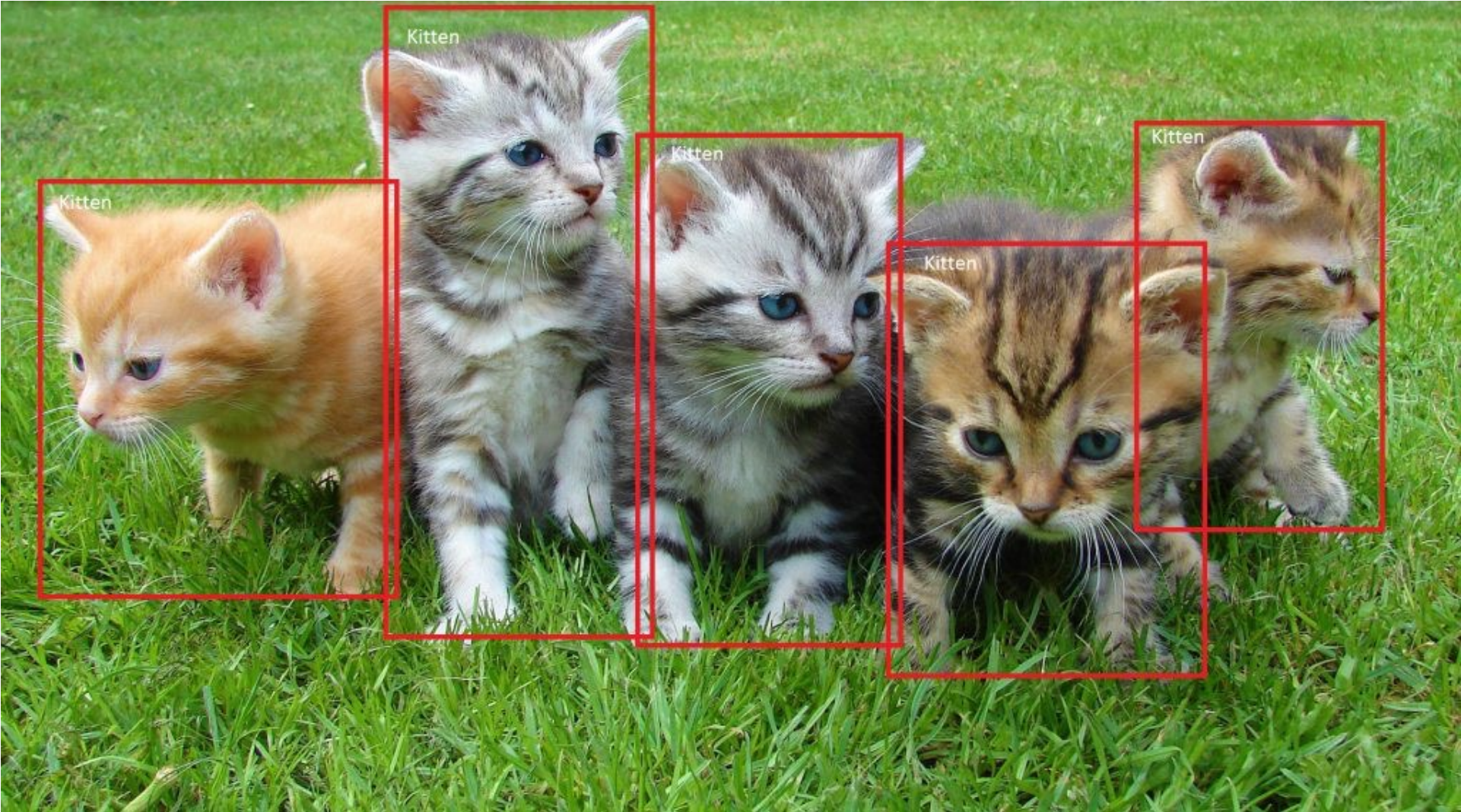


(assume given set of discrete labels)  
{dog, cat, truck, plane, ...}



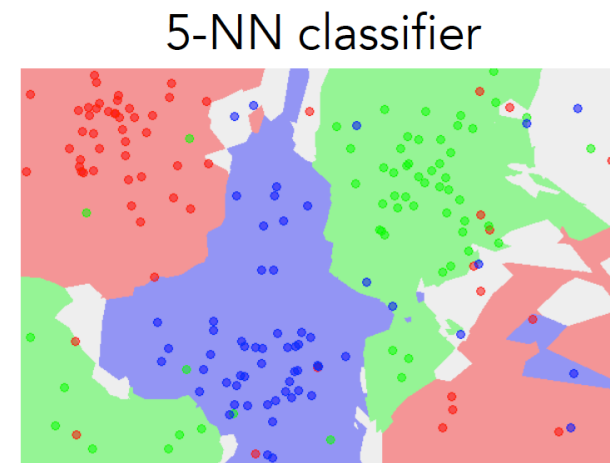
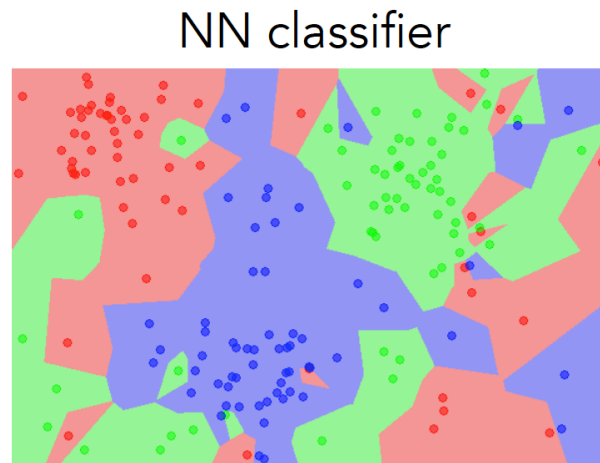
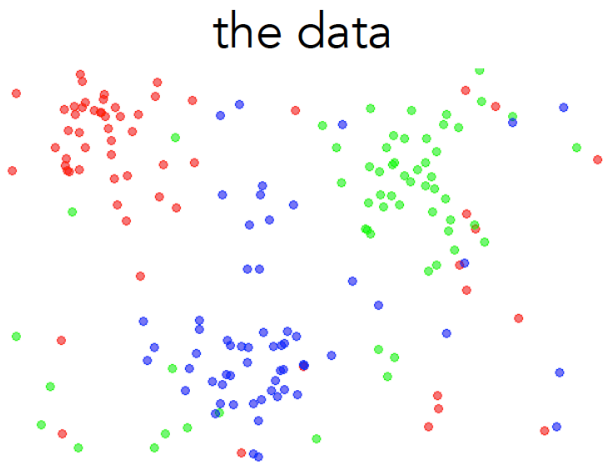
cat

# Object detection



# k-nearest neighbor

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



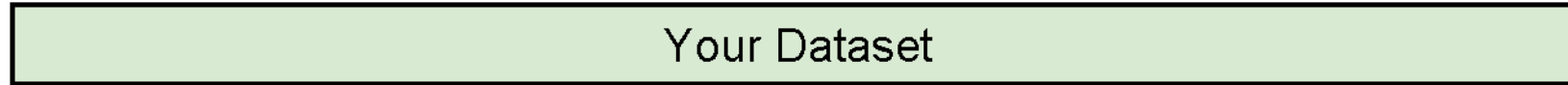
# Hyperparameters

- 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



**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

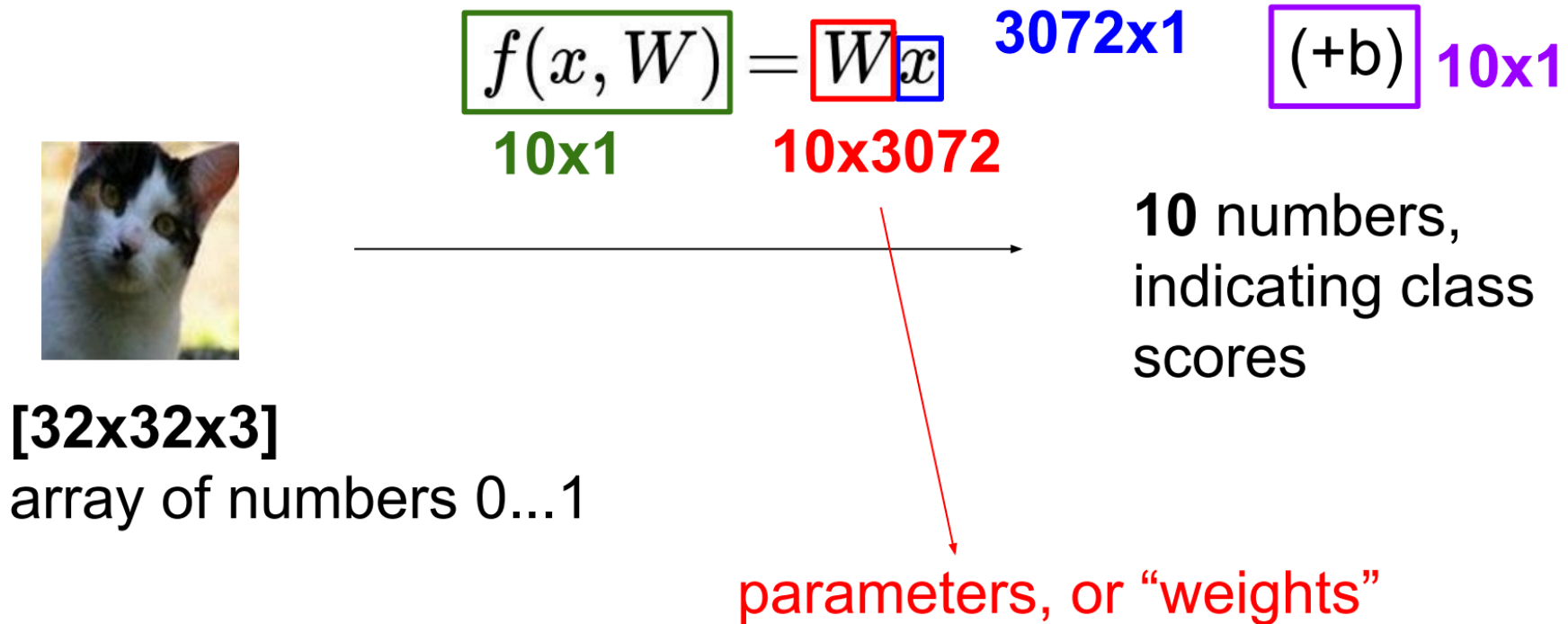


**Idea #3:** Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

**Better!**



# Parametric approach: Linear classifier



# 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  $\mathbf{W}$  that minimizes the loss function**

# Softmax classifier

$$f(x_i, W) = Wx_i \quad \text{score function is the same}$$

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

softmax function

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

Interpretation: squashes values into range 0 to 1

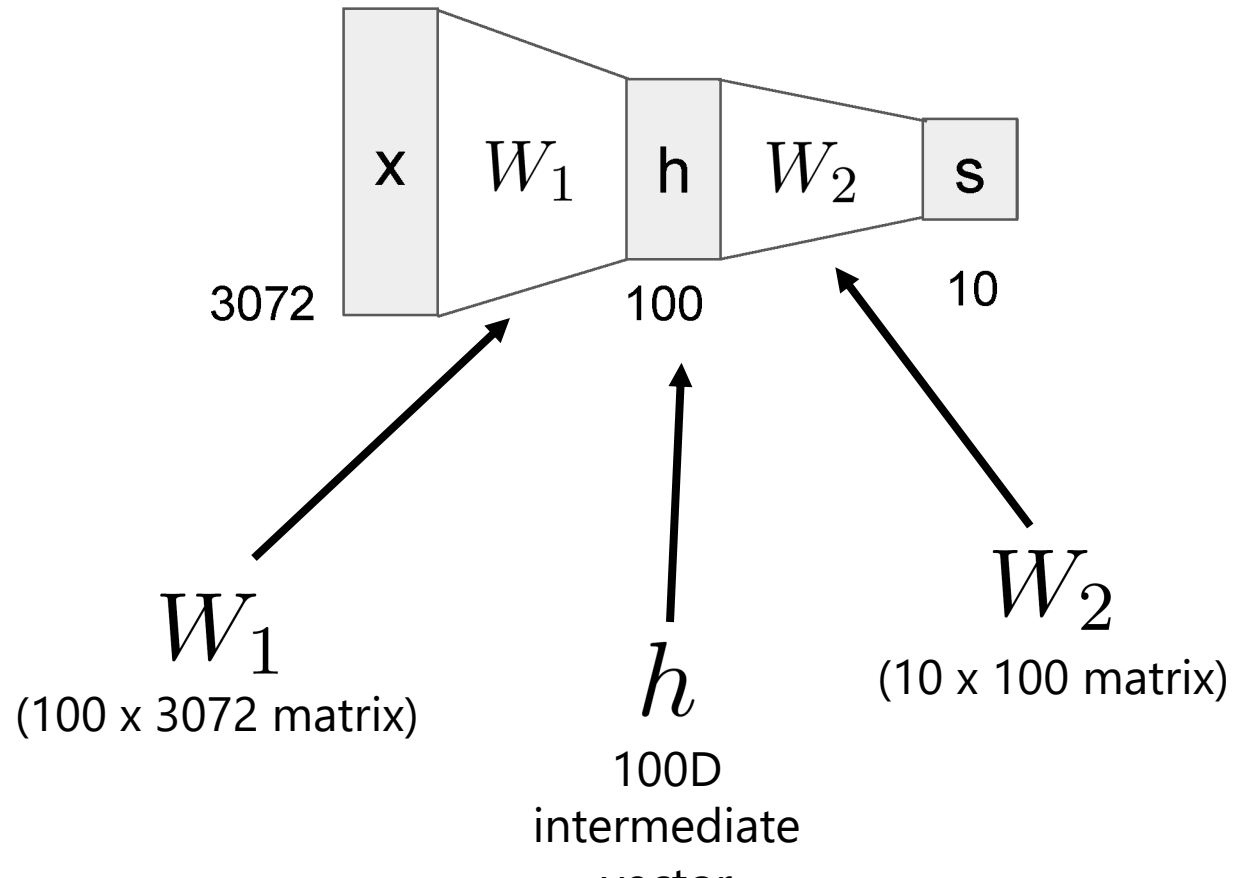
$$P(y_i | x_i; W)$$



# Neural networks

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

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



# Convolutional neural networks

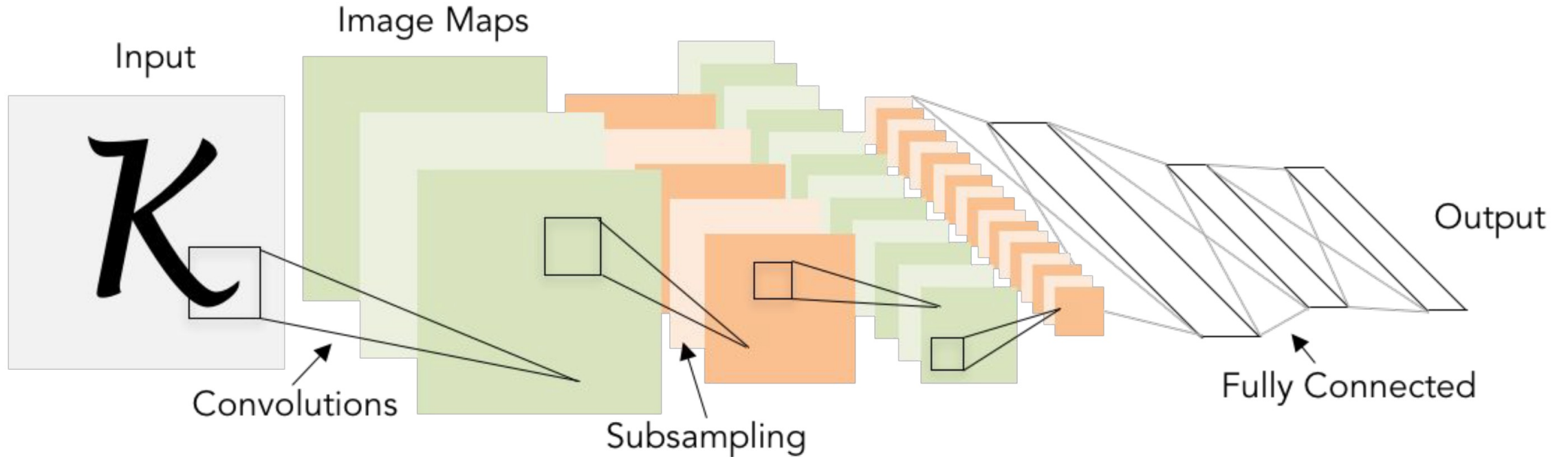


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

# Training deep networks – things to adjust during training

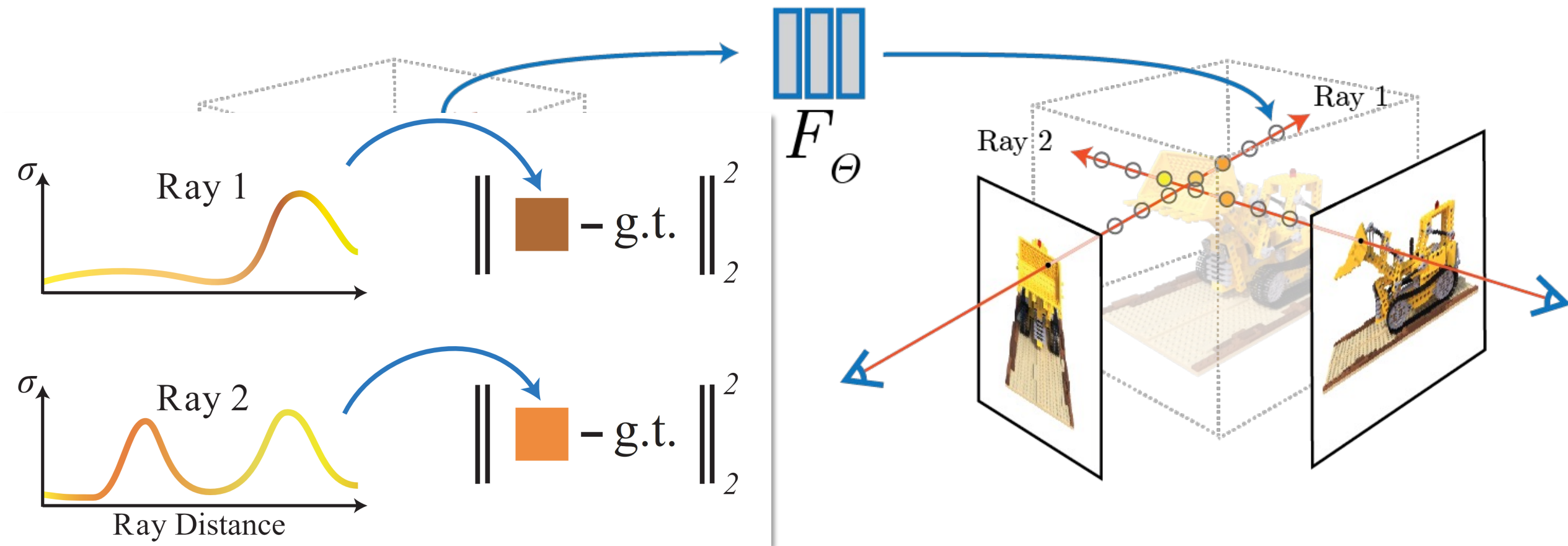
- Network architecture
- Learning rate, decay schedule, update type
- Regularization (L2, L1, maxnorm, dropout, ...)
- Loss function (softmax, SVM, ...)
- Weight initialization

**Goal:** good generalization to unseen data without overfitting on training data

Neural network parameters



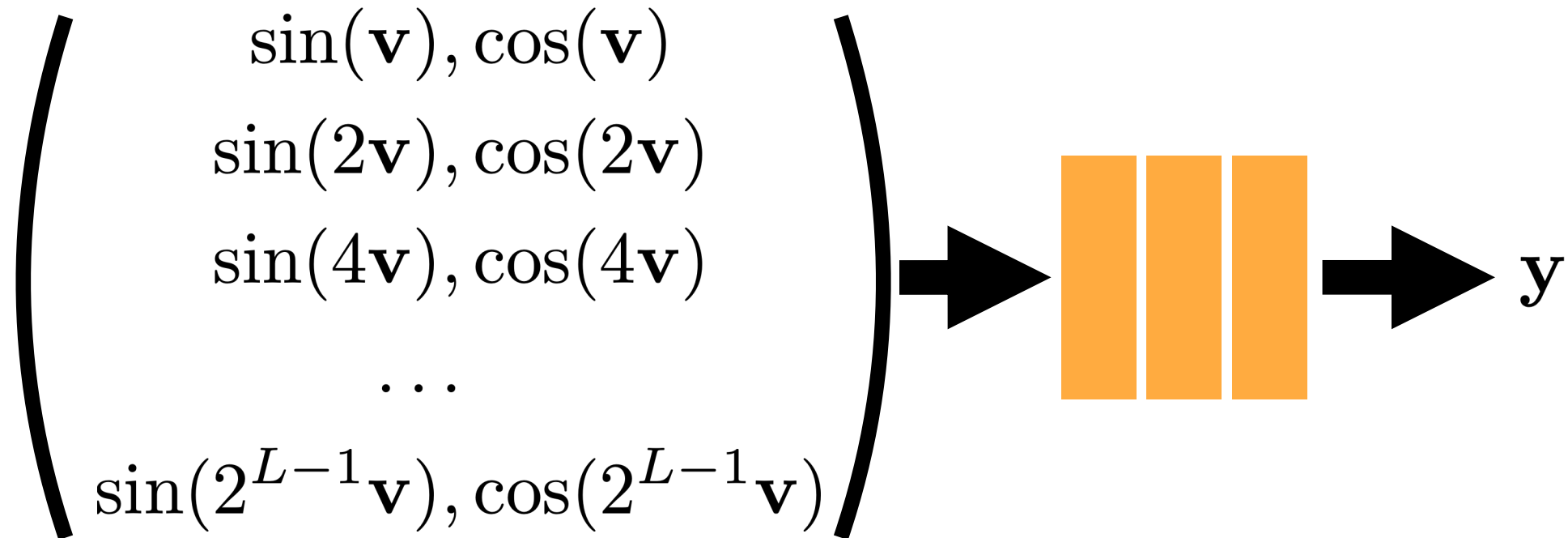
# NeRF: Full Neural 3D reconstruction



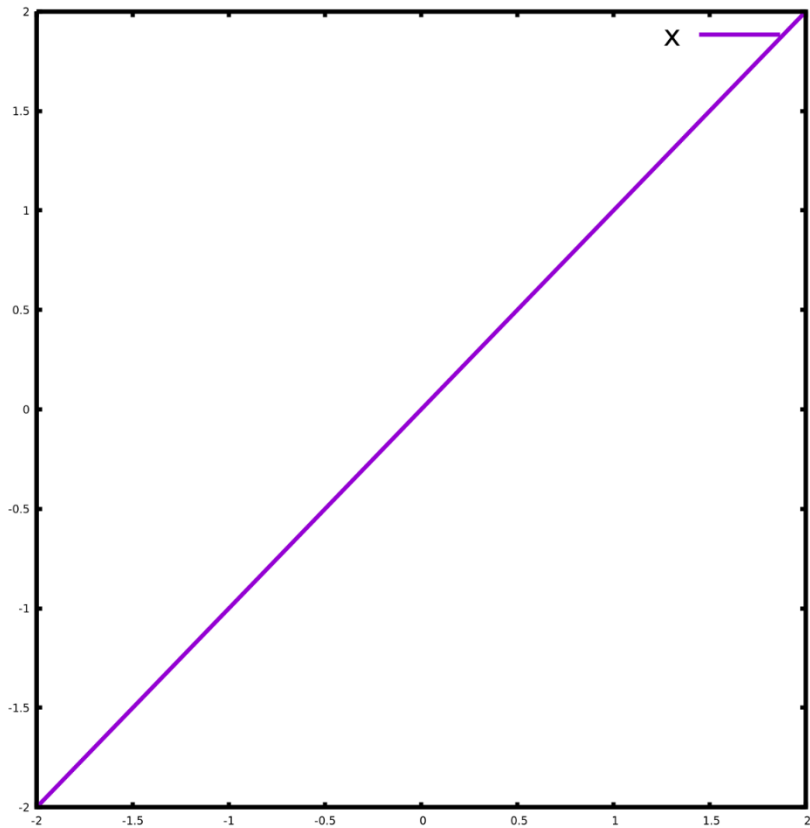
Ben Mildenhall\*, Pratul P. Srinivasan\*, Matthew Tancik\*, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV 2020.

<https://www.matthewtancik.com/nerf>

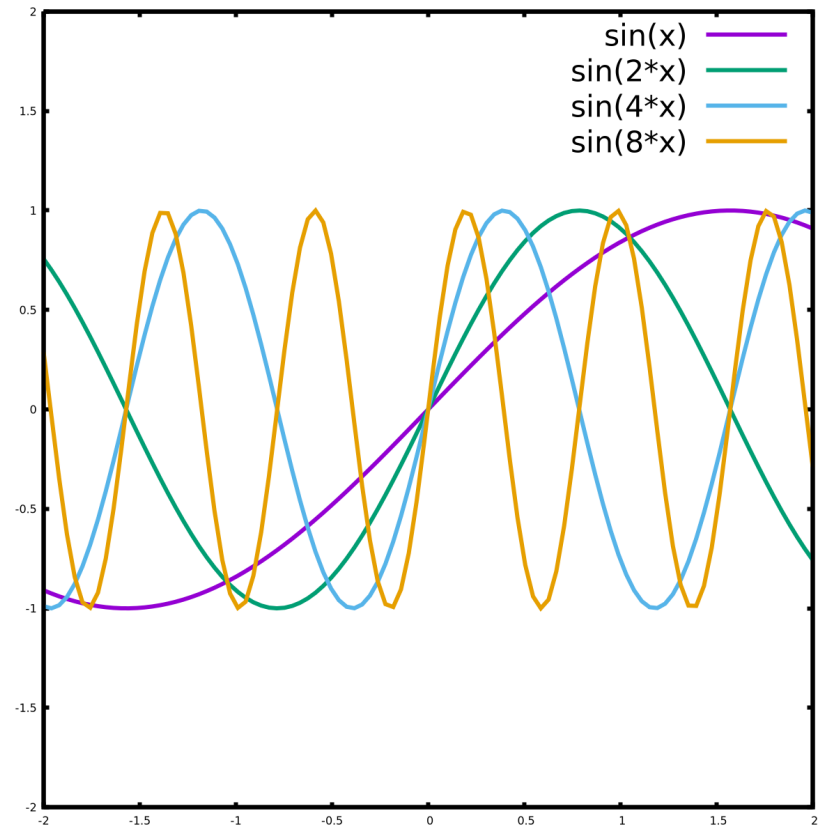
# Positional encoding



# Positional encoding

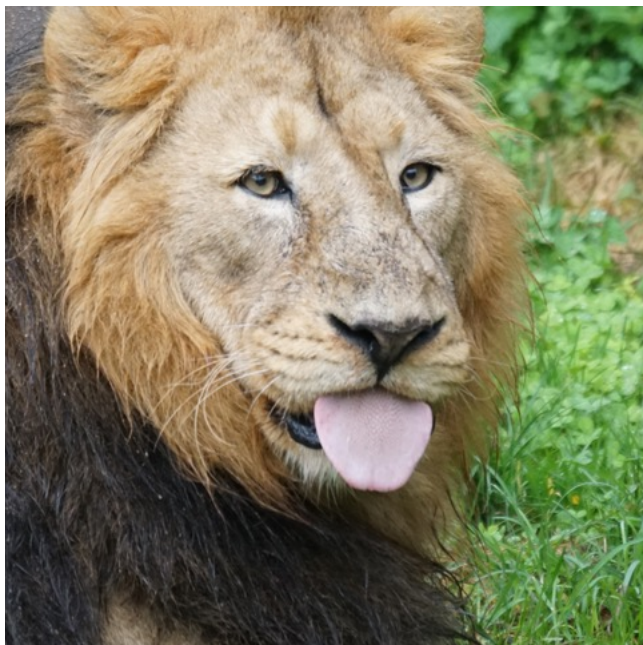


Raw encoding of a number



"Positional encoding" of a number

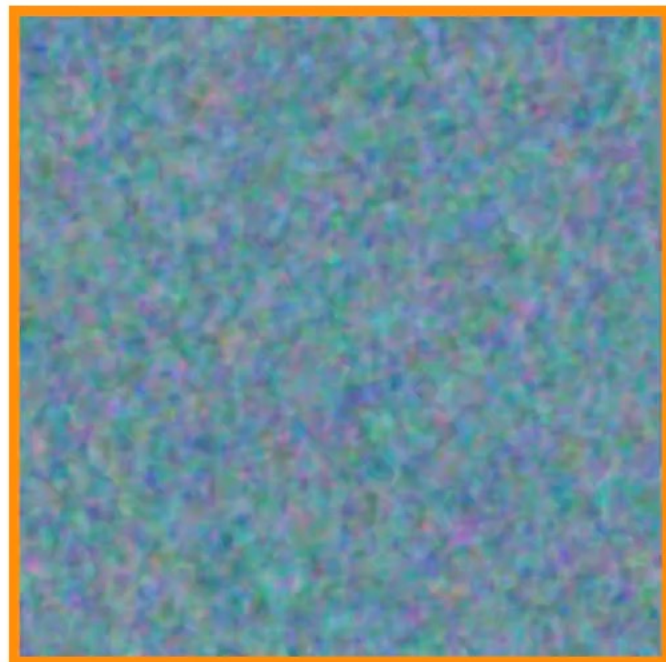
# Fitting high-resolution signals with neural networks (MLPs) via positional encoding



Ground truth image



Neural network output without high frequency mapping

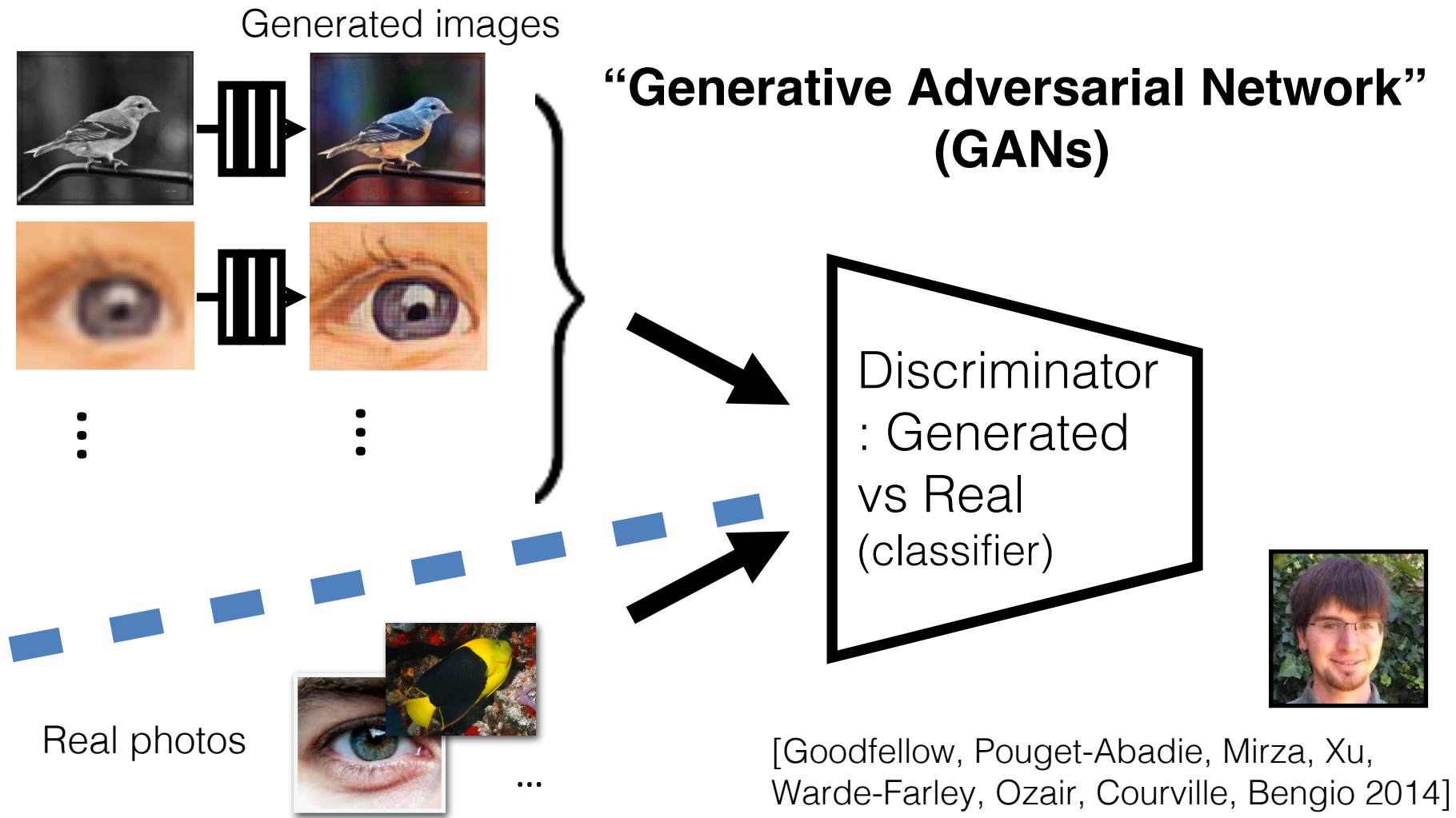


Neural network output with high frequency mapping

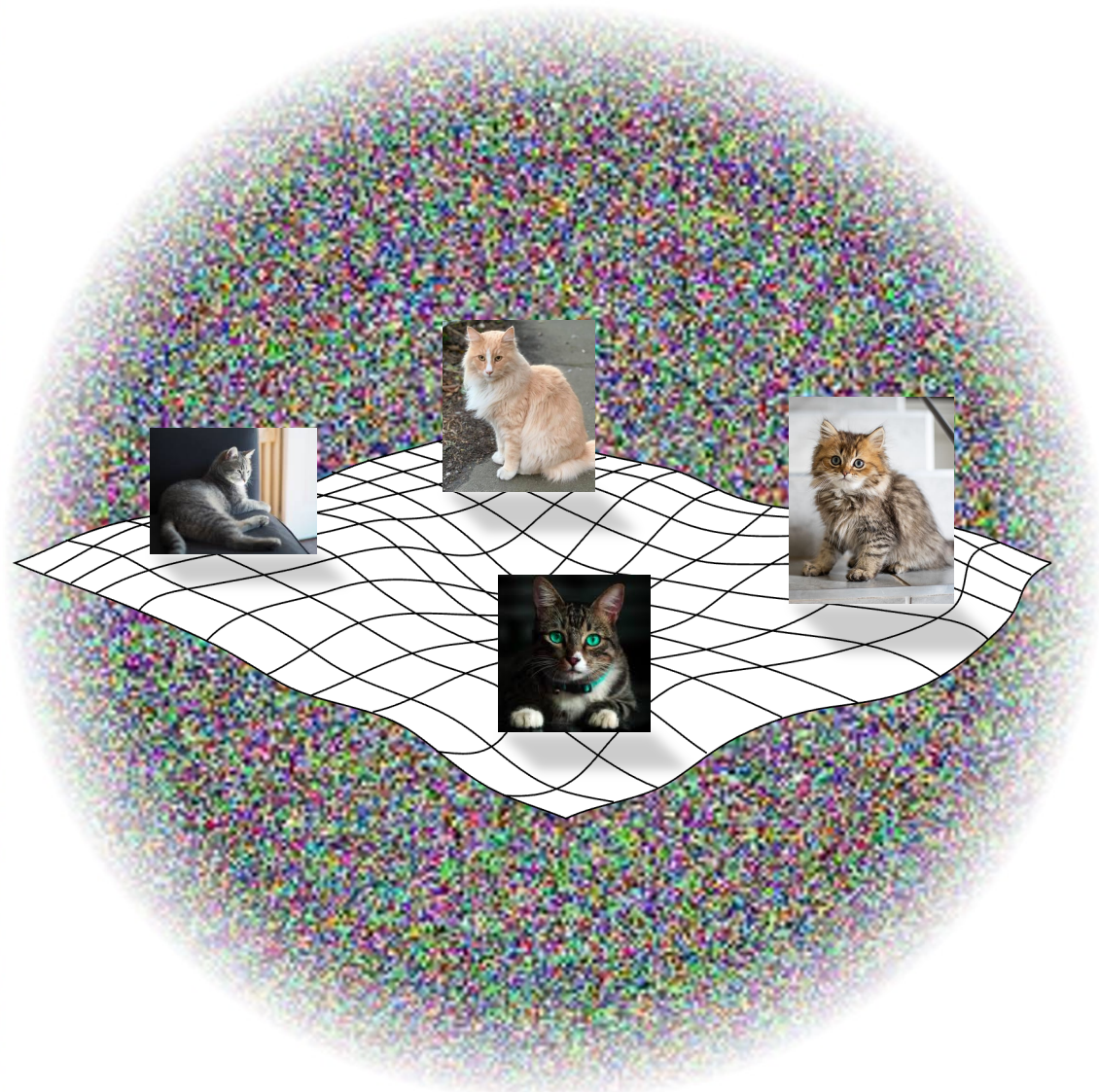
# NeRF Results







[Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio 2014]



**The Space of All Images**



StyleGAN2 [2020]

# BW $\rightarrow$ Color

Input

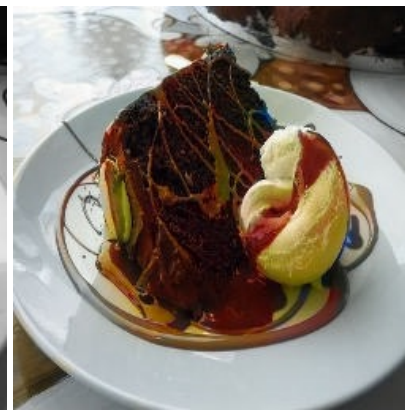
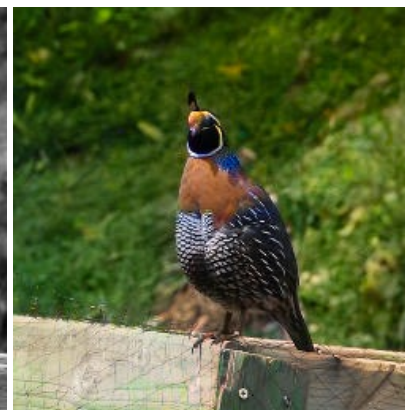
Output

Input

Output

Input

Output

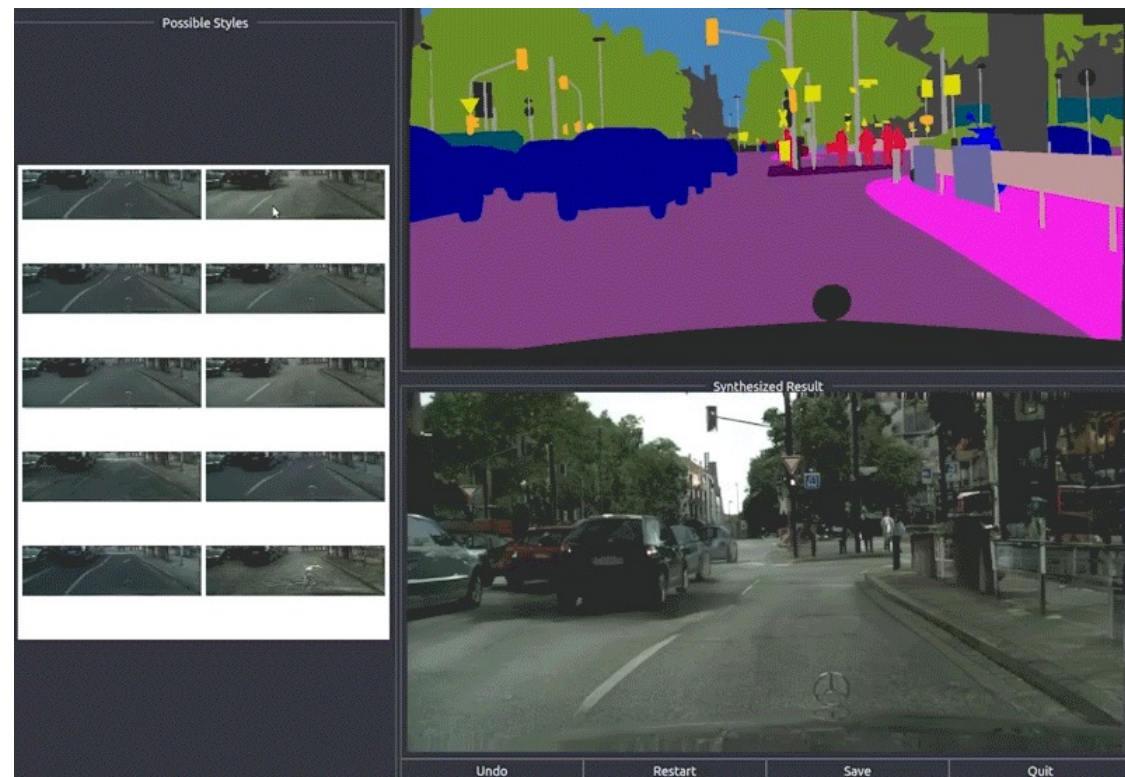


# Labels $\rightarrow$ Street Views

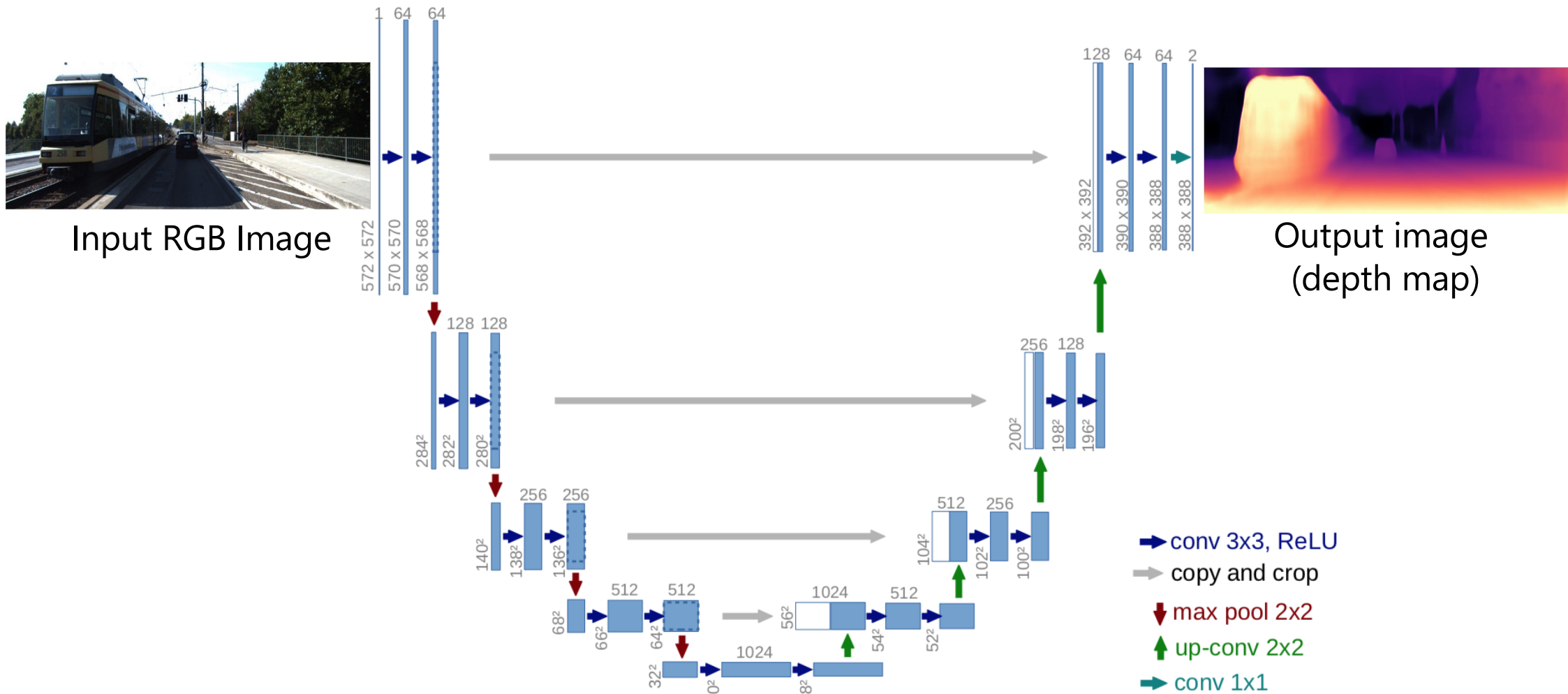
Input labels



Synthesized image



# Mapping images to images with the UNet architecture



# Datasets – Potential Ethical Issues

- Licensing and ownership of data
- Consent of photographer and people being photographed
- Offensive content
- Bias and underrepresentation
  - Including amplifying bias
- Unintended downstream uses of data

# Diffusion models

# Questions?

- Good luck!