3D Vision 4

Week 10

Mid-level Vision / Two-view Geometry: Optical Flow

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

Announcements

- **Assignment 3** due 11:59pm 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
  - Follow the instructions under Submission Requirements
- **Practice Exam Papers** will be released on Wattle tonight
- Assignment figures/screenshots: must be sufficient resolution for the markers to read and interpret

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

Survey 2 Feedback

- I feel immensly unprepared for a written **final exam**. The assignment's have not given me confidence in being prepared for this course. Assignments have taken a long time to complete to a saticfactory level. I feel like there should be an assignment or mid sem which would be able to prepare us for the final exam as this course has a lot of material which requires deep understanding to execute.
- I feel like the questions asked in tutorials have not been very insightful/interesting and feel apprehensive that the final exam will feature questions of similar difficulty (that is, **tutorial questions seem too easy**). Furthermore, the requirement of us to copy code into the reports but also have us **document the code** seems to place unnecessary additional work onto us, and in my experience, has taken away from the enjoyment of completing the assignments.
- **Feedback** was provided but wasn't in depth, poorly formatted and generic. Assignments are cooked and are too far from what is taught in class / tutorials.
- 1. Some of the questions in the first assignment of this course have been slightly adjusted, but the **grading criteria** are still the same. This resulted in unfair grading due to one of the lectures' lack of effective communication and **refusal to re-grade**.
  2. The coursework places too much emphasis on **rigid format** rather than substance.

Survey 2 Feedback

- The **programming** part of this course is **extremely hard** for engineering student. As a major course of engineering student, it didn't consider the capability for engineering student. The **perquisite** of the course listed on the website doesn't seem to have connection with this course.
- It was told that if we havent completed the course **'introduction to machine learning'** that we should drop this course and do that one first however for all the engineering students having to take this as a mandatory course, there is no room in our degree to go back and retake. The assumed knowledge from the previous engineering courses have not prepared us in the slightest for the content in this course and from speaking with fellow engineering students it seems that the **coding** aspect of this course has proven more challenging than expected.
- I feel as if the **weightings** of the assignments has been **too low** given how much work they are. This is particularly true for us ENGN students who are not expert programmers .

Survey 2 Feedback

- The course's lab training **lacked explanation** and did not help me understand it better. The course content has a lot of knowledge and is a bit overwhelming to review, I need to gather a lot of **information outside of class** to understand the course and assignments.
- Well, I find this course require a lot of knowledge We need to **learn outside the lectures**, it is quite challenging for me.
- I hope there are some sample questions or **past papers** to help us prepare for the final exam
- I cannot understand the second half part of course by Dylan.
- I don't understand lectures at all, also the lab or homework at all.
- This course is difficult. And if we can get solutions and explanations for assessments, it would be better to learn it.
- The replies from ed are **always so slow** that students cannot get timely answers. Besides, the description of some questions in the qssignment is very abstract for us to understand. Usually, I cannot get a useful and meaningful answer if I have questions from tutors in the lab time.
- **workload for assignment** is so large that I could not spend enough time to catch lectures

Outline

1. Mid-level Vision / Two-view Geometry: Optical Flow
2. Mid-level Vision / Single-view Geometry: Shape-from-X

Optical Flow

Mid-level Vision / Two-view Geometry

Optical Flow

- **Problem:** Compute a flow field that takes pixels in the first image to their location in the second image.

$$(dx, dy) = f\big(I(t), I(t + dt)\big)_{(x,y)}$$

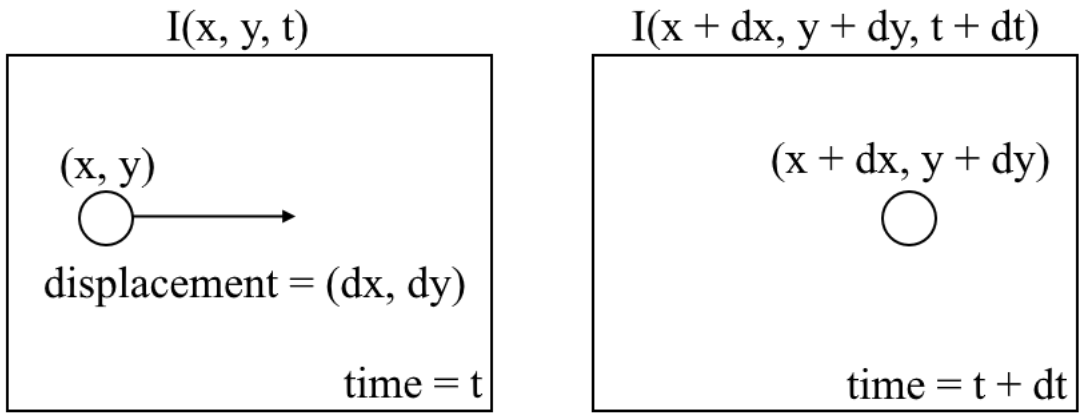


Image Credit: Chuan-en Lin

Optical Flow

- **Problem:** Compute a flow field that takes pixels in the first image to their location in the second image.

Image Credit: MPI Sintel dataset

Optical Flow

- **Problem:** Compute a flow field that takes pixels in the first image to their location in the second image.

Image Credit: MPI Sintel dataset

Optical Flow

- **Problem:** Compute a flow field that takes pixels in the first image to their location in the second image.

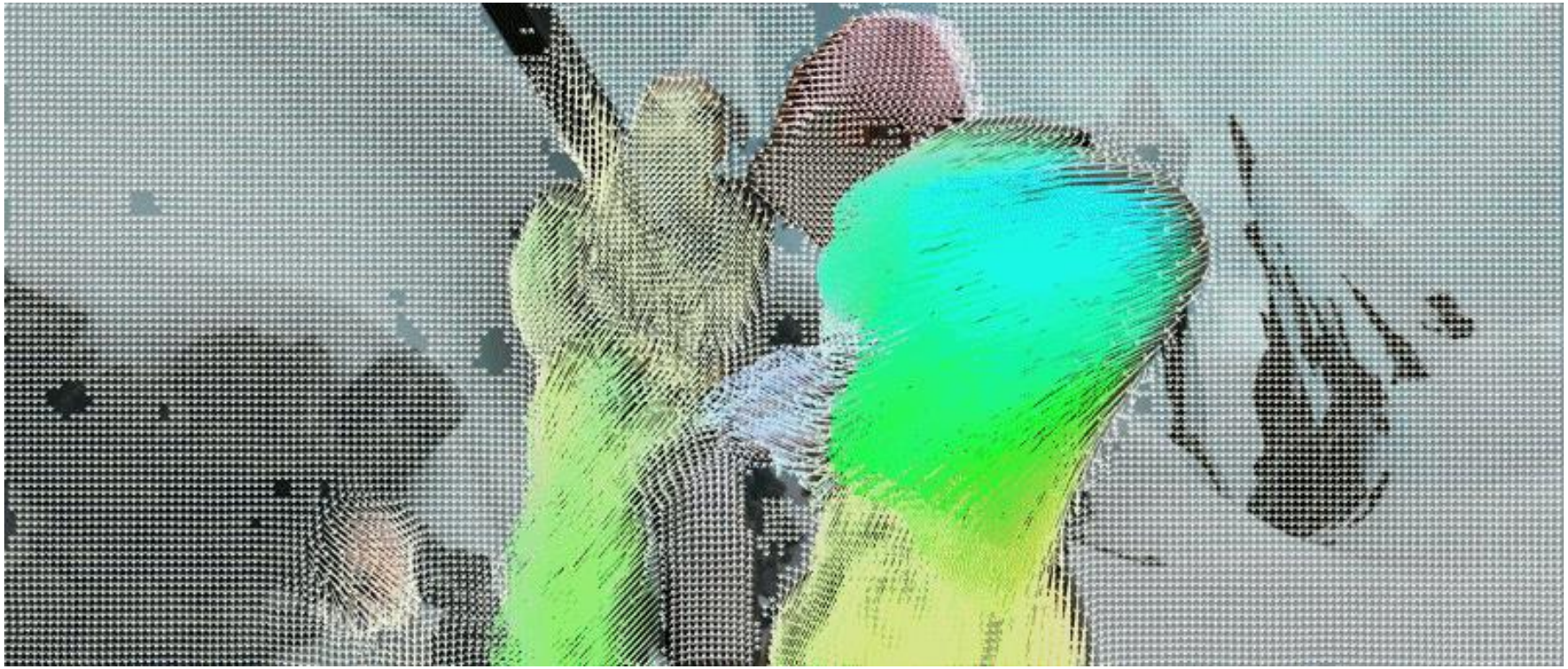


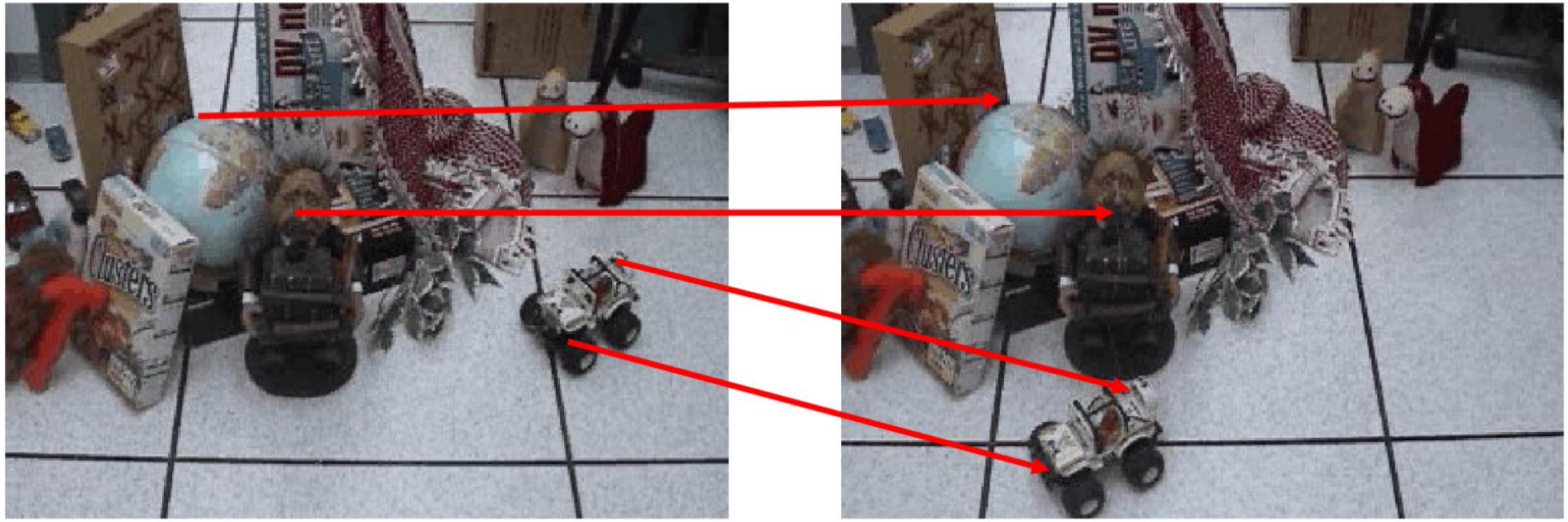
Image Credit: Credit: Yang & Ramanan, “Volumetric Correspondence Networks for Optical Flow”, NeurIPS, 2019

Optical Flow

- **Ideal case:** Projection of the 3D motion field onto the 2D image
- **What is it for?** Object detection and tracking, motion segmentation, action classification, navigation and odometry (e.g., an optical mouse)
- **Types:** Rigid, piecewise rigid, and non-rigid flow
- **Possible Assumptions:** Colour/brightness constancy, smoothness, static scene (camera motion only)
- **Challenges:** Aperture problem; if you only use local information (which is necessary for non-rigid flows), then information related to the component of motion parallel to any edge is lost

Optical Flow Estimation

- Where does each pixel in image 1 go to in image 2?

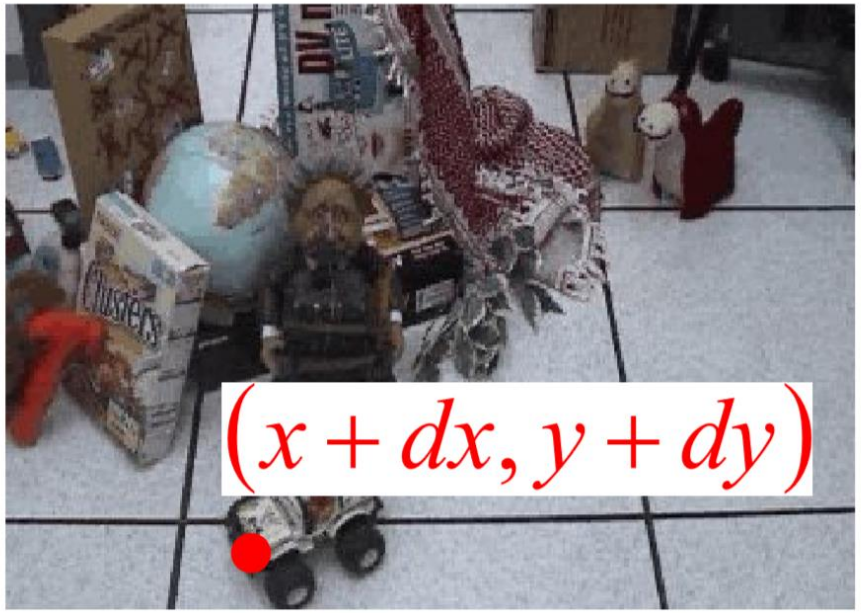


Optical Flow Estimation

- Assume that the image brightness of the corresponding point is constant over two frames

Time = t

Time = $t+dt$

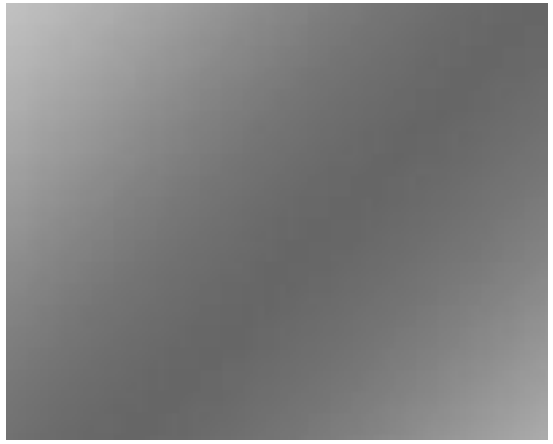


$$I(x,y,t) = I(x+dx, y+dy, t+dt)$$

Brightness/Colour Constancy Assumption

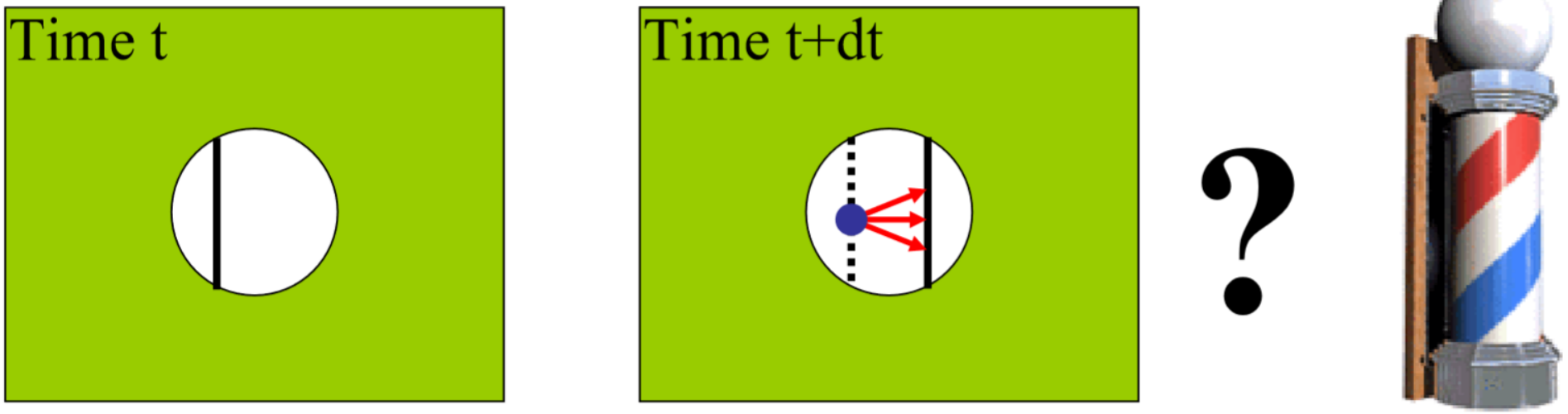
$$I_x u + I_y v = -I_t \leftrightarrow \frac{\partial I}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial I}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{\partial I}{\partial t}\frac{\mathrm{d}t}{\mathrm{d}t}$$

- From first-order Taylor expansion of $I(x, y, t) = I(x + \mathrm{d}x, y + \mathrm{d}y, t + \mathrm{d}t)$
  - Note that all partial derivatives are evaluated at (x, y, t), e.g., $I_x(x, y, t)$
- We have one equation and two unknowns (u, v): the optical flow
- Assumes a smooth visual gradient over the area of motion



The Aperture Problem

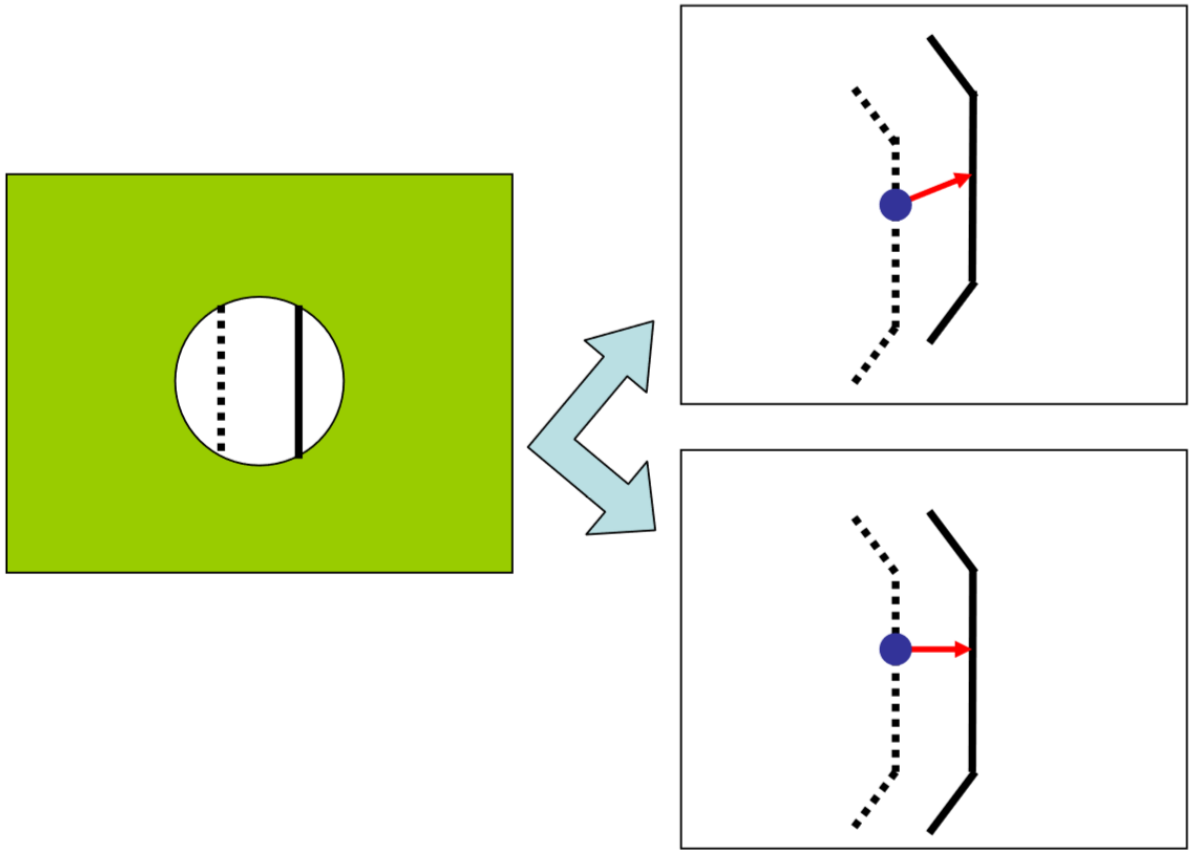
- For points on a line of fixed intensity we can only recover the normal flow (normal to max gradient)



- Where did the blue point move to?
- We need additional constraints

The Aperture Problem: A Remedy

- Use local information: sometimes enlarging the aperture can help

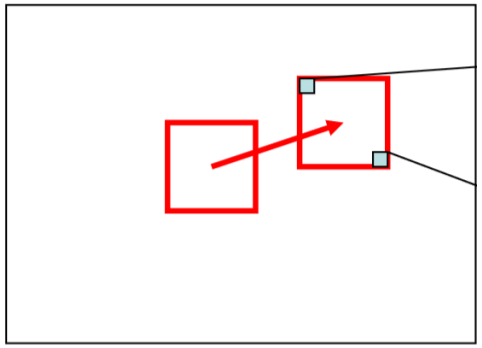


Lucas–Kanade Optical Flow Algorithm

$$I_x u + I_y v = -I_t \leftrightarrow \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t$$

- Assumes **constant motion** within a **small neighbourhood** (patch)
- A 5 × 5 image patch gives us 25 equations

$$I_x(\boldsymbol{p}_1, t)u + I_y(\boldsymbol{p}_1, t)v = -I_t(\boldsymbol{p}_1, t)$$
$$\vdots$$
$$I_x(\boldsymbol{p}_{25}, t)u + I_y(\boldsymbol{p}_{25}, t)v = -I_t(\boldsymbol{p}_{25}, t)$$
$$\therefore A\boldsymbol{u} = \boldsymbol{b}$$



- Least squares solution: $\boldsymbol{u}^\star = \text{argmin}_{\mathbf{u}} \|A\boldsymbol{u} - \boldsymbol{b}\|^2$
  - $\boldsymbol{u}^\star = \left(A^\top A\right)^{-1} A^\top \boldsymbol{b}$ [why?]

Lucas & Kanade, 1981

Lucas–Kanade Optical Flow Algorithm

$$[I_x \quad I_y] \begin{bmatrix} u \\ v \end{bmatrix} = -I_t$$

- Least squares solution: $\boldsymbol{u}^\star = \operatorname{argmin}_{\mathbf{u}} \|A\boldsymbol{u} - \boldsymbol{b}\|^2 = \left(A^\top A\right)^{-1} A^\top \boldsymbol{b}$
- $A^\top A = \begin{bmatrix} \sum_{\boldsymbol{p}\in\mathcal{P}} I_x I_x & \sum_{\boldsymbol{p}\in\mathcal{P}} I_x I_y \\ \sum_{\boldsymbol{p}\in\mathcal{P}} I_y I_x & \sum_{\boldsymbol{p}\in\mathcal{P}} I_y I_y \end{bmatrix} \in \mathbb{R}^{2\times 2}$
- $A^\top \boldsymbol{b} = -\begin{bmatrix} \sum_{\boldsymbol{p}\in\mathcal{P}} I_x I_t \\ \sum_{\boldsymbol{p}\in\mathcal{P}} I_y I_t \end{bmatrix} \in \mathbb{R}^{2\times 1}$
  where the sum is over pixels $\boldsymbol{p}$ in patch $\mathcal{P}$
- You saw this in Harris corner detector!

It denotes that corners are better points to be tracked.

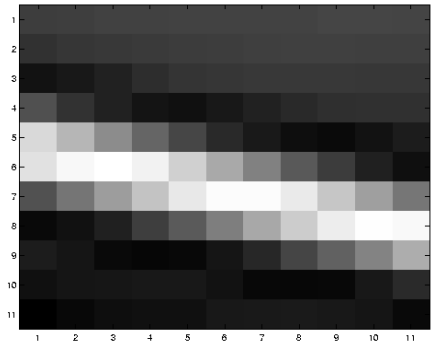
Lucas & Kanade, 1981

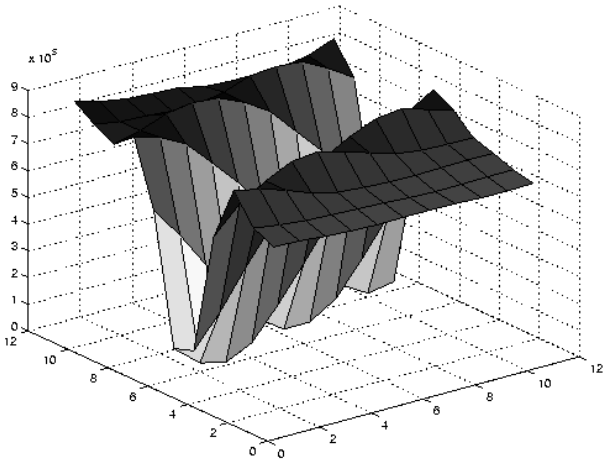
Lucas–Kanade Optical Flow Algorithm

- When is $A^\mathsf{T}A\boldsymbol{u} = A^\mathsf{T}\boldsymbol{b}$ solvable?
  - $A^\mathsf{T}A$ invertible:
    - Determinant non-zero
  - $A^\mathsf{T}A$ not too small, otherwise estimate is sensitive to noise:
    - Eigenvalues λ_1 and λ_2 of $A^\mathsf{T}A$ should not be too small
  - $A^\mathsf{T}A$ well-conditioned:
    - λ_1/λ_2 should not be too large (λ_1: larger eigenvalue)
- Implications:
  - Harris corners are where λ_1 and λ_2 are both big; this is also when Lucas–Kanade optical flow estimation works best
    - Corners are good places to compute optical flow

Lucas & Kanade, 1981

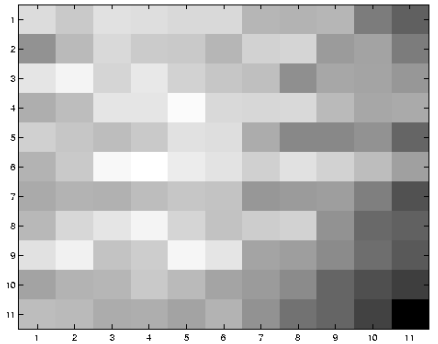
Issues: Edges

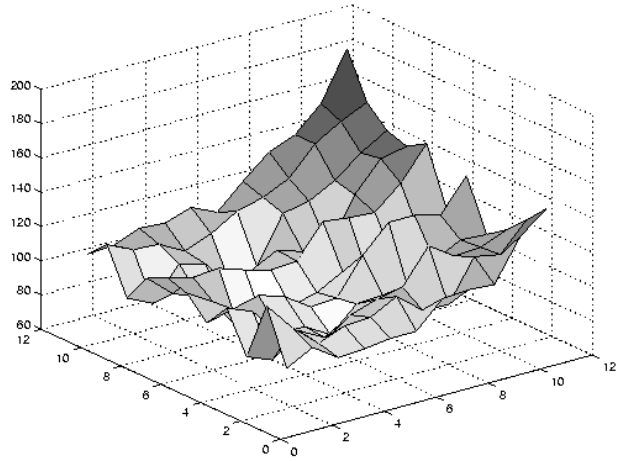




- $A^\top A$:
  - Large gradients, all the same
  - Large λ_1, small λ_2

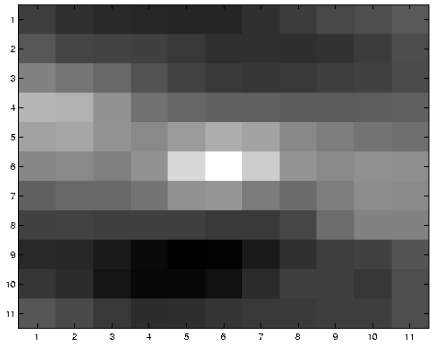
Issues: Low Texture Regions

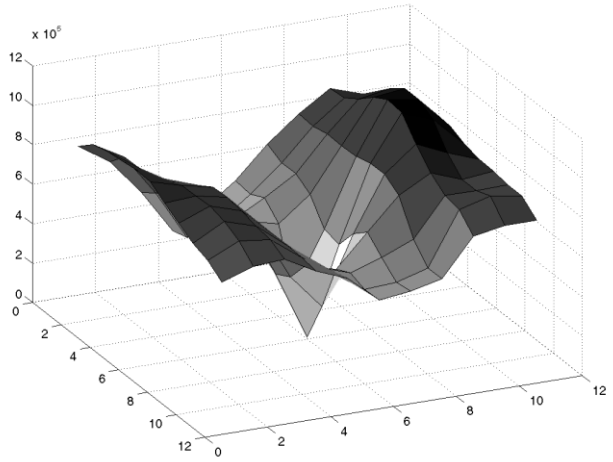




- $A^\top A$:
  - Small magnitude gradients
  - Small λ_1, small λ_2

Ideal: High Texture Regions





- $A^\top A$:
  - Large magnitude, different gradients
  - Large λ_1, large λ_2

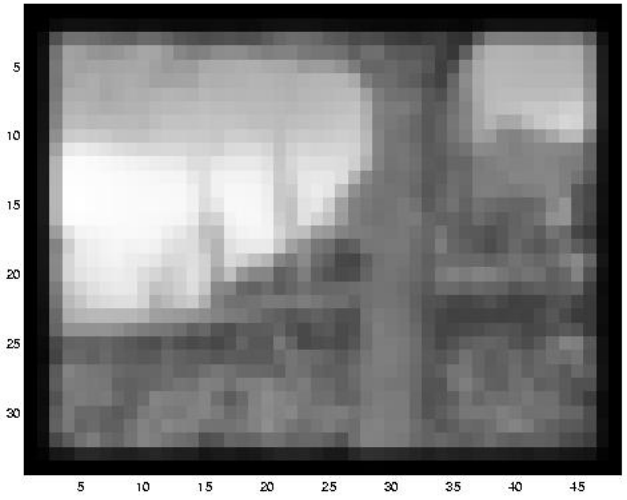
Lucas–Kanade Optical Flow Algorithm: Errors

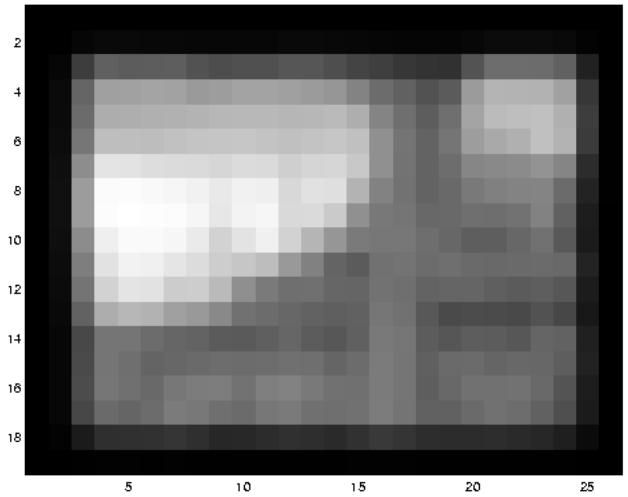
- What are the potential causes of errors in this procedure?
  - Suppose $A^\top A$ is easily invertible
  - Suppose there is not much noise in the image
- When our assumptions are violated:
  - Brightness constancy is not satisfied
  - The motion is not small
  - A point does not move like its neighbours
    - Window size is too large
    - What is the ideal window size?

Revisiting the Small Motion Assumption

- Is this motion small enough?
  - Probably not—it's much larger than one pixel (2^{nd}-order terms dominate)
  - How might we solve this problem?

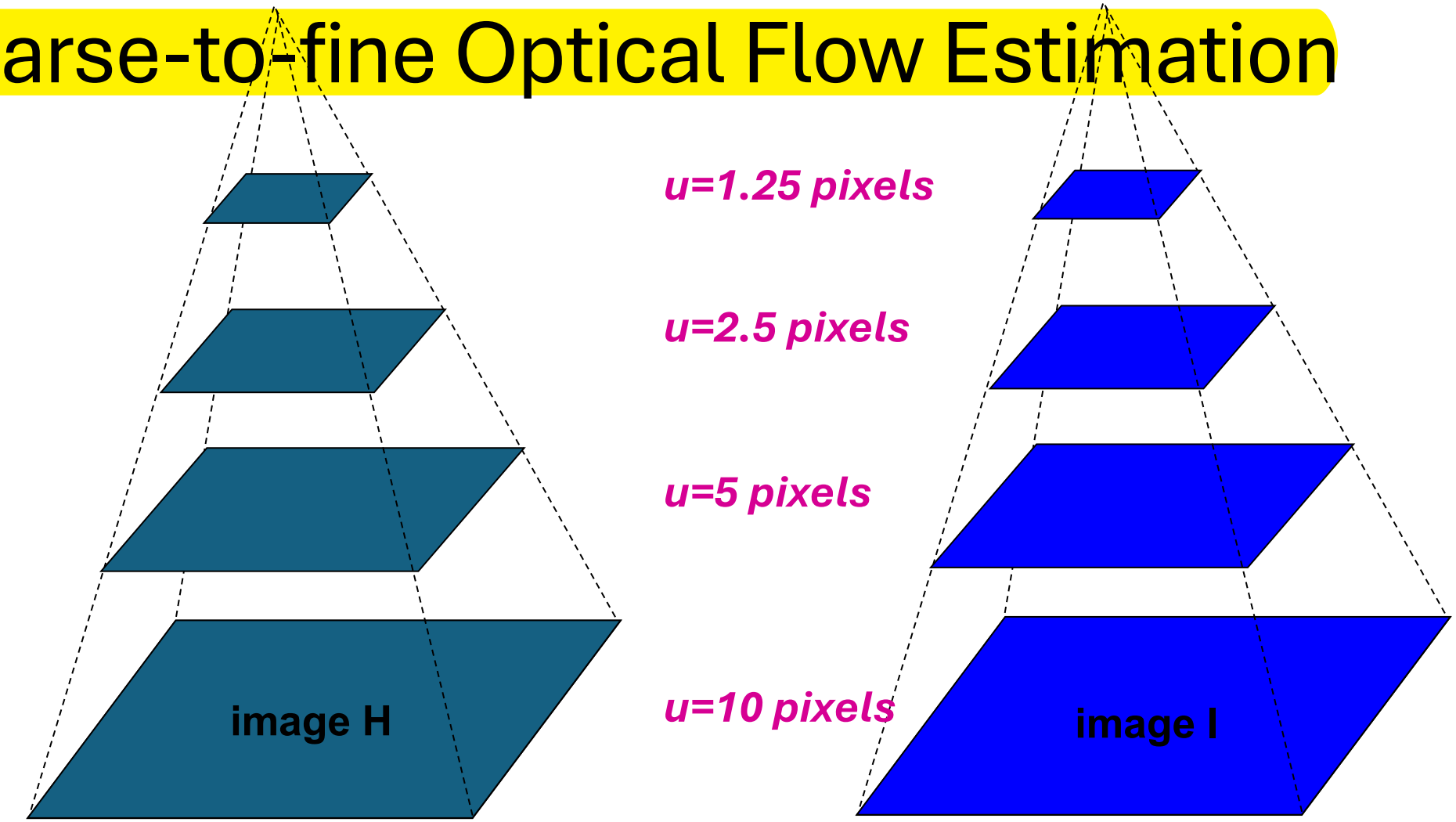
Reduce the Resolution!





All motions are less than 1 pixel

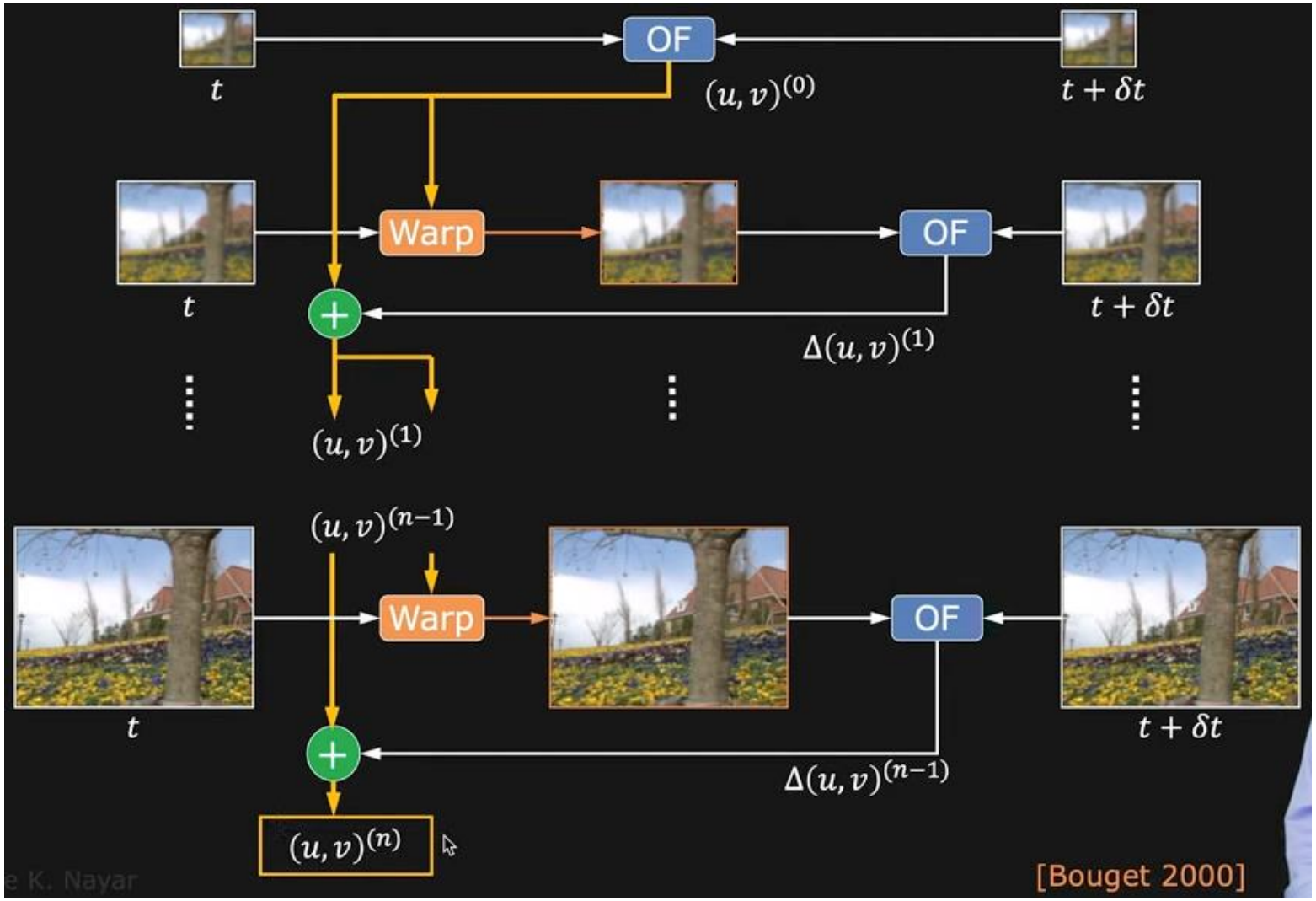
Coarse-to-fine Optical Flow Estimation



Gaussian pyramid of image H **Gaussian pyramid of image I**

- The motion shift is 10 pixels at the lowest level, and is halved each time it goes to an upper level; in this way we reduce the amount of motion that is present

Coarse-to-fine Optical Flow Estimation



Optical Flow Estimation: Summary

- A flow field that takes pixels in the first image to their location in the second image: ideally, projection of the 3D motion field into 2D
- Solve: assume constant motion within a local patch → least squares LK algorithm (or neural networks designed to do similar)

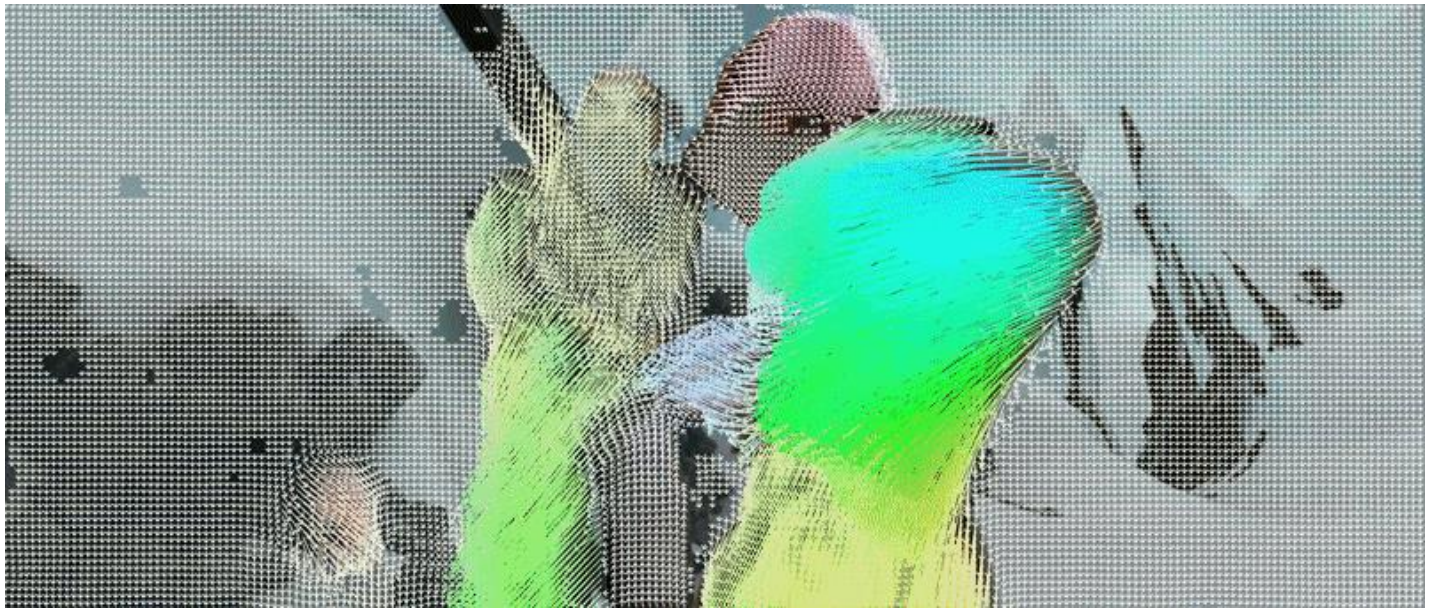


Image Credit: Credit: Yang & Ramanan, “Volumetric Correspondence Networks for Optical Flow”, NeurIPS, 2019

Shape-from-X

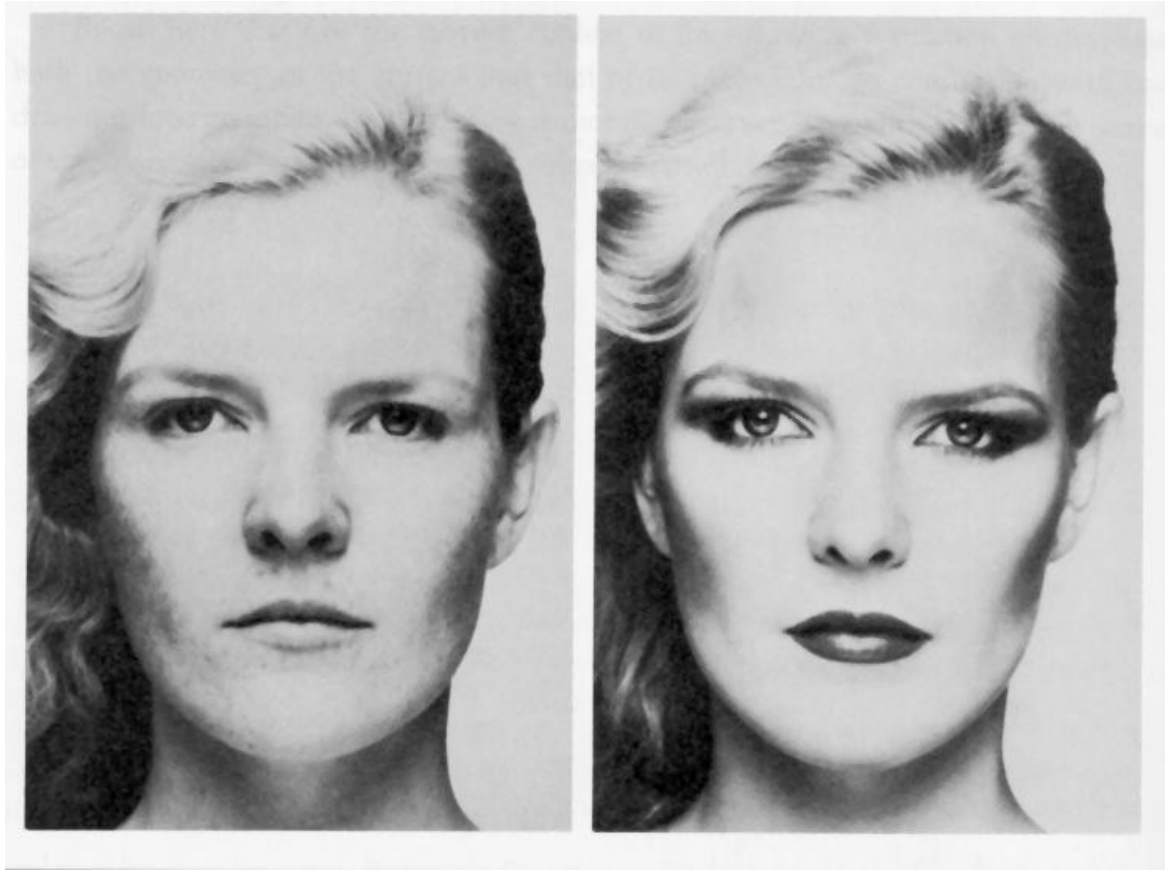
Mid-level Vision / Single-view Geometry

Shape-from-“X”

- X: different **visual cues** that can aid 3D depth perception

Visual Cues: Shape-from-

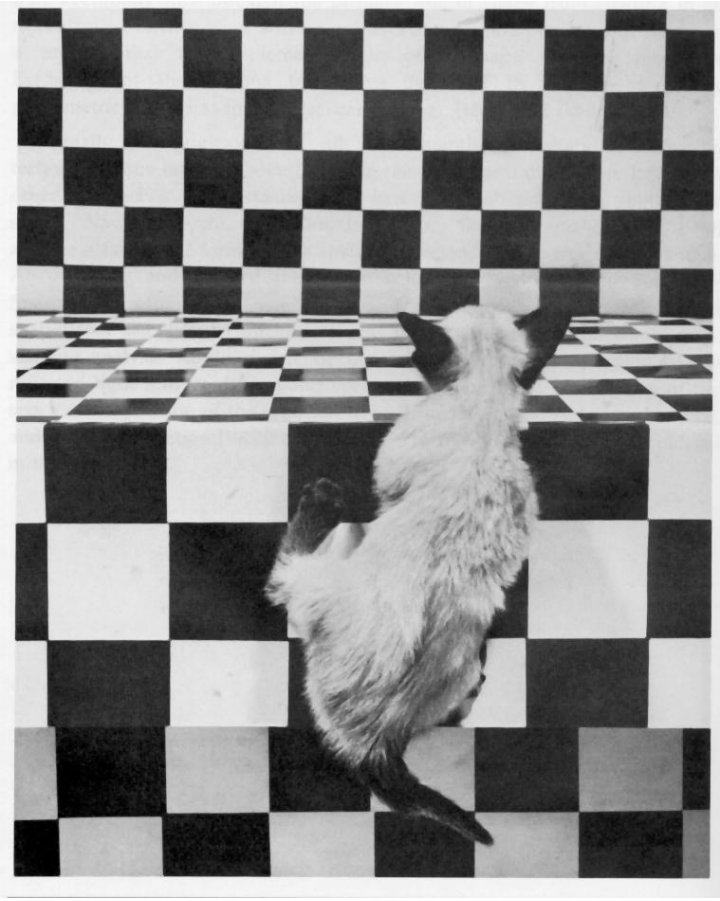
- Shading



Merle Norman Cosmetics, Los Angeles

Visual Cues: Shape-from-

- Shading
- Texture



***The Visual Cliff*, by William Vandivert, 1960**

Visual Cues: Shape-from-

- Shading
- Texture

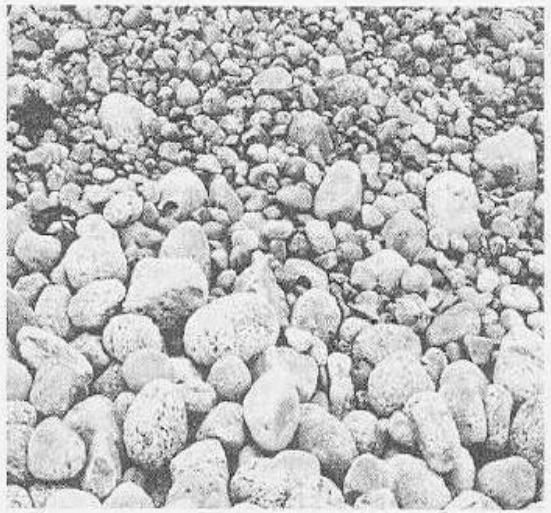


FIGURE 8.27
Texture gradients provide information about depth. (Frank Siteman/Stock, Boston.)
© Frank Sitman/Stock Boston

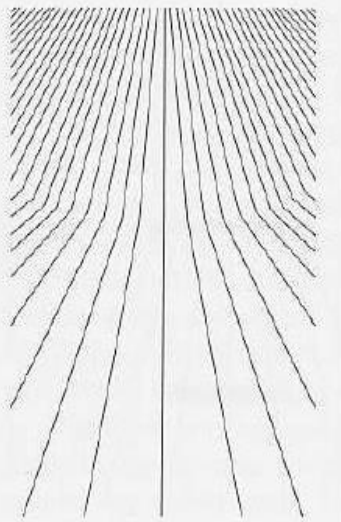
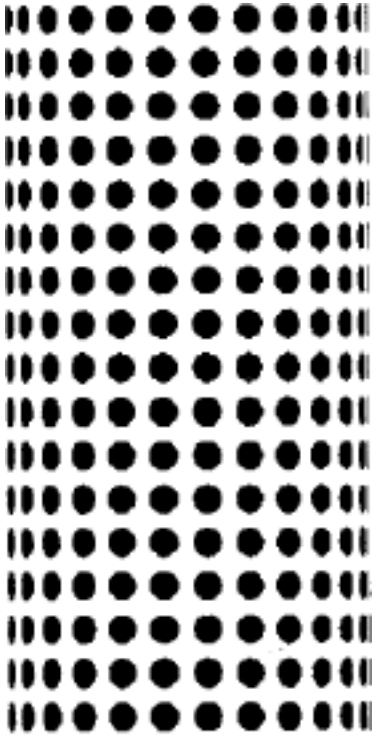


FIGURE 8.28
Texture discontinuity signals the pre corner.



A Witkin. Recovering Surface Shape and Orientation from Texture (1981)

Visual Cues: Shape-from-

- Shading
- Texture
- Focus

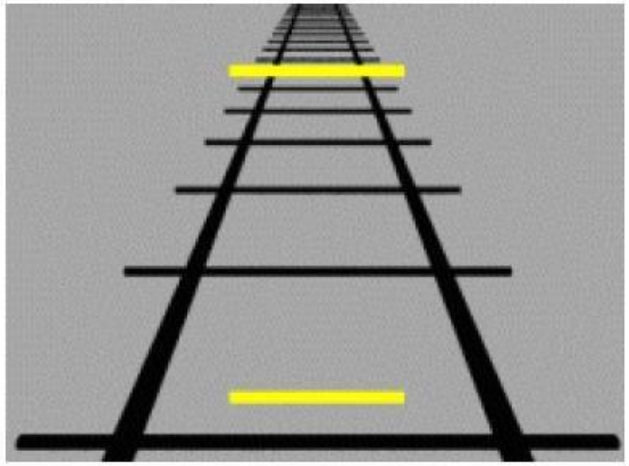
From *The Art of Photography*, Canon

Visual Cues: Shape-from-

- Shading
- Texture
- Focus
- Motion

Visual Cues: Shape-from-

- Shading
- Texture
- Focus
- Motion



Ponzo's illusion

- Perspective distortion

Visual Cues: Shape-from-

- Shading
- Texture
- Focus
- Motion
- Perspective distortion
- Colour

Atmospheric distortion

Visual Cues: Shape-from-

- Shading
- Texture
- Focus
- Motion
- Perspective distortion
- Colour
- Size

Ames Room

Visual Cues: Shape-from-

- Shading
- Texture
- Focus
- Motion
- Perspective distortion
- Colour
- Size
- Occlusion

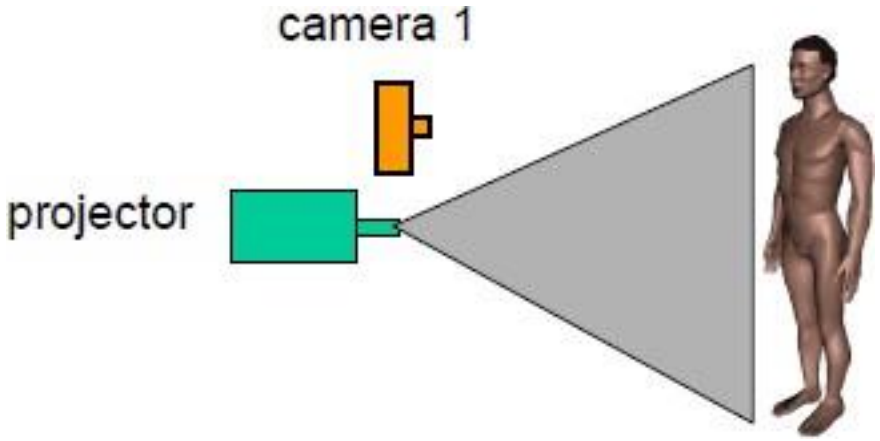
Which is closer?

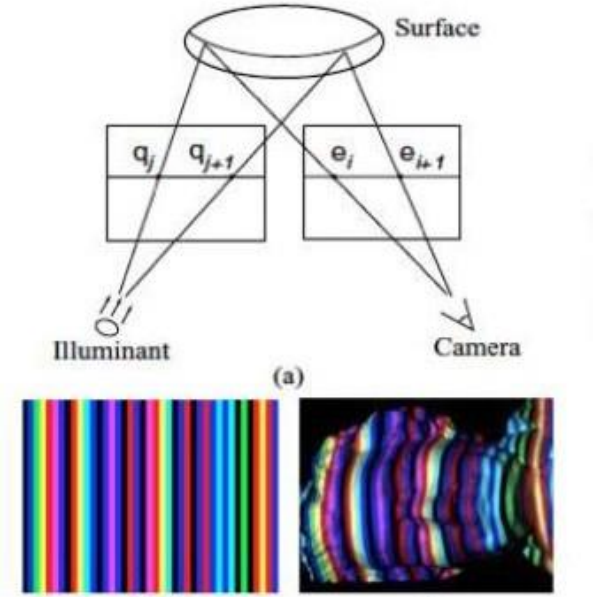
Visual Cues: Shape-from-

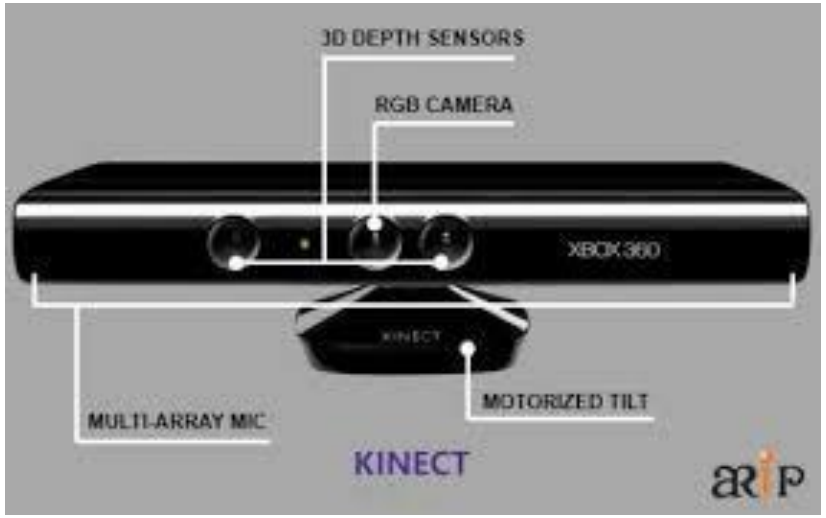
- Shading
- Texture
- Focus
- Motion
- Perspective distortion
- Colour
- Size
- Occlusion

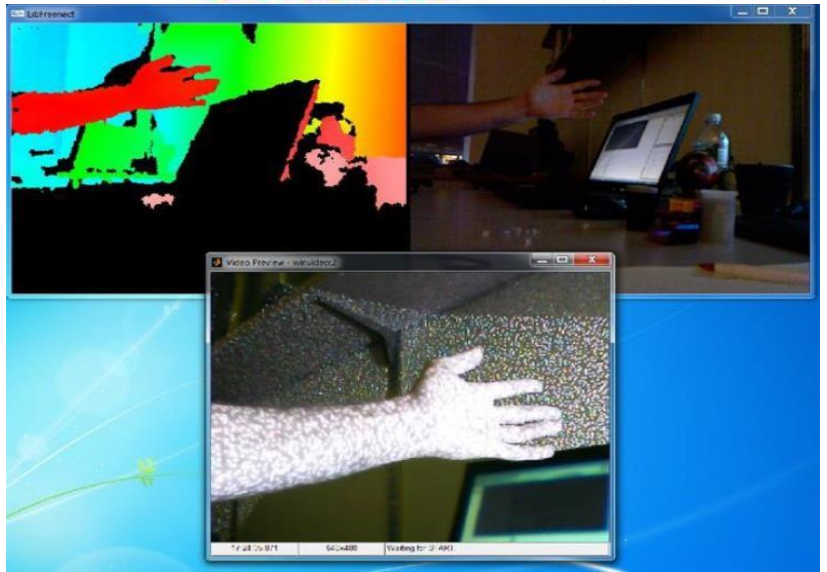
- Stereo
- Specular highlights
- Inter-reflections
- Symmetry
- Light Polarisation
- **Structured light (active)**
- **Time-of-flight (active)**
- **Shadow**
- **Silhouette**

Shape-from-Structured-Light



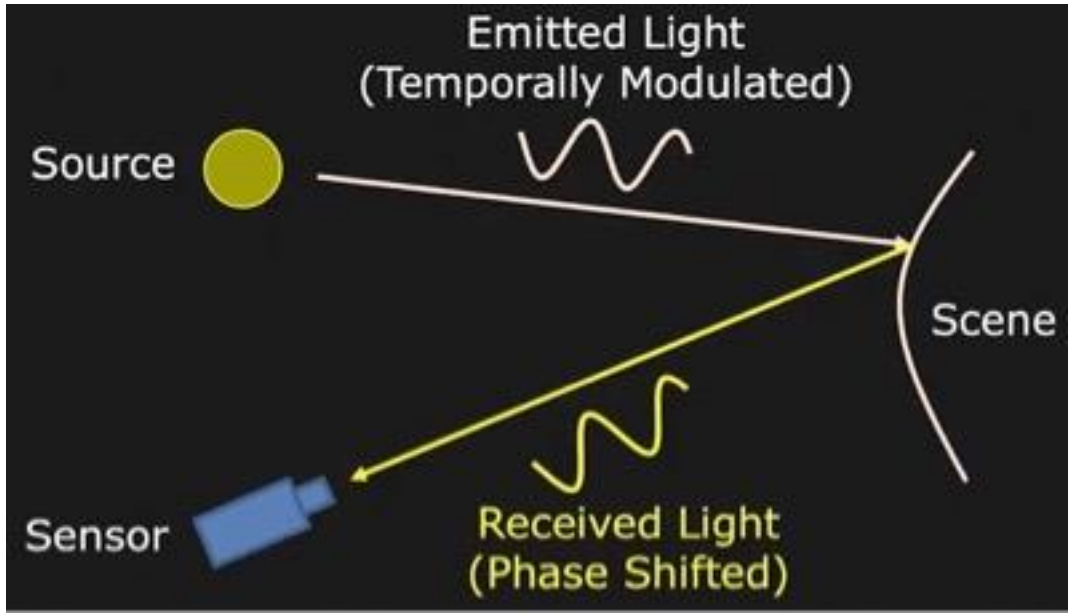






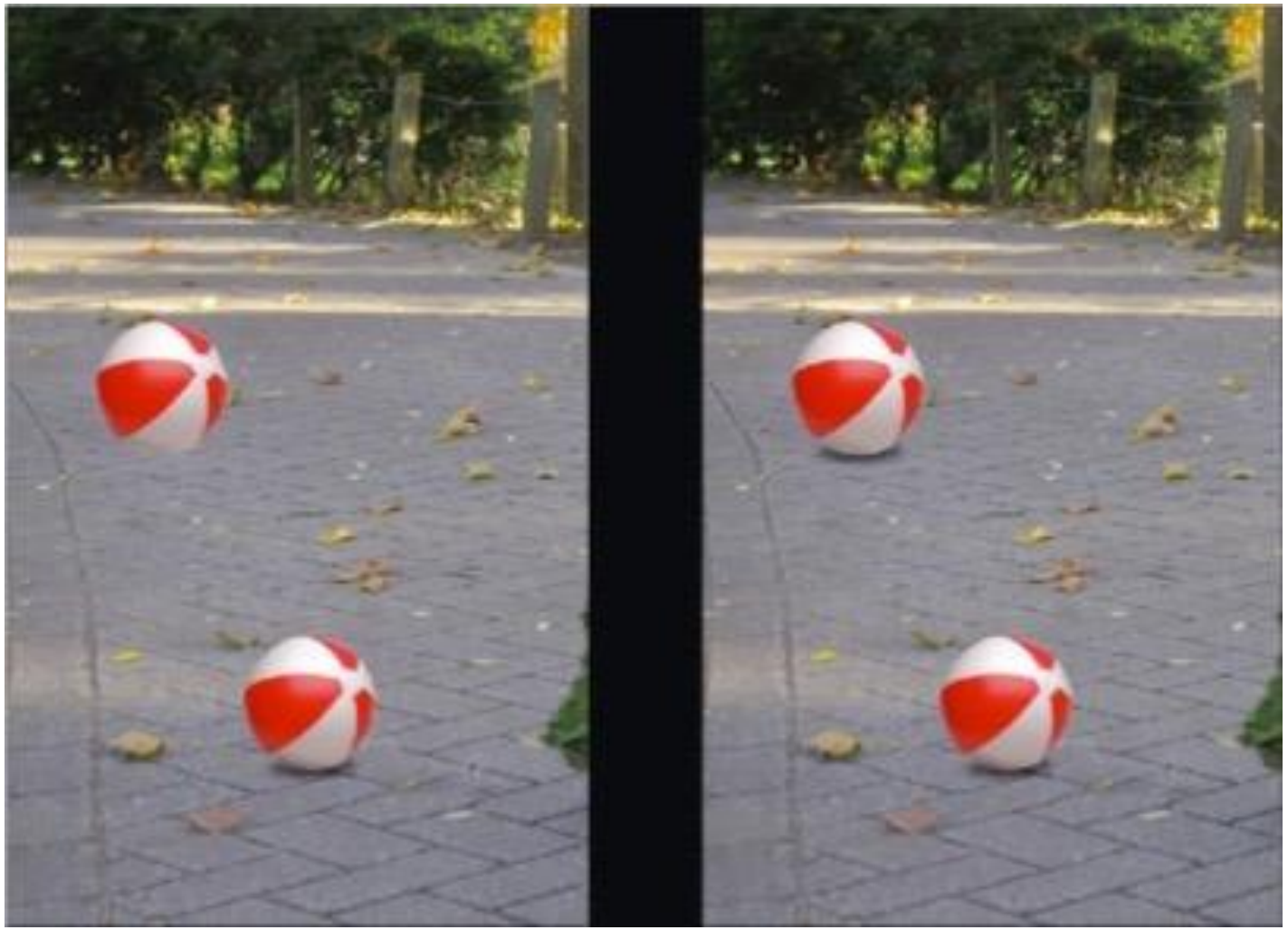
Shape-from-Time-of-Flight (ToF Camera)

- E.g., the Kinect-v2
- Find the phase difference between the emitted and received waves → time delay
- Return trip distance = speed of light × time delay

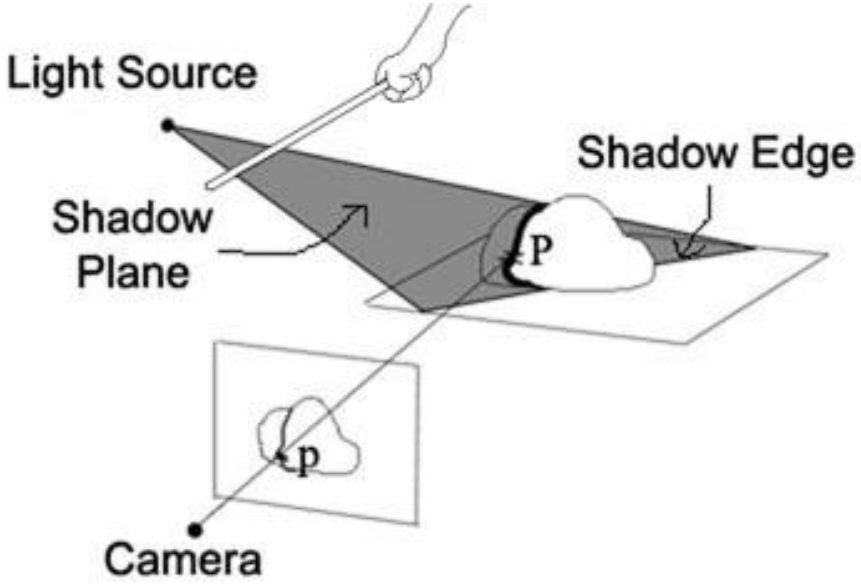


Shape-from-Shadow

Shape-from-Shadow



Shape-from-Shadow



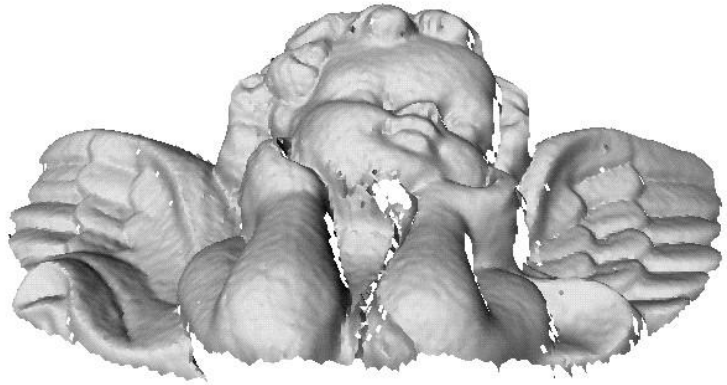
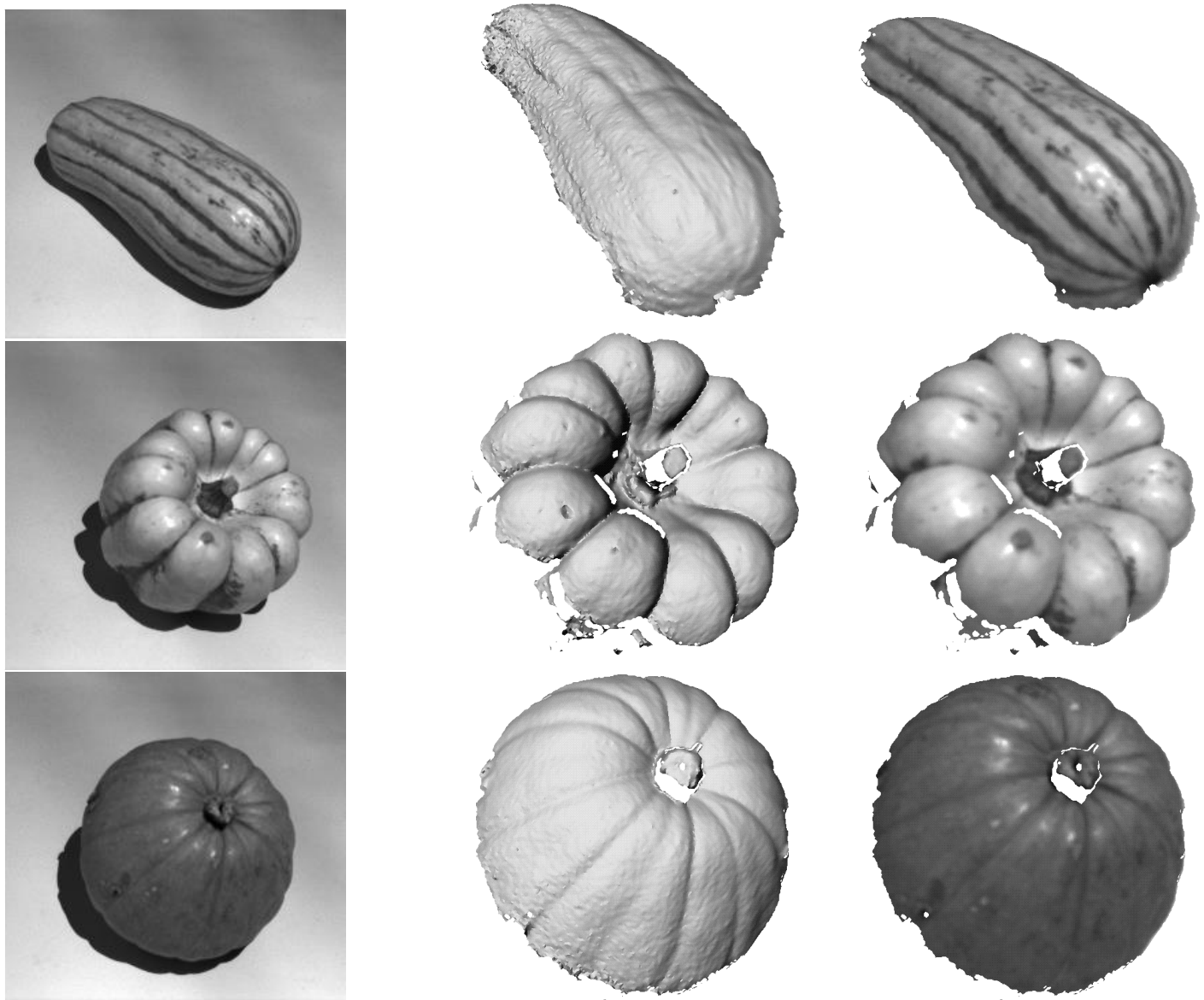
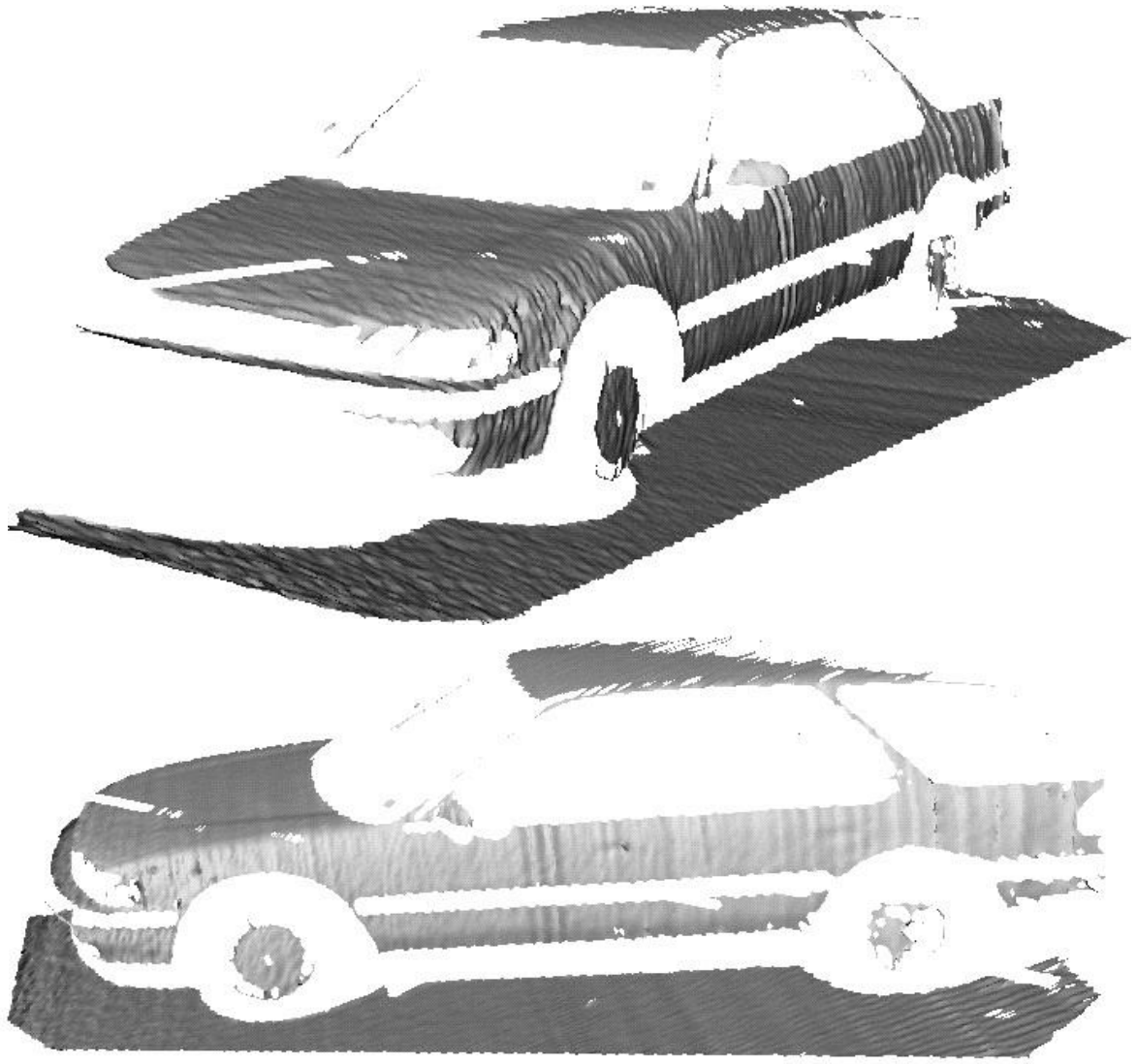


Image Credits: 3D Photography on your desk - ICCV98 - Jean-Yves Bouguet

Shape-from-Shadow: Results

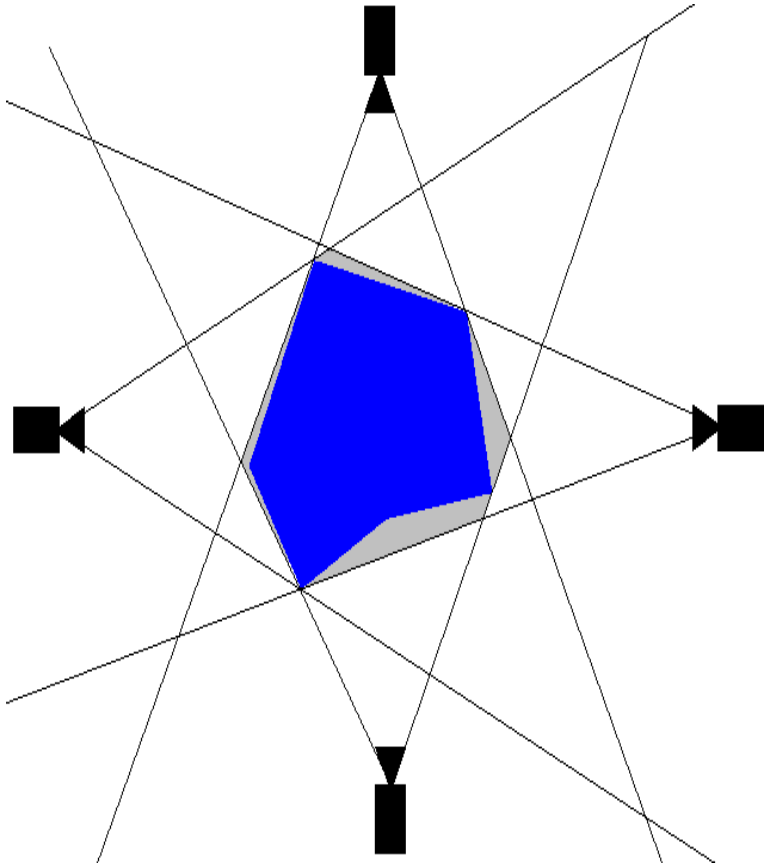


Shape-from-Shadow: Scanning with the Sun



Accuracy: 1cm over 2m
(~ 0.5% error)

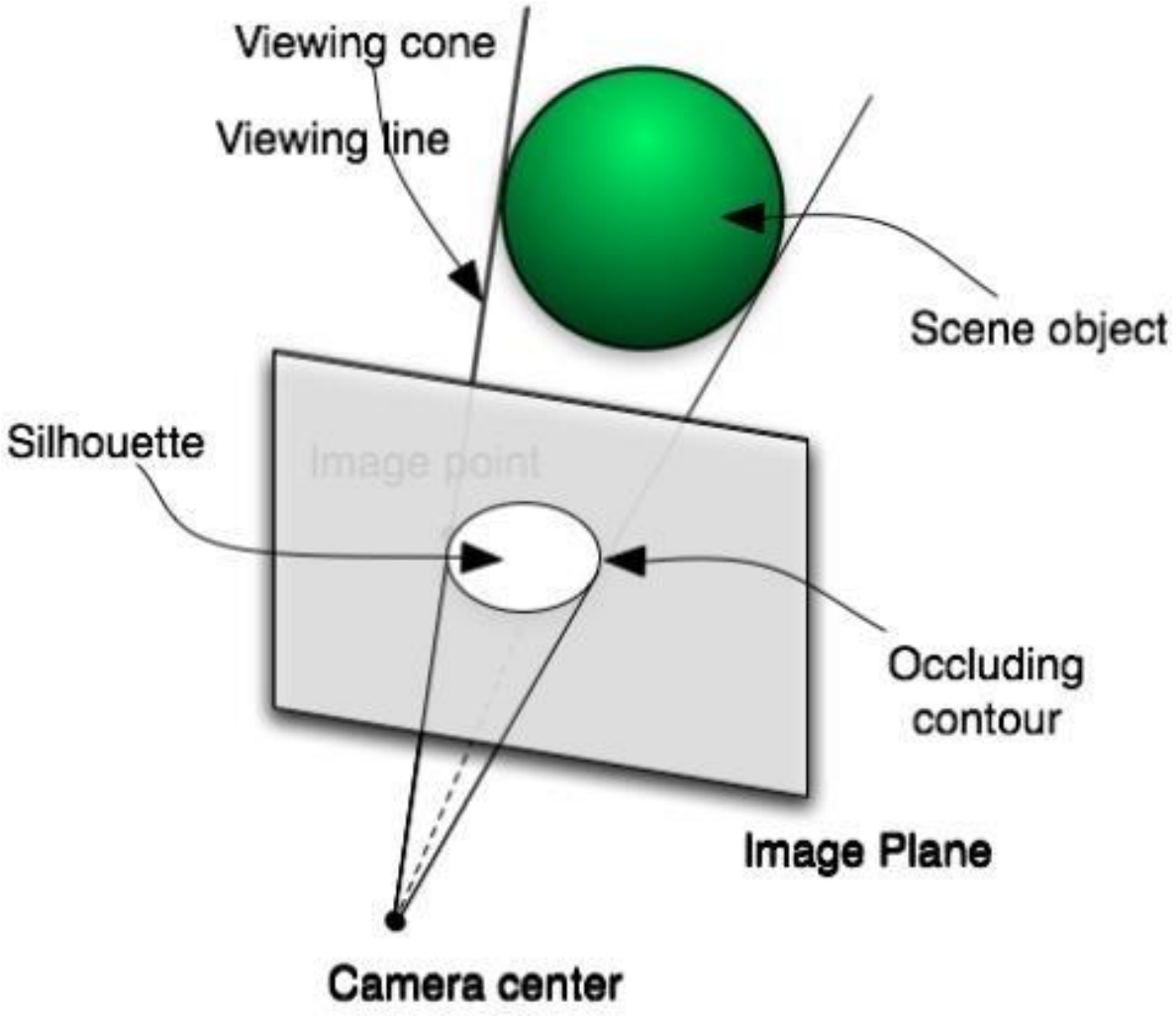
Shape-from-Silhouette (Visual Hull)



(a) Input images.

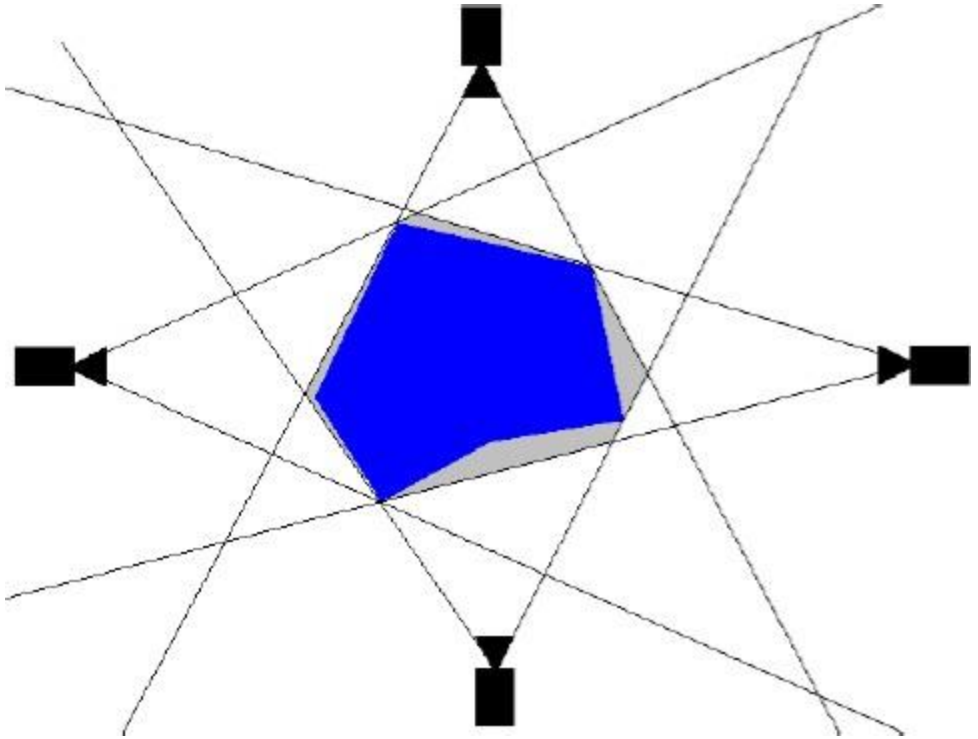
(b) Visual hull reconstruction.

Shape-from-Silhouette (Visual Hull)



Shape-from-Silhouette (Visual Hull)

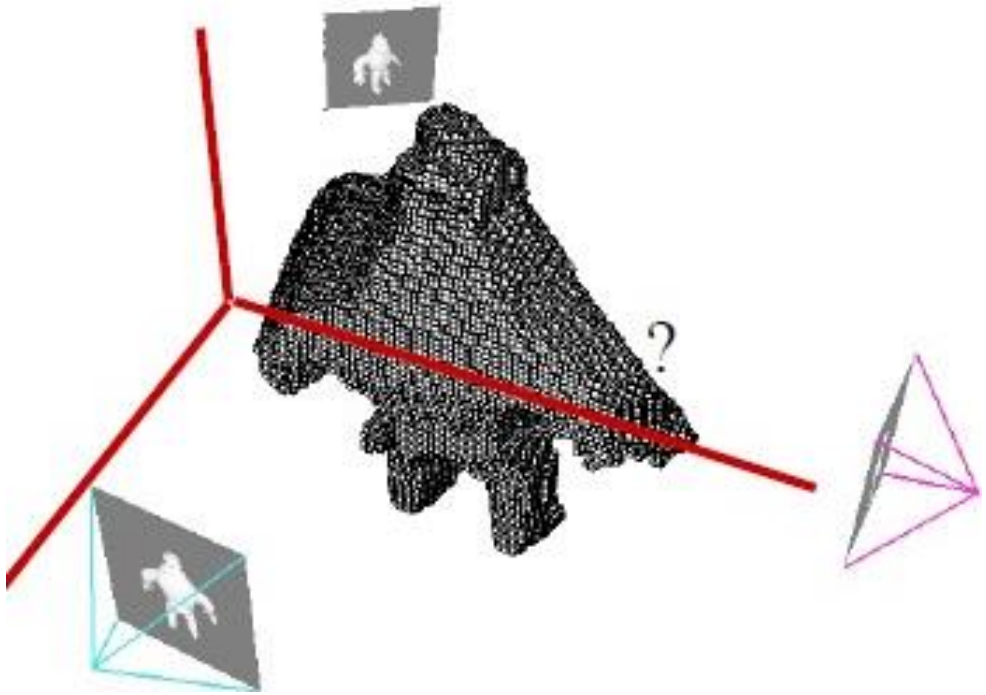
- From multiple views of the same object, intersect the *generalised cones* generated by each silhouette to construct a volume guaranteed to contain the object
  - *Visual hull* of the object (the smallest limiting volume obtainable in this way)



Shape-from-Silhouette (Visual Hull): Algorithm

1. Divide object space into a 3D grid of voxels
2. Intersect voxel grid with each silhouette volume
3. Retain voxels that lie inside **all** silhouette volumes

- Simple, but poor precision/computation time trade-off



Shape-from-Shading

Photometric Stereo

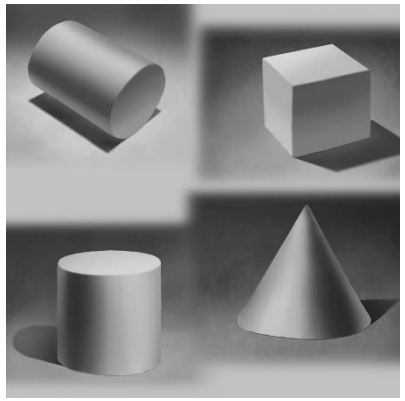
What is Shading?

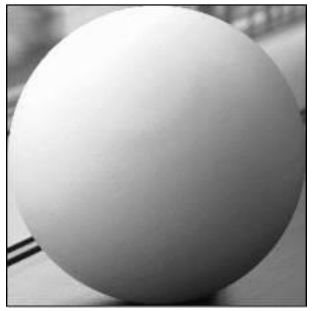
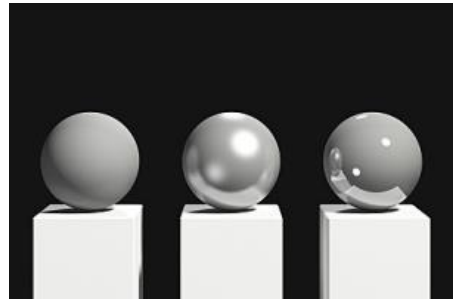
- Well… shading is not shadow…

- We can’t reconstruct shape from one shadow…

What is Shading?

- Variable levels of intensity (greyscale)

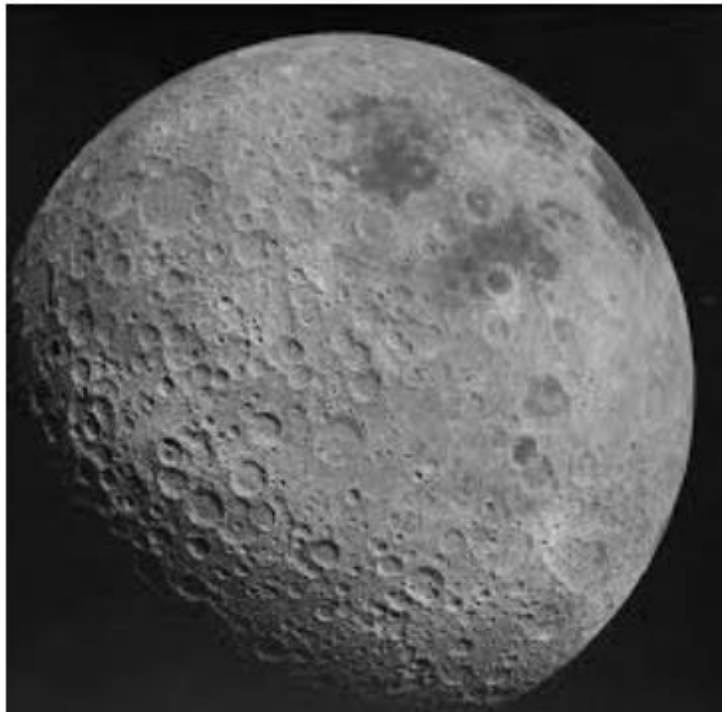


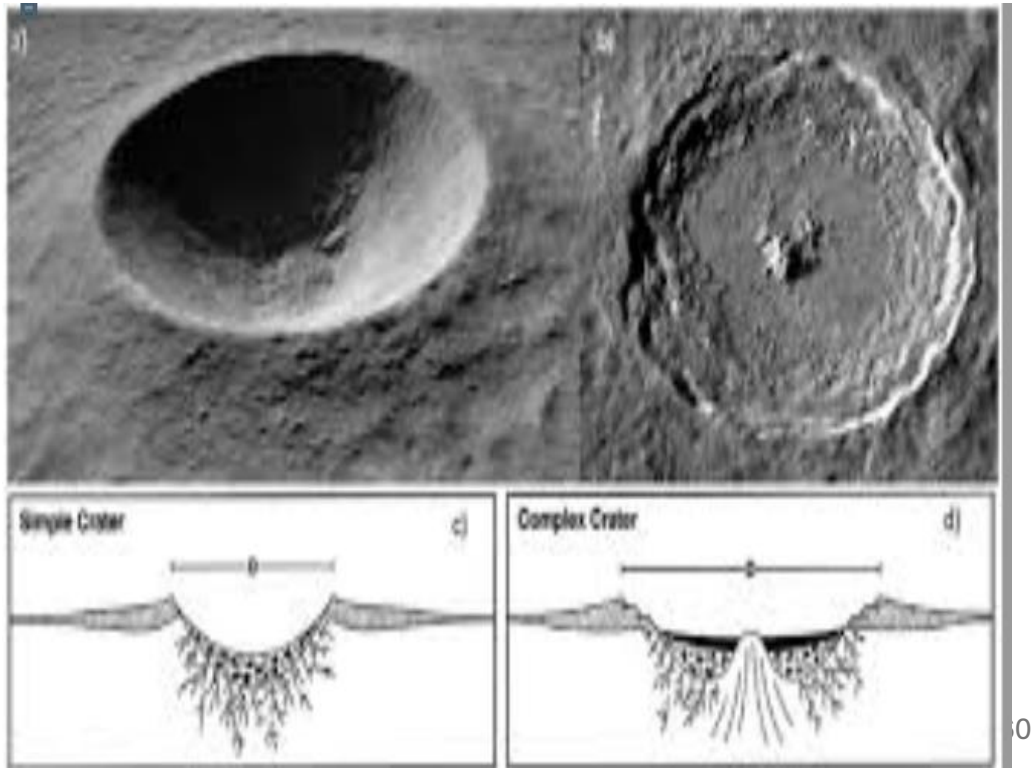
- Gives a cue for the actual 3D shape
- There is a relation between intensity and shape

Represent a 3D Shape via Shading

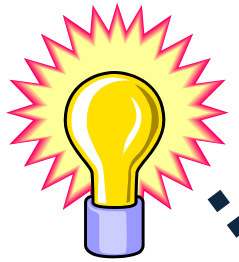
Infer 3D Geometry from Shading



- Shading gives a cue for the actual 3D shape: intensity (shading) & shape are related



Photometric Image Formation



Source emits photons

Photons travel
in a straight line

And then some
reach the
eye/camera

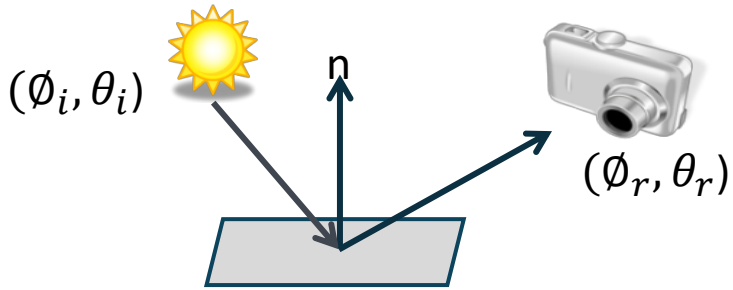
When they hit an object, they:

- bounce off in a new direction
- or are absorbed

(exceptions later)

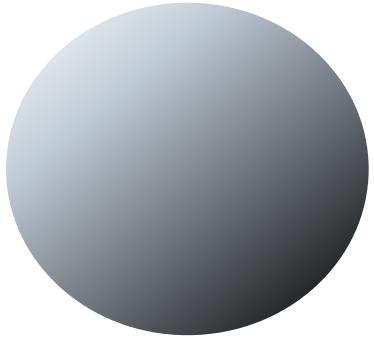
What Determines Image Shading (Intensity)?

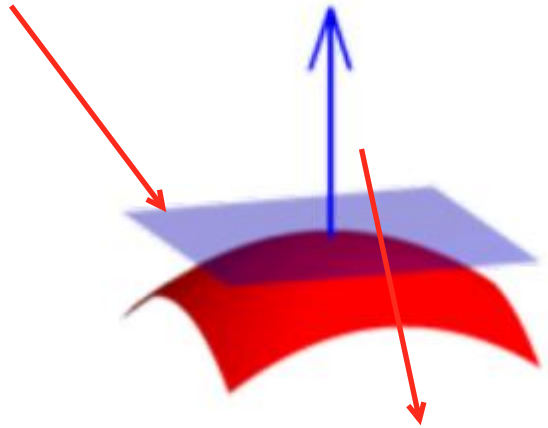
- The amount of light that falls on the surface
- The fraction of light that is reflected (albedo)
- Geometry of light reflection
  - Shape of surface
  - Viewpoint



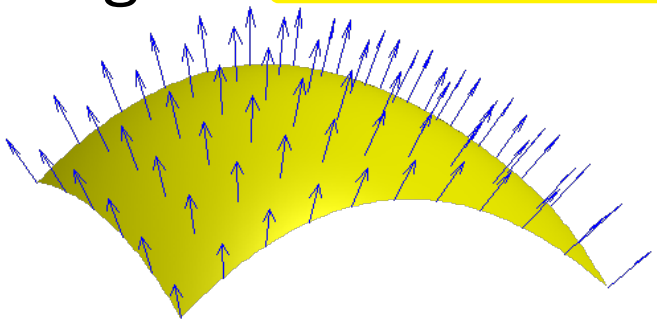
3D Surface Geometry is Defined by the Surface Normal at Every Point

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



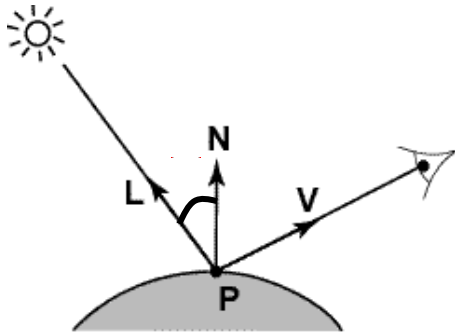


- We can fit the model using the surface normal at every point



Modeling Photometric Image Formation

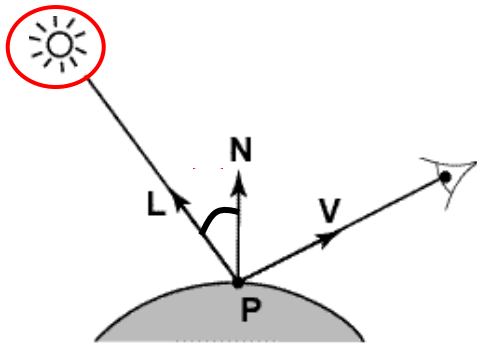
- Now we need to reason about:
  - How light interacts with the scene
  - How a pixel value is related to light energy in the world



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

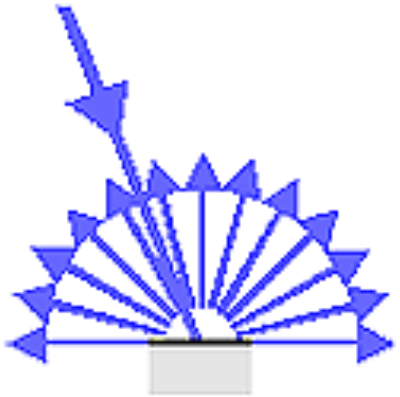
Assume Far-field Directional Lighting

- 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)

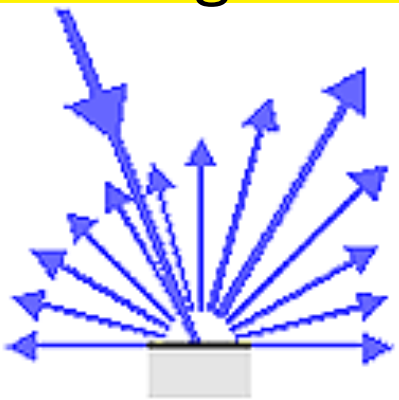


An Ideal Model: Lambertian Reflection

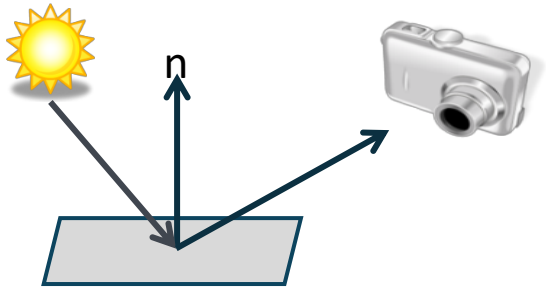
- Appears equally bright from all viewing directions
- Reflects all light without absorbing
- Matte surface, no “shiny” spots
- Brightness of the surface as seen from camera is linearly correlated to the amount of light falling on the surface



Ideal diffuse reflection
(Lambertian surface)

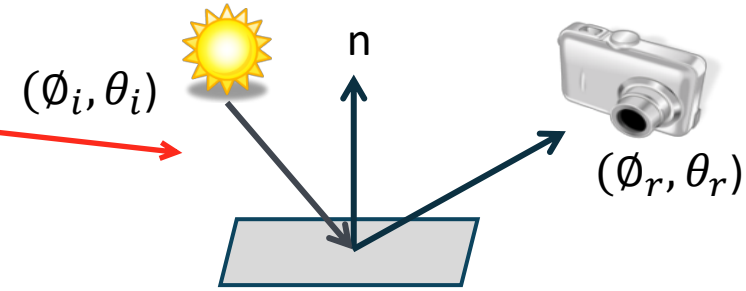


Diffuse reflection with
directional component

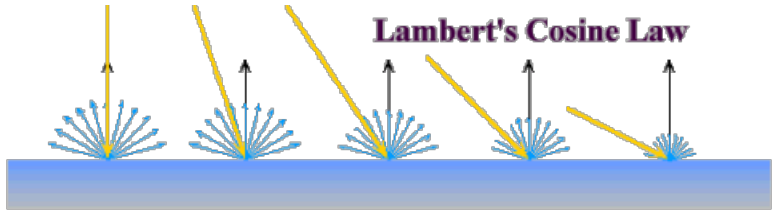


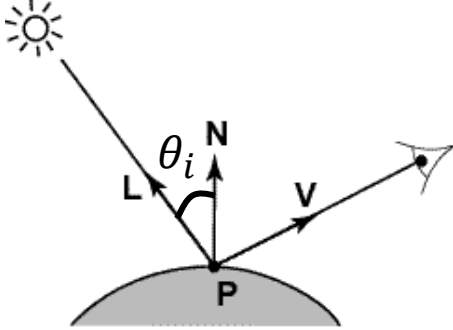
Lambertian Surface

- 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



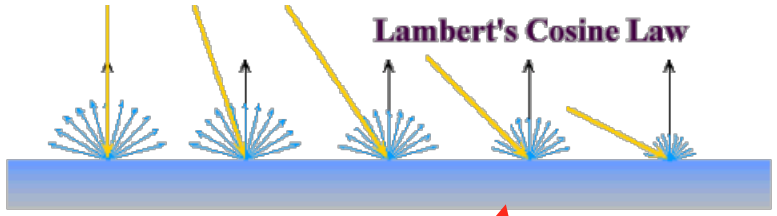


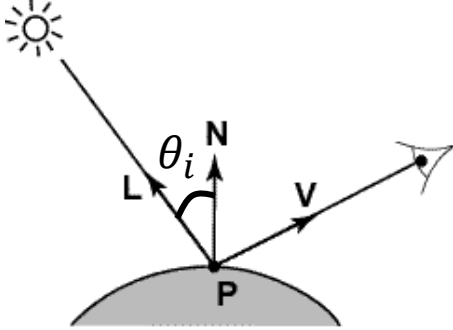
$$R_r = \rho N \cdot L R_i$$
$$I = \rho N \cdot L$$
$$\boxed{I = N \cdot L = \cos\theta_i}$$

- 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





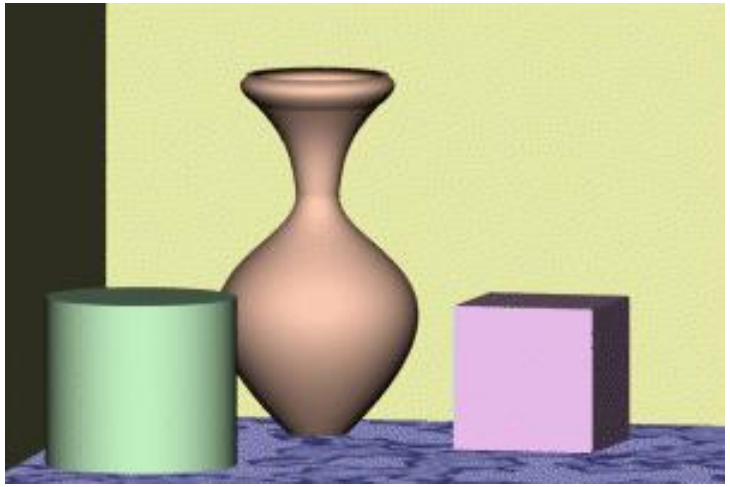
$$R_r = \rho N \cdot L R_i$$
$$I = \rho N \cdot L$$
$$\boxed{I = N \cdot L = \cos\theta_i}$$

- 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

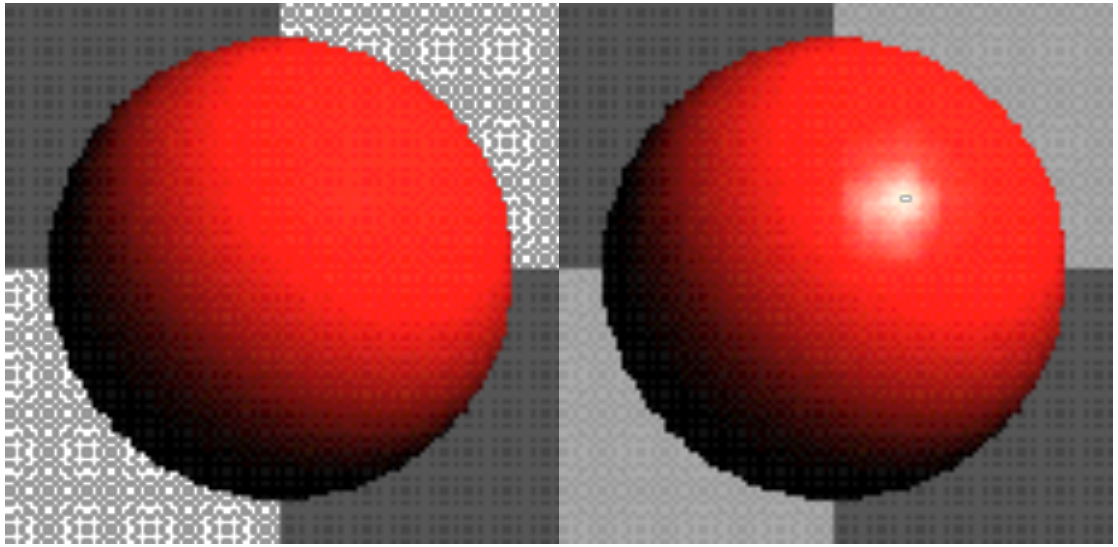


(Oren and Nayar)

- Lambertian sphere as the light moves

(Steve Seitz)

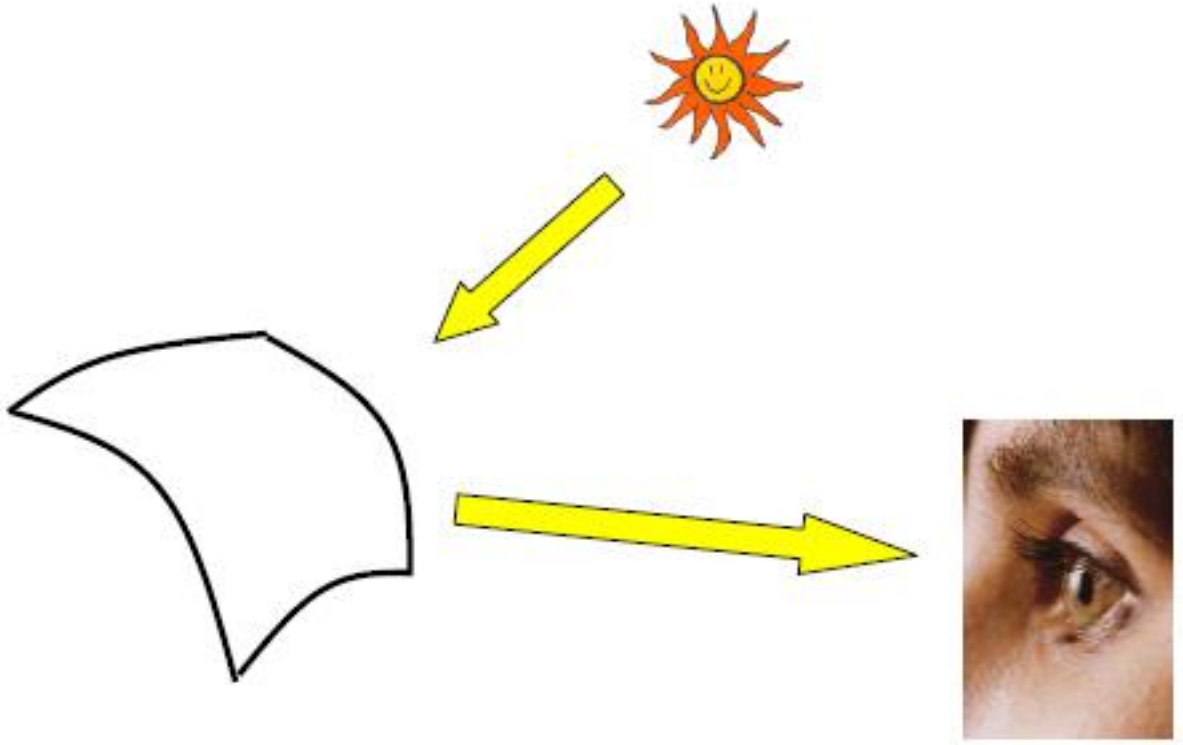
More Realistic: Lambertian + Specular



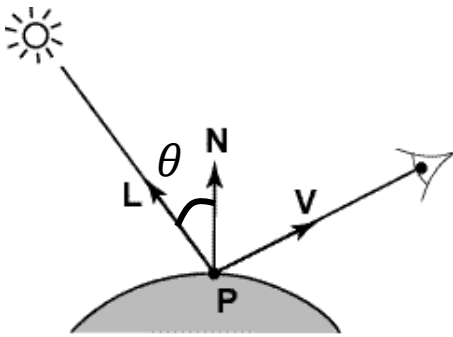
Images from: http://graphics.cs.ucdavis.edu/GraphicsNotes/Shading/Shading.html

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

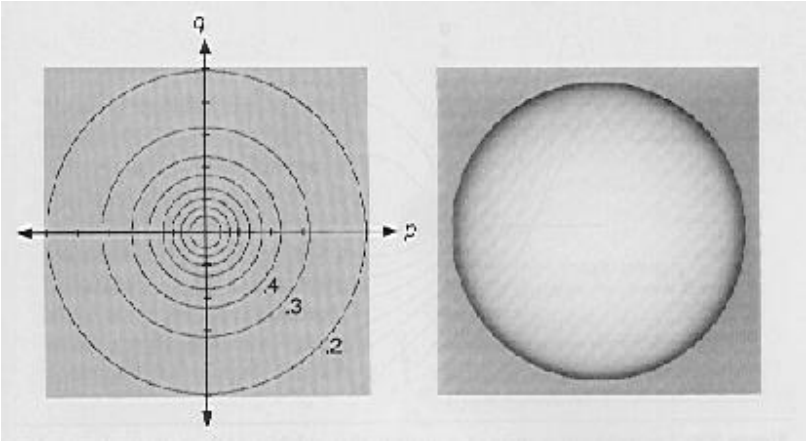
- Given a 3D surface $z(x, y)$, lighting and viewing direction, we can compute the grey level of pixel $I(x, y)$ on the surface
- Assume we know the surface normal map, then we can deduce the image from irradiance
- Use $I(x, y) = R(p, q) = \hat{n}^\top \hat{l} = \frac{1+pp_s+qq_s}{\sqrt{1+p^2+q^2}\sqrt{1+{p_s}^2+{q_s}^2}}$ 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)

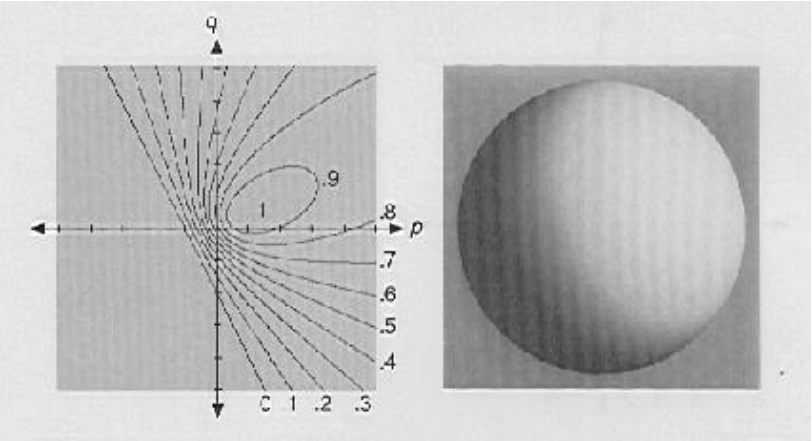
$$R(p,q) = \frac{1 + p_s p + q_s q}{\sqrt{1 + p^2 + q^2}\sqrt{1 + {p_s}^2 + {q_s}^2}}$$

Lambertian Reflectance Map

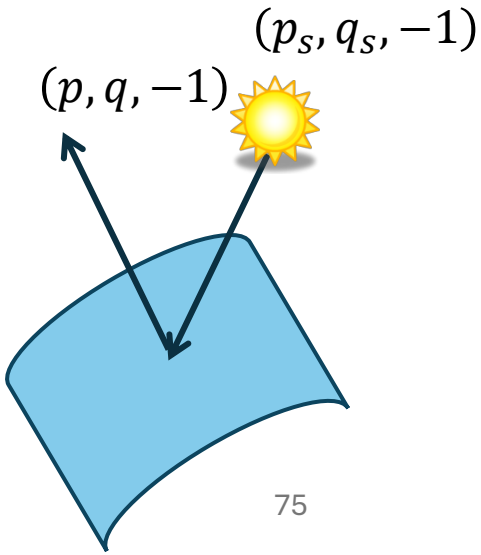
- Local surface orientation that produces equivalent intensities are quadratic conic section contours in gradient space
- Iso-intensity maps:



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

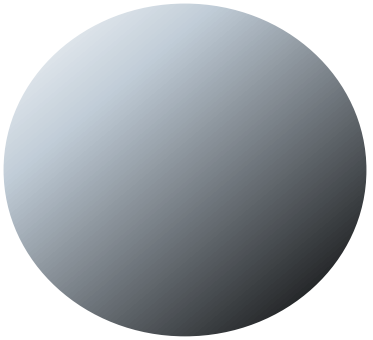


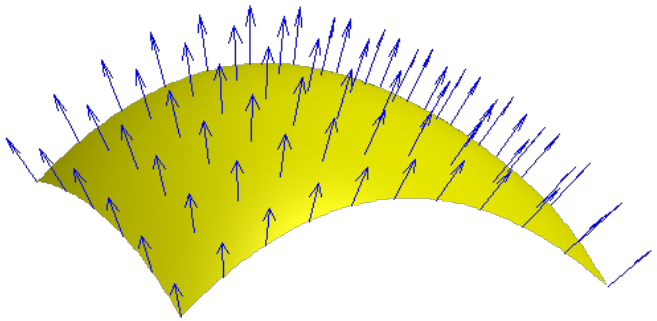
$(p_s, q_s) = (-2, -1)$



Shape-from-Shading

- Given a grayscale 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

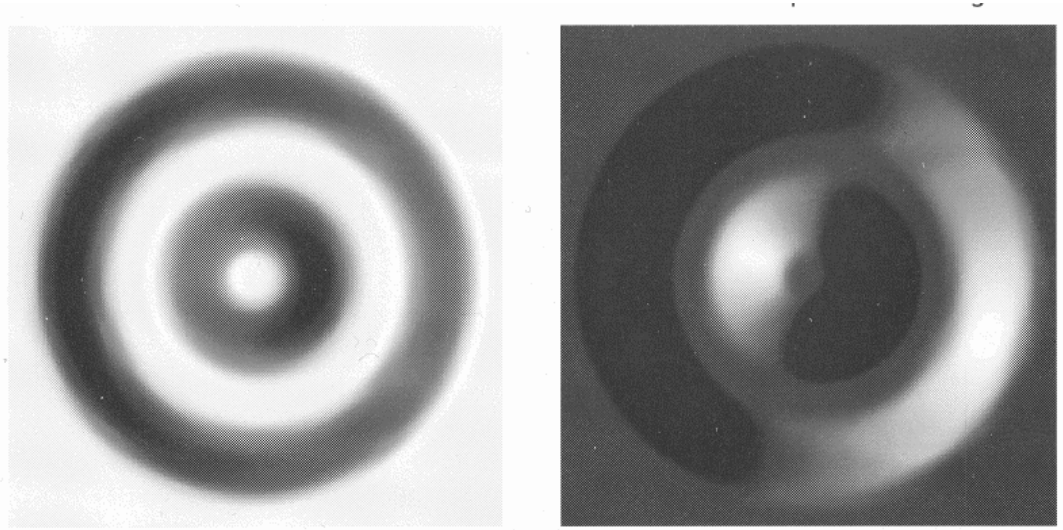
$$E = \int \left(\left(I(x,y) - R(p,q) \right)^2 + \lambda \left(\frac{\partial p}{\partial x}^2 + \frac{\partial p}{\partial y}^2 + \frac{\partial q}{\partial x}^2 + \frac{\partial q}{\partial y}^2 \right) \right) \mathrm{d}x\mathrm{d}y$$

Penalise errors between the image brightness and reflectance map

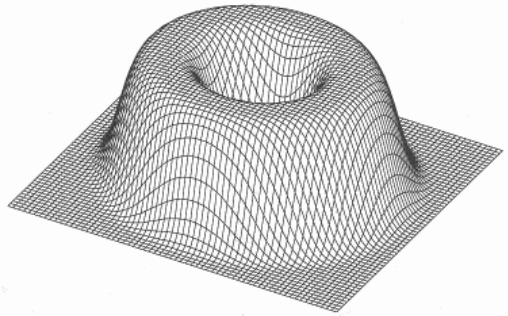
Penalise rapid changes in p and q

- 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

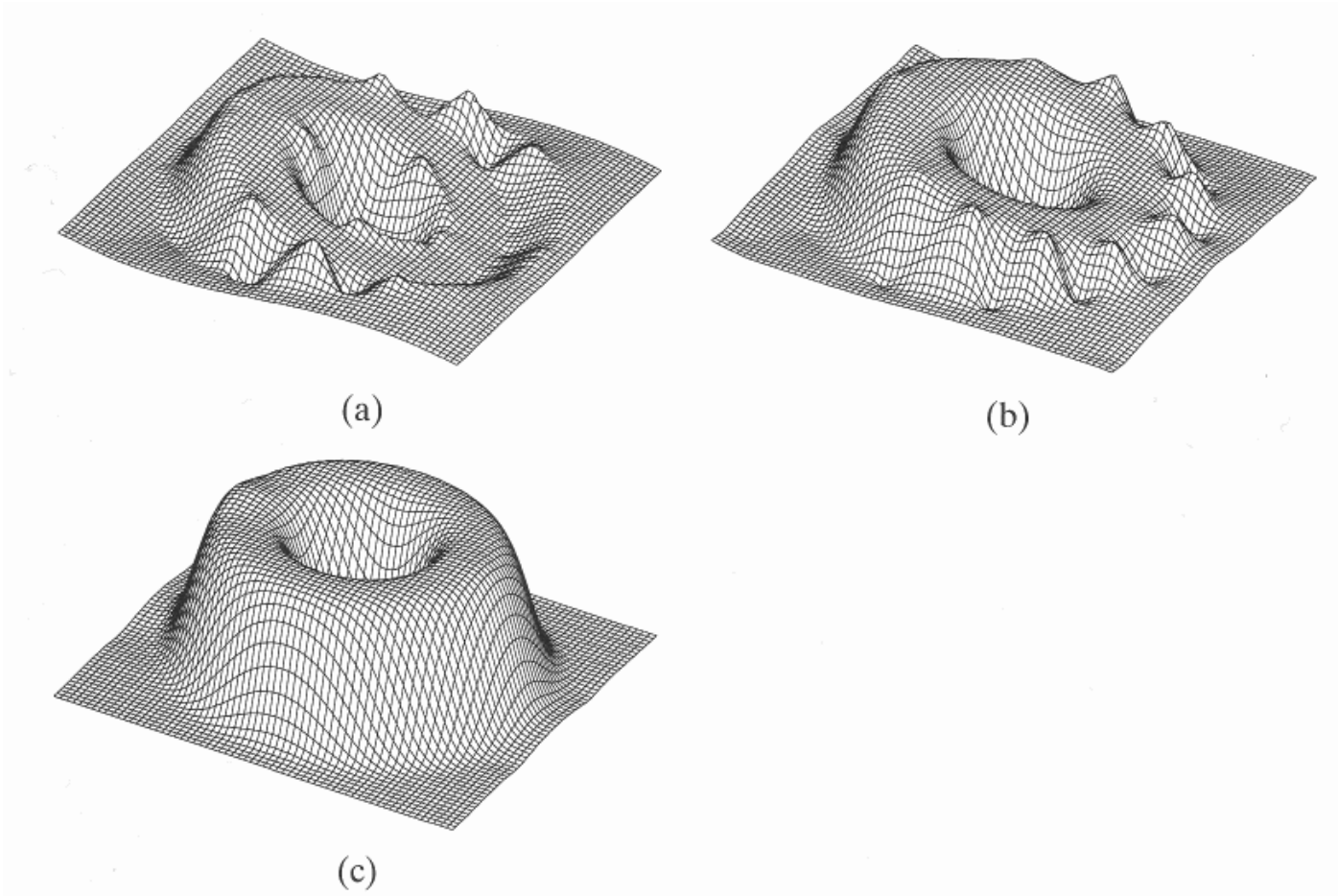
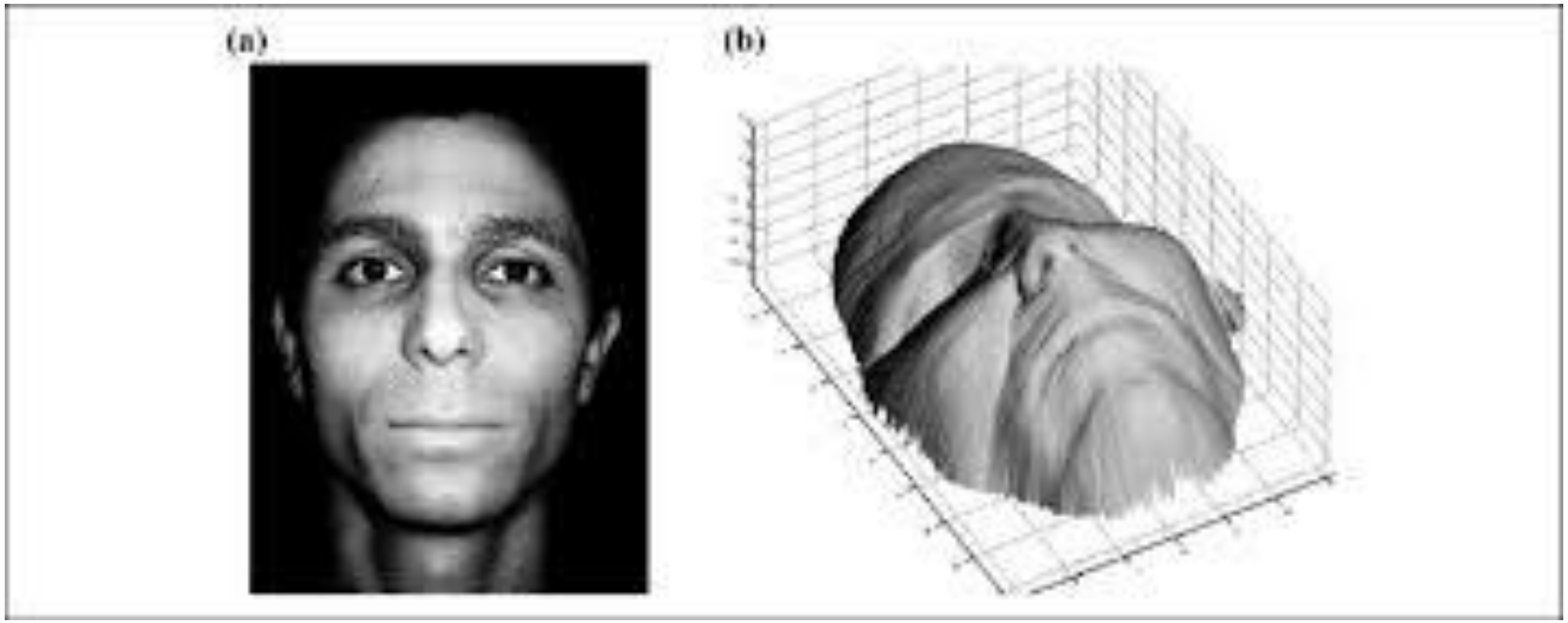
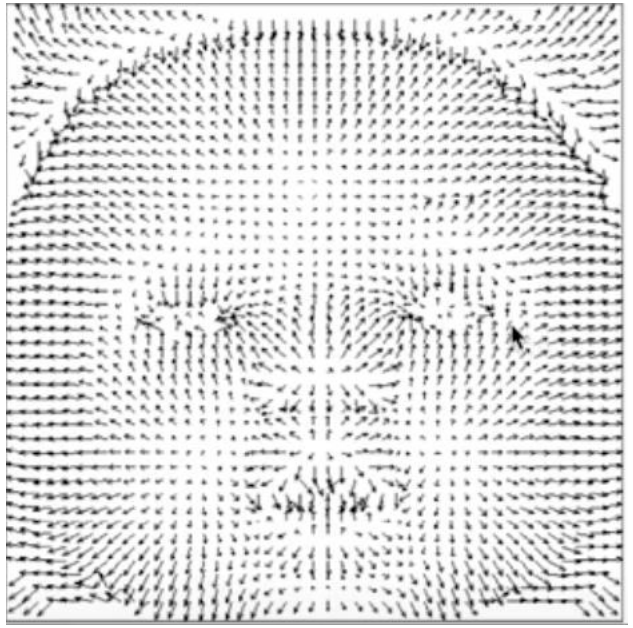


Figure 9.4 Reconstructions of the surface in Figure 9.2 after 100 (a), 1000 (b) and 2000 (c) iterations. The initial surface was a plane of constant height. The asymmetry of the first two reconstruction is due to the illuminant direction.

Shape-from-Shading: Results



Shape-from-Shading: From Normals to Shape



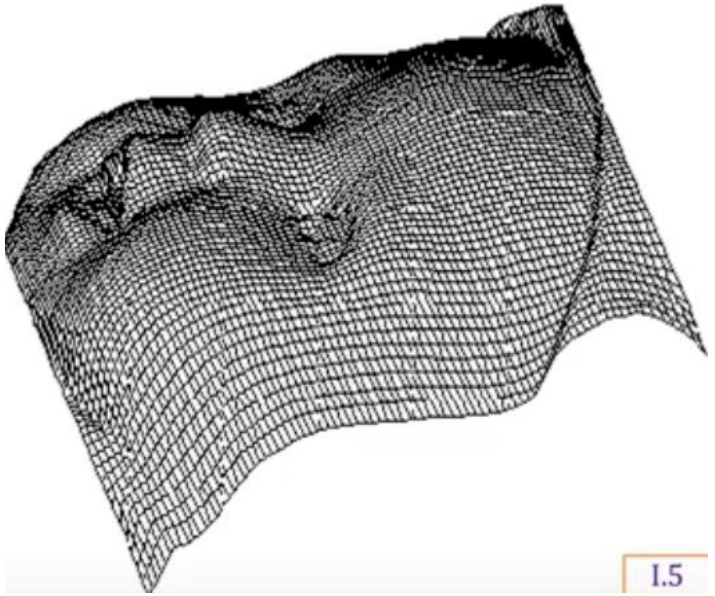
Gradient / normal map

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

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

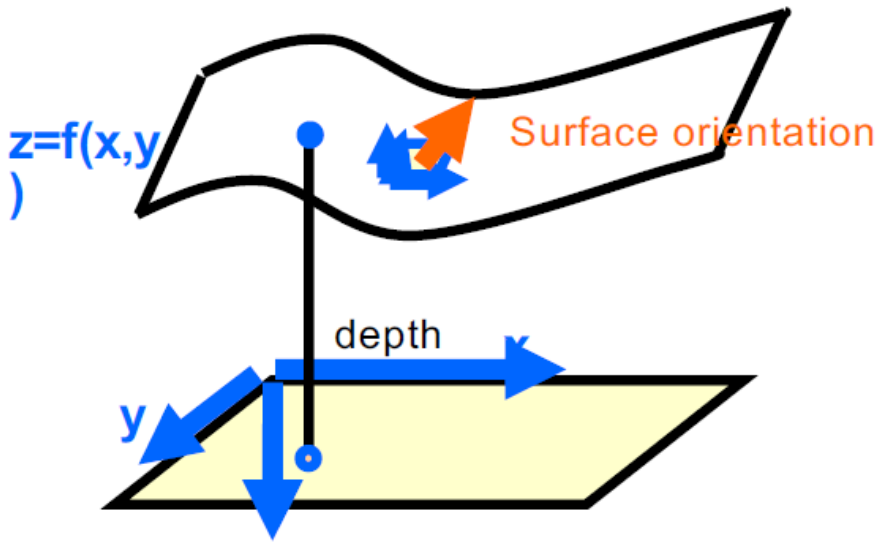
differentiation

integration



Shape / depth map z(x,y)

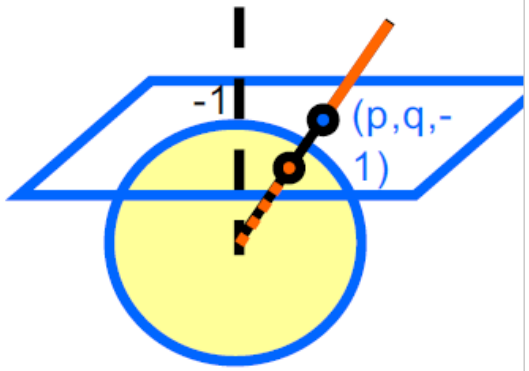
Surface Parametrisation



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

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

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

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

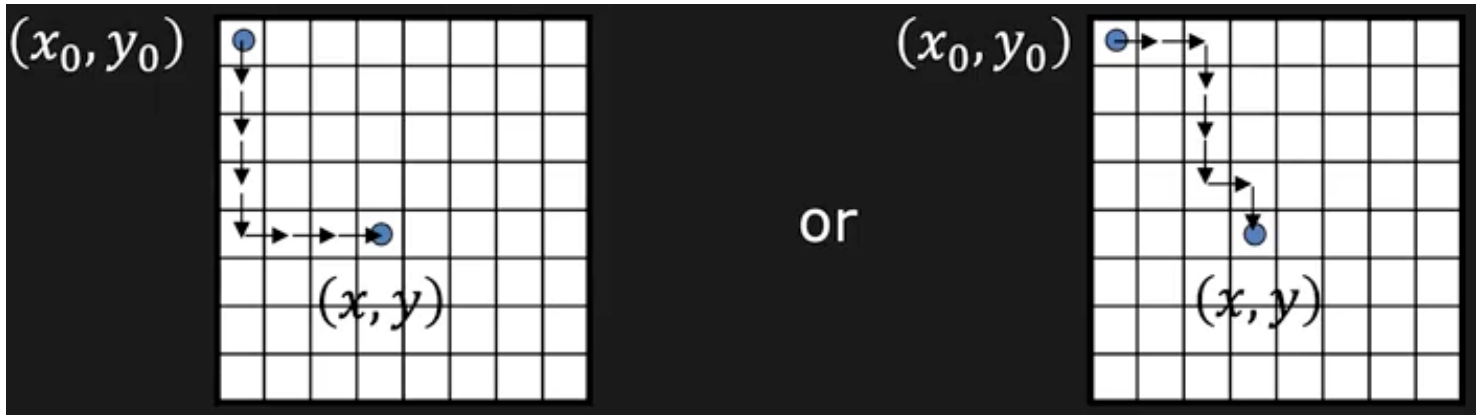
Shape-from-Shading: From Normals to Shape

- 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 $z(x_0, y_0) = 0$

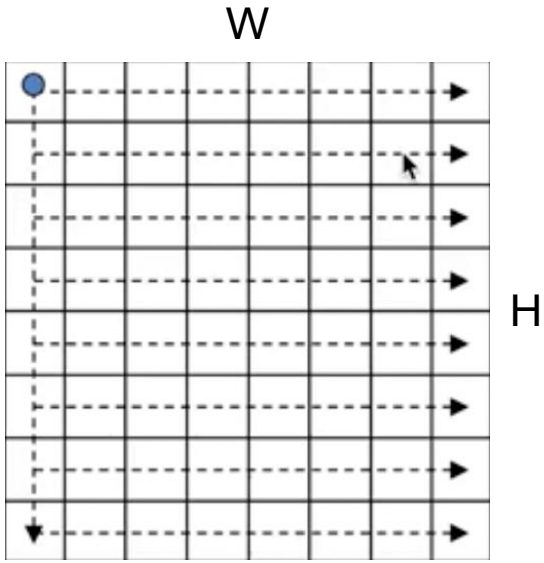
- $z(x,y)$ obtained by integration along any path from (x_0, y_0) should be the same



Shape-from-Shading: From Normals to Shape

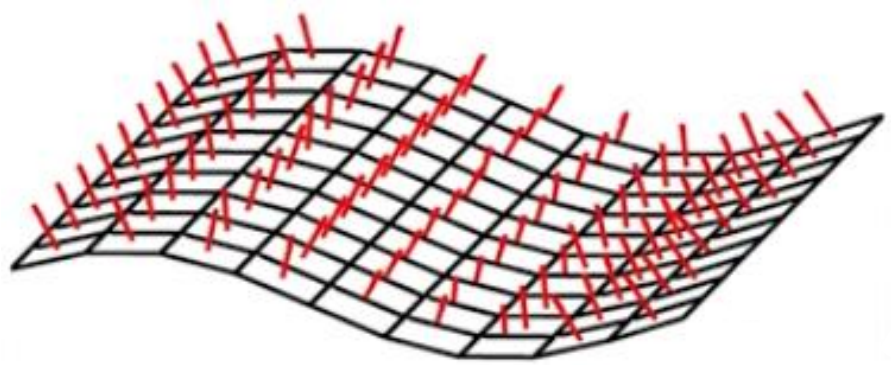
A naïve algorithm for estimating depth map:

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 $(W - 1)$
     $z(x, y) = z(x - 1, y) + p(x, y)$

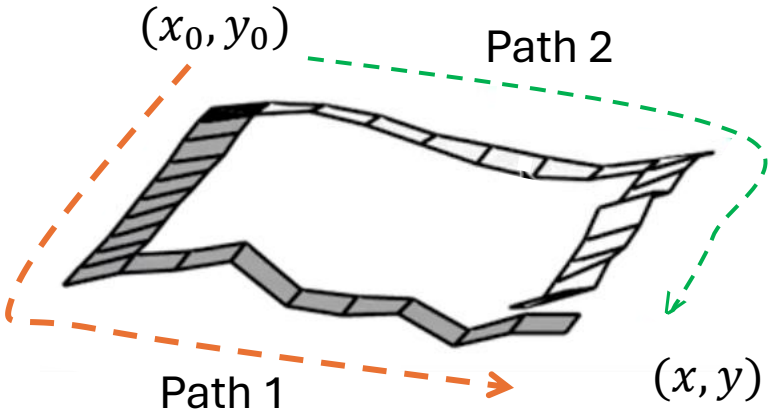


Computed depth map

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

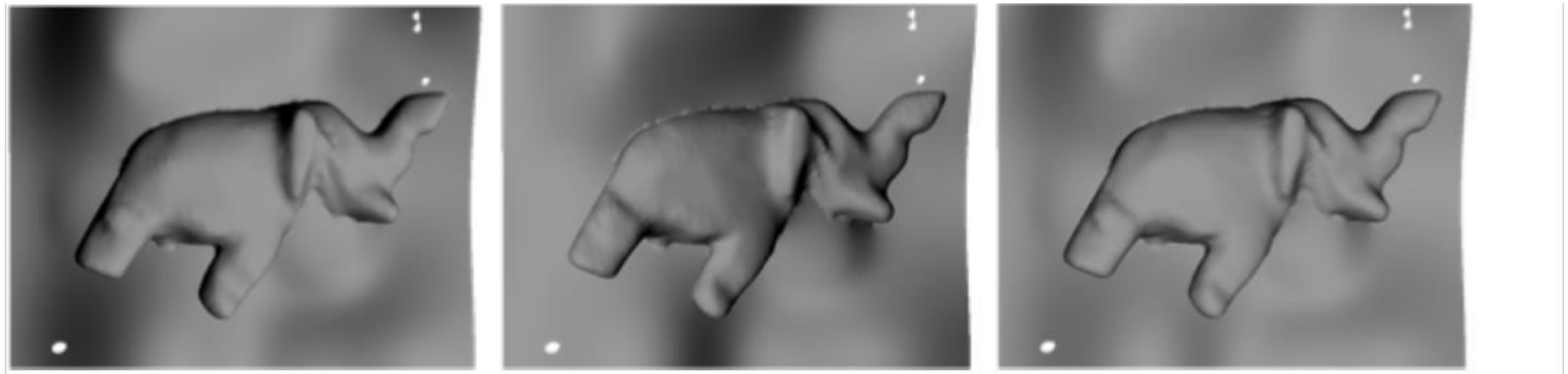
- Find a surface that minimises the errors between measured surface gradients (p, q) and gradients of estimated surface $z(x, y)$
- Error measure:

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

  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)

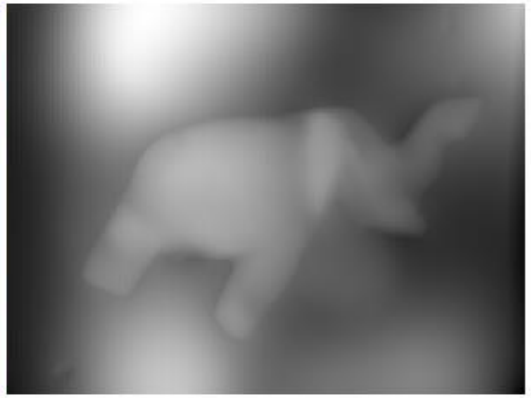
[Horn, 1990]

Shape-from-Shading: Results

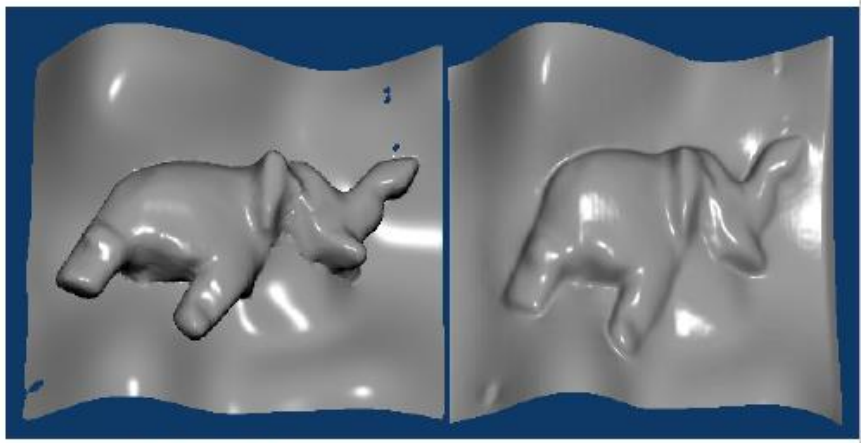


images with different light

normals



Integrated depth



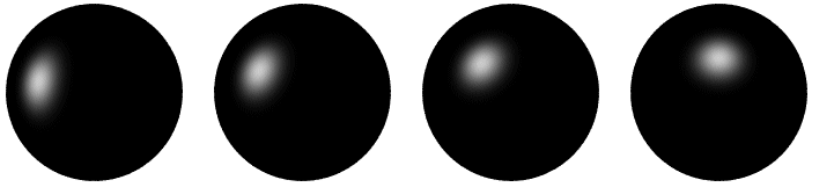
original surface

reconstructed

[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

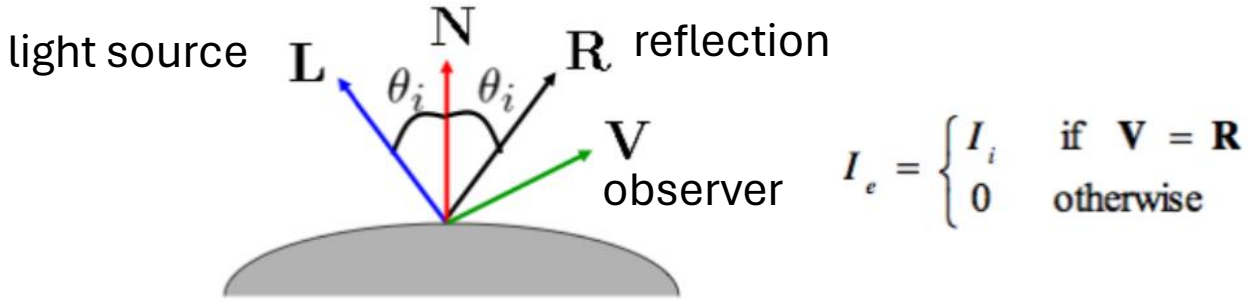
Uffizi Gallery

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

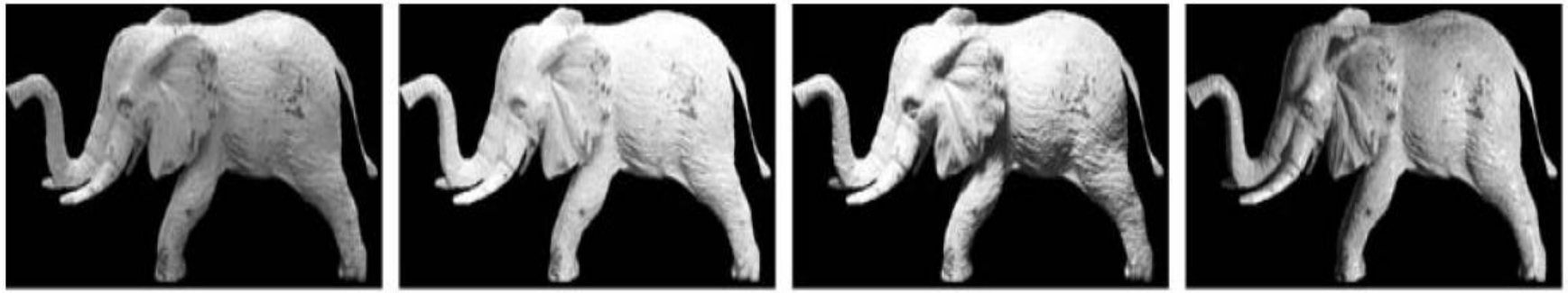


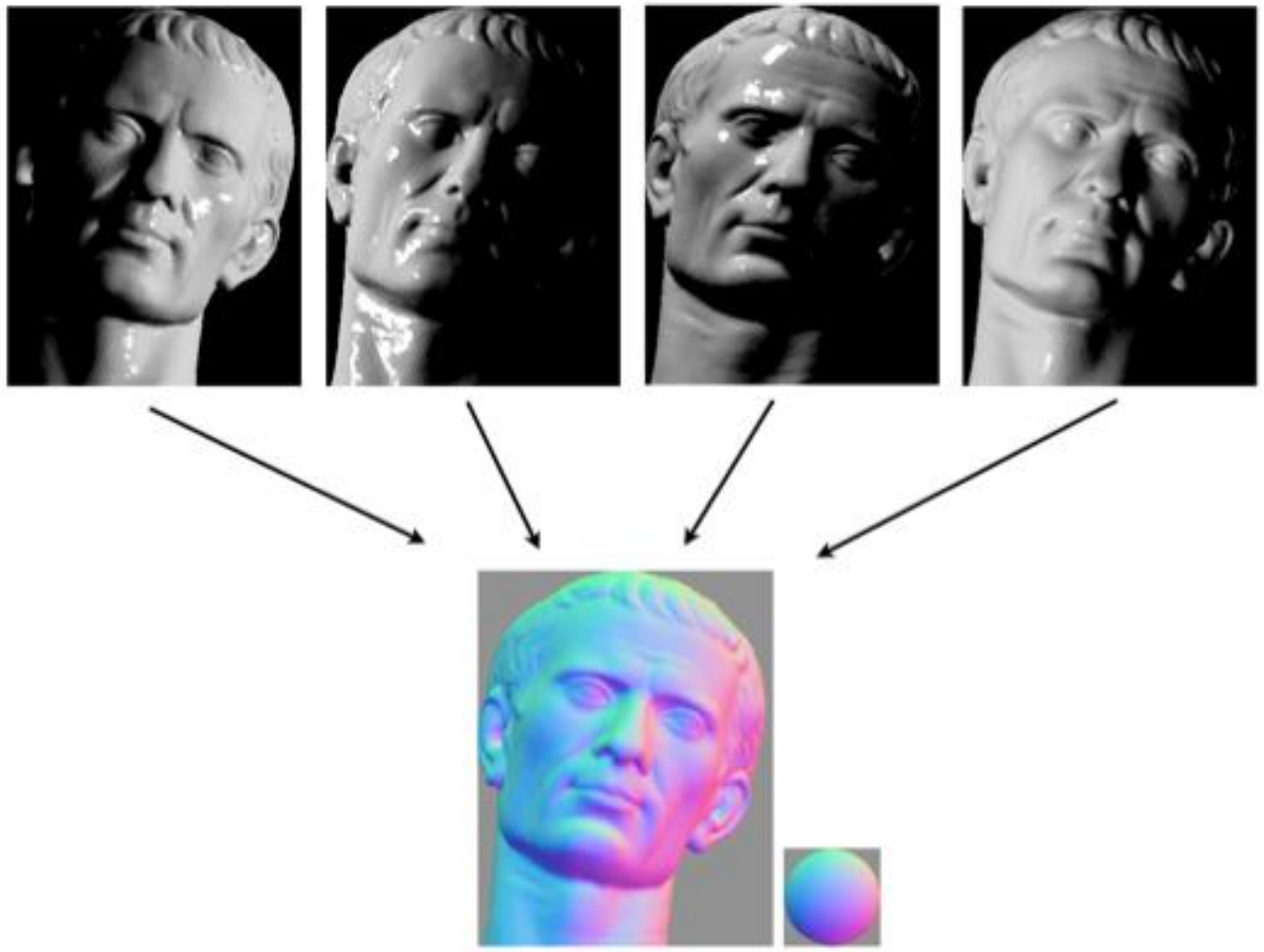
Image Credit: R. Basri et al. “Photometric Stereo with General, Unknown Lighting”, IJCV 2006

Photometric Stereo

Photometric Stereo

Photometric Stereo

Photometric Stereo



Photometric Stereo

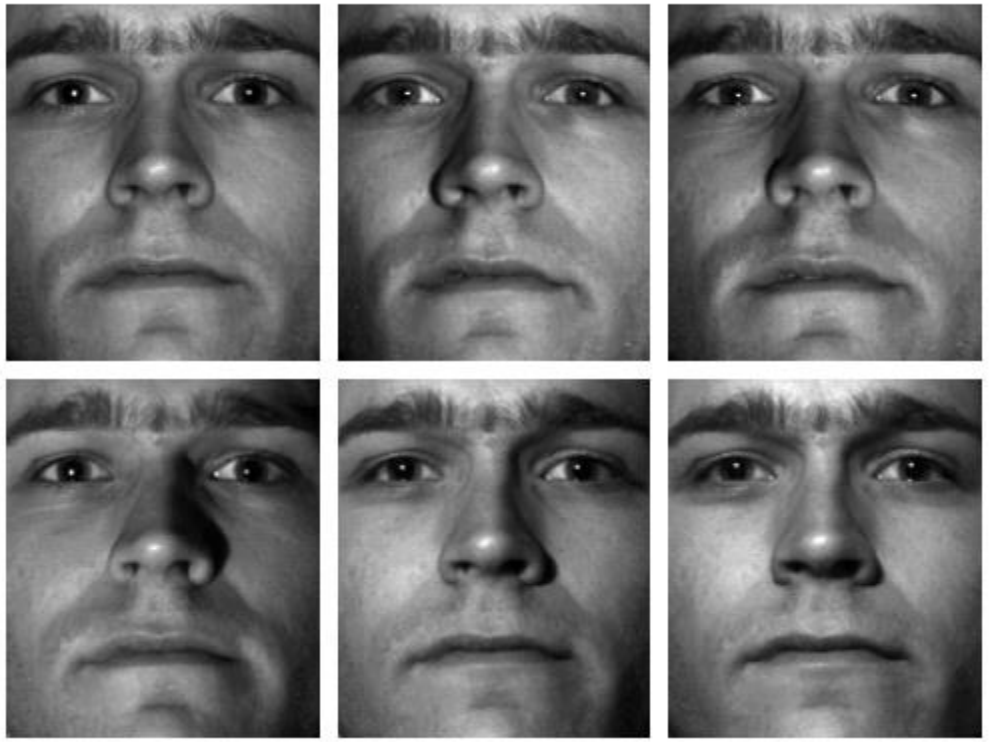
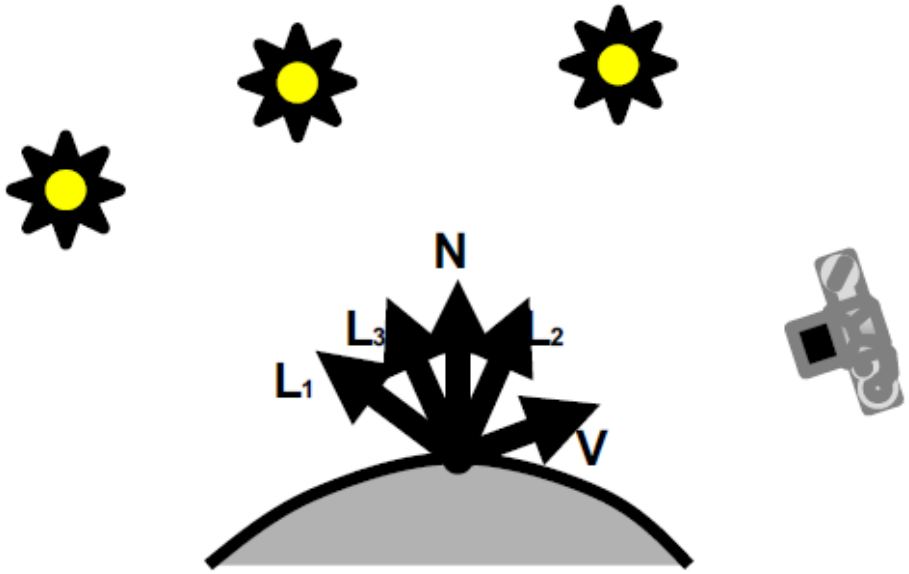


Image Credit: Athos Georghiades http://cvc.yale.edu/people/Athos.html

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:

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

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

Photometric Stereo: Solving the Equations

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

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

Next Week

- ~~High-level vision: Self-supervised Learning~~
- High-level vision: Detection
- High-level vision: Segmentation