High-Level Vision 4

Week 11

Mid-level Vision / Single-view Geometry: Shape-from-X

High-level Vision: Object Detection

Announcements

- **Drop-in Sessions:** Please bring your assignment/exam questions to the following locations
  - 14:00-16:00 Wednesday 15 and 22 May, 1.23 Hanna Neumann (Hoang)
  - 15:00-17:00 Friday 17 and 24 May, 1.23 Hanna Neumann (Yiran)
- **No Labs/Tutorials in Weeks 11-12:**
  - The lab space is still available for you to use for Assignment 3 / exam prep

Announcements

- **Assignment 3** due 11:59pm **this Friday** (17 May)
  - **Zero** marks if either report or code submitted late (unless extension)
    - Submit early; you can always resubmit an updated version later
    - Depending on your internet connection and load on the TurnItIn servers, uploading can sometimes be slow, so please factor this into your submission schedule
  - Submit your report (PDF) and code (ZIP file) **separately under the correct tab** in the submission box, or you will receive a mark of zero
  - Follow the instructions under Submission Requirements
  - Assignment figures/screenshots: must be sufficient resolution for the markers to read and interpret
- **Practice Exam Papers** released last Friday

Weekly Study Plan: Overview

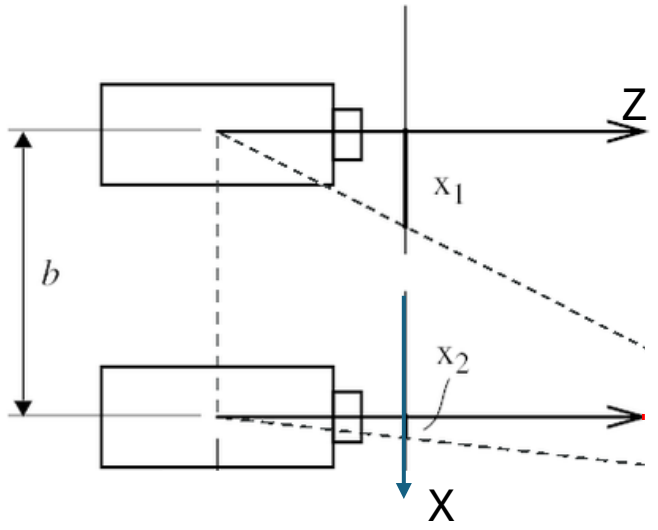
Wk	Starting	Lecture	Lab	Assessment
1	19 Feb	Introduction	✗	
2	26 Feb	Low-level Vision 1	1	
3	4 Mar	Low-level Vision 2	1	
		Mid-level Vision 1		
4	11 Mar	Mid-level Vision 2	1	CLab1 report due Friday
		High-level Vision 1		
5	18 Mar	High-level Vision 2	2	
6	25 Mar	High-level Vision 3[1]	2	
	1 Apr	Teaching break	✗	
	8 Apr	Teaching break	✗	
7	15 Apr	3D Vision 1	2	CLab2 report due Friday
8	22 Apr	3D Vision 2	3	
9	29 Apr	3D Vision 3	3	
10	6 May	3D Vision 4	3	
		Mid-level Vision 3		
11	13 May	High-level Vision 4	✗	CLab3 report due Friday
12	20 May	Course Review	✗	

Weekly Study Plan: Part B

Wk	Starting	Lecture	By
7	15 Apr	3D vision: introduction, camera model, single-view geometry	Dylan
8	22 Apr	3D vision: camera calibration, two-view geometry (homography)	Dylan
9	29 Apr	3D vision: two-view geometry (epipolar geometry, triangulation, stereo)	Dylan
10	6 May	3D vision: multiple-view geometry	Weijian
		Mid-level vision: optical flow, shape-from-X	Dylan
11	13 May	High-level vision: self-supervised learning, detection, segmentation	Dylan
12	20 May	Course review	Dylan

Homogeneous Coordinates: Points at Infinity

- A homogeneous 3-vector can represent points at infinity
- $\lim_{a \to \infty} \begin{bmatrix} a \\ 2a \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$
- Application: epipoles
  - What is the pattern of epipolar lines in the bottom camera?
    - Horizontal parallel lines
  - What is the coordinate of the epipole in the bottom camera coordinates?
    - $\begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$

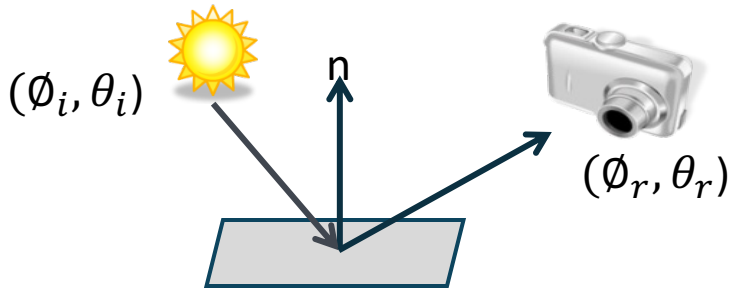


Shape-from-Shading

Photometric Stereo

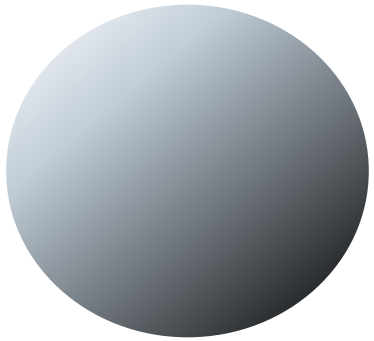
What Determines Image Shading (Intensity)?

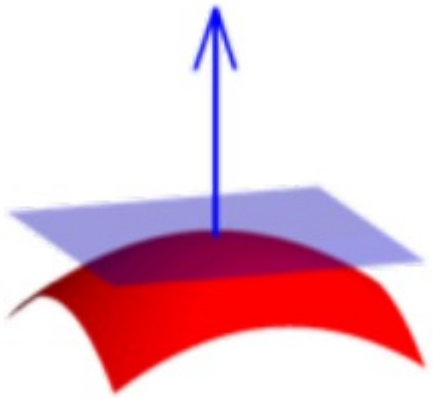
- The amount of light that falls on the surface
- The fraction of light that is reflected (albedo)
- Geometry of light reflection
  - Shape of surface
  - Viewpoint



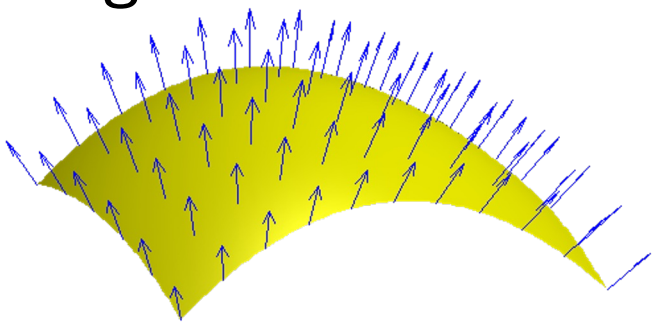
3D Surface Geometry is Defined by the Surface Normal at Every Point

- Convenient notation for surface orientation
- A smooth surface has a local tangent plane at every point



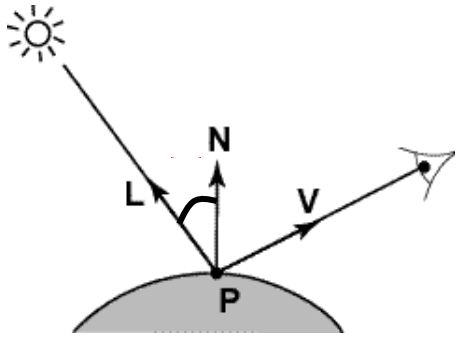


- We can fit the model using the surface normal at every point



Modeling Photometric Image Formation

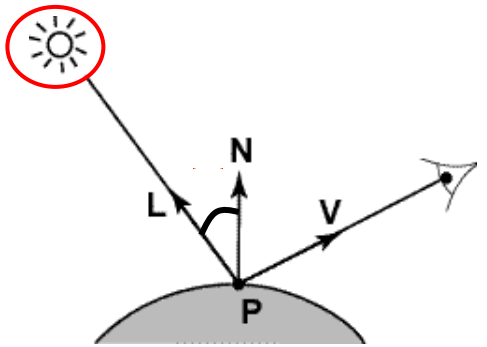
- Now we need to reason about:
  - How light interacts with the scene
  - How a pixel value is related to light energy in the world



  - Track a ray of light all the way from the light source to the image CMOS/CCD sensor

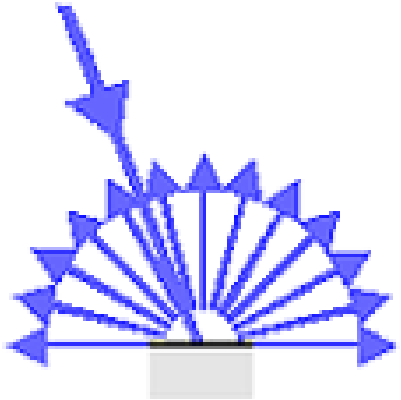
Assume Far-field Directional Lighting

- Key property: all rays are parallel
- Equivalent to an infinitely distance point source
- (Other lighting models are possible, but this is the one we will study)

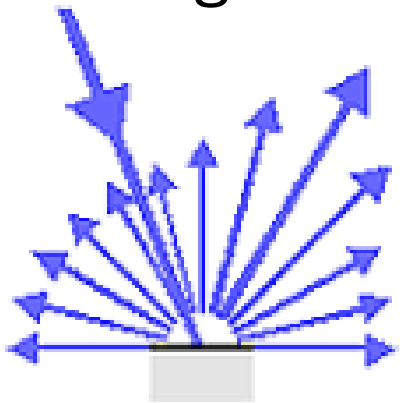


An Ideal Model: Lambertian Reflection

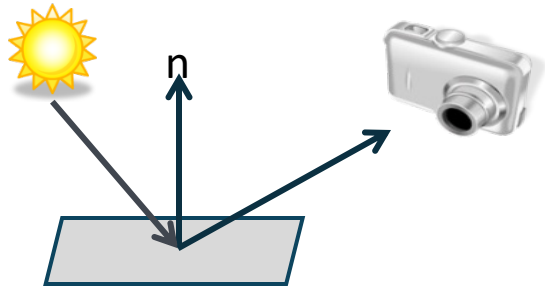
- Appears equally bright from all viewing directions
- Reflects all light without absorbing
- Matte surface, no “shiny” spots
- Brightness of the surface as seen from camera is linearly correlated to the amount of light falling on the surface



Ideal diffuse reflection
(Lambertian surface)

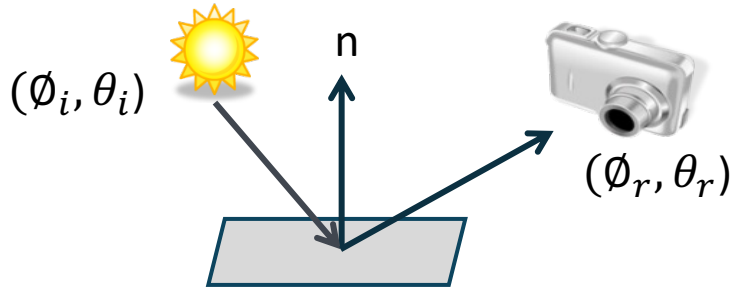


Diffuse reflection with
directional component

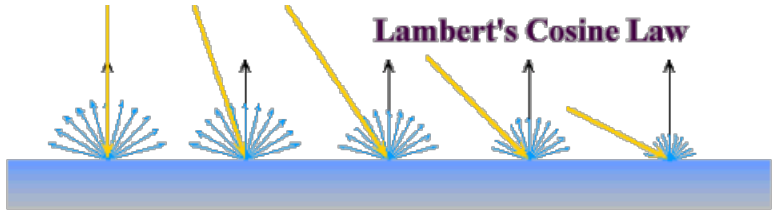


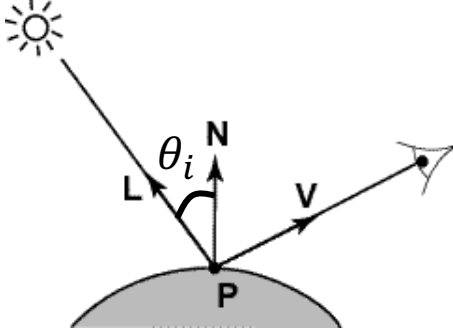
Lambertian Surface

- A point on a Lambertian surface appears equally bright in all directions
  - It has the same surface radiance no matter which direction you observe it from
- $f_L(\phi_i, \theta_i; \phi_r, \theta_r) = \bar{f} = \frac{\rho}{\pi} = \text{const.}$
  - For a fixed incident direction, $(\emptyset_r, \theta_r)$ is the observer
  - $\rho \in [0, 1]$: albedo (proportion of incident light that is reflected by a surface)
    - A constant that depends on the surface material
    - $\rho = 0$: perfectly black surface
    - $\rho = 1$: perfectly white surface, reflecting all light



Lambertian Reflection



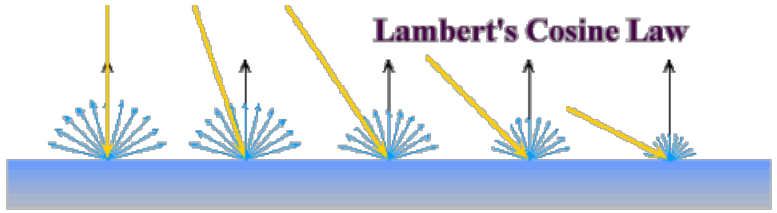


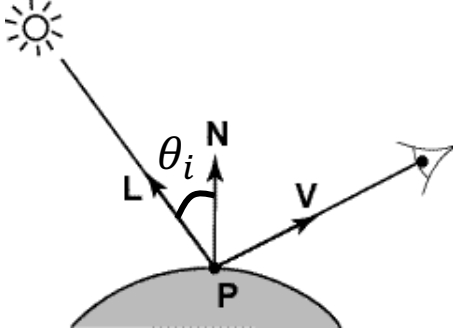
$$R_r = \rho N \cdot L R_i$$
$$I = \rho N \cdot L$$
$$\boxed{I = N \cdot L = \cos\theta_i}$$

- R_i: incident light intensity
- R_r: observed light intensity
- L: unit illuminant direction
- N: unit normal direction
- I: image intensity at point **P**
- ρ: material albedo

- Simplifying assumptions
  - $I = R_r$: camera response function f is the identity function
    - Can also assume linearly proportional
    - (If required, can perform radiometric calibration)
  - $R_i = 1$: light source intensity is 1
    - Can achieve this by dividing each pixel in the image by R_i
  - $\rho = 1$: material albedo is 1

Lambertian Reflection





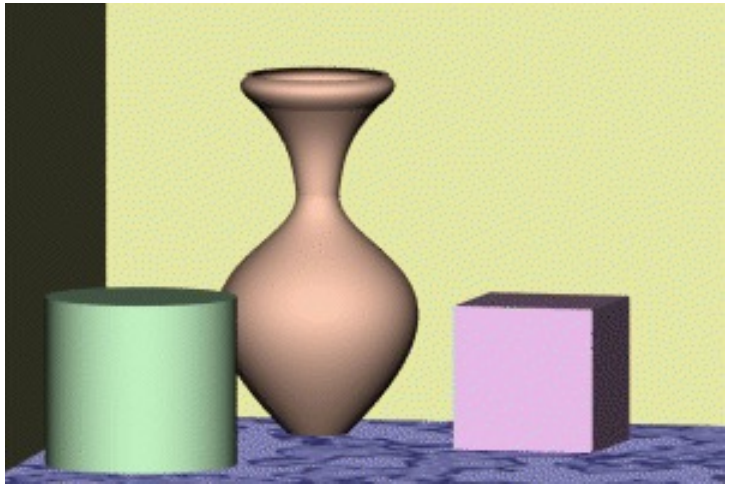
$$R_r = \rho N \cdot L R_i$$
$$I = \rho N \cdot L$$
$$\boxed{I = N \cdot L = \cos\theta_i}$$

- R_i: incident light intensity
- R_r: observed light intensity
- L: unit illuminant direction
- N: unit normal direction
- I: image intensity at point **P**
- ρ: material albedo

Therefore:

- Greatest image intensity when θ_i is 0
- Reduces as θ_i increases

Lambertian Surfaces in Computer Graphics

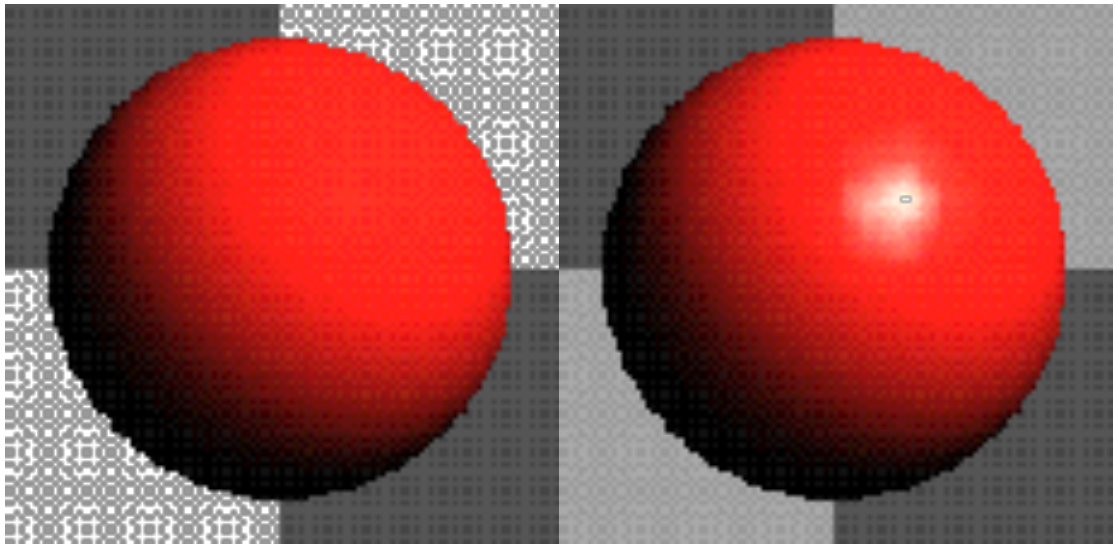


(Oren and Nayar)

- Lambertian sphere as the light moves

(Steve Seitz)

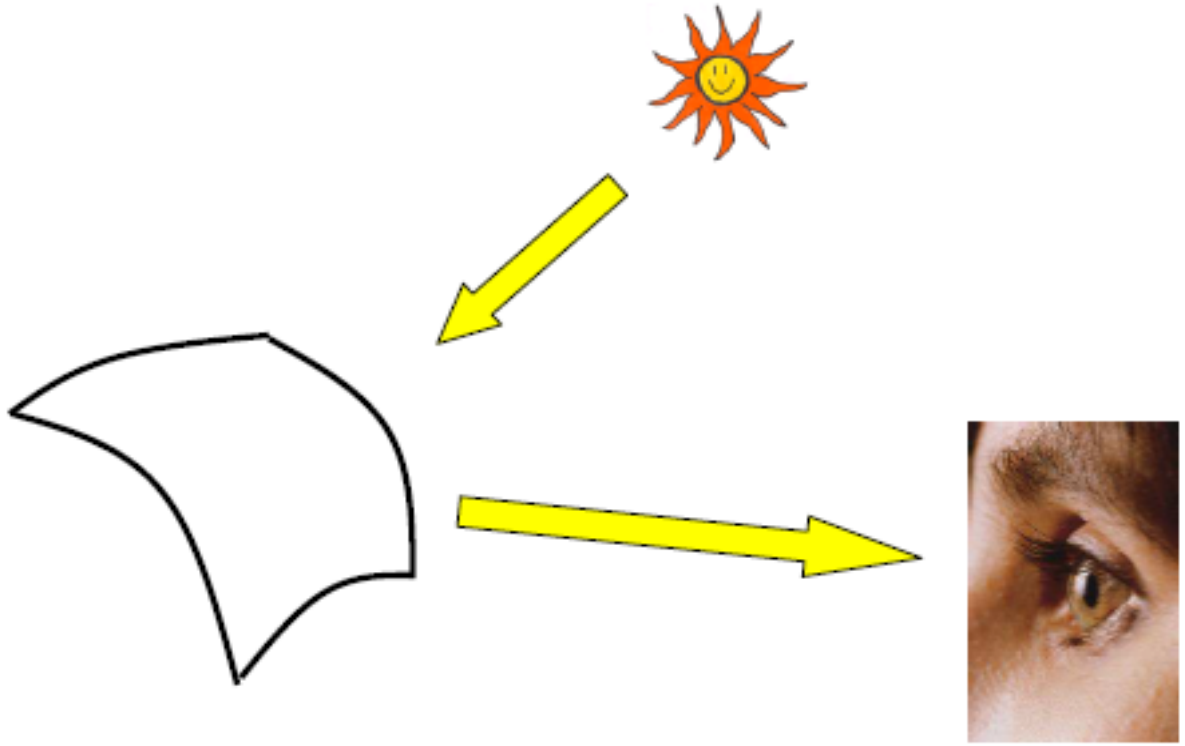
More Realistic: Lambertian + Specular



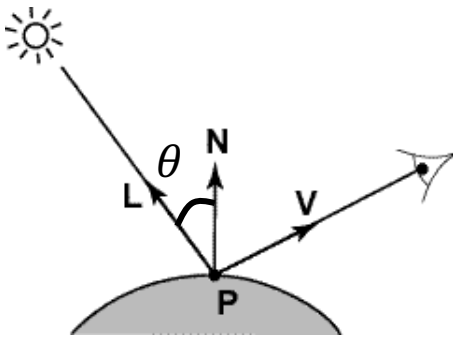
Images from: http://graphics.cs.ucdavis.edu/GraphicsNotes/Shading/Shading.html

Shape-from-Shading

- Inverting the image formation process



Shape-from-Shading



- From $I = N^\top L = \cos\theta$, we see that you can directly measure the angle between the normal and the light source
- Not enough information to compute normal:
  - 1 equation, 2 degrees of freedom for a unit surface normal
  - Normal could be any vector rotating around the light vector with angle θ
- But can be enough if you add some additional information, e.g.:
  - Assume some of the normals are known (e.g., along silhouette)
  - Constraints on neighbouring normals—"integrability"
  - Smoothness
- Hard to get to work well in practice
  - How many real objects have constant albedo anyway?

$$\hat{x} = \frac{x}{||x||}$$

Computer Graphics: Forward Rendering

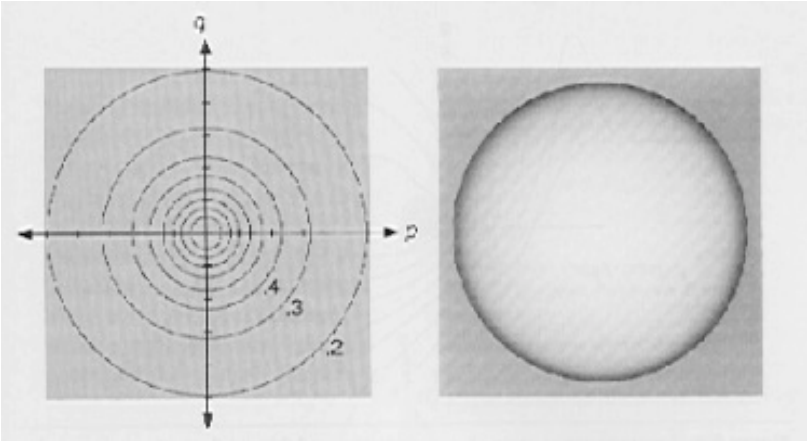
- Given a 3D surface $z(x, y)$, lighting and viewing direction, we can compute the grey level of pixel $I(x, y)$ on the surface
- Assume we know the surface normal map, then we can deduce the image from irradiance
- Use $I(x, y) = R(p, q) = \hat{n}^\top \hat{l} = \frac{1+pp_s+qq_s}{\sqrt{1+p^2+q^2}\sqrt{1+{p_s}^2+{q_s}^2}}$ to determine the grey level
  - $n = (p, q, -1)$: surface normal
  - $l = (p_s, q_s, -1)$: light source direction
- In CV, we do the opposite: assume I is known and try to work out n

R. Woodham, *Photometric Method for Determining Surface Orientation from Multiple Images*. Optical Engineering 19(1)139-144 (1980). (PDF)

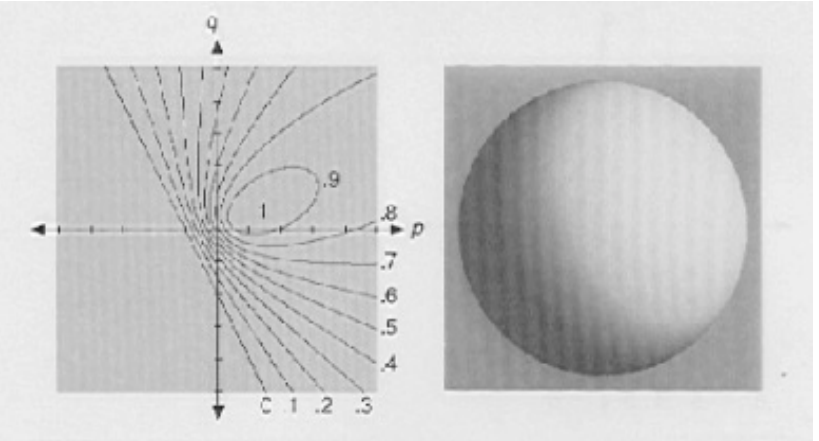
$$R(p,q) = \frac{1 + p_s p + q_s q}{\sqrt{1 + p^2 + q^2}\sqrt{1 + {p_s}^2 + {q_s}^2}}$$

Lambertian Reflectance Map

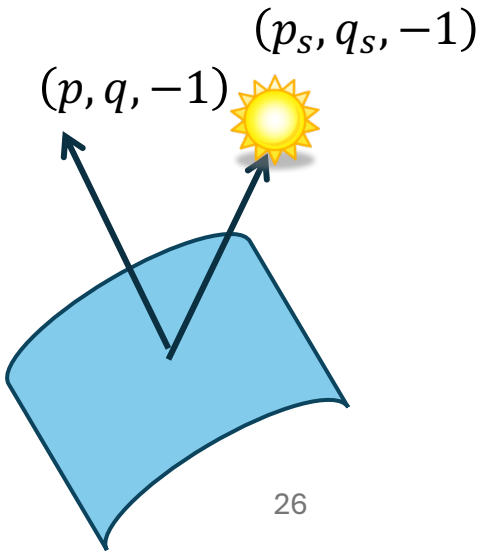
- Local surface orientation that produces equivalent intensities are quadratic conic section contours in gradient space
- Iso-intensity maps:



$(p_s, q_s) = (0,0)$

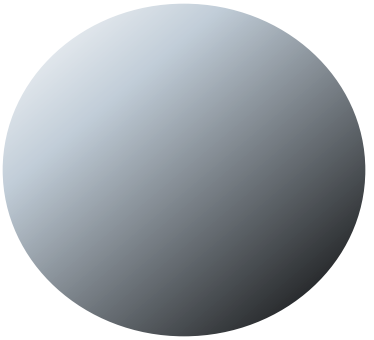


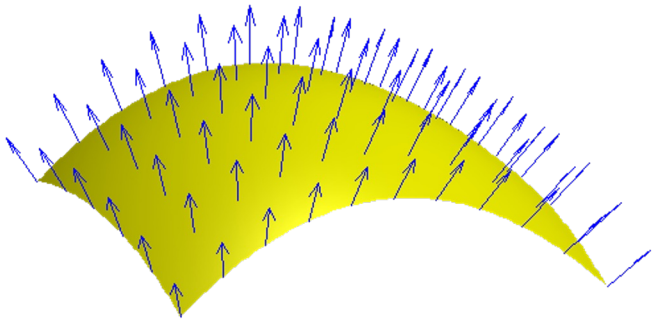
$(p_s, q_s) = (-2, -1)$



Shape-from-Shading

- Given a greyscale image
  - albedo
  - point light source direction
- Reconstruct scene geometry
  - Can be modelled by surface normals





Shape-from-Shading: Solving It

- We have $R(p,q) = \frac{1+pp_s+qq_s}{\sqrt{1+p^2+q^2}\sqrt{1+{p_s}^2+{q_s}^2}}$
- Discretise: one nonlinear equation per pixel with two unknowns
  - For N pixels → *N* equations in 2*N* unknowns…
- Cannot be solved if you treat each point independently
- We have to somehow tie these points together so that we can propagate them and let them constrain each other in some way

Variational Shape-from-Shading

- Approach: add smoothness constraint (p and q change slowly)
  - Form an energy minimisation problem
- Given observed brightness I(x,y), find {(p,q)} that minimises energy

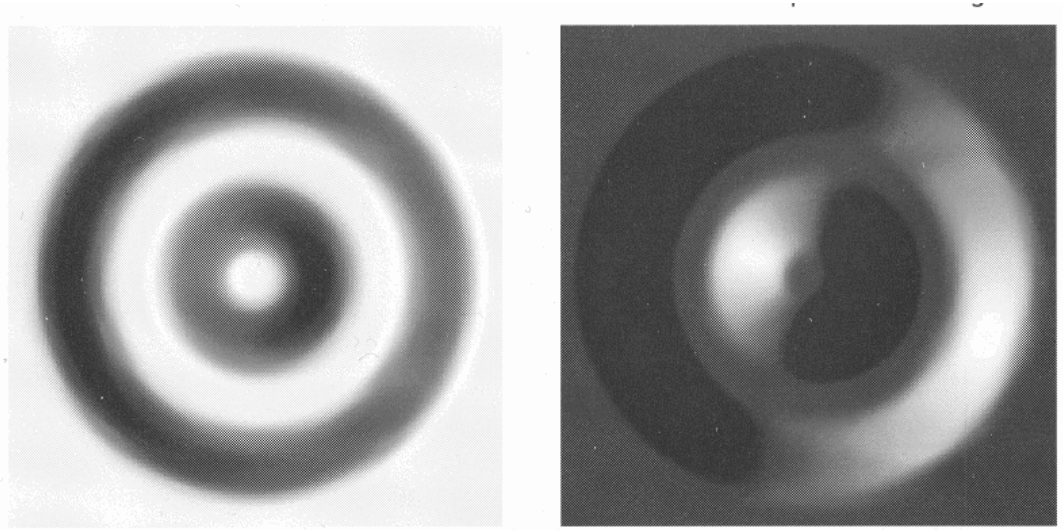
$$E = \int \left(\left(I(x,y) - R(p,q) \right)^2 + \lambda \left(\frac{\partial p}{\partial x}^2 + \frac{\partial p}{\partial y}^2 + \frac{\partial q}{\partial x}^2 + \frac{\partial q}{\partial y}^2 \right) \right) \mathrm{d}x\mathrm{d}y$$

Penalise errors between the image brightness and reflectance map

Penalise rapid changes in p and q

- Regularisation: minimise surface curvature
- Use Euler–Lagrange equations to solve (not in course)

Shape-from-Shading: Results



(b)

(a)

Figure 9.2 Two images of the same Lambertian surface seen from above but illuminated from different directions and 3-D rendering of the surface. Practically all the points in the top left image receive direct illumination ($\mathbf{i} = [0.20, 0, 0.98]^\top$); some regions of the top right image are in the dark due to self-shadowing effects ($\mathbf{i} = [0.94, 0.31, 0.16]^\top$).

Image Credit: Trucco & Verri

Shape-from-Shading: Results

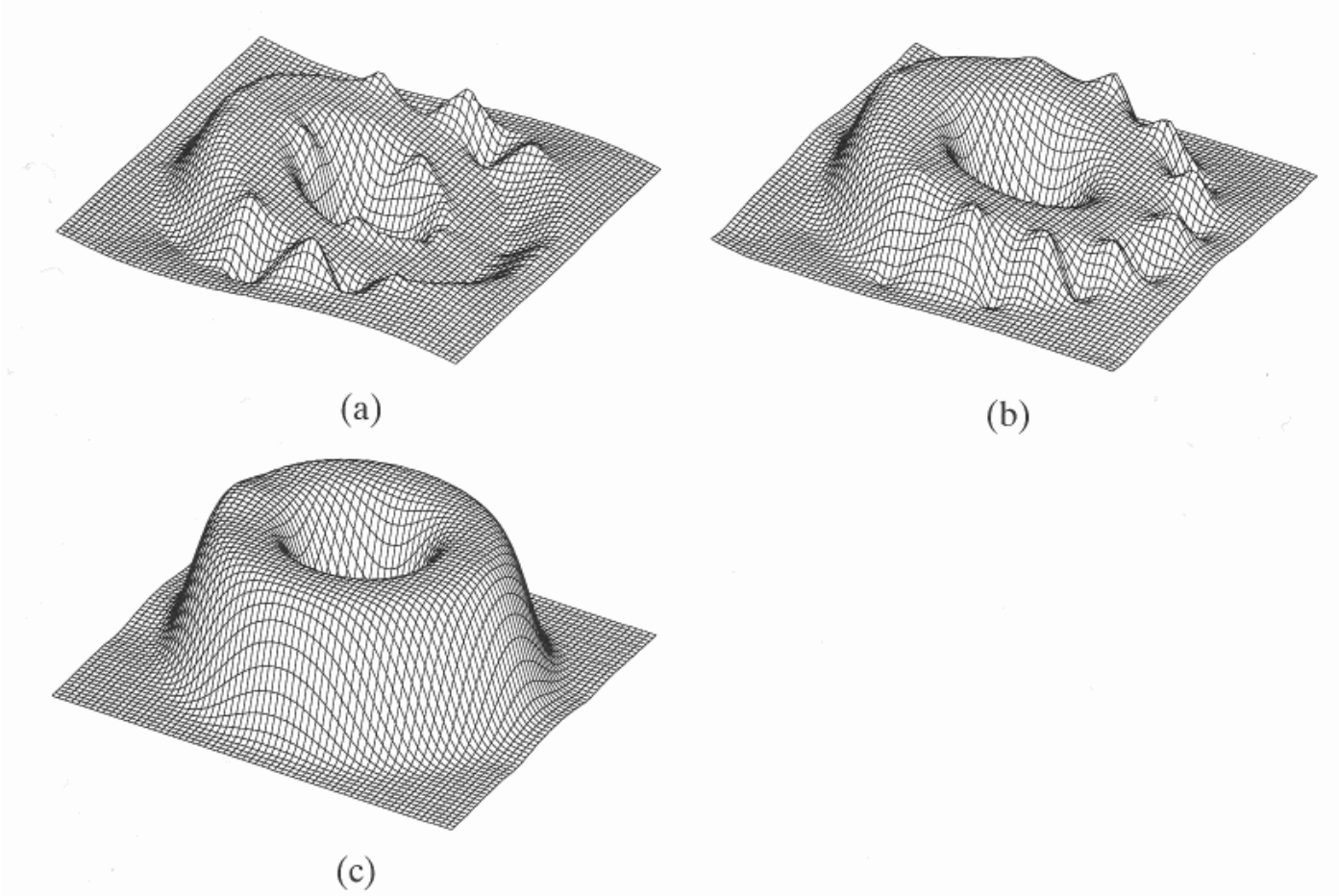
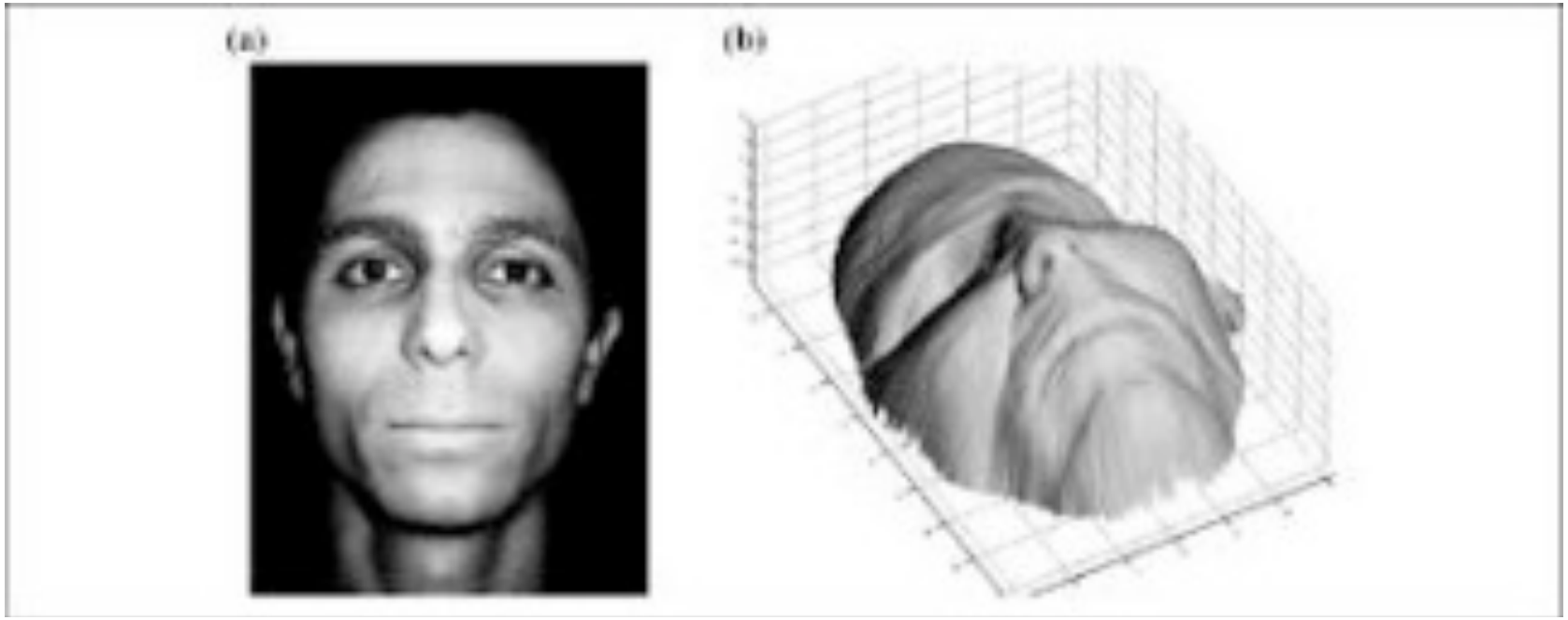
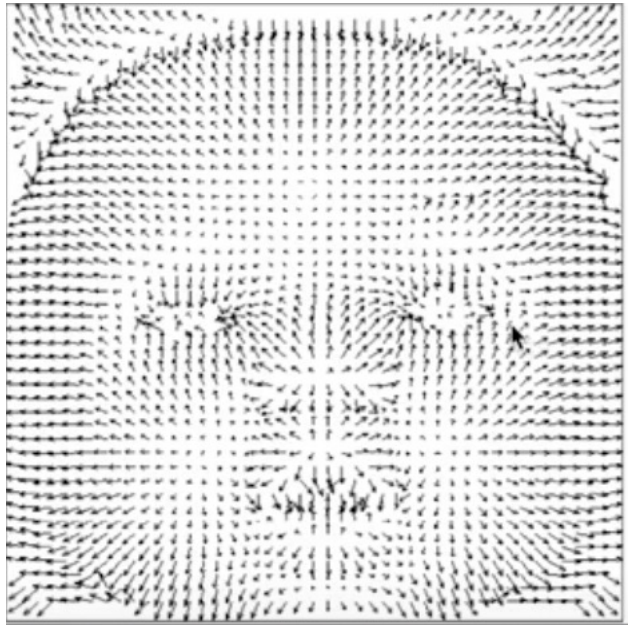


Figure 9.4 Reconstructions of the surface in Figure 9.2 after 100 (a), 1000 (b) and 2000 (c) iterations. The initial surface was a plane of constant height. The asymmetry of the first two reconstruction is due to the illuminant direction.

Shape-from-Shading: Results



Shape-from-Shading: From Normals to Shape



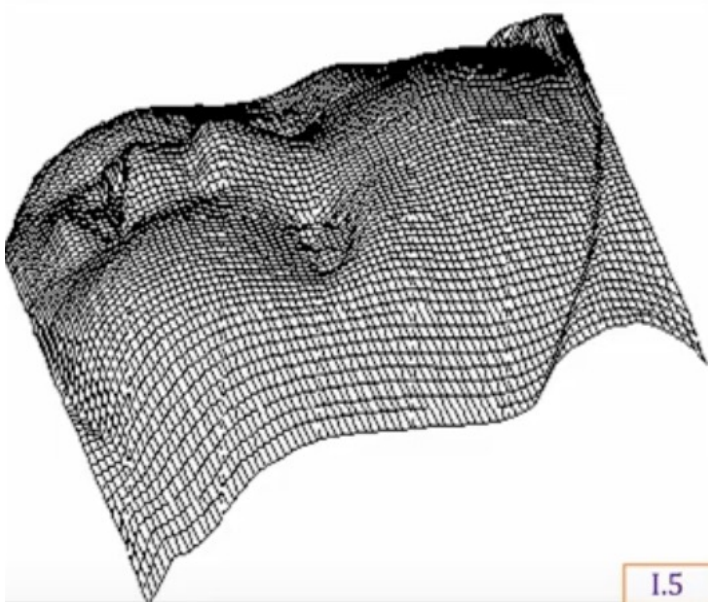
Gradient / normal map

$$[p(x,y), q(x,y), 1]$$

$$(p,q) = \left(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}\right)$$

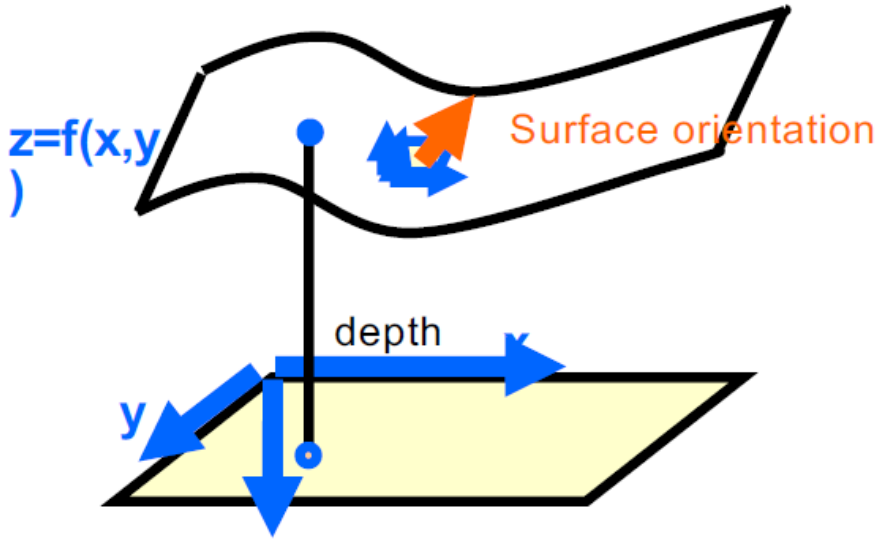
differentiation

integration



Shape / depth map z(x,y)

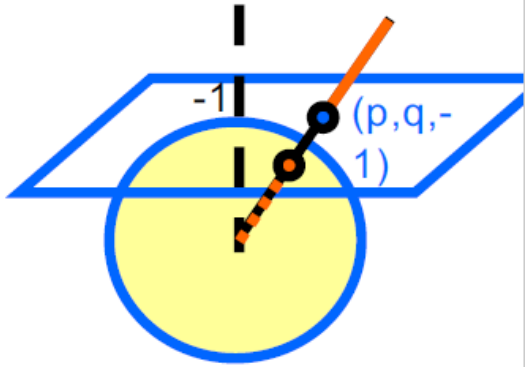
Surface Parametrisation



Surface $s(x,y) = (x, y, f(x,y))$

Tangent plane $\frac{\partial s}{\partial x} = \left(1, 0, \frac{\partial f}{\partial x}\right)^T \quad \frac{\partial s}{\partial y} = \left(0, 1, \frac{\partial f}{\partial y}\right)^T$

Normal vector $\mathbf{n} = \frac{\partial s}{\partial x} \times \frac{\partial s}{\partial y} = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, -1\right)^T$



Gradient space

$$p = \frac{\partial f}{\partial x} \quad q = \frac{\partial f}{\partial y}$$

$$\mathbf{n} = (p, q, -1)$$

$$\hat{\mathbf{n}} = \frac{1}{\sqrt{p^2 + q^2 + 1}}(p, q, -1)$$

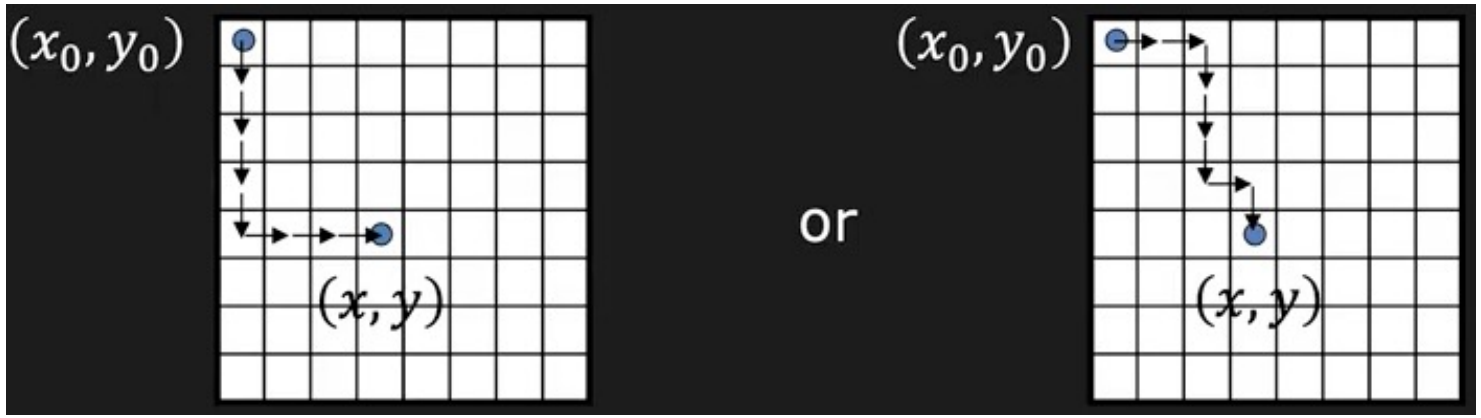
Shape-from-Shading: From Normals to Shape

- We now have surface normal, but how do we get depth or shape?
- Estimate surface by integrating surface gradient

$$z(x,y) = z(x_0,y_0) + \int_{(x_0,y_0)}^{(x,y)} (pdx + qdy)$$

  where (x_0, y_0) is a reference point and $z(x_0, y_0) = 0$

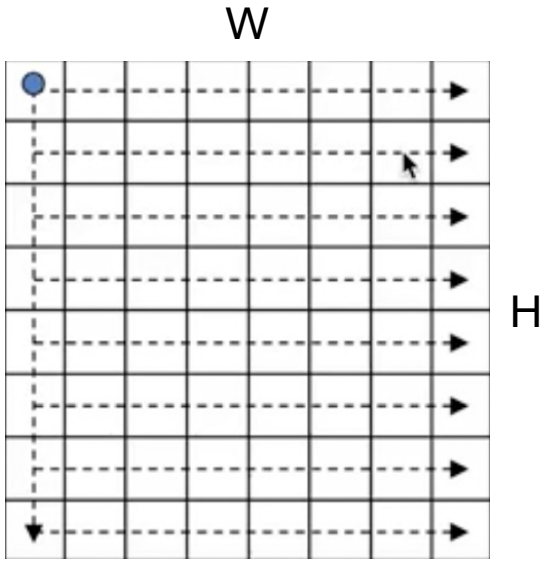
- $z(x,y)$ obtained by integration along any path from (x_0, y_0) should be the same



Shape-from-Shading: From Normals to Shape

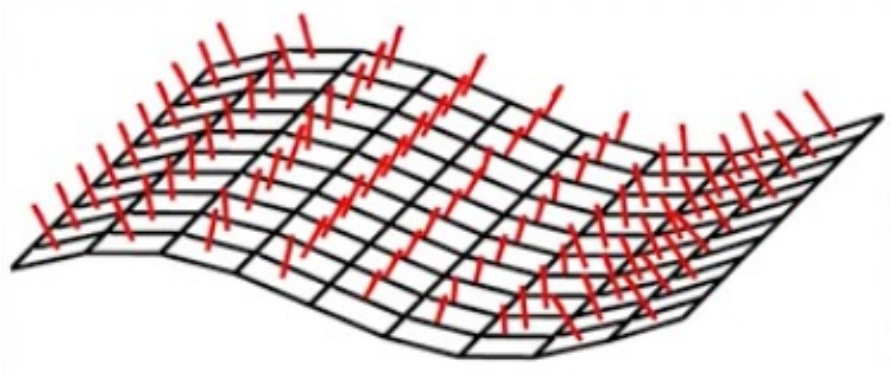
A naïve algorithm for estimating depth map:

1. Initialise reference depth
   - $z(0,0) = 0$
2. Compute depth for first column
   - for $y = 1$ to $(H - 1)$
     $z(0, y) = z(0, y - 1) + q(0, y)$
3. Compute depth for each row
   - for $y = 0$ to $(H - 1)$
     for $x = 1$ to $(W - 1)$
     $z(x, y) = z(x - 1, y) + p(x, y)$

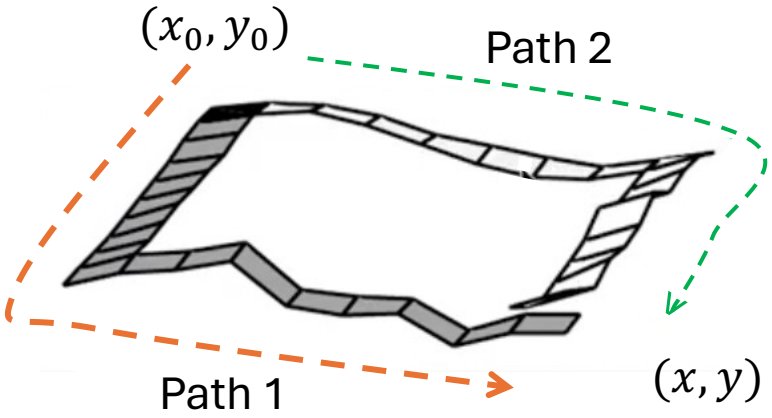


Computed depth map

Noise Sensitivity of Computed Surface



Actual surface shown with noisy estimates of surface gradient



Depth computed from noisy gradients depends on the integration path

- An ad-hoc solution: compute depth maps using different paths. Then, find average of computed depth maps to reduce error
- It turns out you can do much better than this…

Estimating Shape using Least Squares

- Find a surface that minimises the errors between measured surface gradients (p, q) and gradients of estimated surface $z(x, y)$
- Error measure:

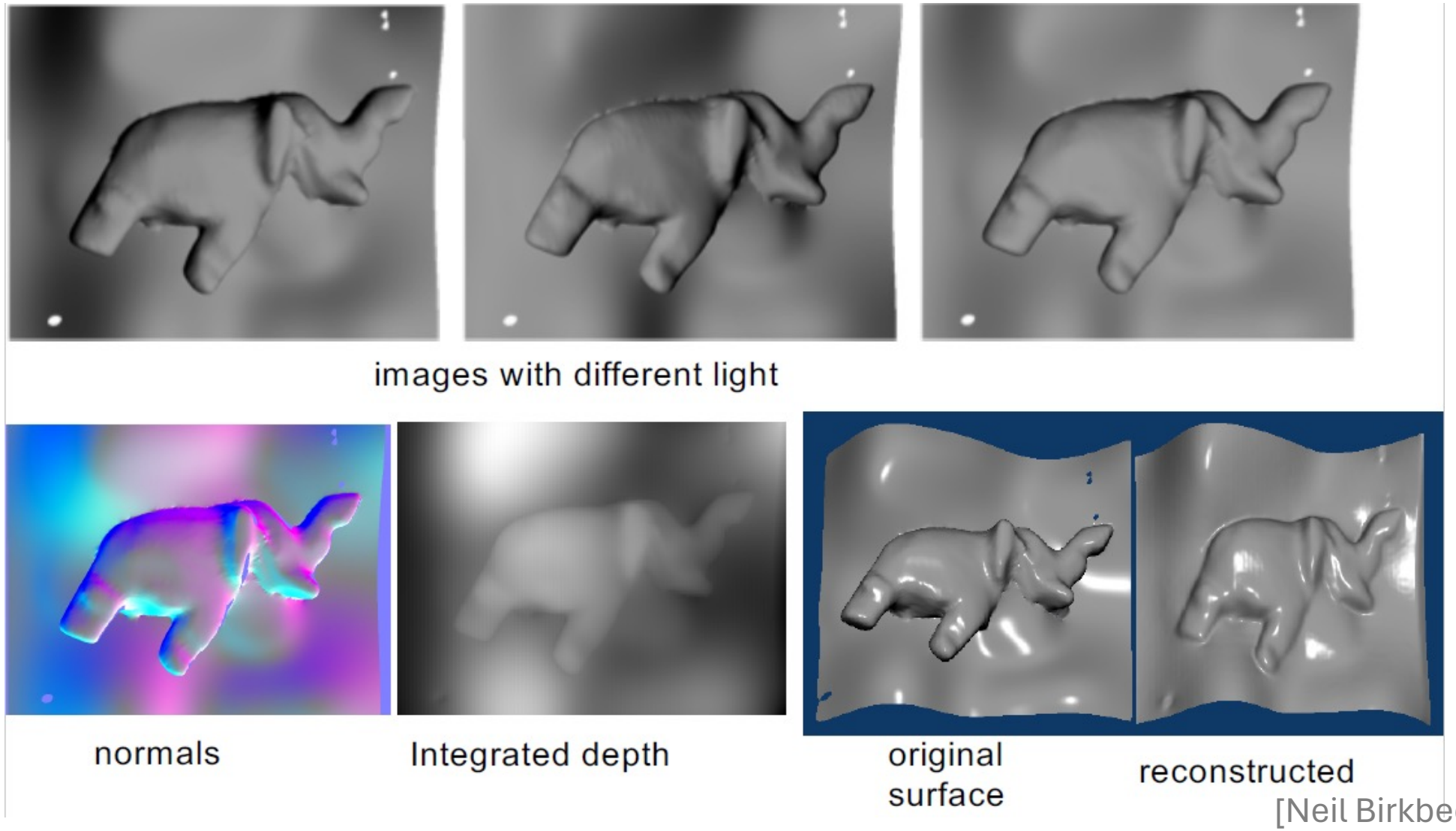
$$D = \iint_{image} \left(\frac{\partial z}{\partial x} - p\right)^2 + \left(\frac{\partial z}{\partial y} - q\right)^2 dxdy$$

  where $\frac{\partial z}{\partial x}$ and $\frac{\partial z}{\partial y}$ are gradients of the estimated surface

- Use the Frankot-Chellappa algorithm [1988] to solve this problem in the Fourier domain (not required in this course)

[Horn, 1990]

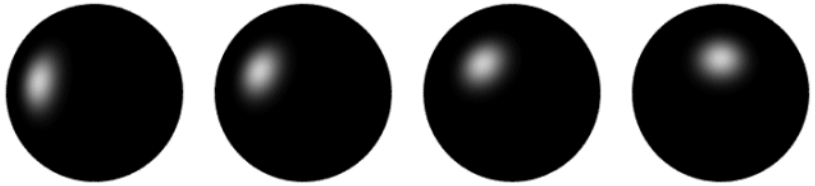
Shape-from-Shading: Results



[Neil Birkbeck]

Determining the Light Source

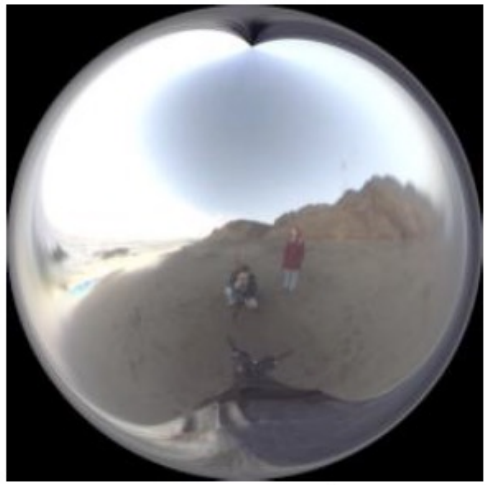
- Trick: Place a mirror ball in the scene



- The location of the highlight is determined by the light source direction

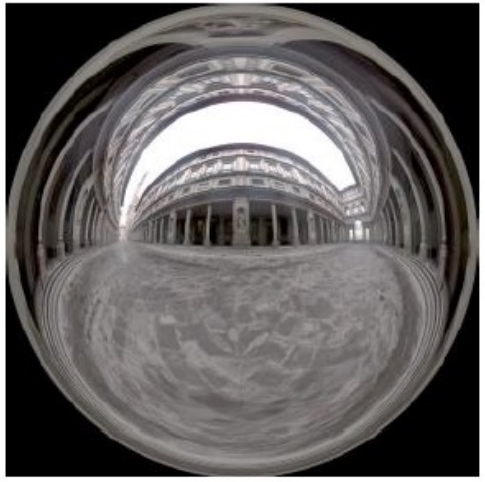
Determining the Light Source

Funston Beach



Eucalyptus Grove

Uffizi Gallery

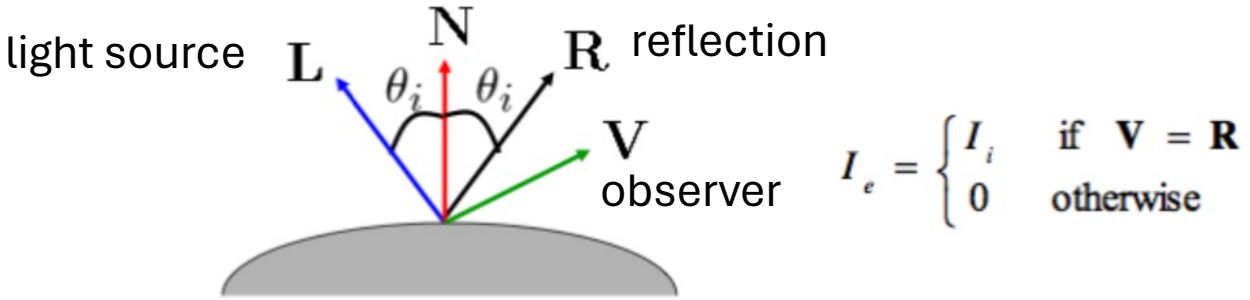


Grace Cathedral

Lighting Environments from the Light Probe Image Gallery:
http://www.debevec.org/Probes/

Determining the Light Source

- For a perfect mirror ball, the light is reflected across N:



- So the light source direction is given by $L = 2(N \cdot R)N - R$
  - $N \cdot R$: Amount of surface reflection projected on the surface normal
  - $(N \cdot R)N$: a vector in the direction of the surface normal, of length $N \cdot R$

Photometric Stereo

- Method of recovering 3D shape / surface normal information from image intensity values measured across multiple light sources
  - Capture images of the scene under different light sources, one at a time
- Key Idea: use intensity differences (shading) to understand shape

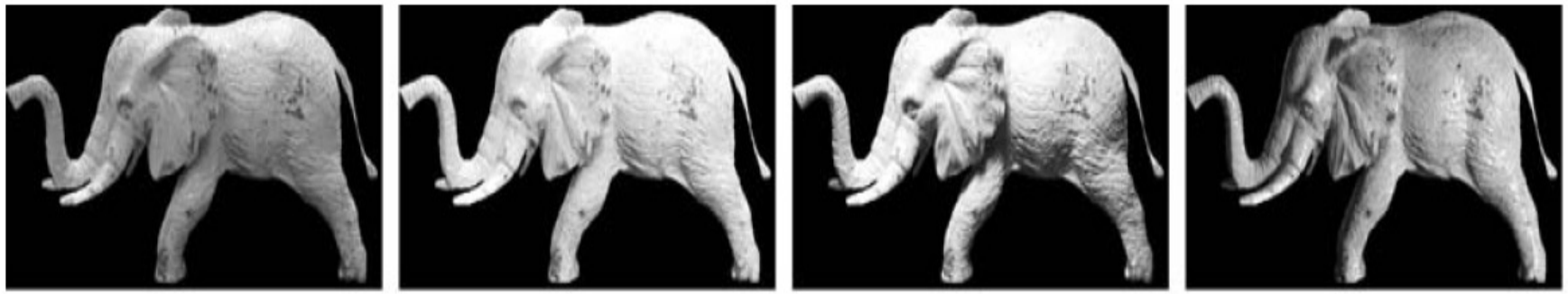


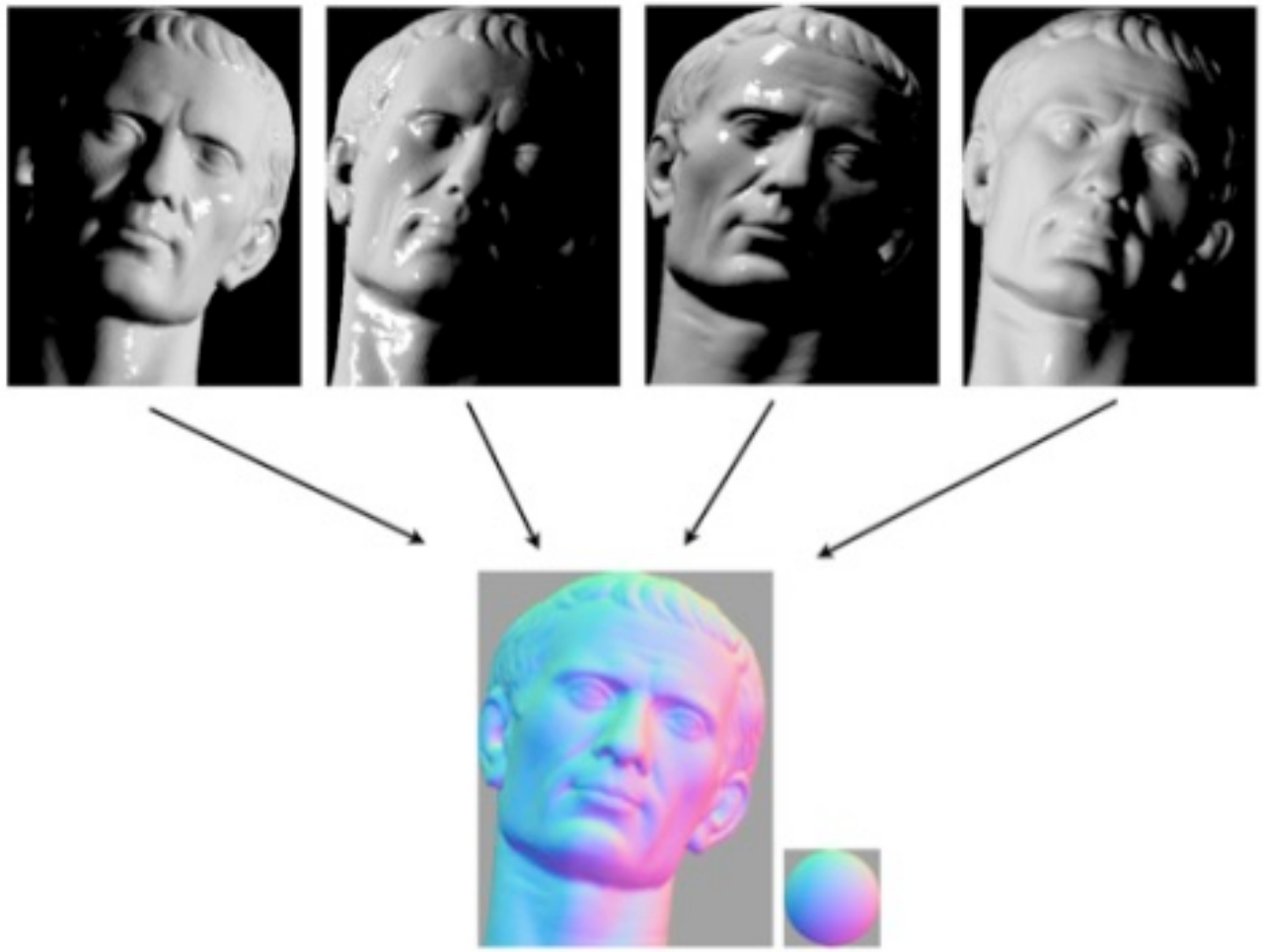
Image Credit: R. Basri et al. “Photometric Stereo with General, Unknown Lighting”, IJCV 2006

Photometric Stereo

Photometric Stereo

Photometric Stereo

Photometric Stereo



Photometric Stereo

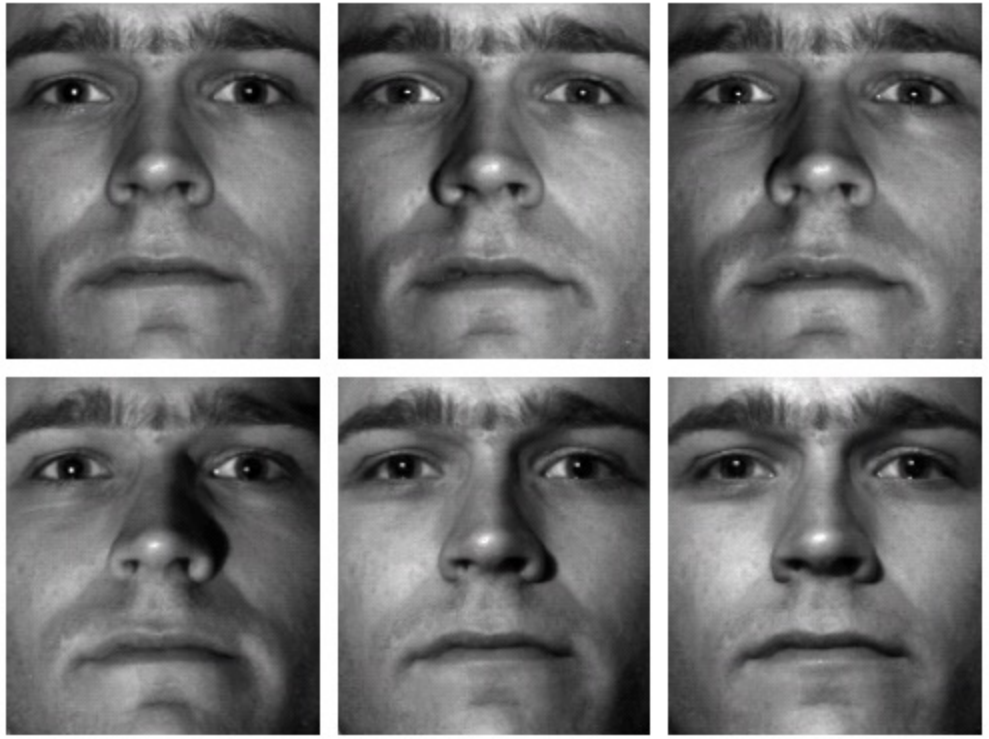
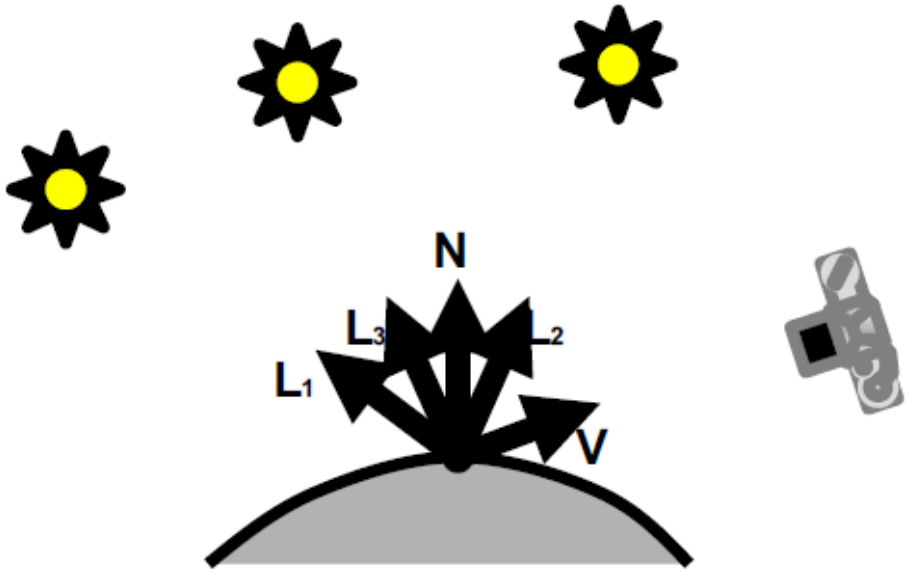


Image Credit: Athos Georghiades http://cvc.yale.edu/people/Athos.html

Photometric Stereo: Algebraic Derivation



$$I_1 = \rho N \cdot L_1$$
$$I_2 = \rho N \cdot L_2$$
$$I_3 = \rho N \cdot L_3$$

- Therefore,
$$[I_1 \;\; I_2 \;\; I_3] = \rho N^{\top} [L_1 \;\; L_2 \;\; L_3]$$

Photometric Stereo: Solving the Equations

$$\underbrace{[I_1 \;\; I_2 \;\; I_3]}_{I} = \underbrace{\rho N^{\top}}_{G} \underbrace{[L_1 \;\; L_2 \;\; L_3]}_{L}$$

Image intensity matrix I is known

I 1x3

G 1x3

L 3x3

Light source matrix L is known

$$G = IL^{-1}$$

- ρ and N are unknowns (ρ may differ across surface)
- Surface normal: $N = \frac{G}{\|G\|}$
- Albedo: $\rho = \|G\|$

Photometric Stereo: Solving the Equations

$$\underbrace{[I_1 \;\; I_2 \;\; I_3]}_{I} = \underbrace{\rho N^\top}_{G}\underbrace{[L_1 \;\; L_2 \;\; L_3]}_{L}$$

I: 1x3 $\quad G$: 1x3 $\quad L$: 3x3

$$G = IL^{-1}$$

- When is L invertible?
  - >= 3 light directions are linearly independent
  - ↔ light direction vectors cannot lie on a plane

Photometric Stereo: Solving the Equations

$$\underbrace{[I_1 \quad \dots \quad I_n]}_{I} = \underbrace{\rho N^{\mathsf{T}}}_{G}\underbrace{[L_1 \quad \dots \quad L_n]}_{L}$$

I: 1xn, G: 1x3, L: 3xn

- More than 3 lights? Solve using least squares:

$$I = GL$$
$$IL^{\mathsf{T}} = GLL^{\mathsf{T}}$$
$$G = (IL^{\mathsf{T}})(LL^{\mathsf{T}})^{-1}$$

- Equivalently use SVD
- Given G, solve for N and ρ as before

Photometric Stereo: Solving the Equations

$$\underbrace{\begin{bmatrix} I_1 & \dots & I_n \end{bmatrix}}_{I} = \underbrace{\rho N^{\top}}_{G} \underbrace{\begin{bmatrix} L_1 & \dots & L_n \end{bmatrix}}_{L}$$

I: mxn, G: mx3, L: 3xn

- More than 1 pixel? Stack into a system and solve as before

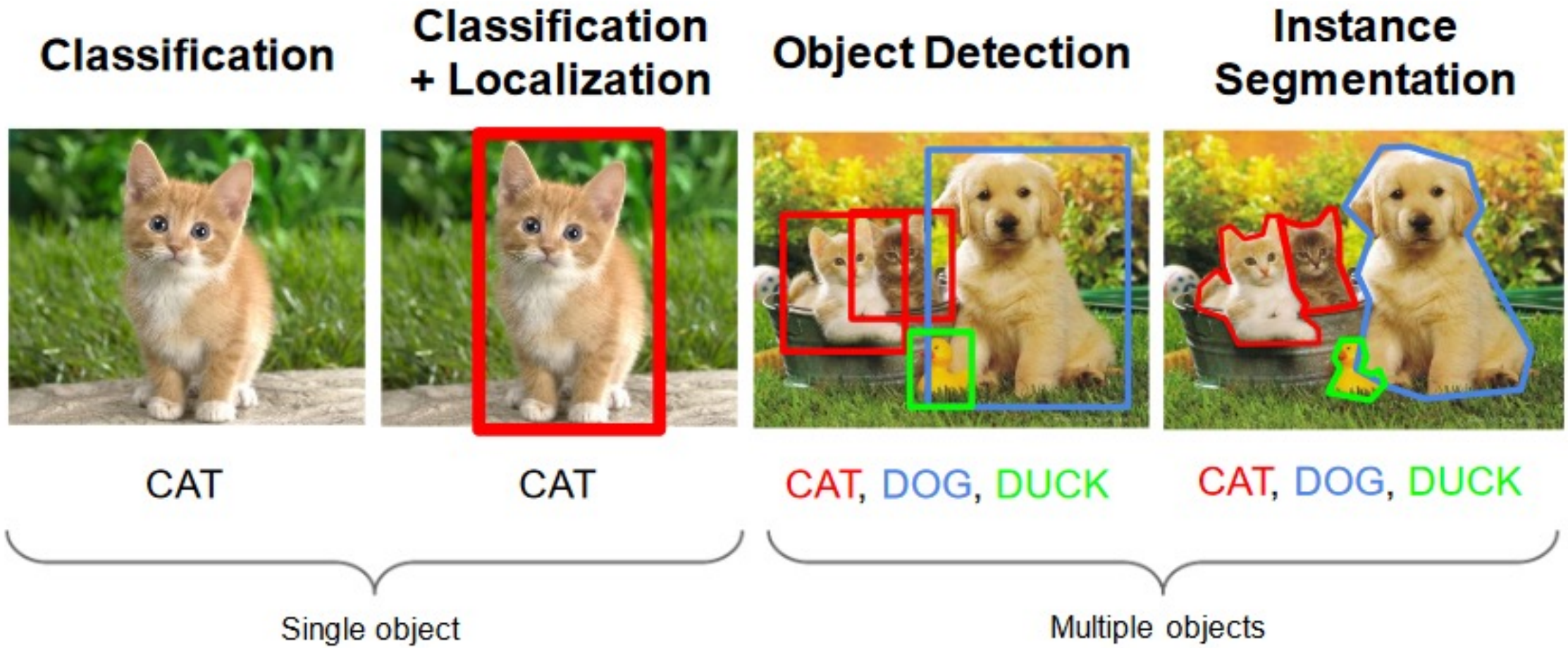
YouTube Video Demo

- https://www.youtube.com/watch?v=tsLTq3MuXNI

Object Detection

High-level Vision

High-level Vision Tasks

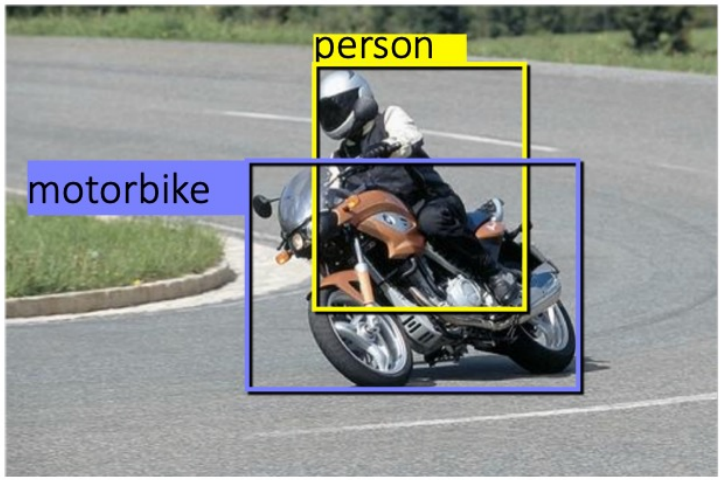


Object Detection

- **Task:** Locate the presence of objects in an image with a bounding box and a class label per object

{ airplane, bird, motorbike, person, sofa }

Input



Desired output

Detection as Regression

DOG, (x, y, w, h)
CAT, (x, y, w, h)
CAT, (x, y, w, h)
DUCK (x, y, w, h)

= 16 numbers

Detection as Regression

DOG, (x, y, w, h)
CAT, (x, y, w, h)

= 8 numbers

Detection as Regression

CAT? NO

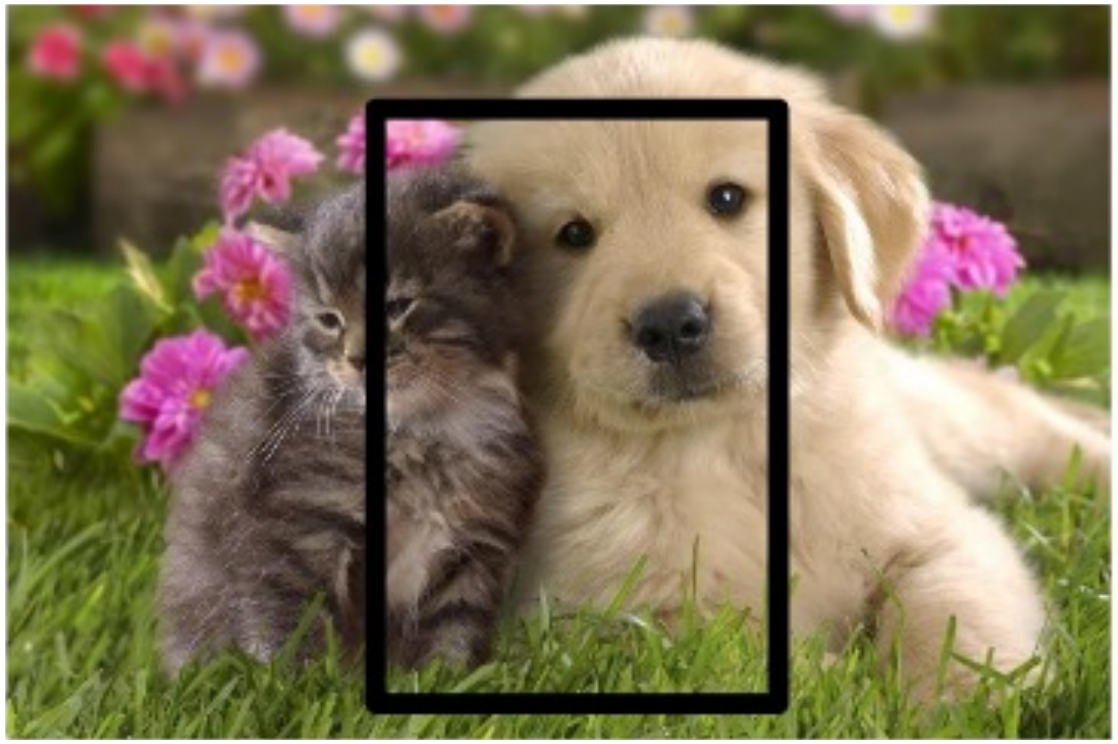
DOG? NO

Detection as Regression

CAT? YES!

DOG? NO

Detection as Regression



CAT? NO

DOG? NO

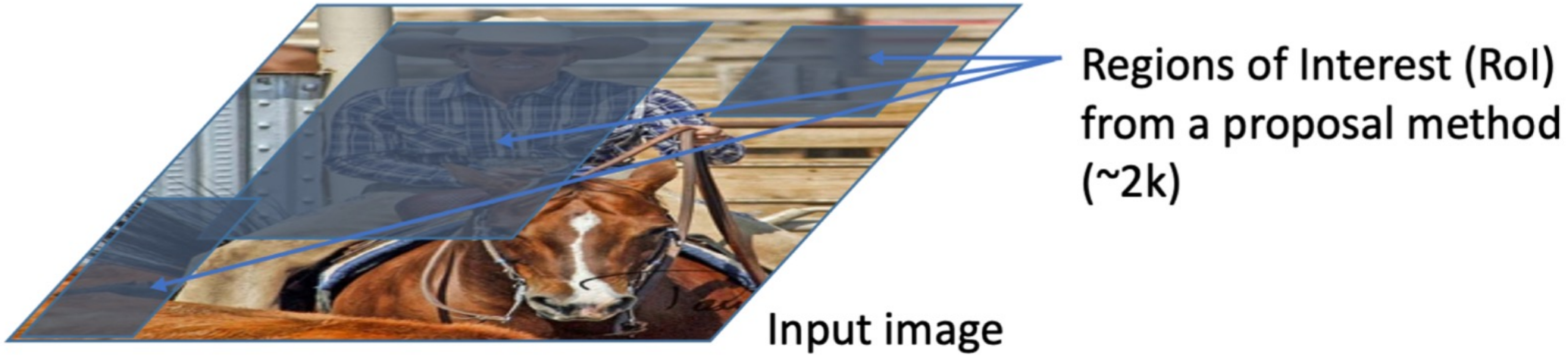
Object Detection: Challenges

- **Multiple Outputs:** Need to output a *variable* numbers of objects per image
- **Multiple Types of Output:** Need to predict “what” (category label) as well as “where” (bounding box)
- **Resolution:** Classification works at 224x224; need higher resolution for detection, often ~800x600

Object Detection

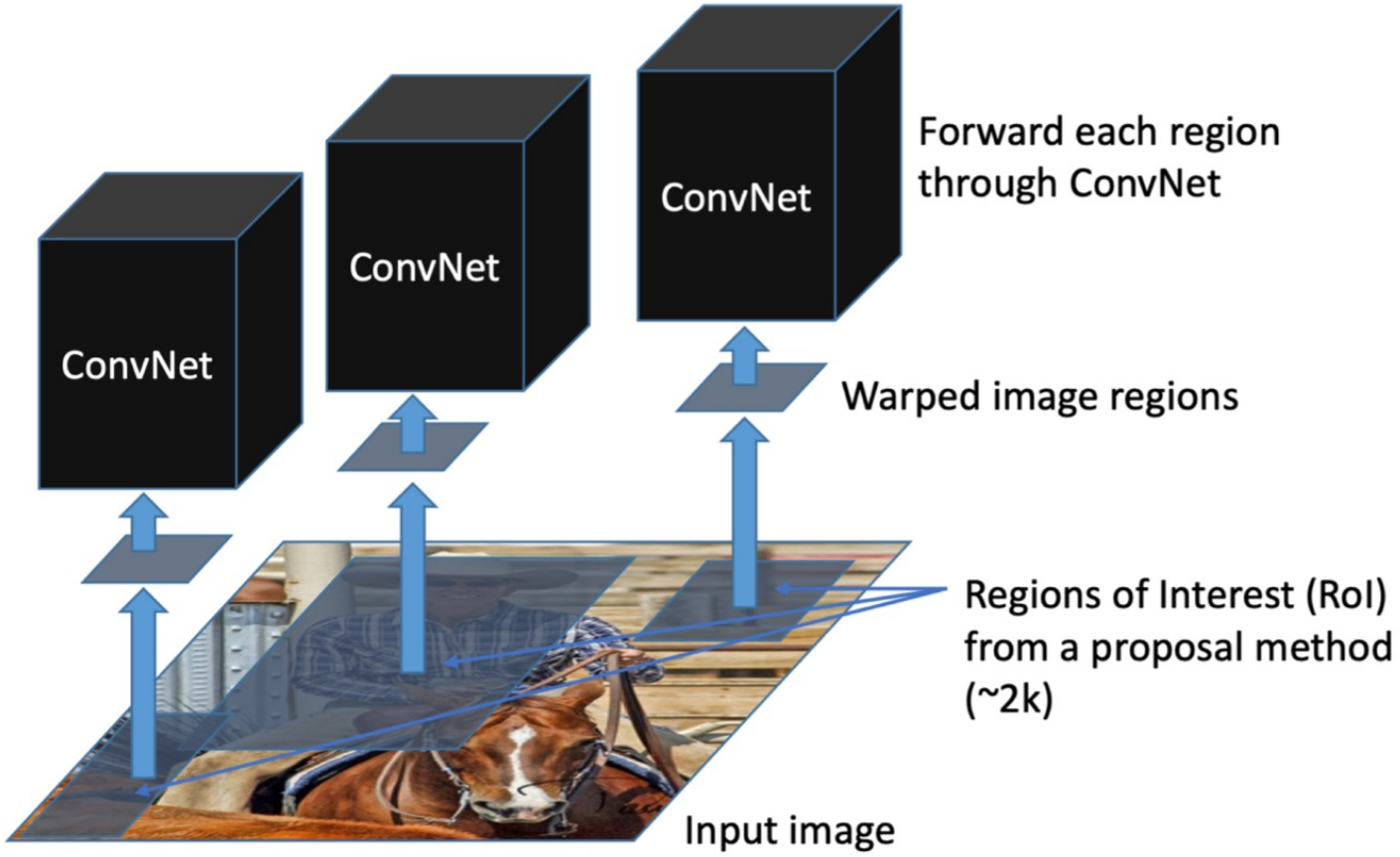
- Problem: Need to test many positions and scales
- Solution: Fast classifier, or region proposals
- Two case studies: RCNN and YOLO

Region CNN (R-CNN) Family

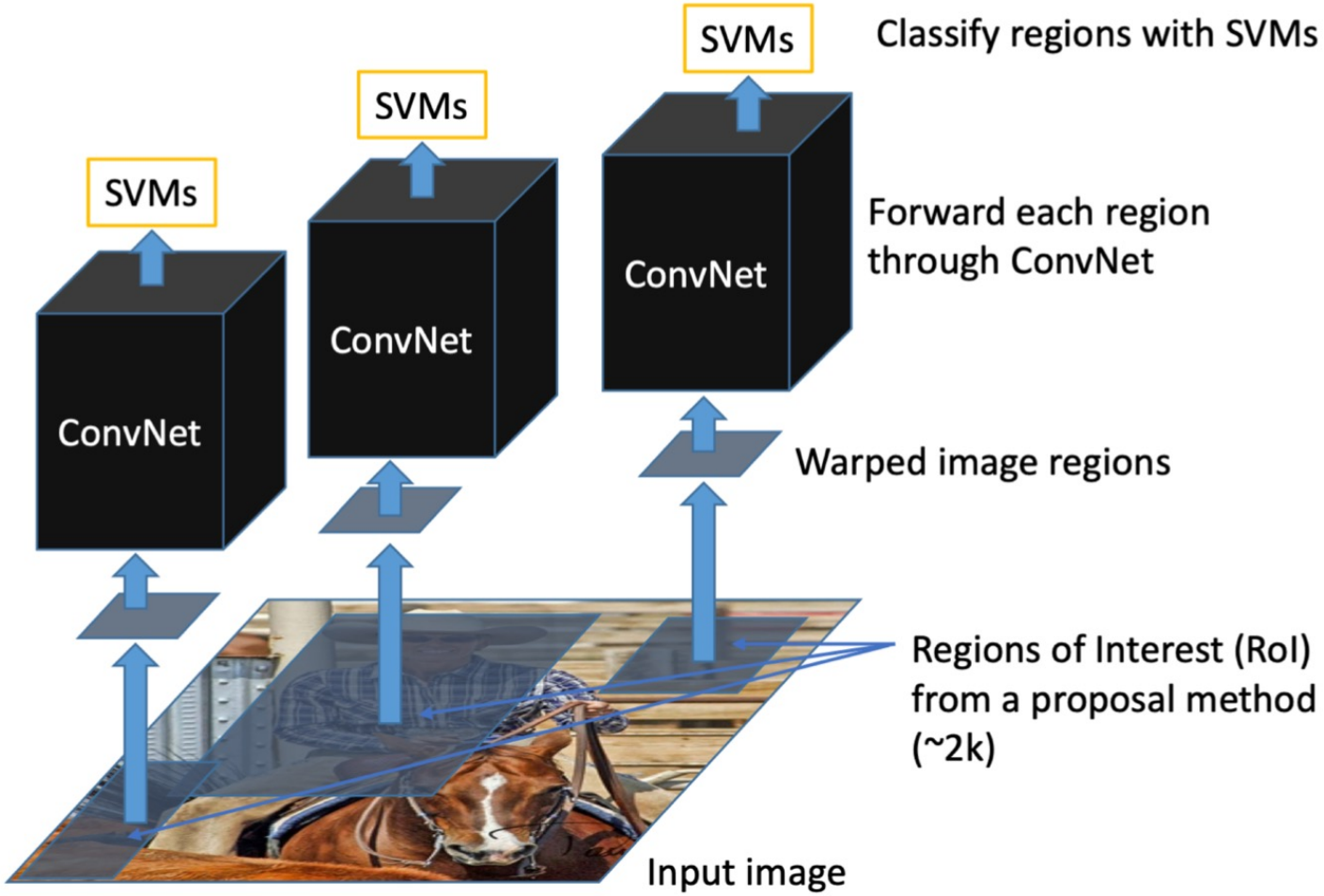


Girshick et al., Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014

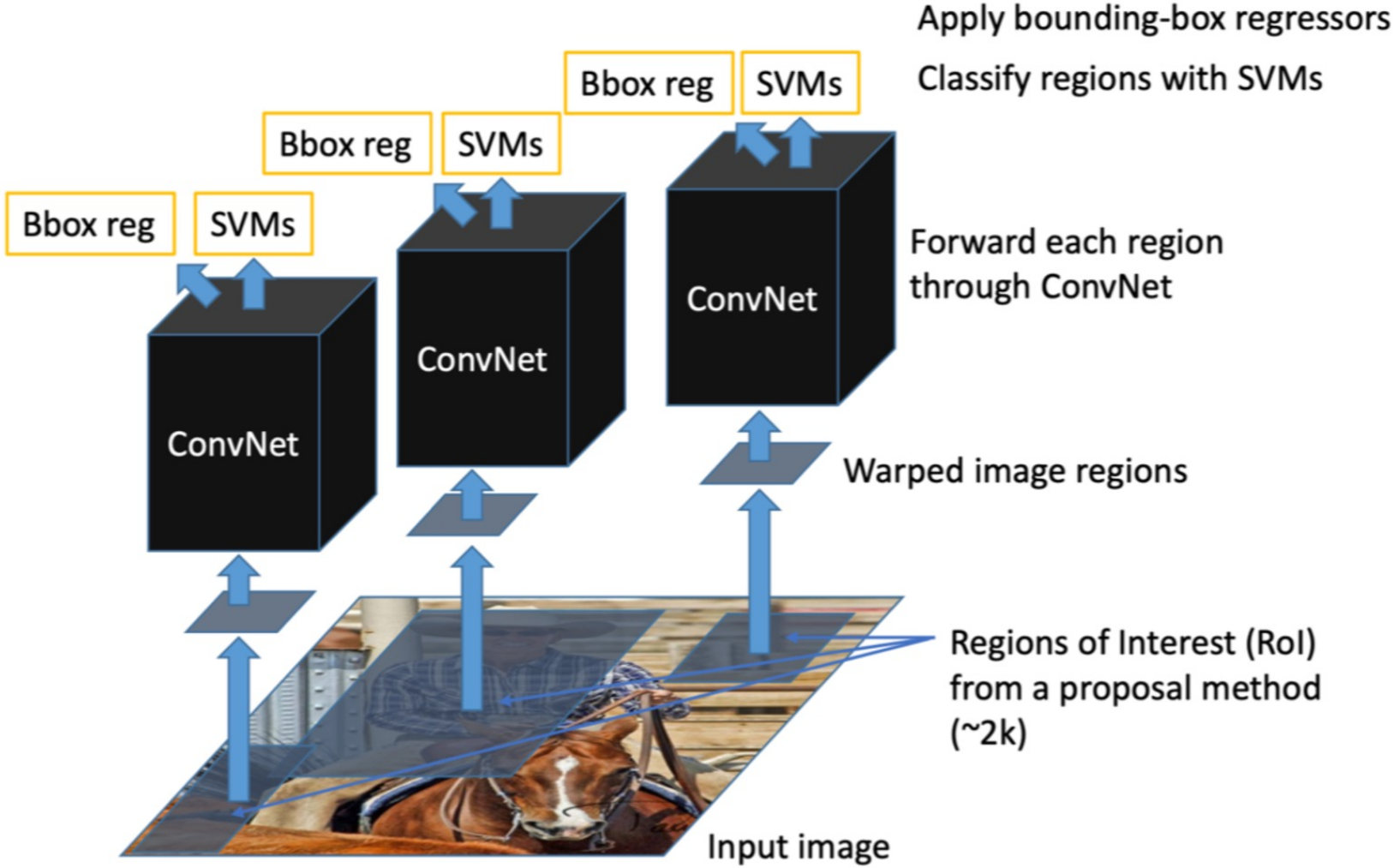
Region CNN (R-CNN) Family



Region CNN (R-CNN) Family

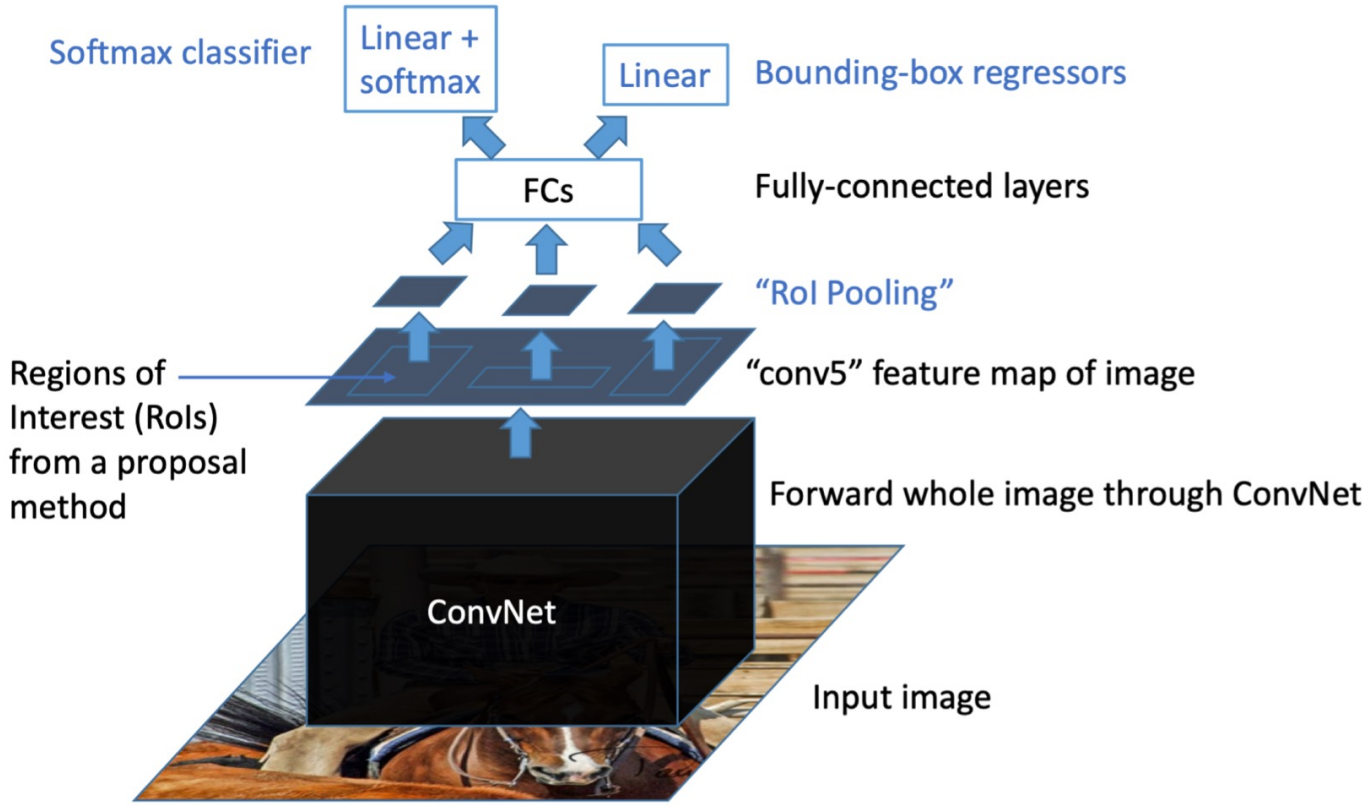


Region CNN (R-CNN) Family



Region CNN (R-CNN) Family: Fast R-CNN

- Single network



Girshick et al., Fast R-CNN, ICCV 2015

Region CNN (R-CNN) Family: Faster R-CNN

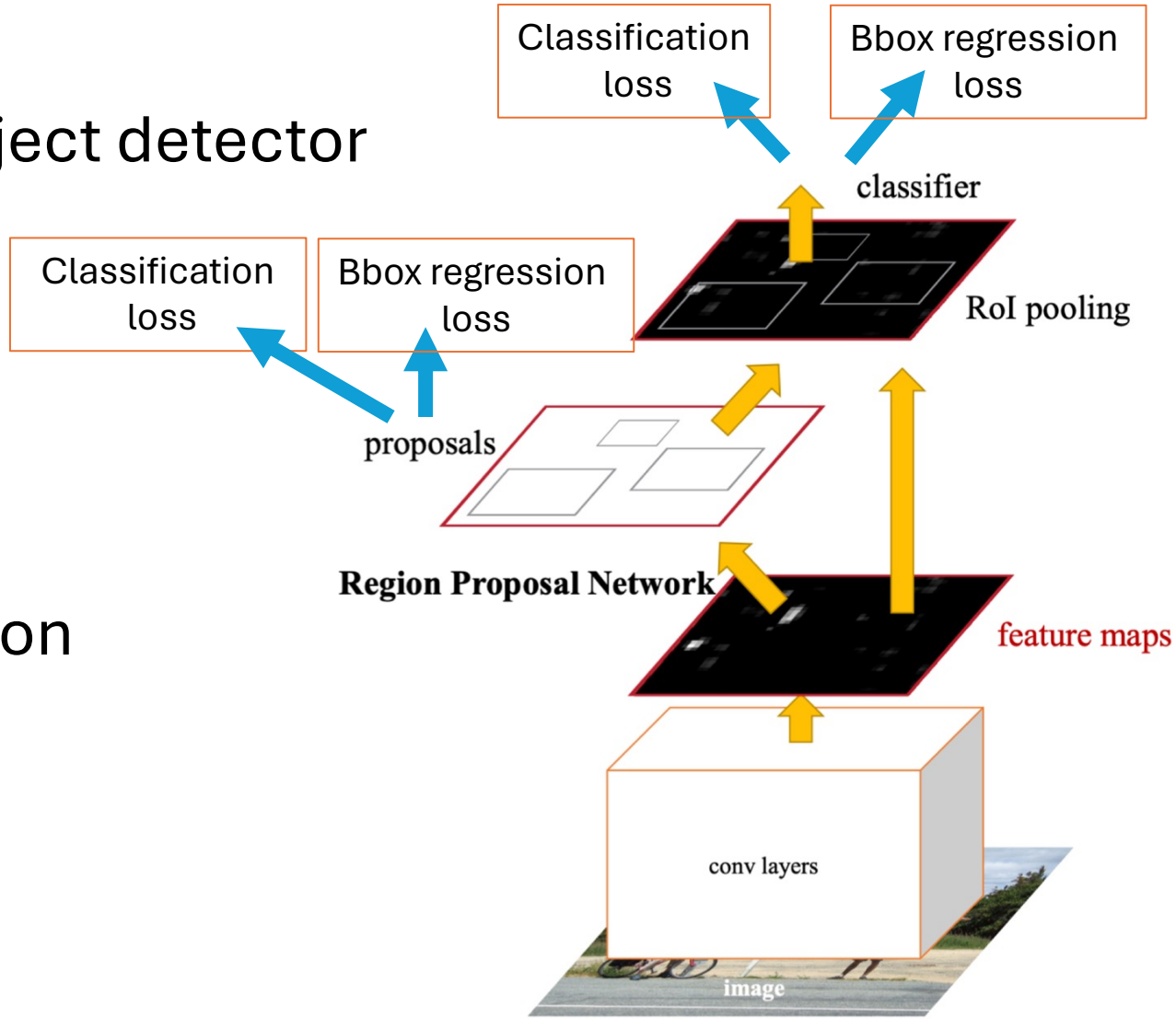
- Faster RCNN is a two-stage object detector

First stage: Run once per image

1. Backbone network
2. Region proposal network

Second stage: Run once per region

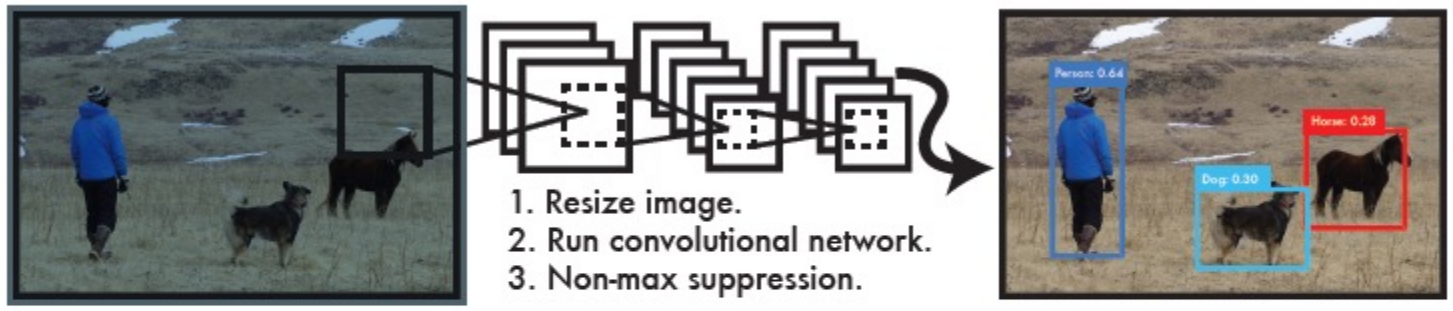
1. Crop features: ROI pool
2. Predict object class
3. Predict bbox offset



Ren et al., Faster R-CNN: Towards real-time object detection with region proposal networks, NeurIPS 2015

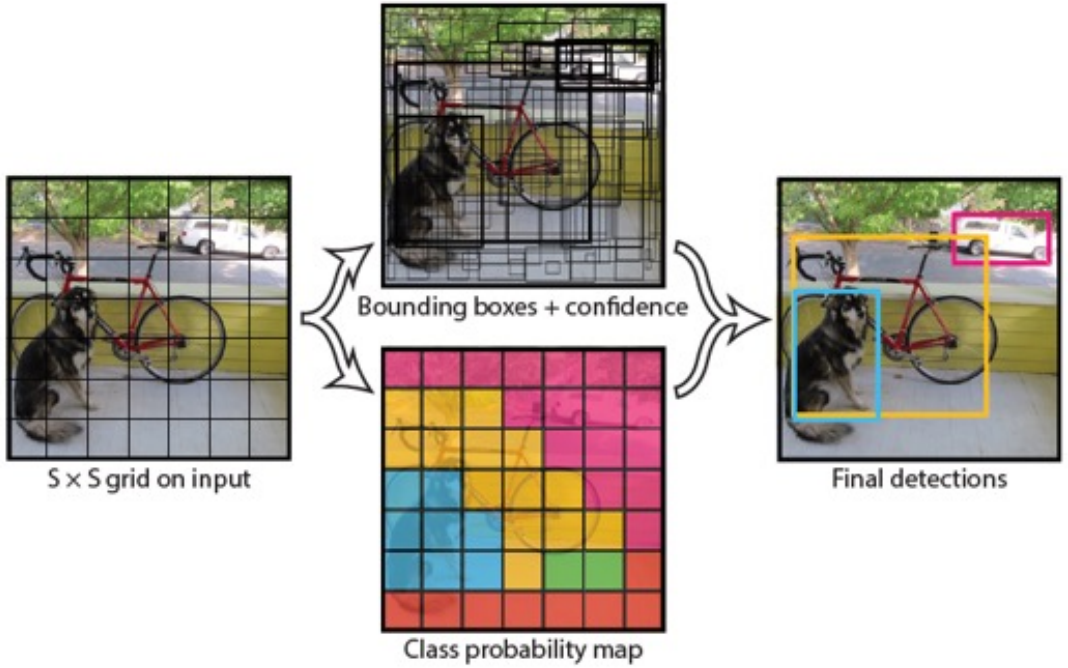
Single-stage Detector: YOLO

- Redmon et al., You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016
- Defines detection as a regression problem
- Also, a general model for handling regression problems



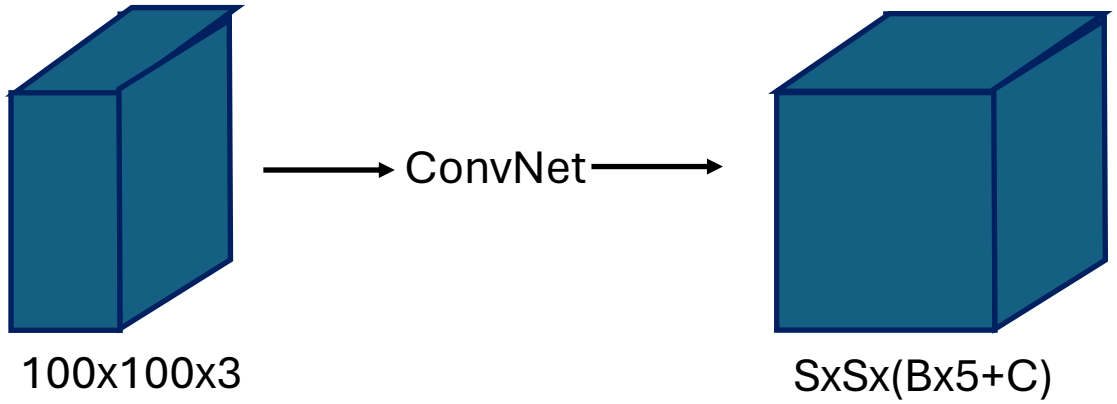
Single-stage Detector: YOLO

- Divides image into an SxS grid
- Each grid cell predicts whether an object centre is in cell
  - Uses whole image to predict bounding box for each object
  - All classes predicted at once
- Each cell predicts:
  - B bounding boxes (x,y,w,h) and confidences (c) for all classes
    - Centre, width, height
  - C class probabilities
    - If no object, all classes predict 0



Single-stage Detector: YOLO

- Output encoding
- Predicted tensor: SxSx(Bx5+C)
  - B bounding boxes: p_b = p(object)
  - C class probabilities: p_c = p(class | object)
  - $[(p_{c1},p_{c2},p_{c3},\ldots, p_{cC}),$
    $(p_{b1},x_1,y_1,w_1,h_1),\ldots,$
    $(p_{bB},x_B,y_B,w_B,h_B)]^T$
- Example: empty cell
  - [(?,?,?, … ?),
    (0, ?, ?, ?, ?), …,
    (0, ?, ?, ?, ?),…]^T



Single-stage Detector: YOLO – Architecture

- Inspired by GoogLeNet
  - 24 convolutional layers, 2 fully connected
- More generally, use features from the convolutional layers of a classification network, and fine tune
  - Pre-train first 20 convolution layers on ImageNet classification, remove classification fully connected layers (transfer learning…)
    - With average pooling and fully connected layers
- Then, add 4 conv layers and 2 fully connected layers
  - Randomly initialised
- Linear activation for final layer (regression, not classification)
- Final layer predicts class probabilities and bound boxes

Single-stage Detector: YOLO – Losses

$\mathbb{1}_i^{obj} = 1$ if an object appears in cell i, otherwise 0.

$\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

$\hat{C}_i$ is the box confidence score of the box j in cell i.

Square root: power normalisation

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Single-stage Detector: YOLO – Losses

- Sum squares
- Square root:
  - Partially normalise for box size
  - Don’t want big bounding boxes to dominate
- Different weight for classification vs localization
  - λ_{coord} is larger
- Different loss if objects present
  - Avoid training null coordinates, just probability
  - λ_{noobj} is smaller

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Single-stage Detector: YOLO – NMS

- Non-maximal suppression (NMS) on output
- Each grid cell only gives best two bounding boxes
- Ignore all low probability bounding boxes
- For each class (e.g., pedestrian):
  - Use NMS on final outputs (limit per class overall)

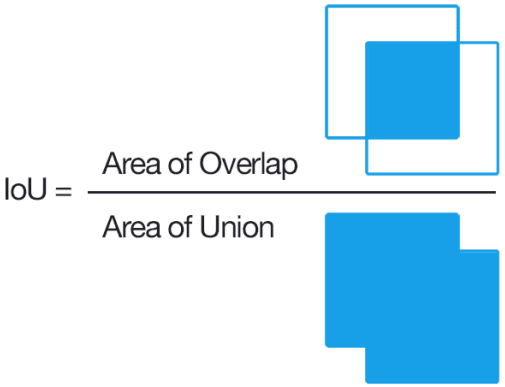
Single-stage Detector: YOLO

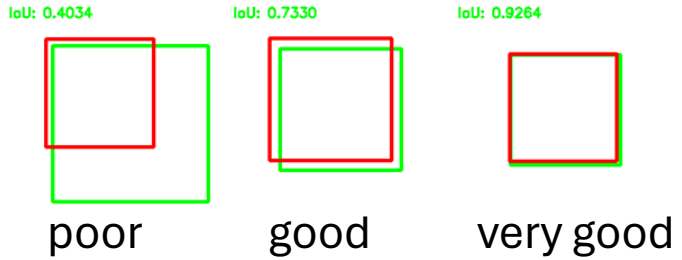
- General approach of bounding boxes is common to all detection architectures
- General approach of regression to estimate continuous numbers in networks
- A good baseline for understanding detection and regression tasks

Evaluating Detectors

- Intersection over Union (IoU)

$$\text{IoU} = \frac{|X \cap Y|}{|X \cup Y|}$$





Next Lecture

- High-level vision: Segmentation