3D Vision-MVS/SFM/Optical Flow

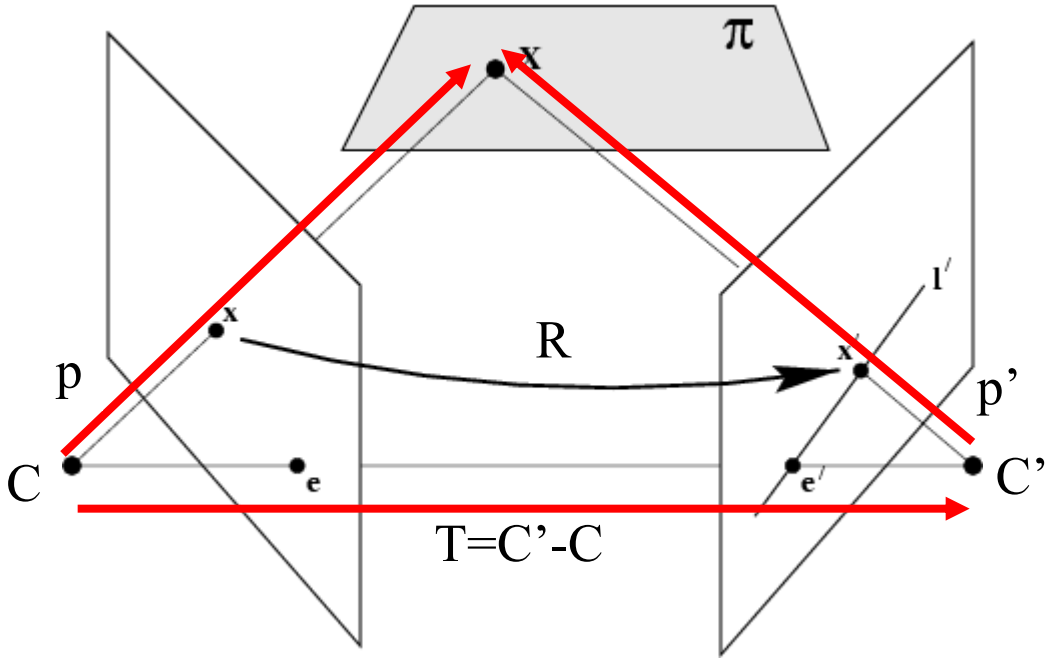
Announcements

- 3D tutorial recording issues
- Second 3D tutorial. Please vote for the tutorial time
- Survey.
  - Assignments take time.
  - More interaction during lecture
  - Weekly lab session.
  - More application-based questions
  - Assignment Difficult

Overview

- Review
- Multiple View Stereo
- Structure from Motion
- Optical Flow

Review: Essential Matrix E



$$\mathbf{p}'^{T}\,\mathbf{E}\mathbf{p} = 0$$

$$\mathbf{p}'^{\mathrm{T}}\,\mathbf{E}\mathbf{p} = 0$$

Let K and K' be the intrinsic matrices, then

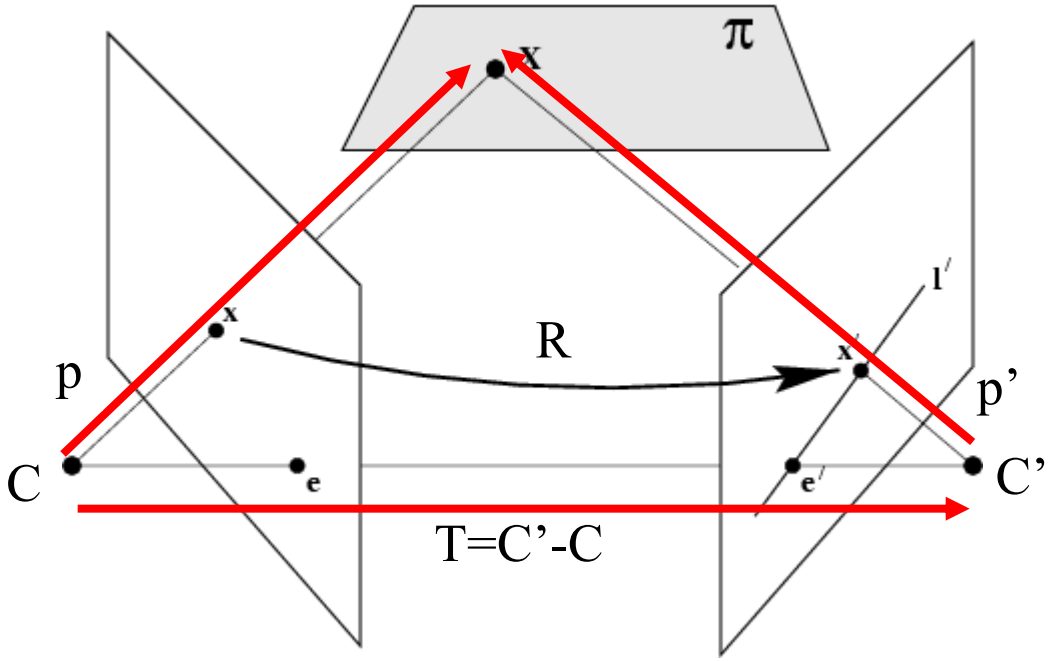
$$\mathbf{p} = \mathbf{K}^{-1}\mathbf{x} \qquad \mathbf{p}' = \mathbf{K}'^{-1}\,\mathbf{x}'$$

$$\Rightarrow \quad (\mathbf{K}'^{-1}\,\mathbf{x}')^{\mathrm{T}}\,\mathbf{E}(\mathbf{K}^{-1}\mathbf{x}) = 0$$

$$\Rightarrow \quad \mathbf{x}'^{\mathrm{T}}\,\boxed{\mathbf{K}'^{-\mathrm{T}}\,\mathbf{E}\mathbf{K}^{-1}}\,\mathbf{x} = 0$$

$$\Rightarrow \quad \mathbf{x}'^{\mathrm{T}}\,\boxed{\mathbf{F}}\,\mathbf{x} = 0 \qquad \text{Fundamental matrix}$$

F vs E



$$\mathbf{p}'^{T}\,\mathbf{E}\mathbf{p} = 0$$

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

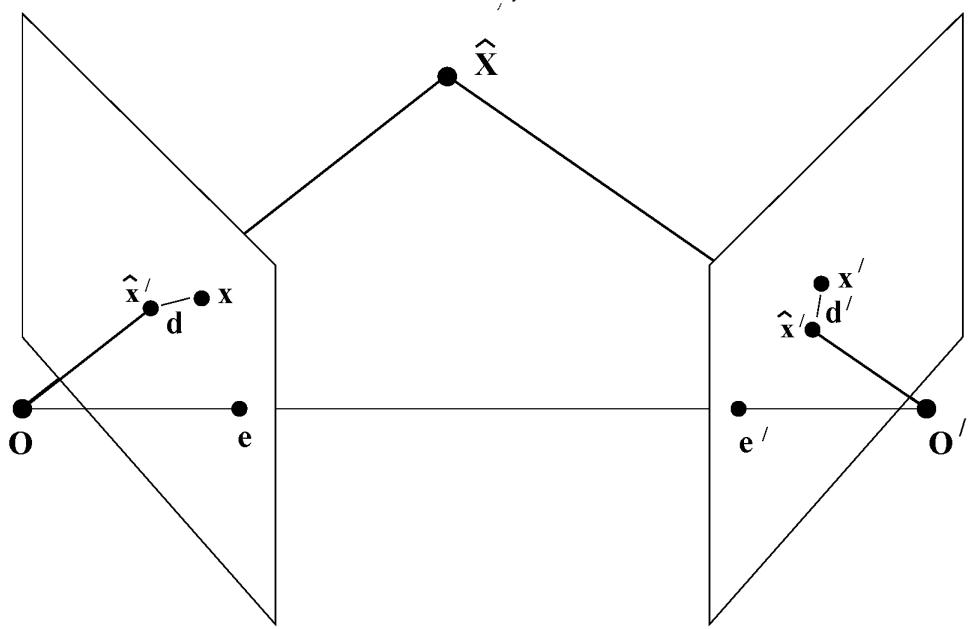
Review: Triangulation

Triangulation

Triangulation :

- Knowing $\mathtt{P}$ and $\mathtt{P}'$
- Knowing $\mathbf{x}$ and $\mathbf{x}'$
- Compute $\mathbf{X}$ such that

$$\mathbf{x} = \mathbf{P}\mathbf{X};\ \mathbf{x}' = \mathbf{P}'\mathbf{X}$$



Linear triangulation methods

- Direct analogue of the linear method of camera resectioning.
- Given equations

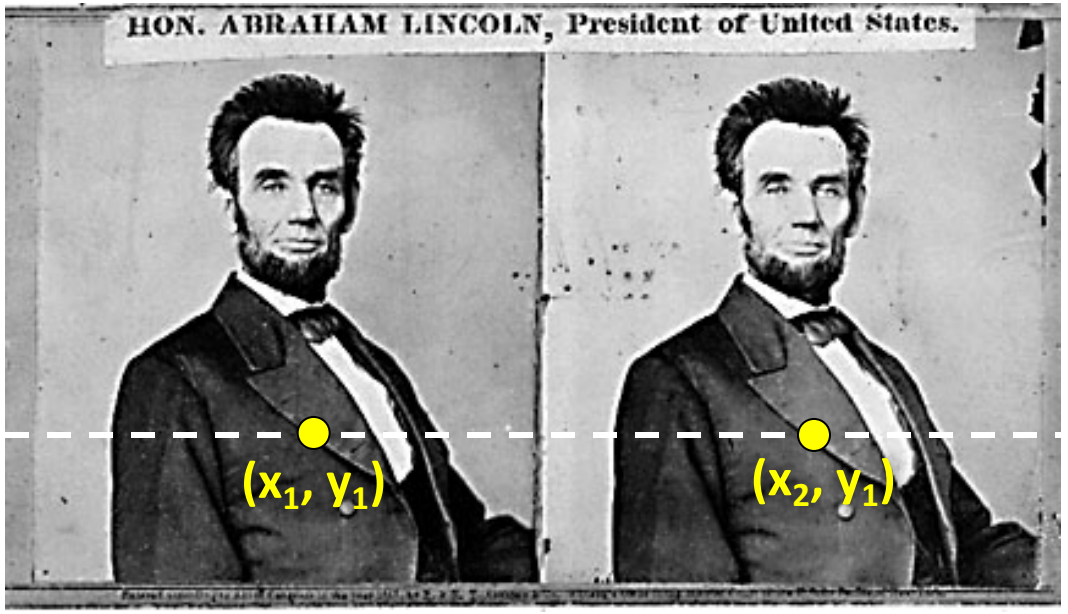
$$\mathbf{x} = \mathbf{P}\mathbf{X};\ \mathbf{x}' = \mathbf{P}'\mathbf{X}$$

- $\mathbf{p}^{i\top}$ are the rows of $\mathtt{P}$.
- Write as linear equations in $\mathbf{X}$

$$\begin{bmatrix} x\mathbf{p}^{3\top} - \mathbf{p}^{1\top} \\ y\mathbf{p}^{3\top} - \mathbf{p}^{2\top} \\ x'\mathbf{p}'^{3\top} - \mathbf{p}'^{1\top} \\ y\mathbf{p}'^{3\top} - \mathbf{p}'^{2\top} \end{bmatrix} \mathbf{X} = 0$$

- Solve for $\mathbf{X}$.
- Generalizes to point match in several images.
- Minimizes no meaningful quantity – not optimal.

Review: Stereo-Epipolar geometry

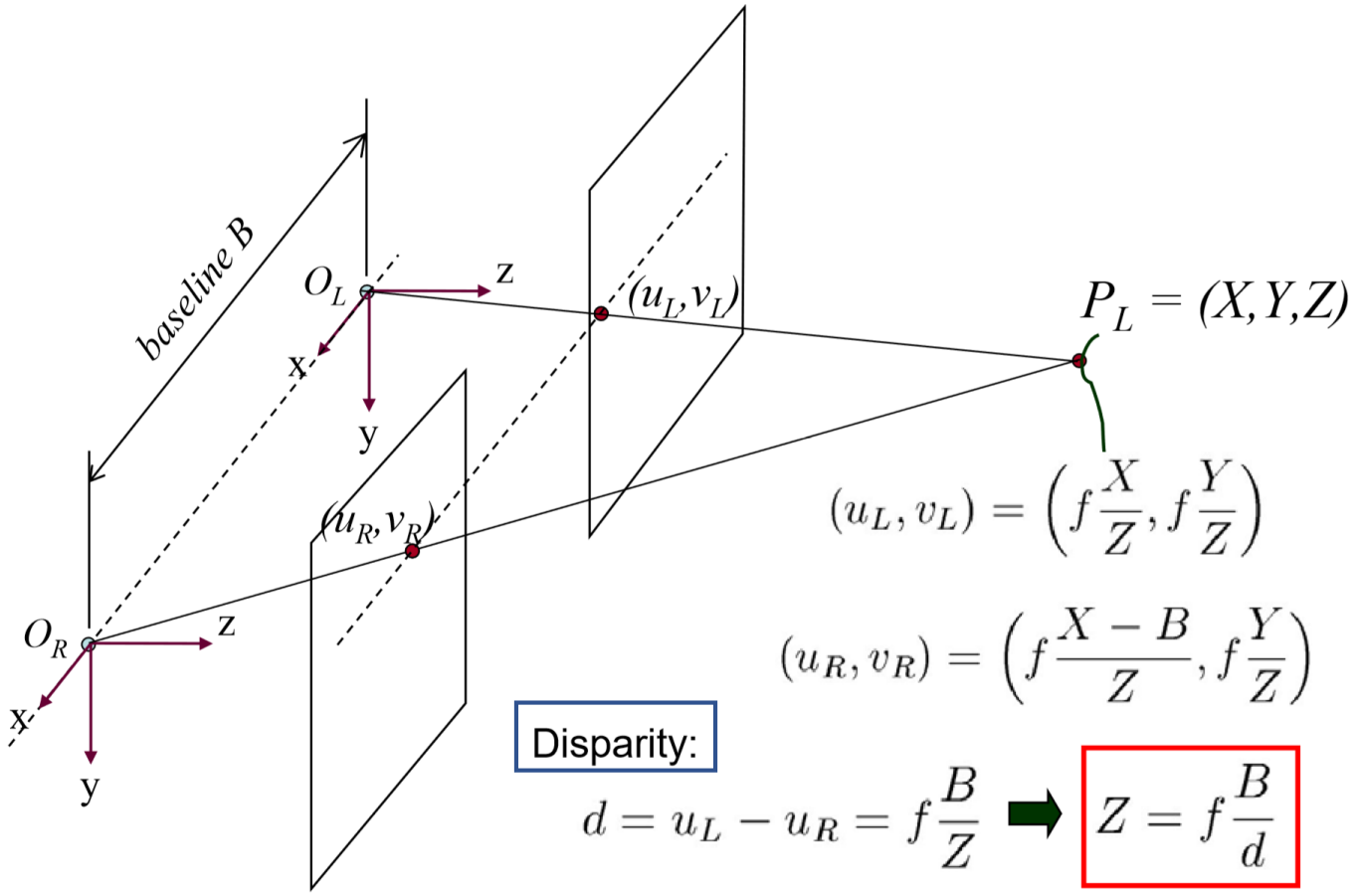


Two images captured by a purely horizontal translating camera (*rectified* stereo pair)

$x_2 - x_1$ = the *disparity* of pixel (x_1, y_1)

Credit: Noah Snavely

Disparity Computation



Multiple View Stereo

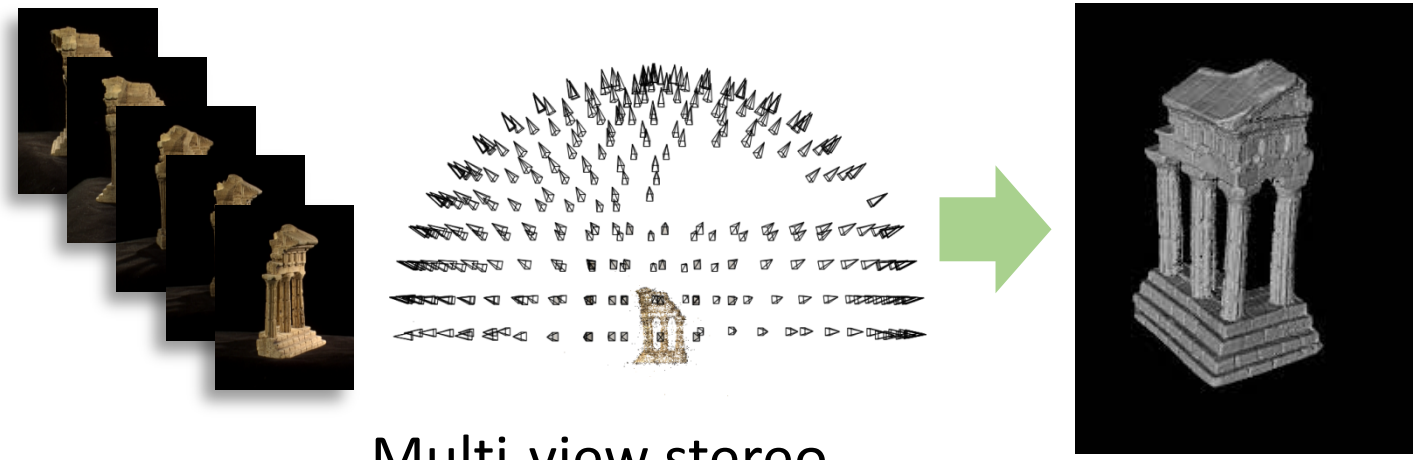
Readings

- *Multi-View Stereo: A Tutorial*, Furukawa and Hernandez, 2015
  - http://carlos-hernandez.org/papers/fnt_mvs_2015.pdf
- Chapter 12.1.2, Computer Vision: Algorithms and Applications

Multi-view Stereo

Problem formulation: given several images of the same object or scene, compute a representation of its 3D shape

Binocular Stereo



Multi-view stereo

Credit: Noah Snavely

Multi-view Stereo

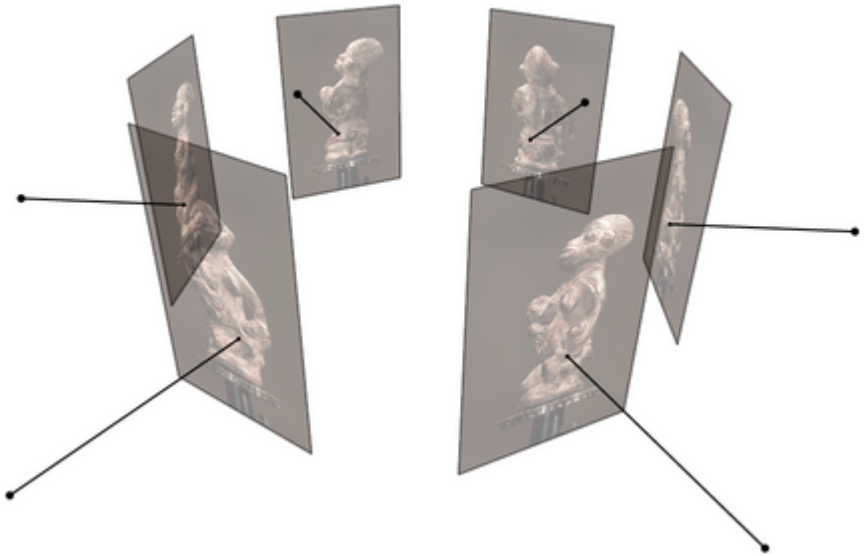
CMU's Panoptic Studio

Credit: Noah Snavely

Multi-view Stereo

Input: calibrated images from several viewpoints (known intrinsics and extrinsics / projection matrices)

Output: 3D object model

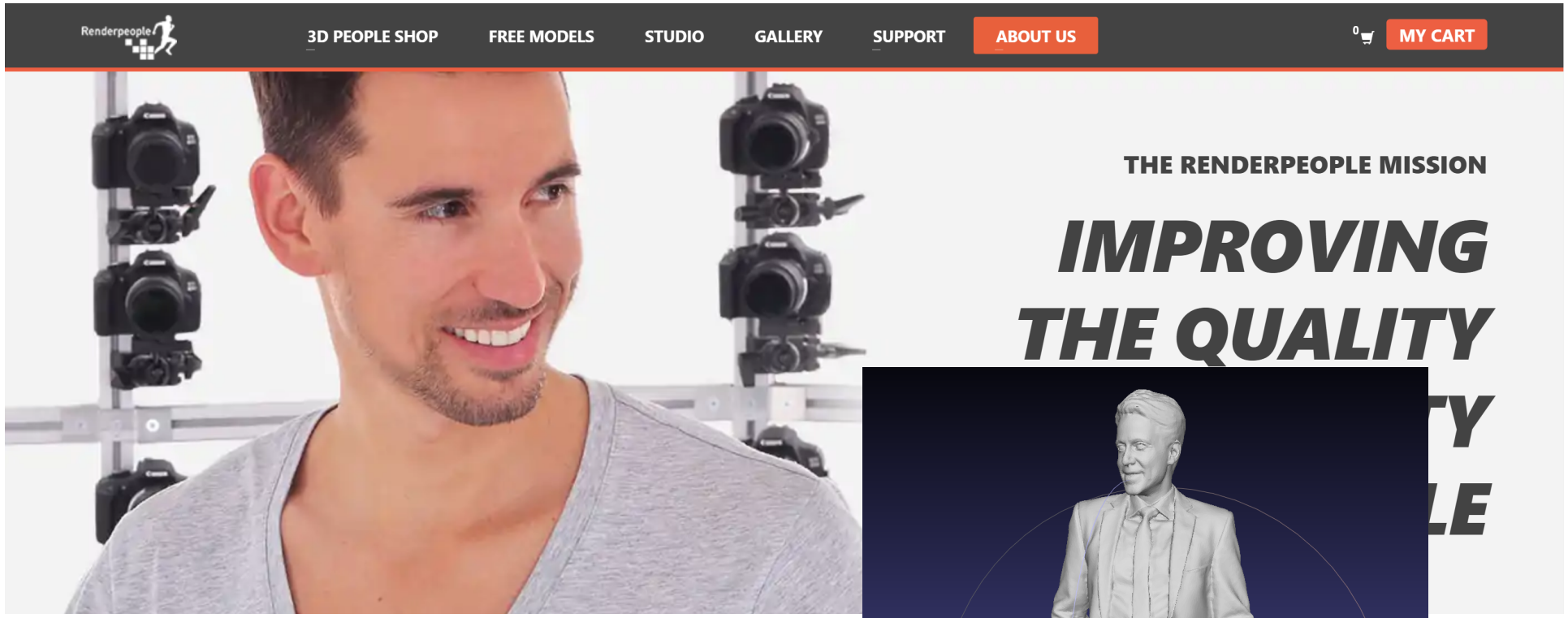


Figures by Carlos Hernandez

Credit: Noah Snavely

Applications

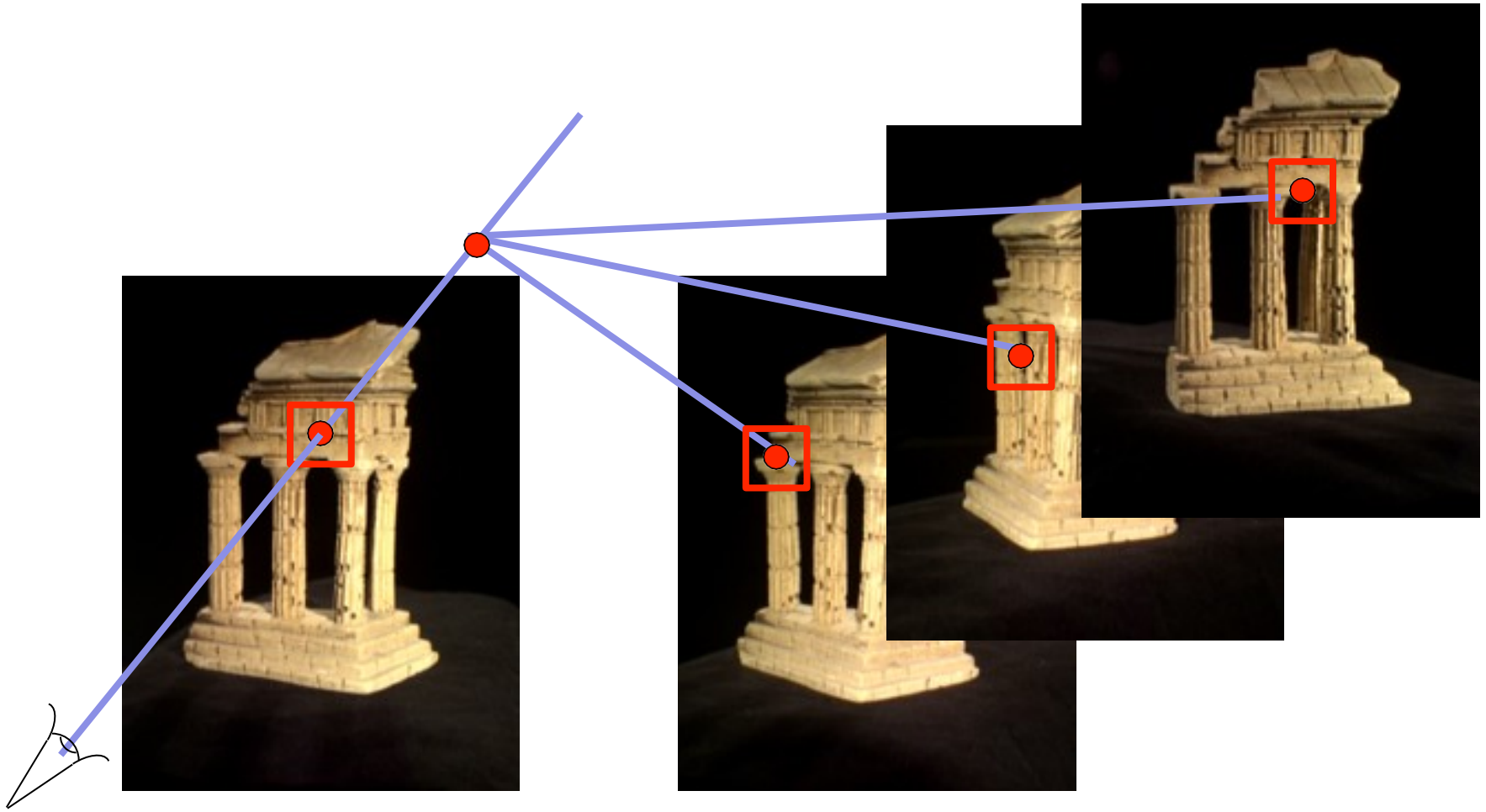
Credit: Noah Snavely



https://renderpeople.com/about-us/

Credit: Noah Snavely

Multi-view stereo: Basic idea

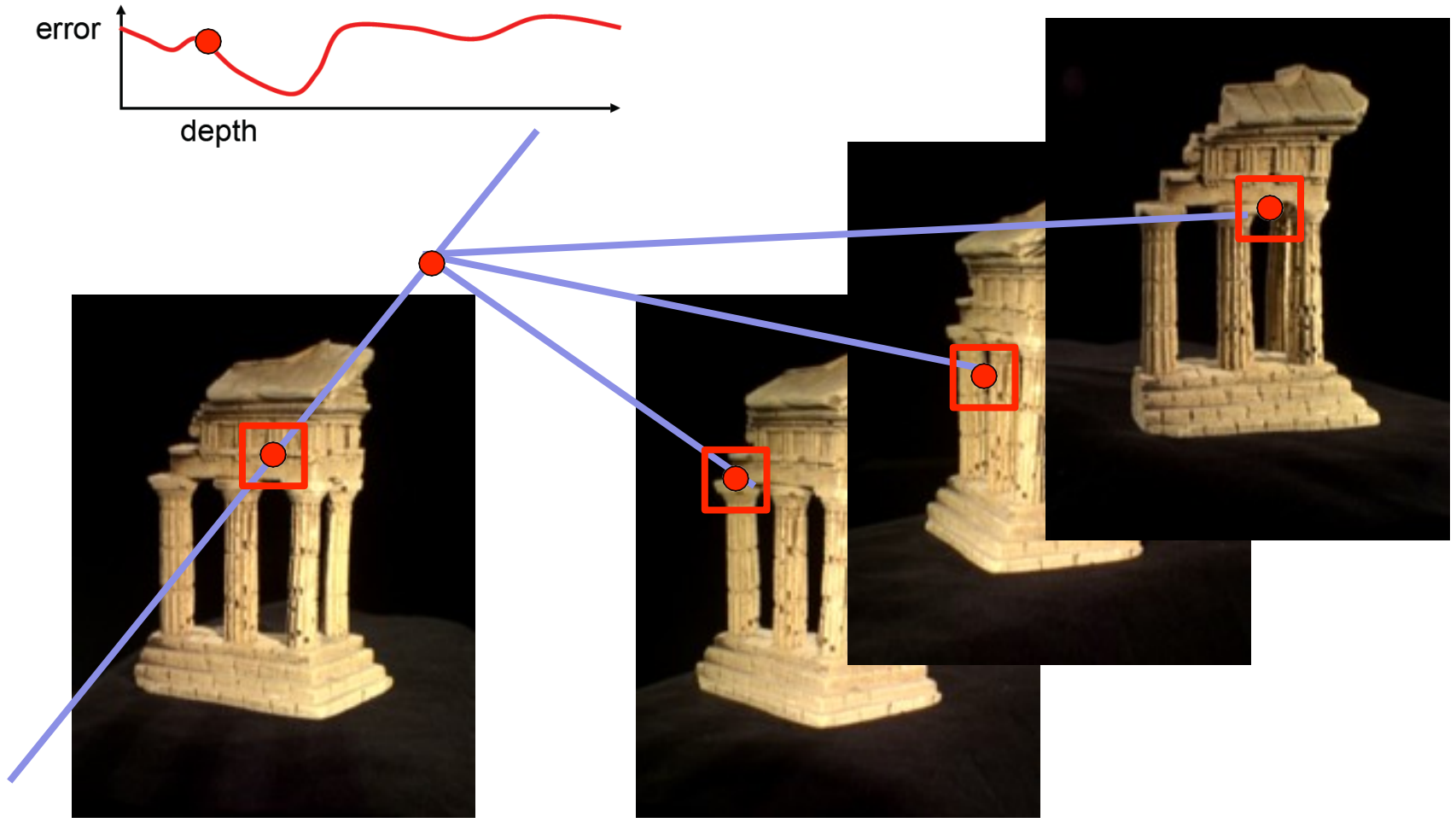


reference view

neighbor views

Source: Y. Furukawa

Multi-view stereo: Basic idea

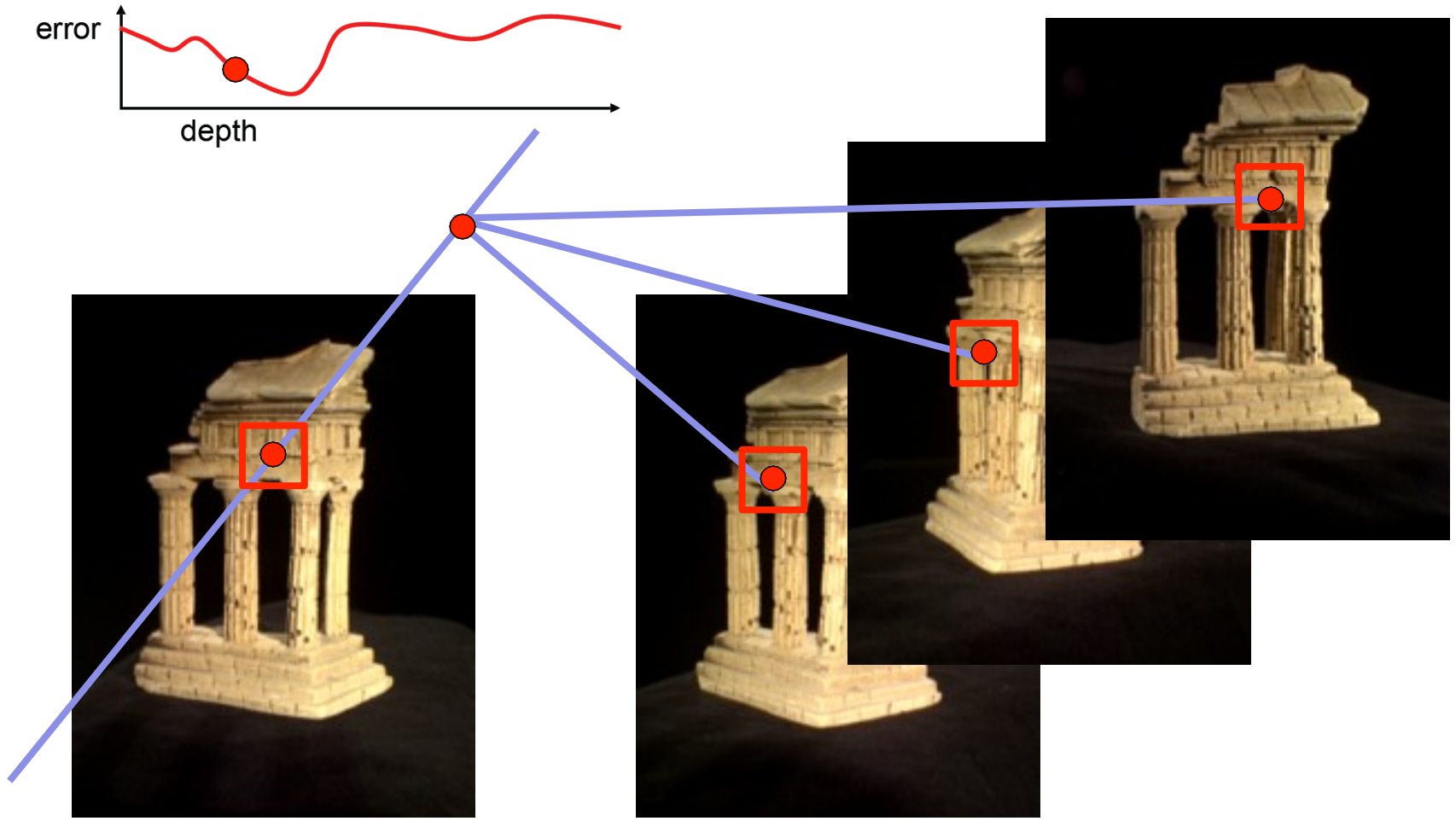


reference view

neighbor views

Source: Y. Furukawa

Multi-view stereo: Basic idea

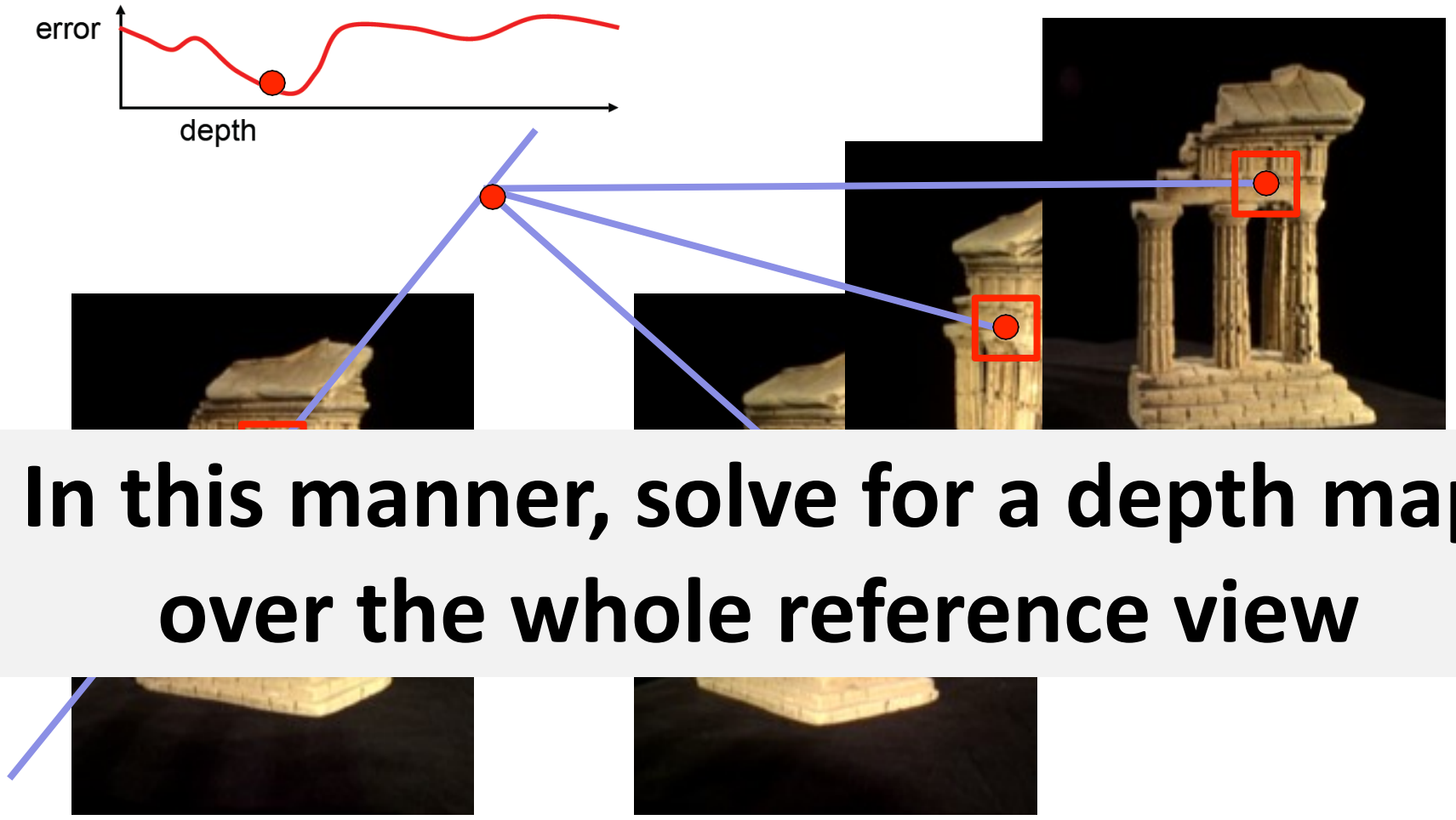


reference view

neighbor views

Source: Y. Furukawa

Multi-view stereo: Basic idea



In this manner, solve for a depth map over the whole reference view

reference view

neighbor views

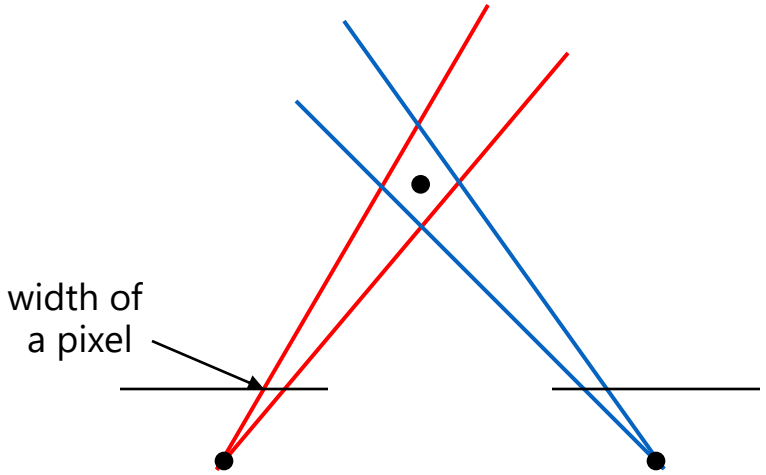
Source: Y. Furukawa

Multi-view stereo: advantages

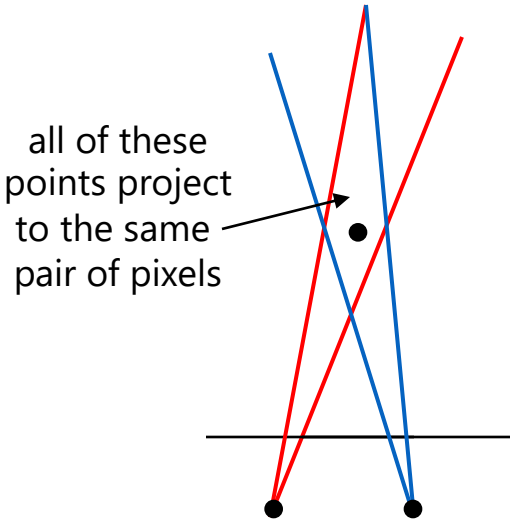
- Can match windows using more than 1 neighbor, giving a **stronger match signal**
- If you have lots of potential neighbors, can **choose the best subset** of neighbors to match per reference image
- Can reconstruct a depth map for each reference frame, and the merge into a **complete 3D model**

Credit: Noah Snavely

Choosing the stereo baseline



Large Baseline



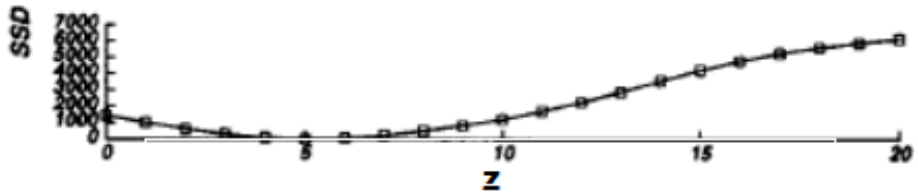
Small Baseline

What's the optimal baseline?

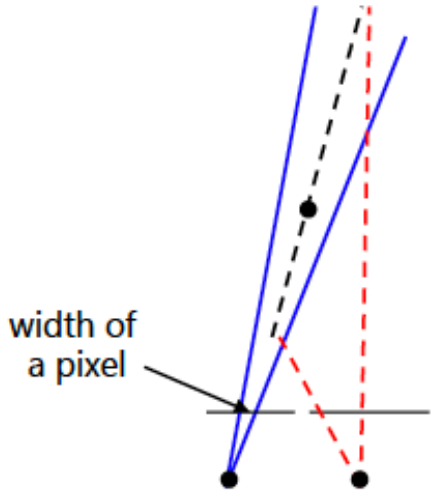
- Too small: large depth error
- Too large: difficult search problem

Credit: Noah Snavely

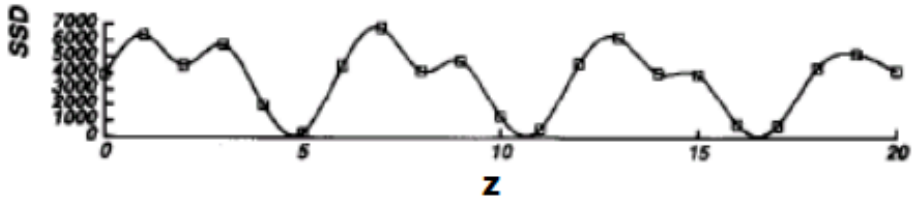
Multiple-baseline stereo



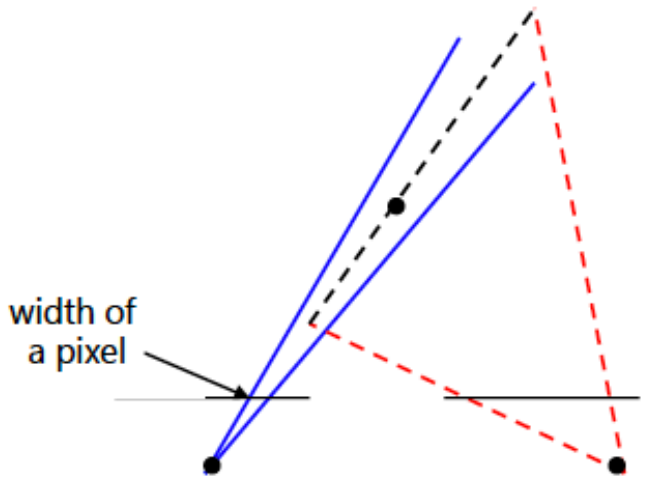
pixel matching score



- For short baselines, estimated depth will be less precise due to narrow triangulation



- For larger baselines, must search larger area in second image



M. Okutomi and T. Kanade, "A Multiple-Baseline Stereo System," IEEE Trans. on Pattern Analysis and Machine Intelligence, 15(4):353-363 (1993).

Credit: Noah Snavely

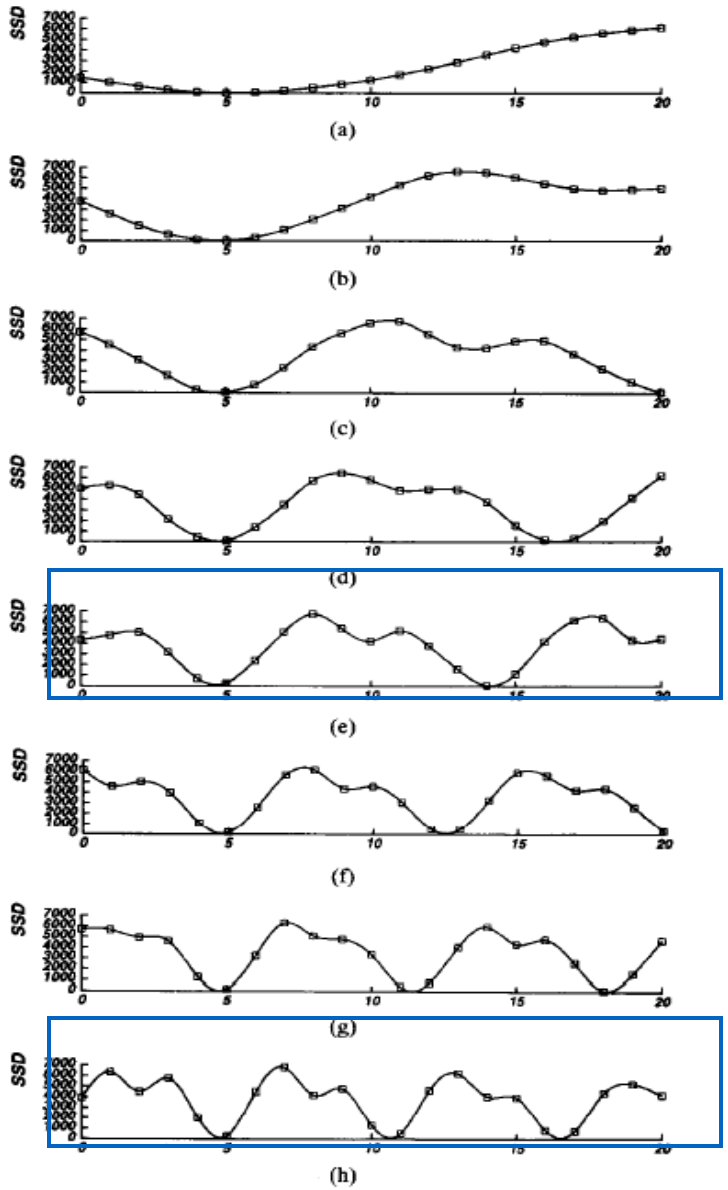


Fig. 5. SSD values versus inverse distance: (a) $B = b$; (b) $B = 2b$; (c) $B = 3b$; (d) $B = 4b$; (e) $B = 5b$; (f) $B = 6b$; (g) $B = 7b$; (h) $B = 8b$. The horizontal axis is normalized such that $8bF = 1$.

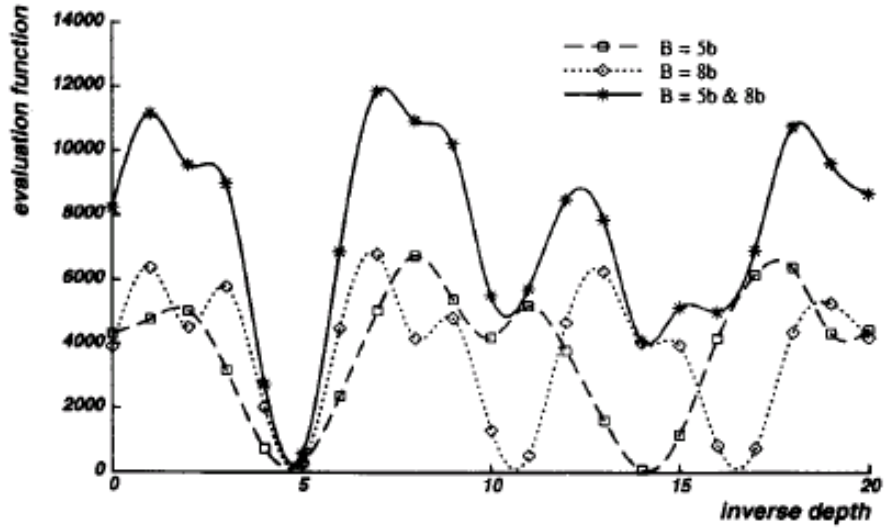


Fig. 6. Combining two stereo pairs with different baselines.

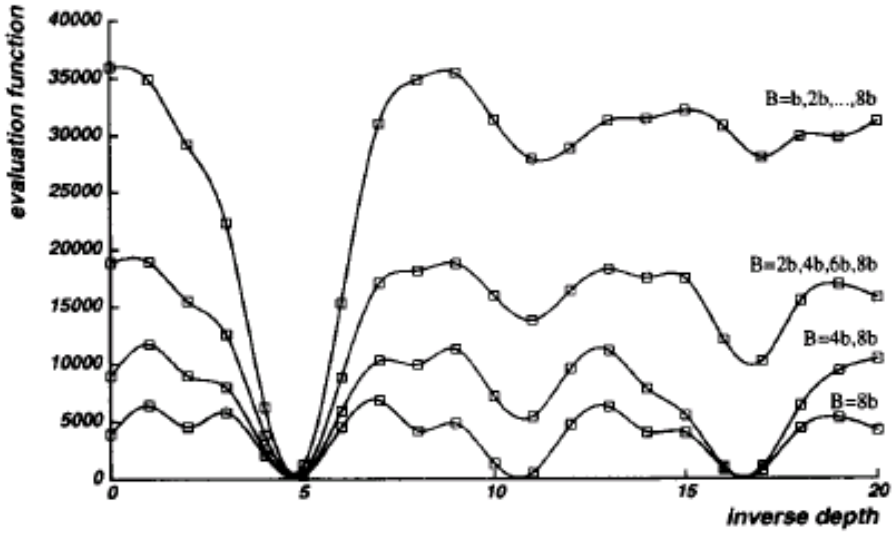
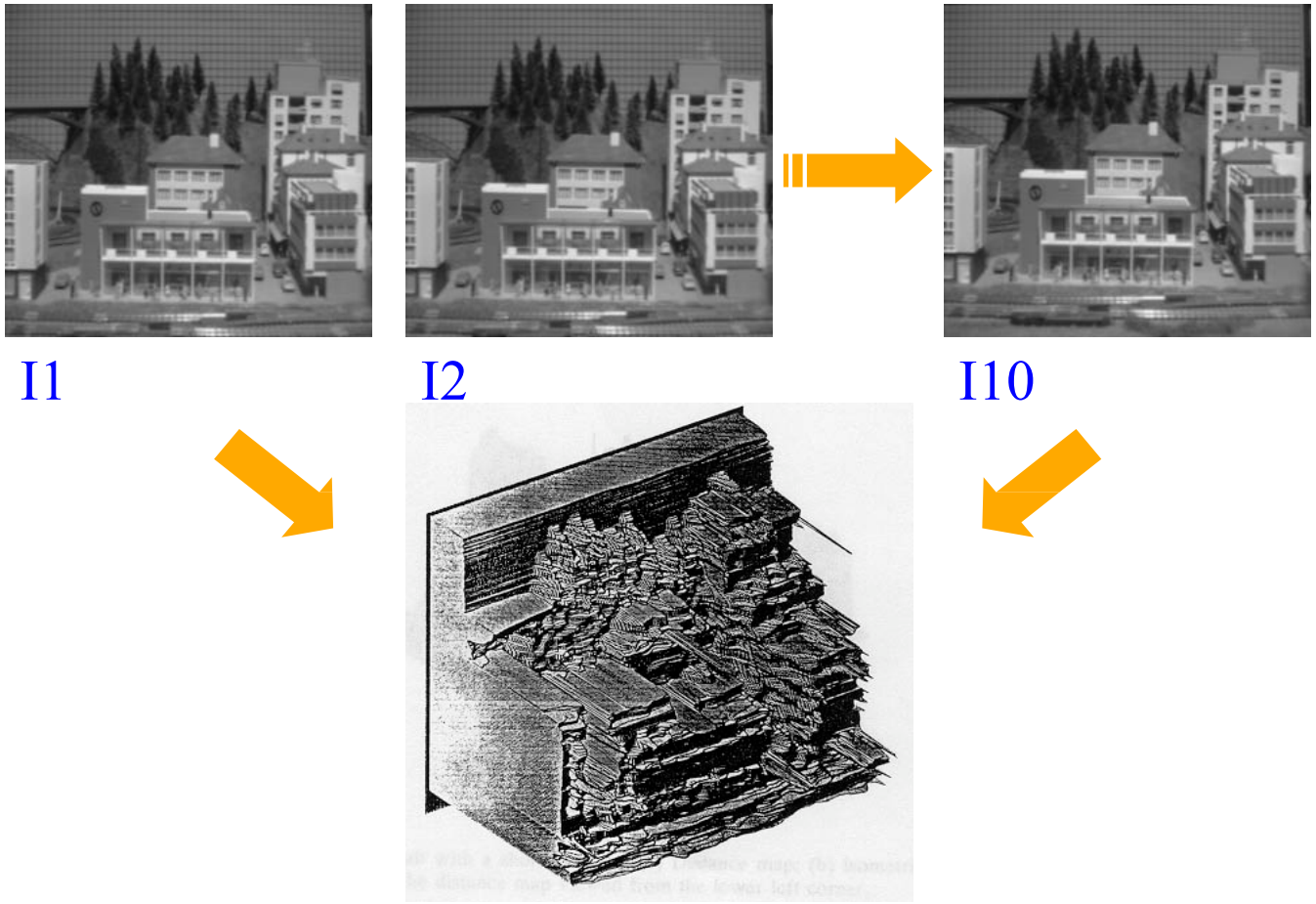


Fig. 7. Combining multiple baseline stereo pairs.

Credit: Noah Snavely

Multiple-baseline stereo results



M. Okutomi and T. Kanade, *A Multiple-Baseline Stereo System,* IEEE Trans. on Pattern Analysis and Machine Intelligence, 15(4):353-363 (1993).

Credit: Noah Snavely

Multibaseline Stereo

Basic Approach

- Choose a reference view
- Use your favorite stereo algorithm BUT
  - replace two-view SSD with **SSSD** over all baselines
  - **SSSD**: the SSD values are computed first for each pair of stereo images, and then add all together from multiple stereo pairs.

Limitations

- Only gives a depth map (not an “object model”)
- Won’t work for widely distributed views.

Credit: Noah Snavely

Problem: *visibility*

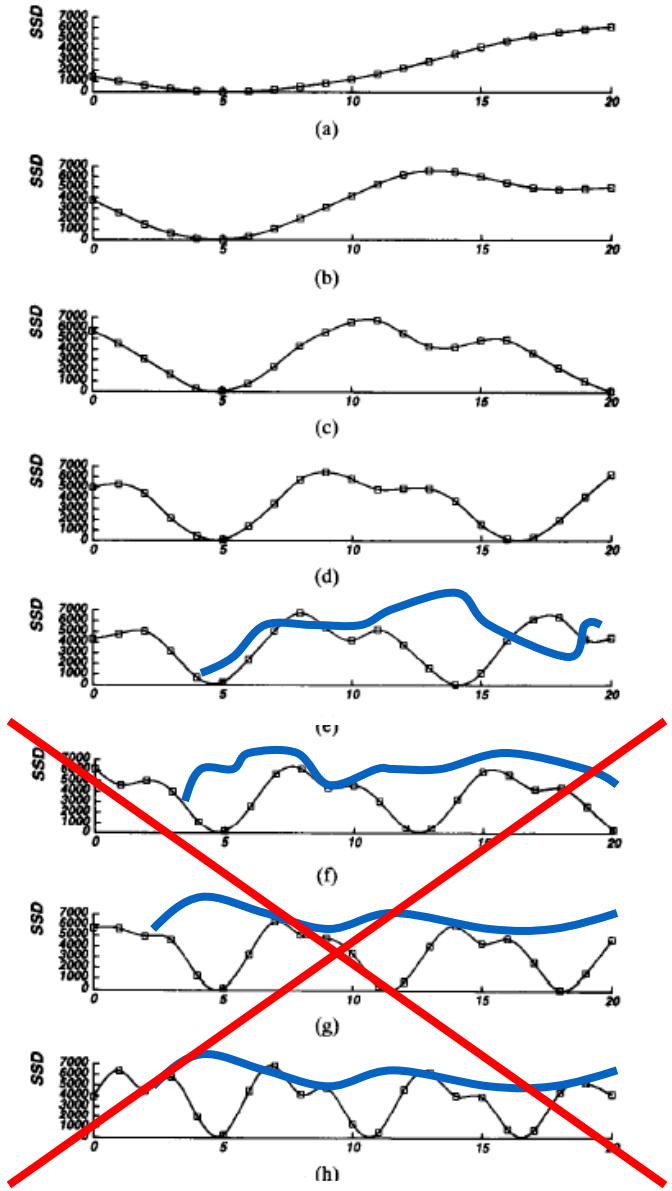


Fig. 5. SSD values versus inverse distance: (a) $B = b$; (b) $B = 2b$; (c) $B = 3b$; (d) $B = 4b$; (e) $B = 5b$; (f) $B = 6b$; (g) $B = 7b$; (h) $B = 8b$. The horizontal axis is normalized such that $8bF = 1$.

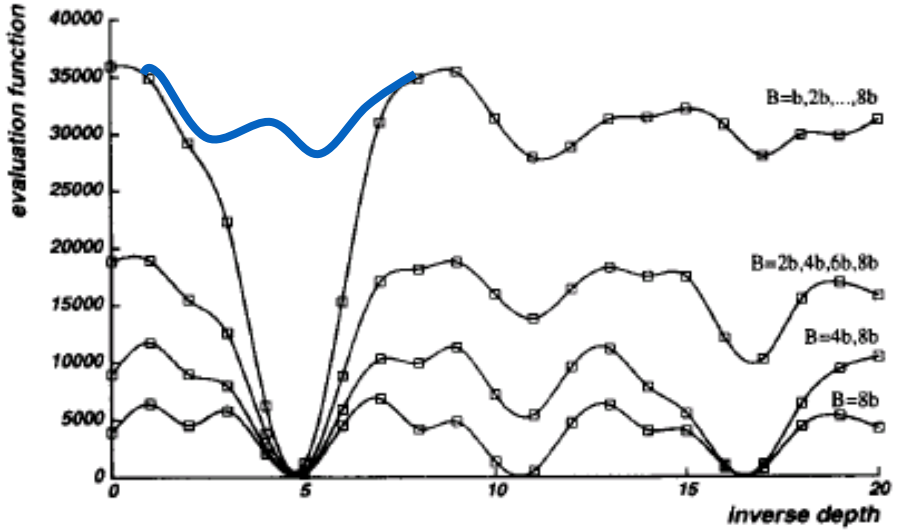


Fig. 7. Combining multiple baseline stereo pairs.

Some Solutions

- Match only nearby photos [Narayanan 98]
- Use NCC instead of SSD,
  Ignore NCC values > threshold
  [Hernandez & Schmitt 03]

Credit: Noah Snavely

Popular matching scores

- SSD (Sum of Squared Differences) $\sum_{x,y} |W_1(x,y) - W_2(x,y)|^2$
- SAD (Sum of Absolute Differences) $\sum_{x,y} |W_1(x,y) - W_2(x,y)|$
- ZNCC (Zero-mean Normalized Cross Correlation)

$$\frac{\sum_{x,y}(W_1(x,y) - \overline{W_1})(W_2(x,y) - \overline{W_2})}{\sigma_{W_1}\sigma_{W_2}}$$

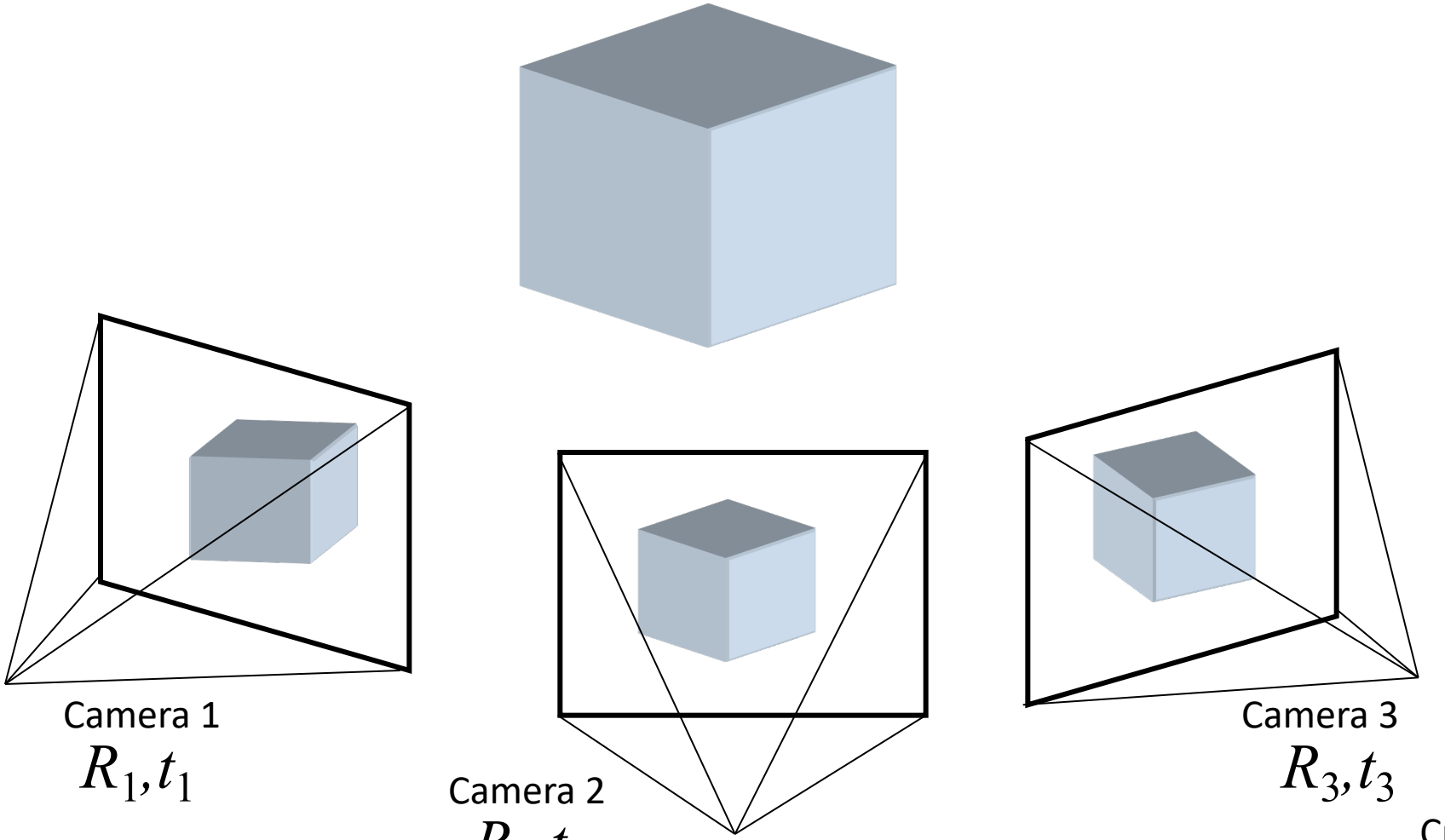
SAD is more robust here, because if there is a very large difference in some image, SSD will focus on the largest error and less effected by the outliers.

The lighting condition might be different, thus change the mean of the image.
NCC can minimize this influence.

- where $\overline{W_i} = \frac{1}{n}\sum_{x,y} W_i$ $\sigma_{W_i} = \sqrt{\frac{1}{n}\sum_{x,y}(W_i - \overline{W_i})^2}$
- what advantages might NCC have?

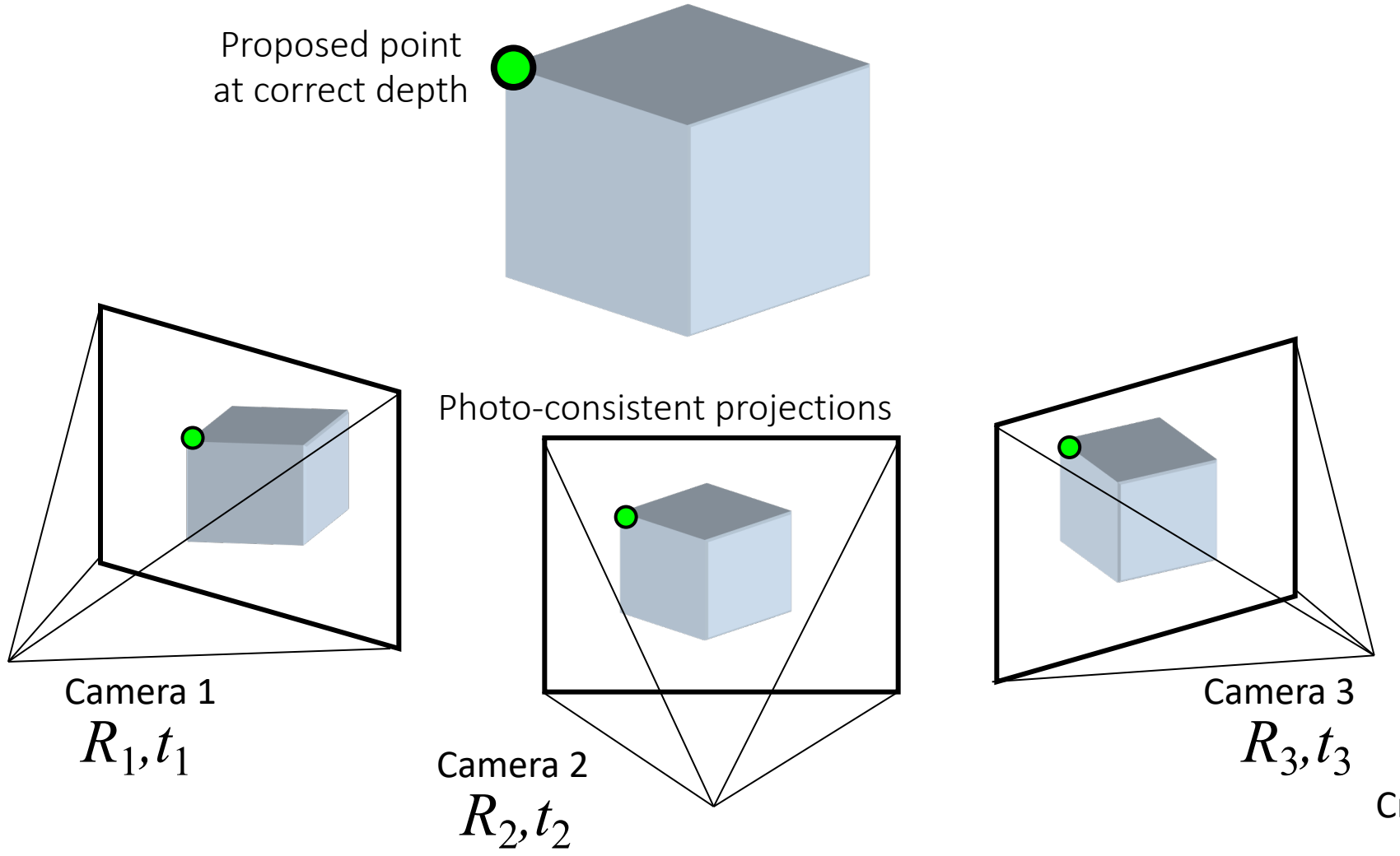
Credit: Noah Snavely

Plane-Sweep Stereo



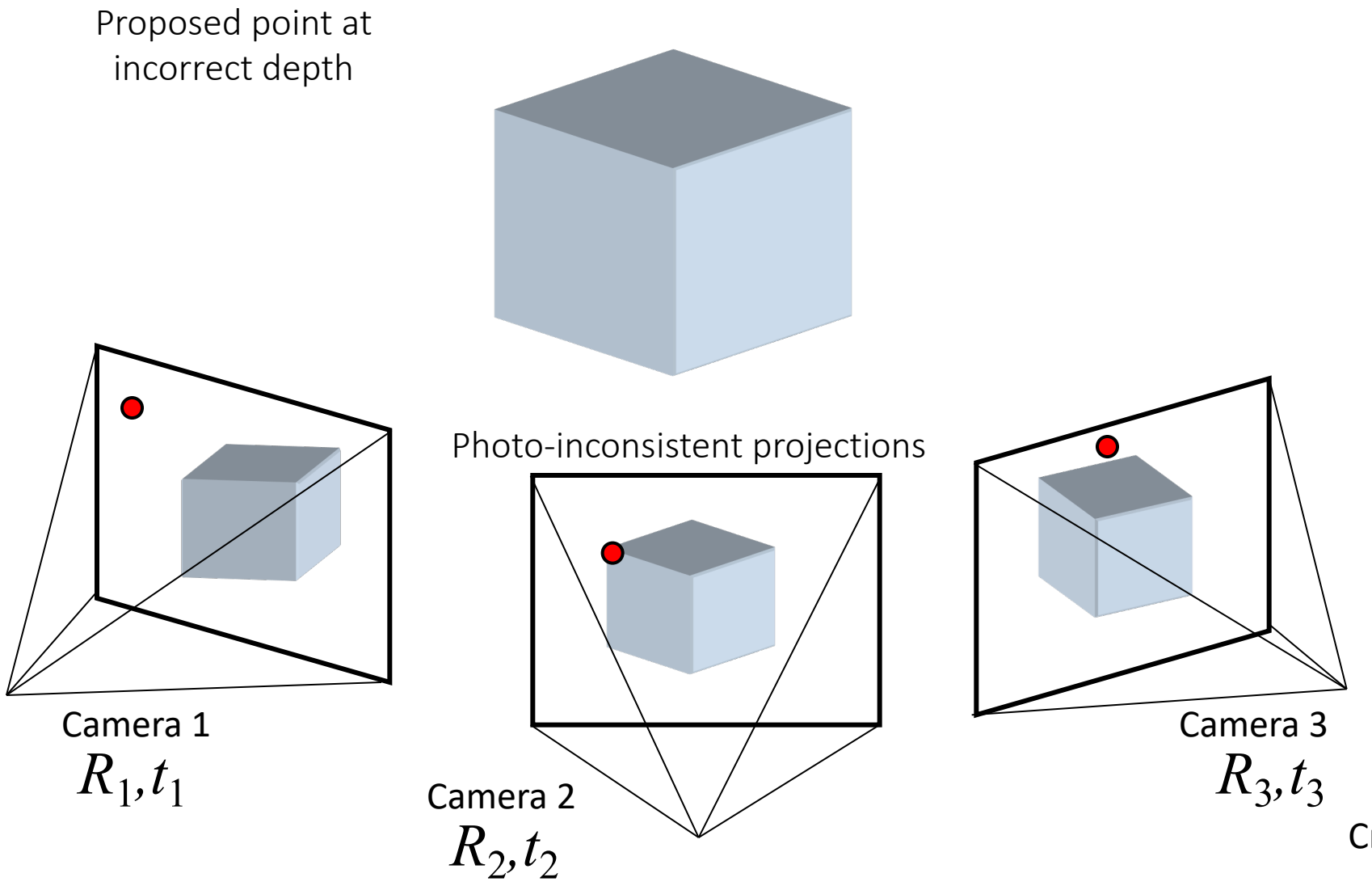
Credit: Noah Snavely

Plane-Sweep Stereo



Credit: Noah Snavely

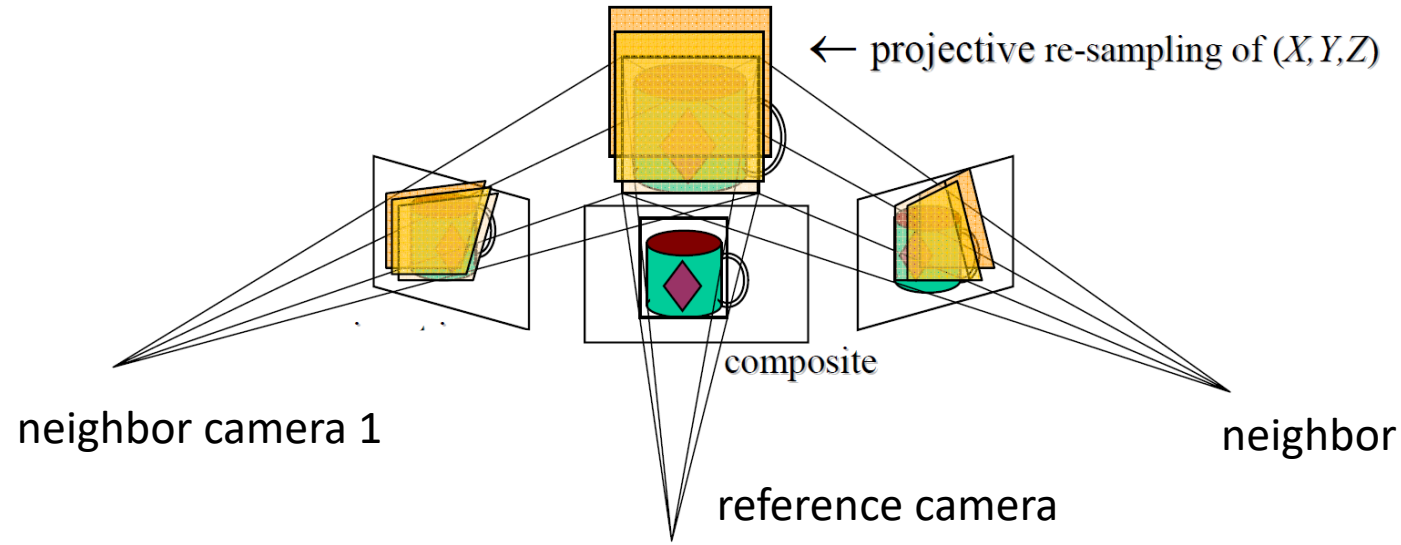
Plane-Sweep Stereo



Credit: Noah Snavely

Plane-Sweep Stereo

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



Credit: Noah Snavely

Plane-Sweep Stereo

Left neighbor

Reference image

Right neighbor

Left neighbor projected into reference image

Average images on each plane

Right neighbor projected into reference image

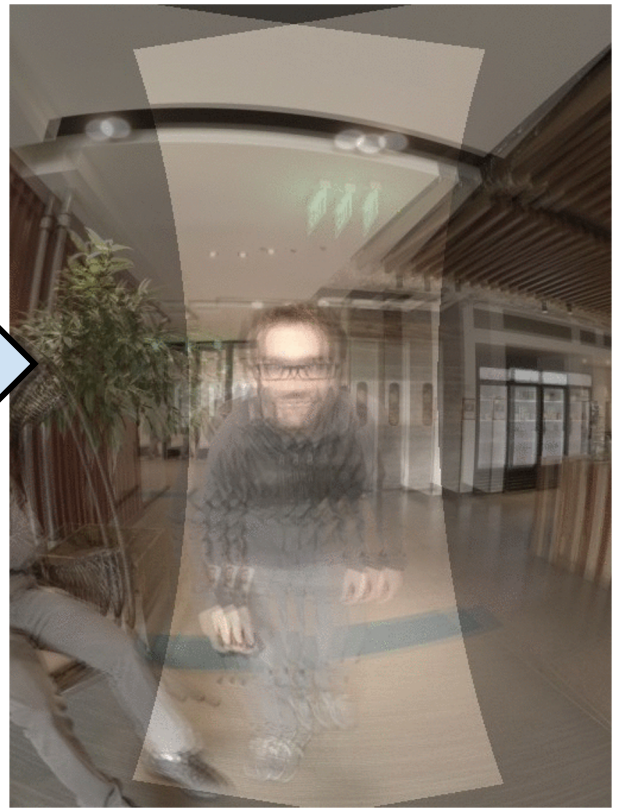
Credit: Noah Snavely

Another example

Left neighbor

Reference image

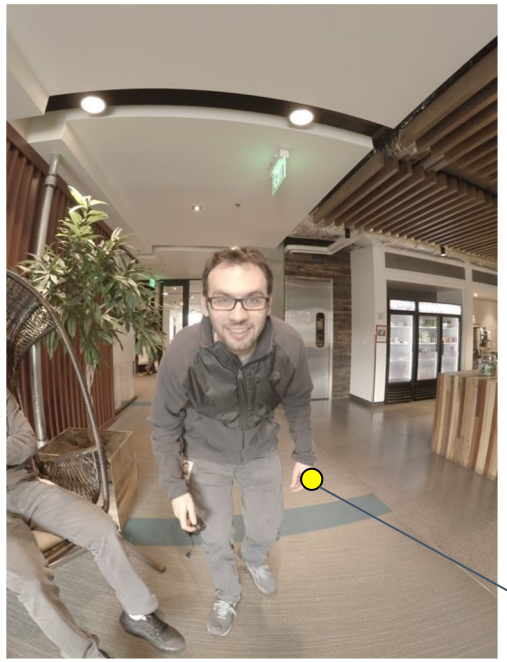
Right neighbor



Planar image reprojections swept over depth (averaged)

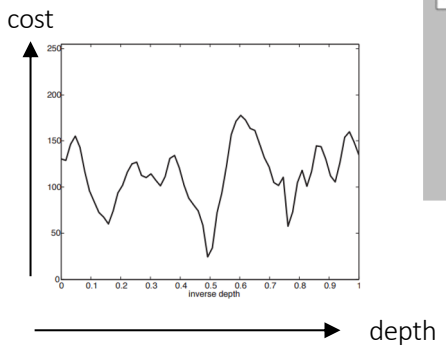
Credit: Noah Snavely

Cost Volumes -> Depth Maps

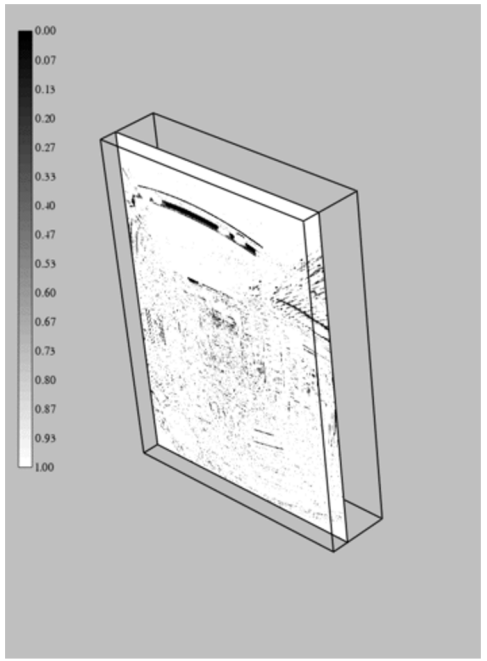


Reference image

Plane sweep



Single pixel's cost profile



Full cost volume

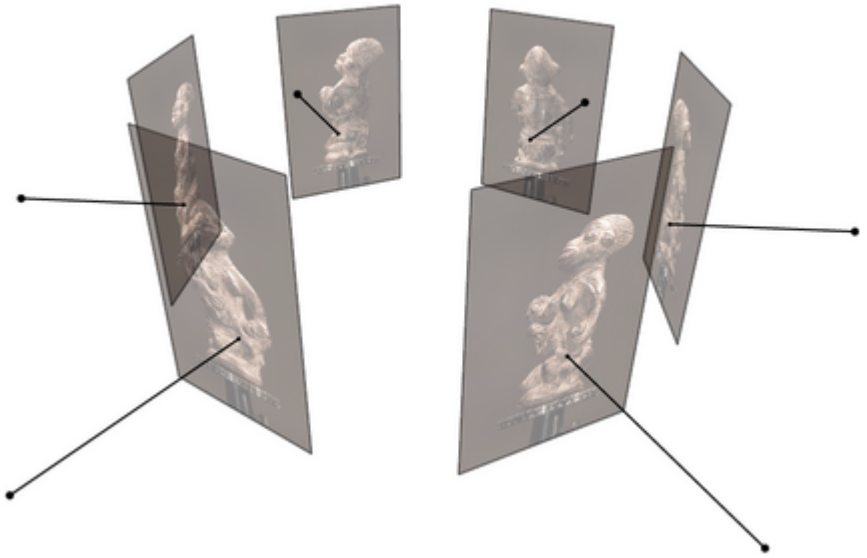
Depth map solver

(Belief propagation, graph cuts, etc.)

Credit: Noah Snavely

Fusing multiple depth maps

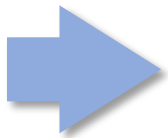
- Compute depth map per image
- Fuse the depth maps into a 3D model



Figures by Carlos Hernandez

Credit: Noah Snavely

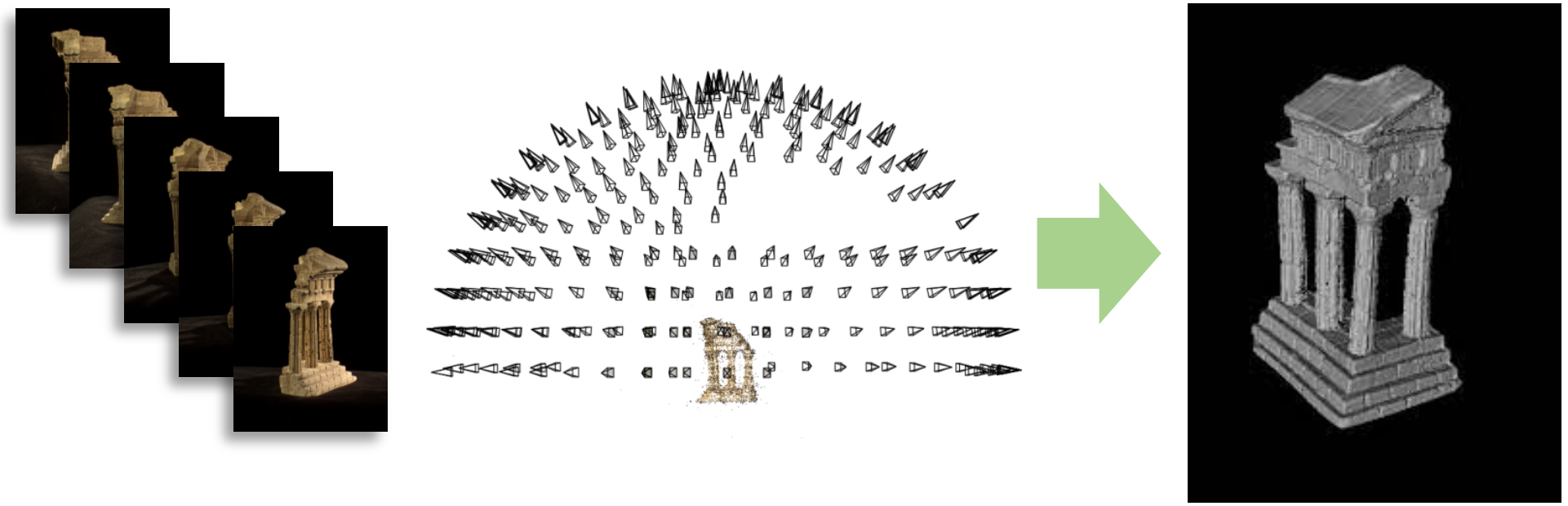
Structure From Motion



Readings

Reminder: multi-view stereo

Problem formulation: given several images of the same object or scene, compute a representation of its 3D shape



Structure from motion

- Multi-view stereo assumes that cameras are calibrated
  - Extrinsics and intrinsics are known for all views

- How do we compute calibration if we don’t know it? In general, this is called *structure from motion*

Large-scale structure from motion

Dubrovnik, Croatia. 4,619 images (out of an initial 57,845).
Total reconstruction time: 23 hours
Number of cores: 352

Two views

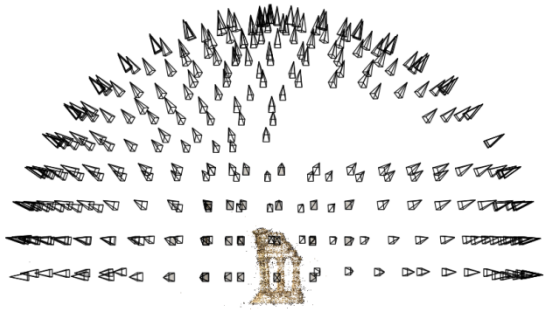
- Solve for Fundamental matrix / Essential matrix
- Factorize into intrinsics, rotation, and translation

Structure from motion

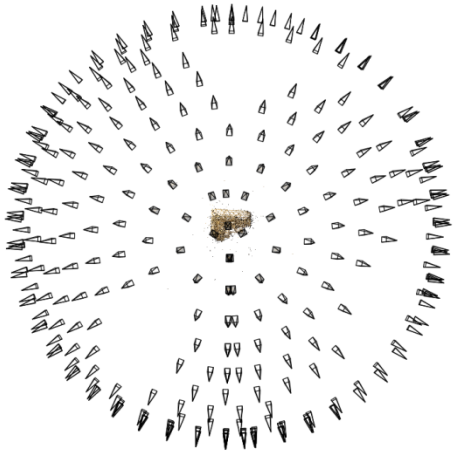
- Given many images, how can we
  a) figure out where they were all taken from?
  b) build a 3D model of the scene?

This is the **structure from motion (SfM)** problem

Structure from motion



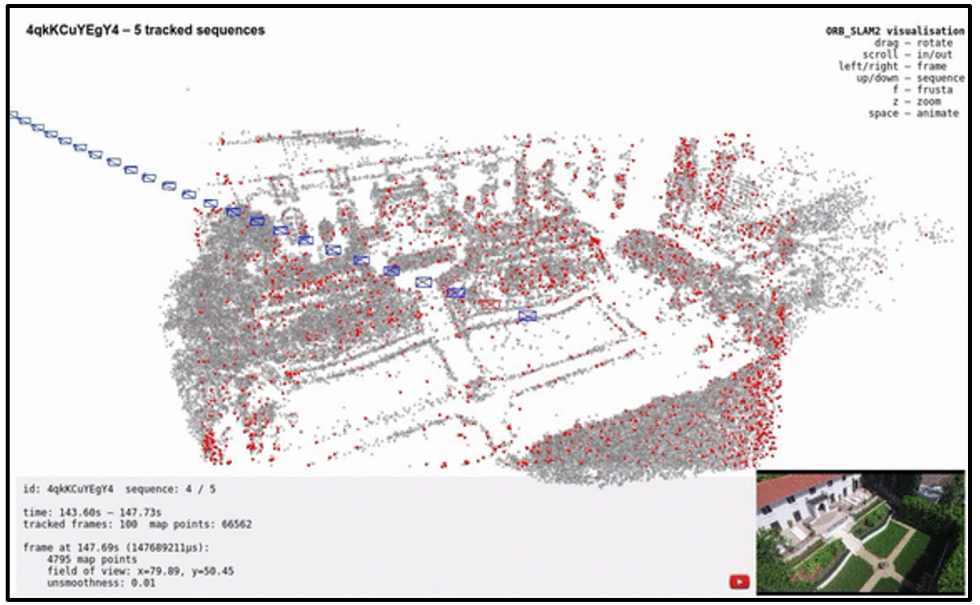
Reconstruction (side)



(top)

- Input: images with 2D points in correspondence
  $p_{i,j} = (u_{i,j}, v_{i,j})$
- Output
  - structure: 3D location $\mathbf{x}_i$ for each point p_i
  - motion: camera parameters $\mathbf{R}_j$, $\mathbf{t}_j$ & possibly $\mathbf{K}_j$
- Objective function: minimize *reprojection error*

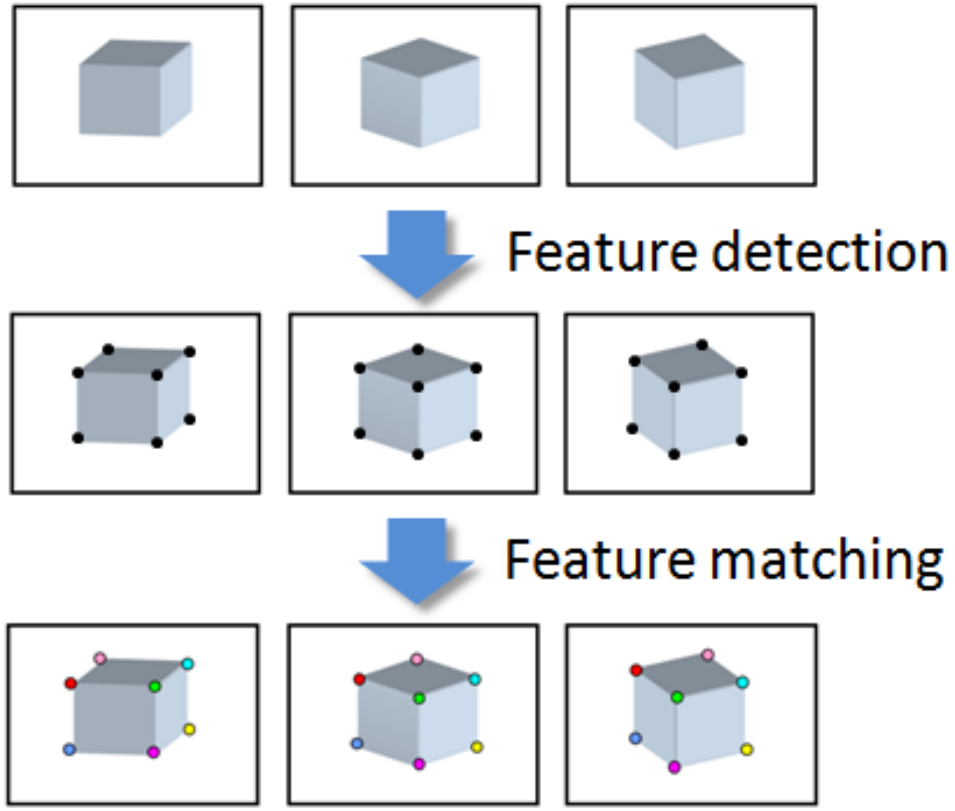
Also doable from video



What we've seen so far…

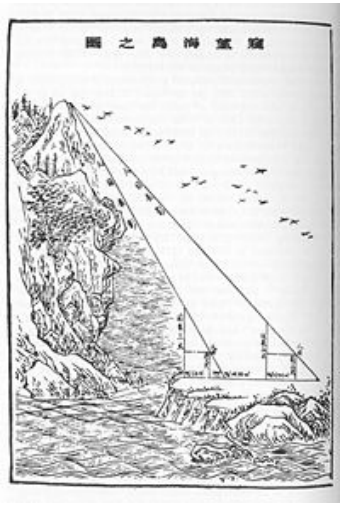
- 2D transformations between images
  - Translations, affine transformations, homographies…
- Fundamental matrices
  - Represent relationships between 2D images in the form of corresponding 2D lines
- **What's new:** Explicitly representing 3D geometry of cameras *and points*

Input



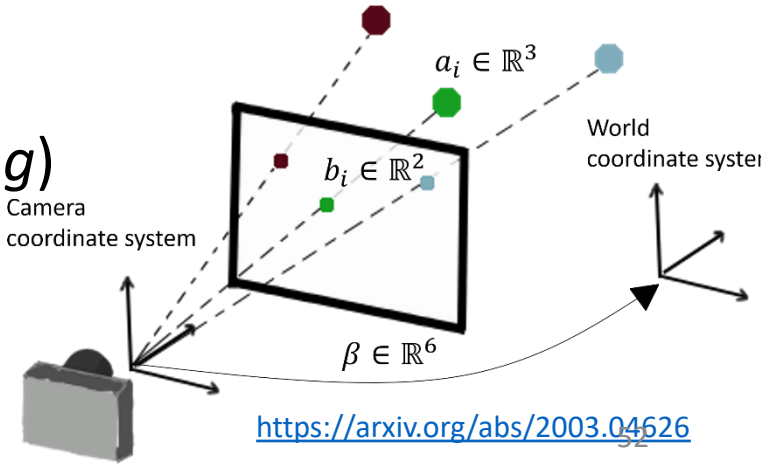
Triangulation & camera calibration

- Suppose we have known camera parameters, each of which observes a point
  - How can we compute the 3D location of that point?
  - This is called *triangulation* (known since ancient times)



Liu Hui (c. 263), How to measure the height of a sea island.

- On the other hand: Suppose we have known 3D points
  - And have matches between these points and an image
  - How can we compute the camera parameters?
  - This is called *camera calibration* (or *camera resectioning*)



https://arxiv.org/abs/2003.04626

Structure from motion

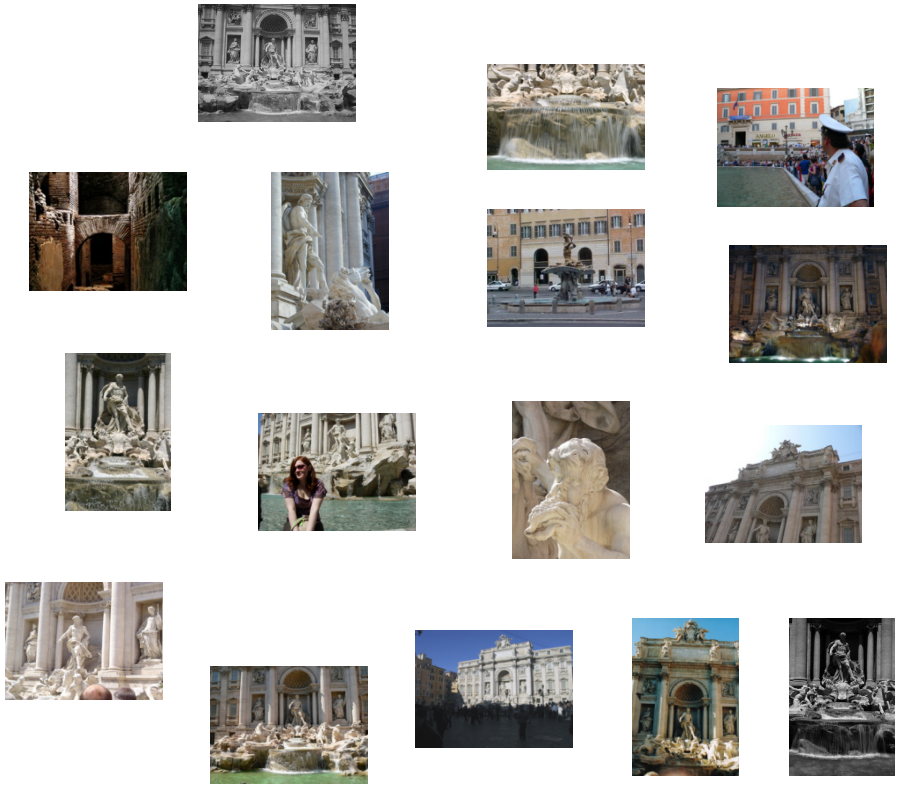
- What if we don’t know 3D points or camera parameters?
- SfM solves both of these problems *at once*
- A kind of chicken-and-egg problem
  - (but solvable)

First step: how to get correspondence?

- Feature detection and matching

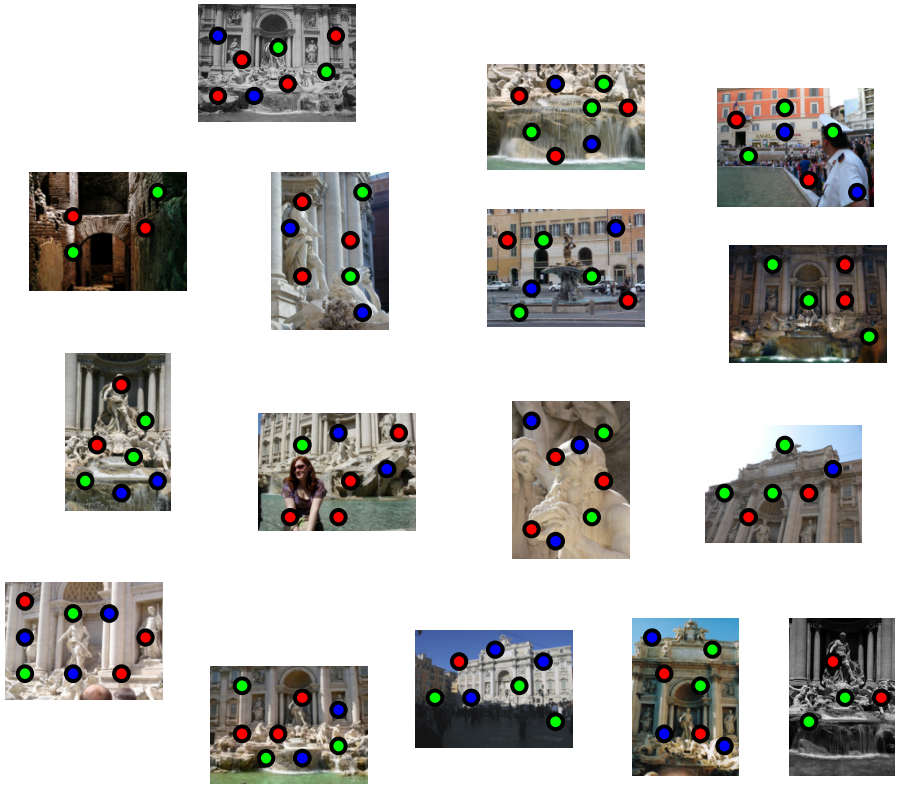
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



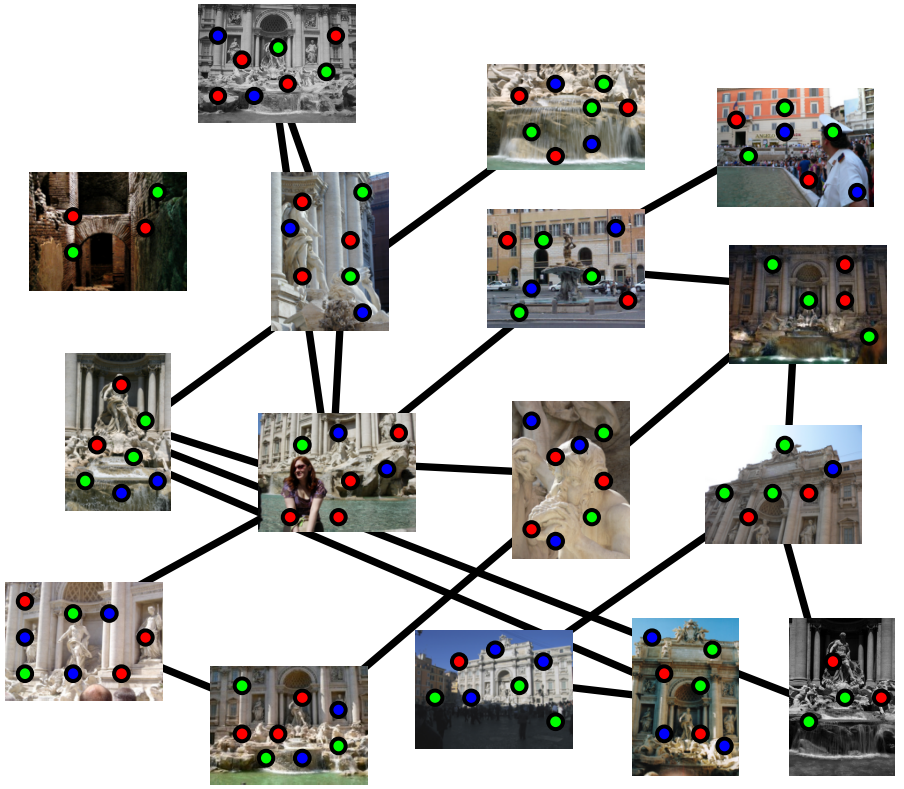
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



Feature matching

Match features between each pair of images



Feature matching

Refine matching using RANSAC to estimate fundamental matrix between each pair

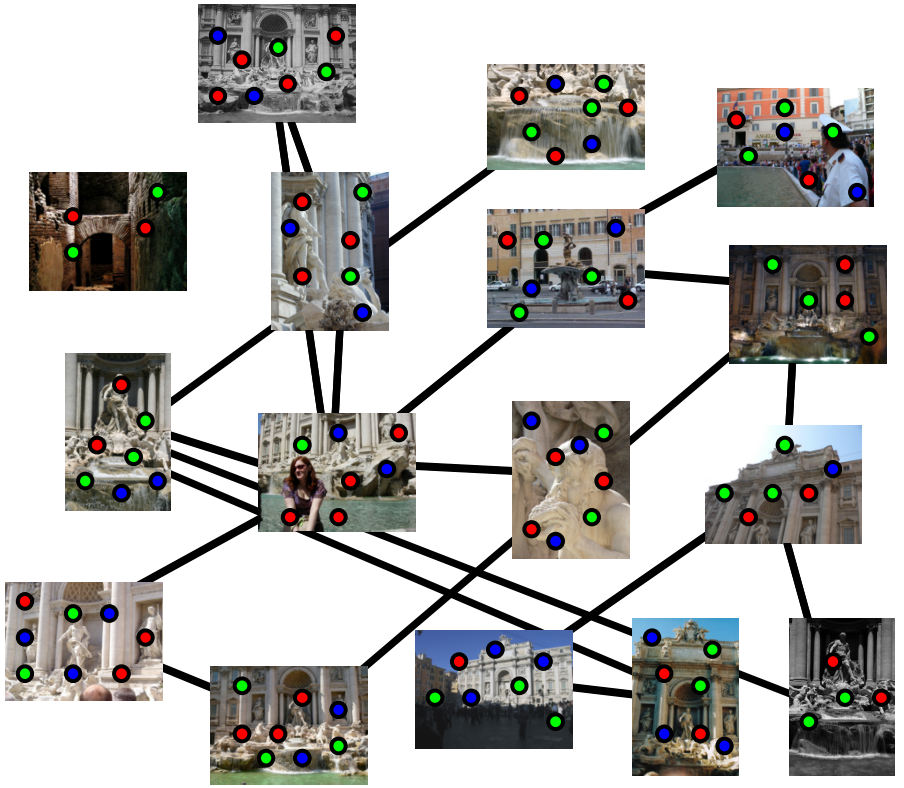
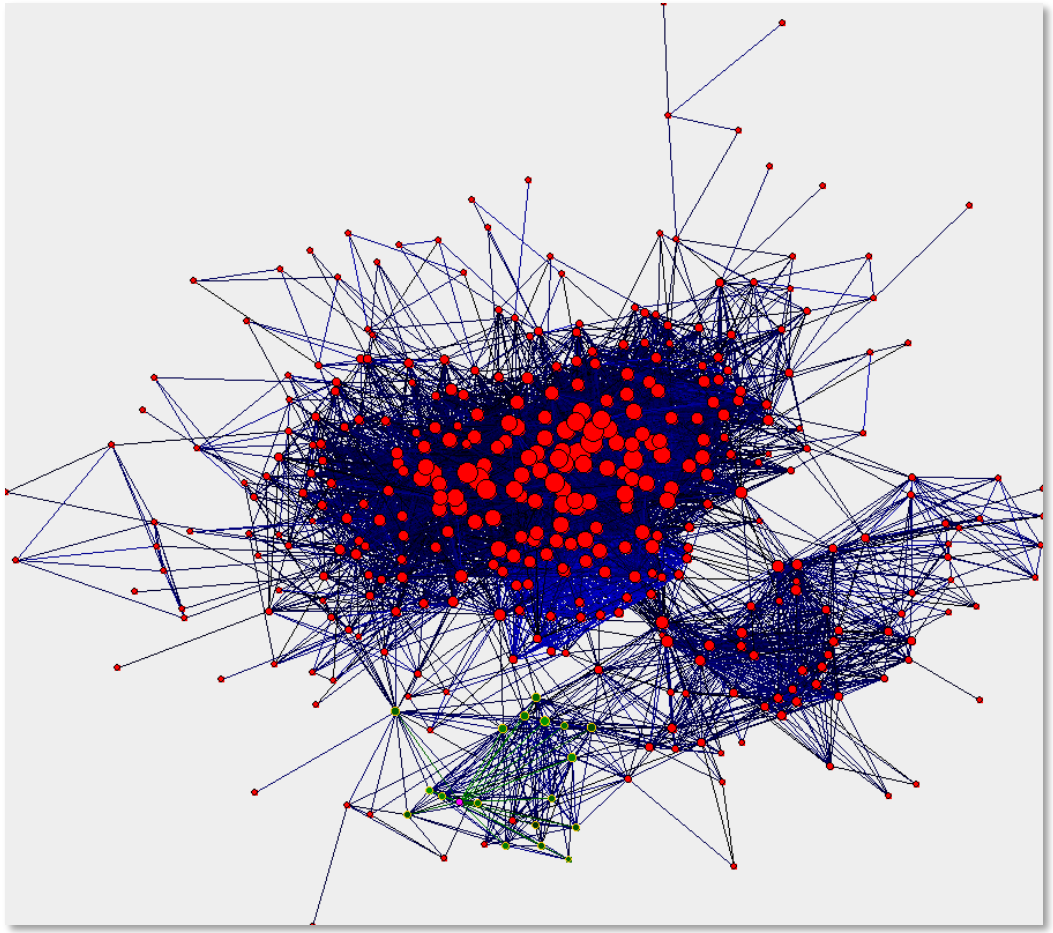


Image connectivity graph



(graph layout produced using the Graphviz toolkit: http://www.graphviz.org/)

Demo

Correspondence estimation

- Link up pairwise matches to form connected components of matches across several images

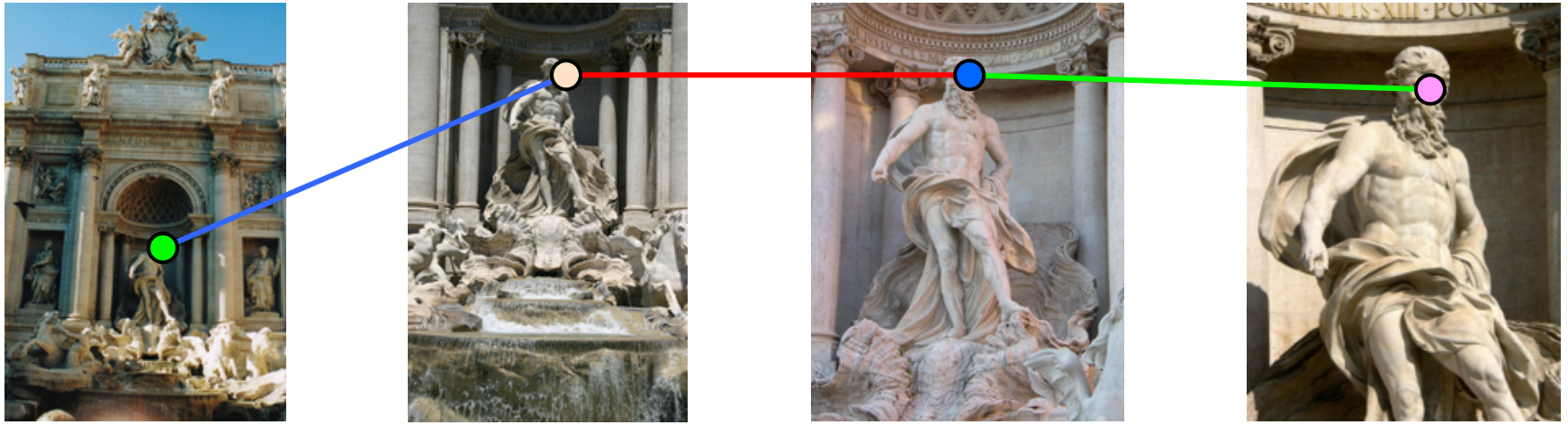
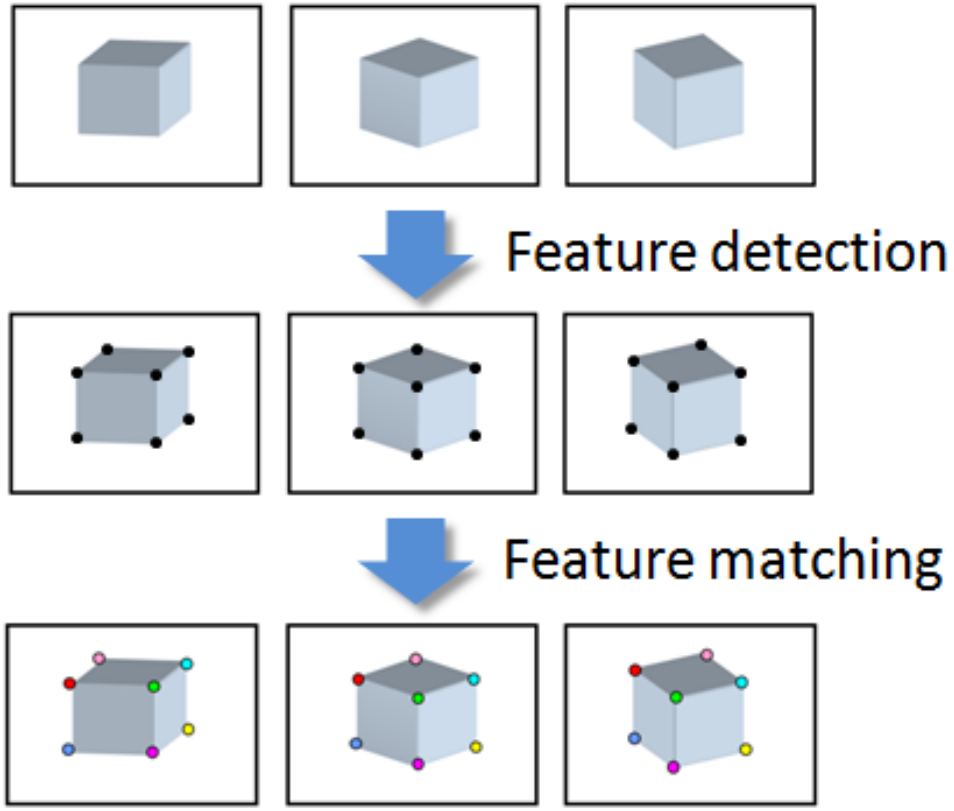
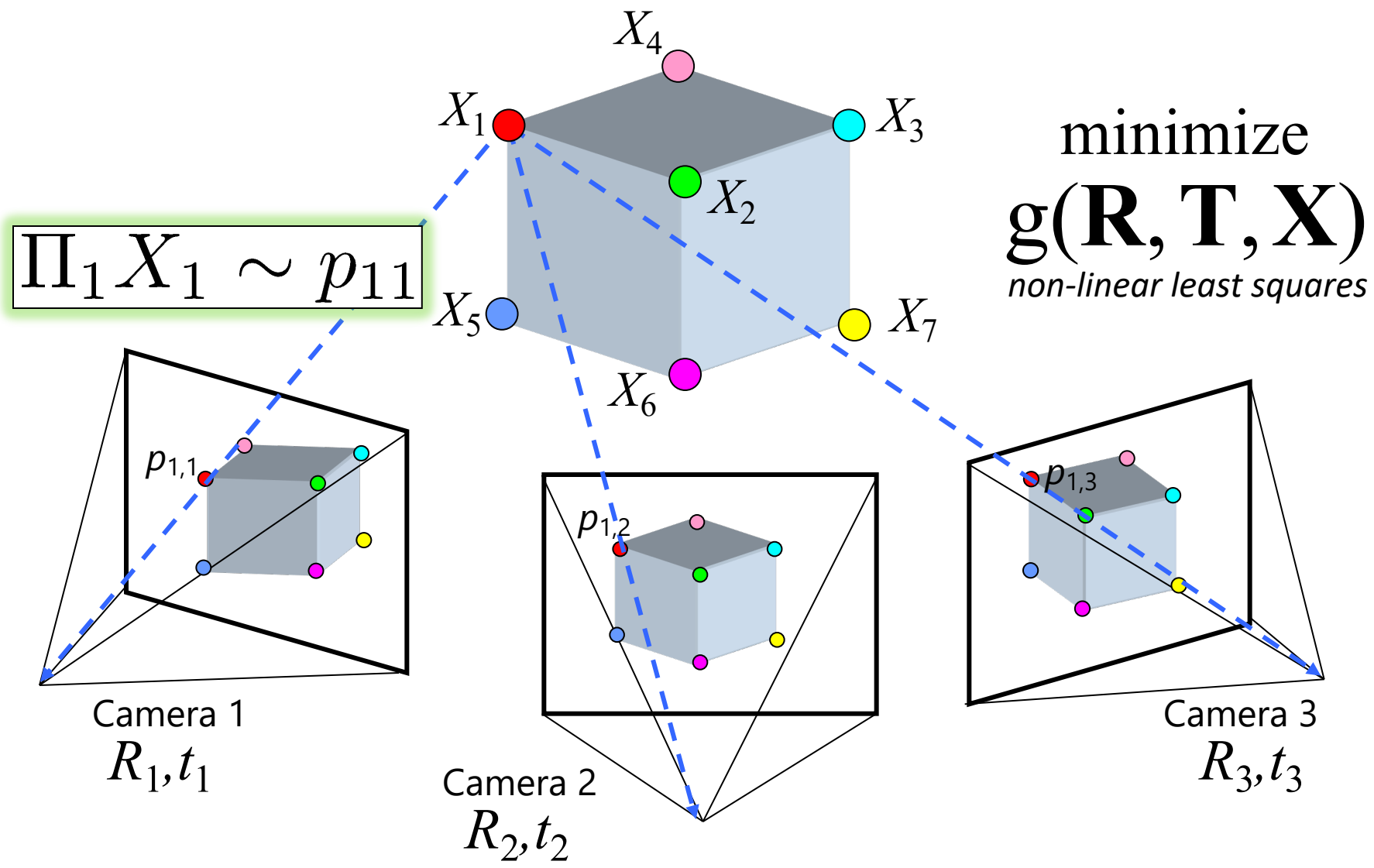


Image 1 Image 2 Image 3 Image 4

Input to Structure from Motion



Structure from motion



Problem size

- What are the variables?
  - Cameras and points
- How many variables per camera?
  - 6 (if calibrated), more if uncalibrated

Rotation matrix (3) and translation vector (3)

- How many variables per point?
  - 3 (X, Y, Z)

- Trevi Fountain collection

  466 input photos

  \+ > 100,000 3D points

  = very large optimization problem

Structure from motion

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2$$

predicted image location

observed image location

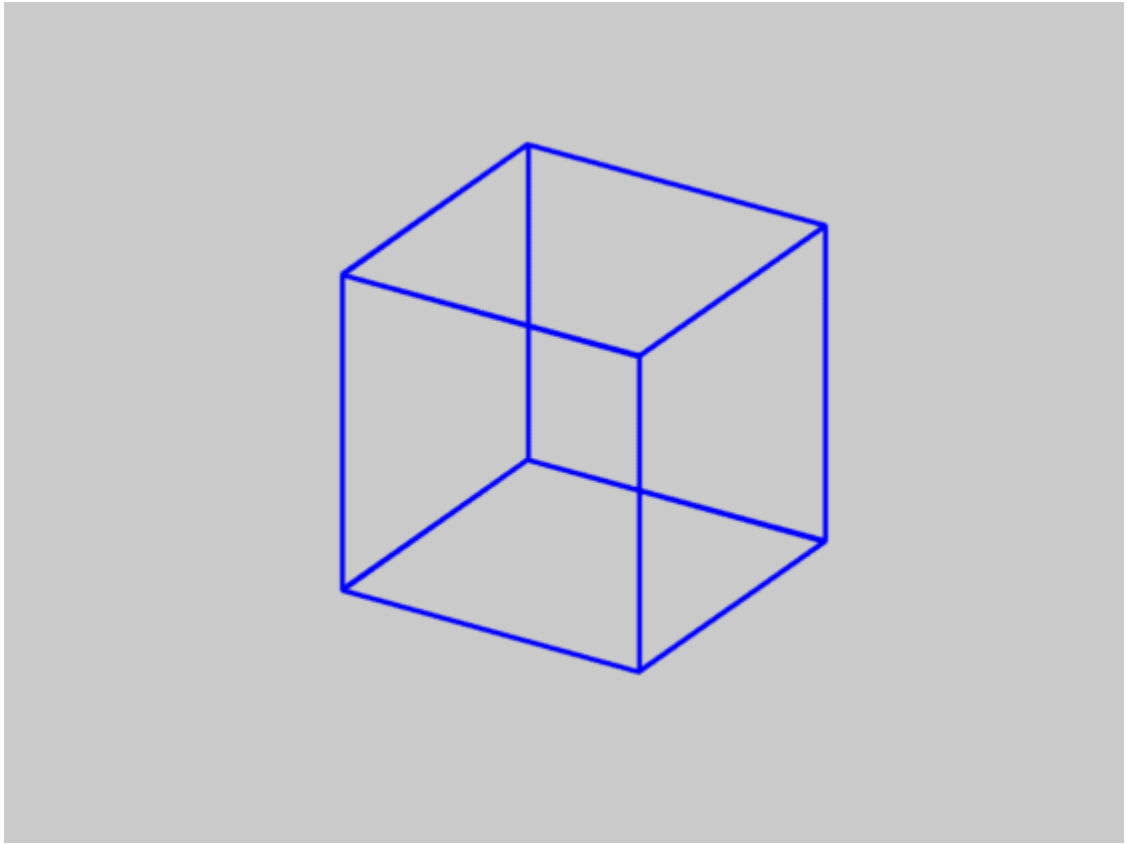
indicator variable: is point *i* visible in image *j* ?

- Minimizing this function is called *bundle adjustment*
  - Optimized using non-linear least squares, e.g. Levenberg-Marquardt algorithm

Is SfM always uniquely solvable?

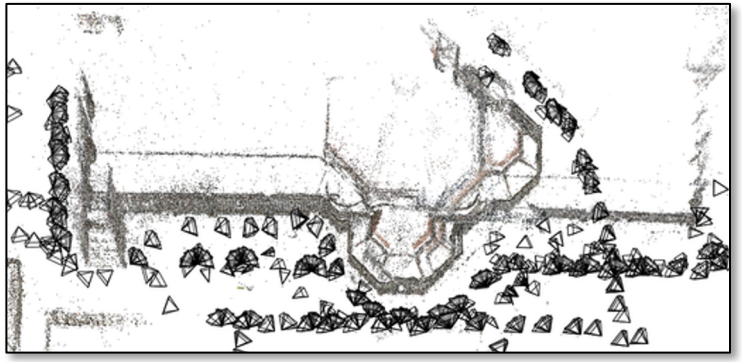
Is SfM always uniquely solvable?

- No…



Structure from Motion – Failure cases

- Repetitive structures: Symmetries in man-made scenes



SfM applications

- 3D modeling
- Surveying
- Robot navigation and mapmaking
- Virtual and augmented reality
- Visual effects (“Match moving”)
  - https://www.youtube.com/watch?v=RdYWp70P_kY

Applications – Hyperlapse

https://www.youtube.com/watch?v=SOpwHaQnRSY

https://www.youtube.com/watch?v=sA4Za3Hv6ng

Applications: Visual Reality & Augmented Reality

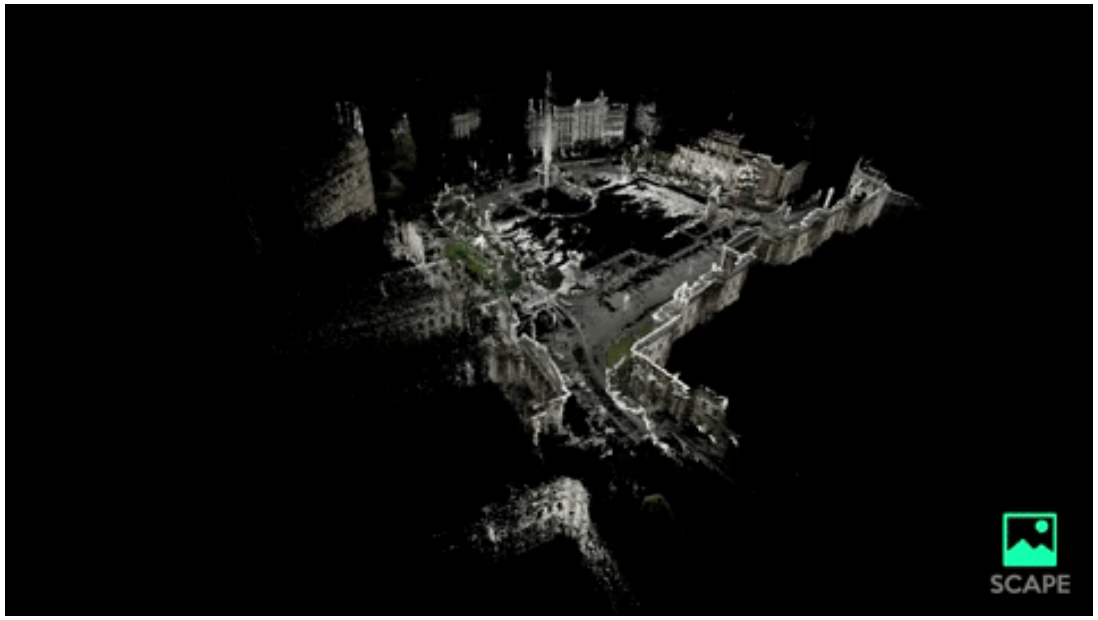
Oculus

https://www.youtube.com/watch?v=KOG7yTz1iTA

Hololens

https://www.youtube.com/watch?v=FMtvrTGnP04

Applications: Visual Reality & Augmented Reality



Scape: Building the ‘AR Cloud’: Part Three —3D Maps, the Digital Scaffolding of the 21st Century
https://medium.com/scape-technologies/building-the-ar-cloud-part-three-3d-maps-the-digital-scaffolding-of-the-21st-century-465fa55782dd

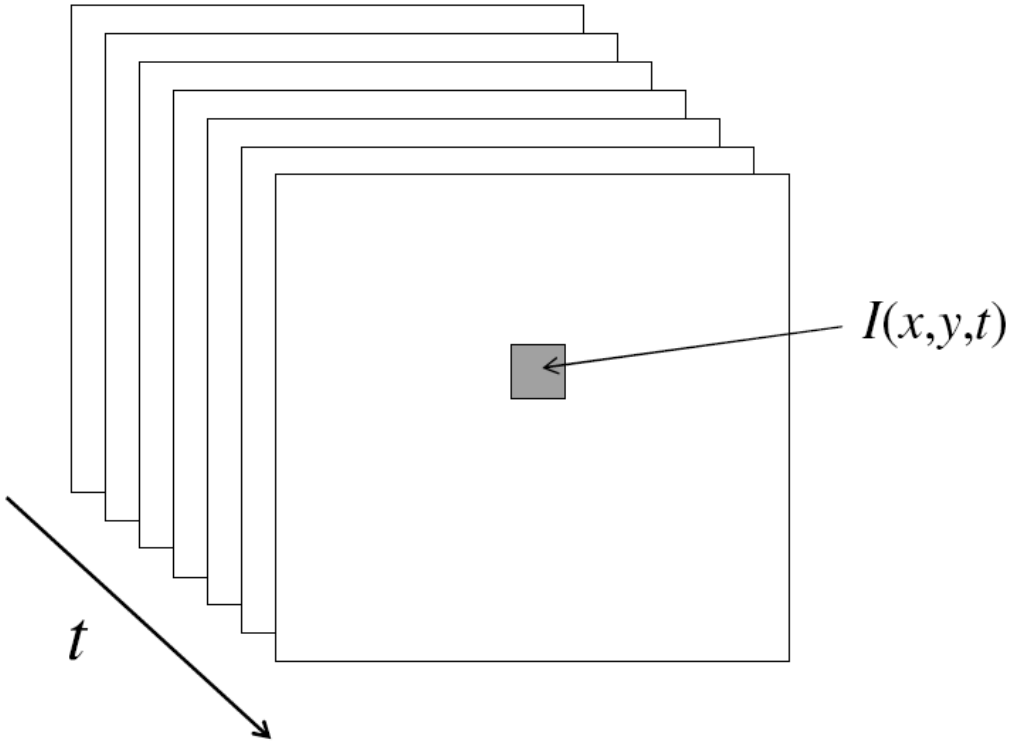
Application: AR walking directions

https://www.theverge.com/2019/8/8/20776247/google-maps-live-view-ar-walking-directions-ios-android-feature

Optical Flow

Video

- A video is a sequence of frames captured over time
- Now our image data is a function of space (x, y) and time (t)



Credit: K. Grauman

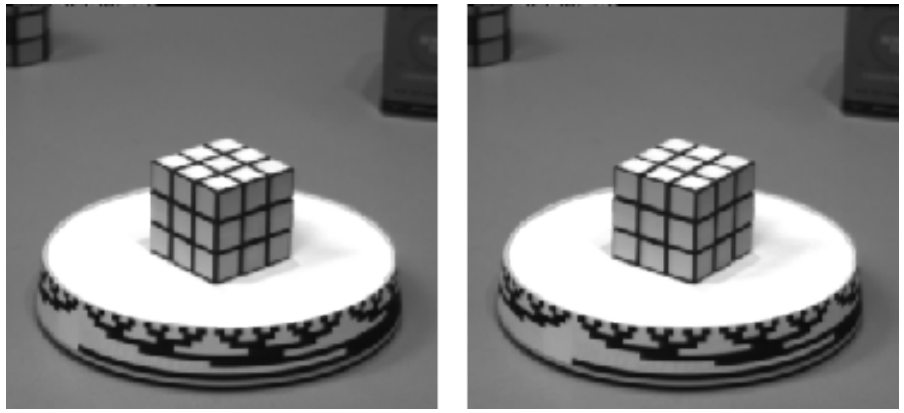
Uses of motion

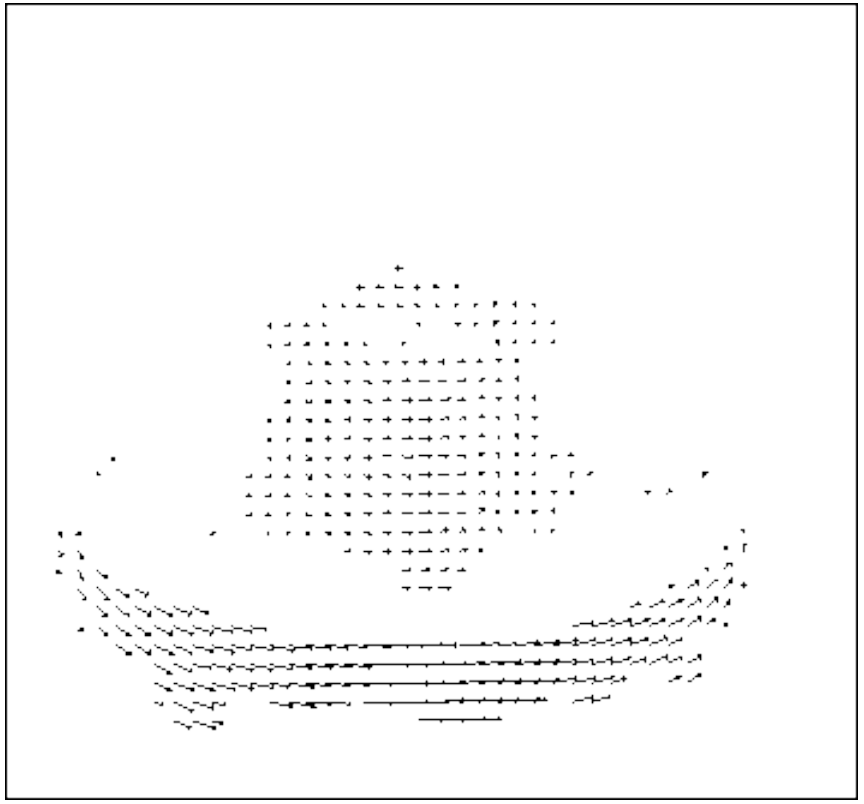
- Estimating 3D structure
- Segmenting objects based on motion cues
- Learning dynamical models
- Recognizing events and activities
- Improving video quality (motion stabilization)

Credit: K. Grauman

Motion field

- The motion field is the projection of the 3D scene motion into the image

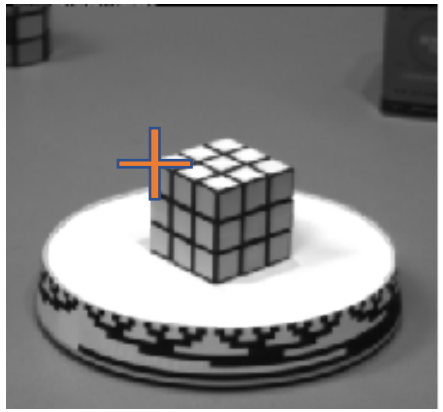




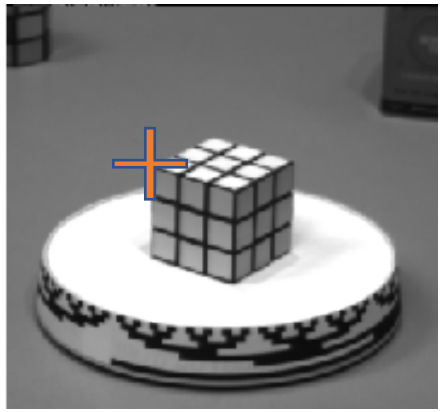
Credit: K. Grauman

Motion field

- The motion field is the projection of the 3D scene motion into the image

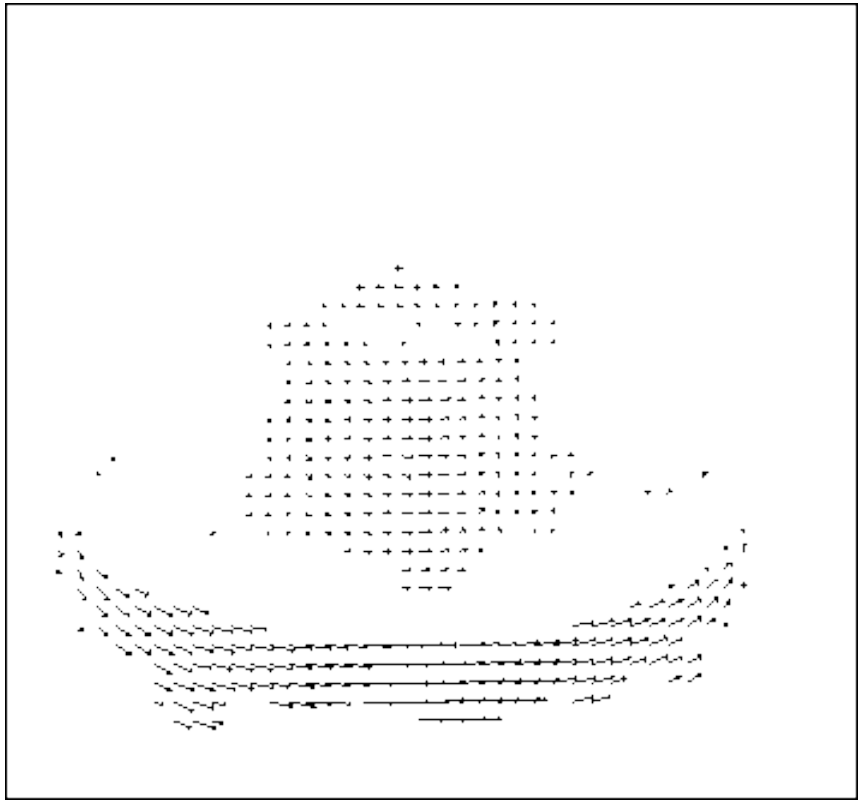


(100, 100)



(101, 102)

(u,v) = (1, 2)



Credit: K. Grauman

Motion field + camera motion

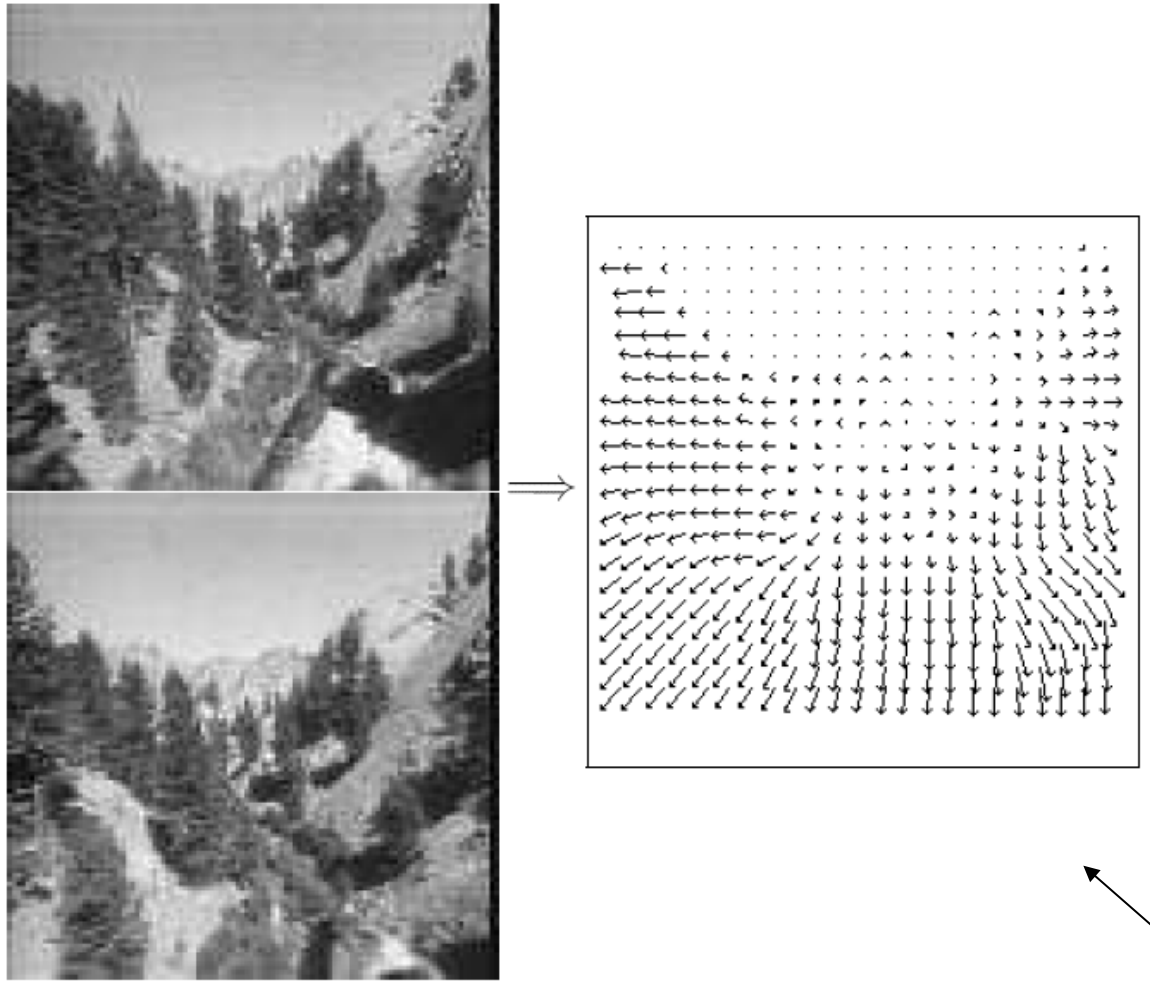


Figure 1.2: Two images taken from a helicopter flying through a canyon and the computed optical flow field.

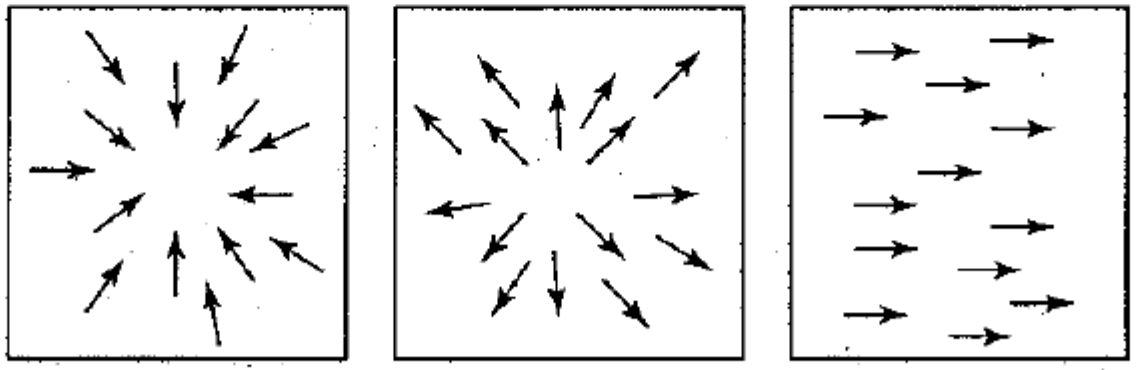
Length of flow vectors inversely proportional to depth Z of 3d point

points closer to the camera move more quickly across the image plane

Figure from Michael Black, Ph.D. Thesis

Credit: K. Grauman

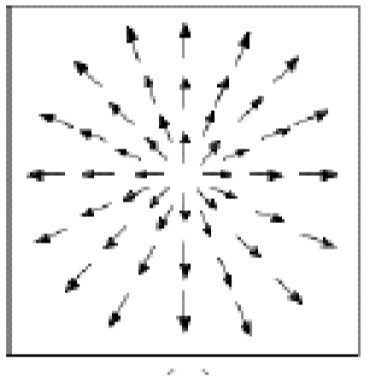
Motion field + camera motion



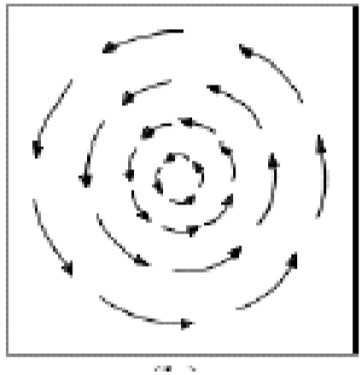
Zoom out **Zoom in** **Pan right to left**

Credit: K. Grauman

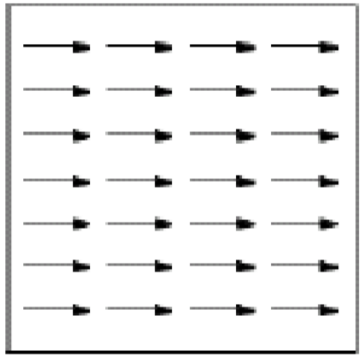
Some Simple Motion Fields



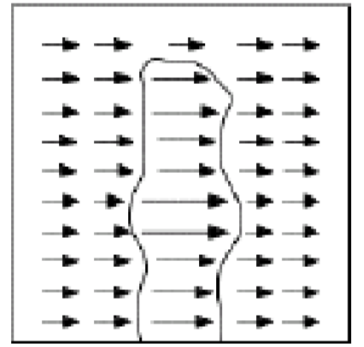
Forward motion



Rotation



Horizontal translation

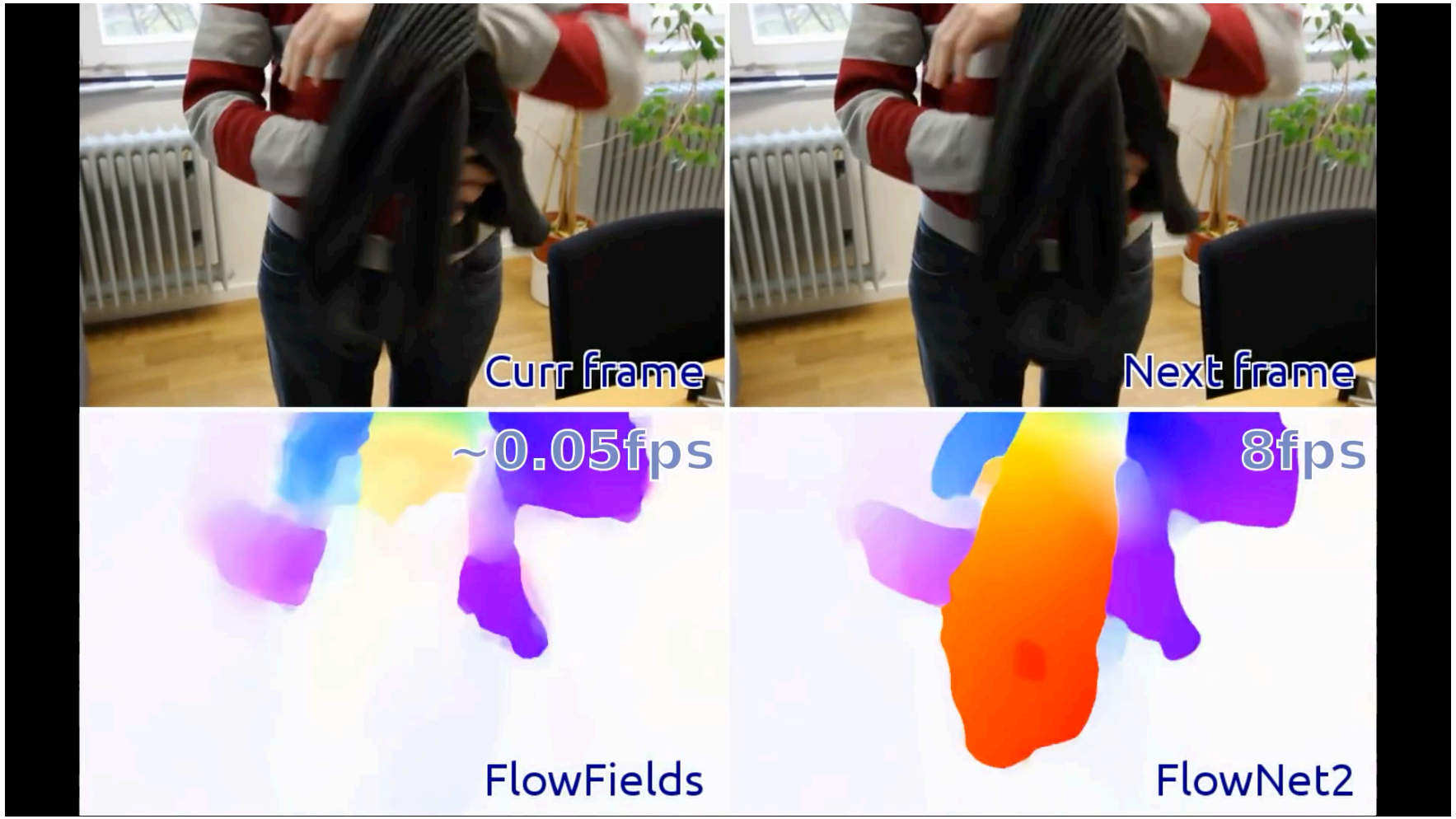


Closer objects appear to move faster!!

Optical flow

- Definition: optical flow is the *apparent* motion of brightness patterns in the image

Credit: K. Grauman



https://www.youtube.com/watch?v=JSzUdVBmQP4 FlowNet

Optical flow

- Definition: optical flow is the *apparent* motion of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field
- Have to be careful: apparent motion can be caused by lighting changes without any actual motion

Credit: K. Grauman

Apparent motion != motion field

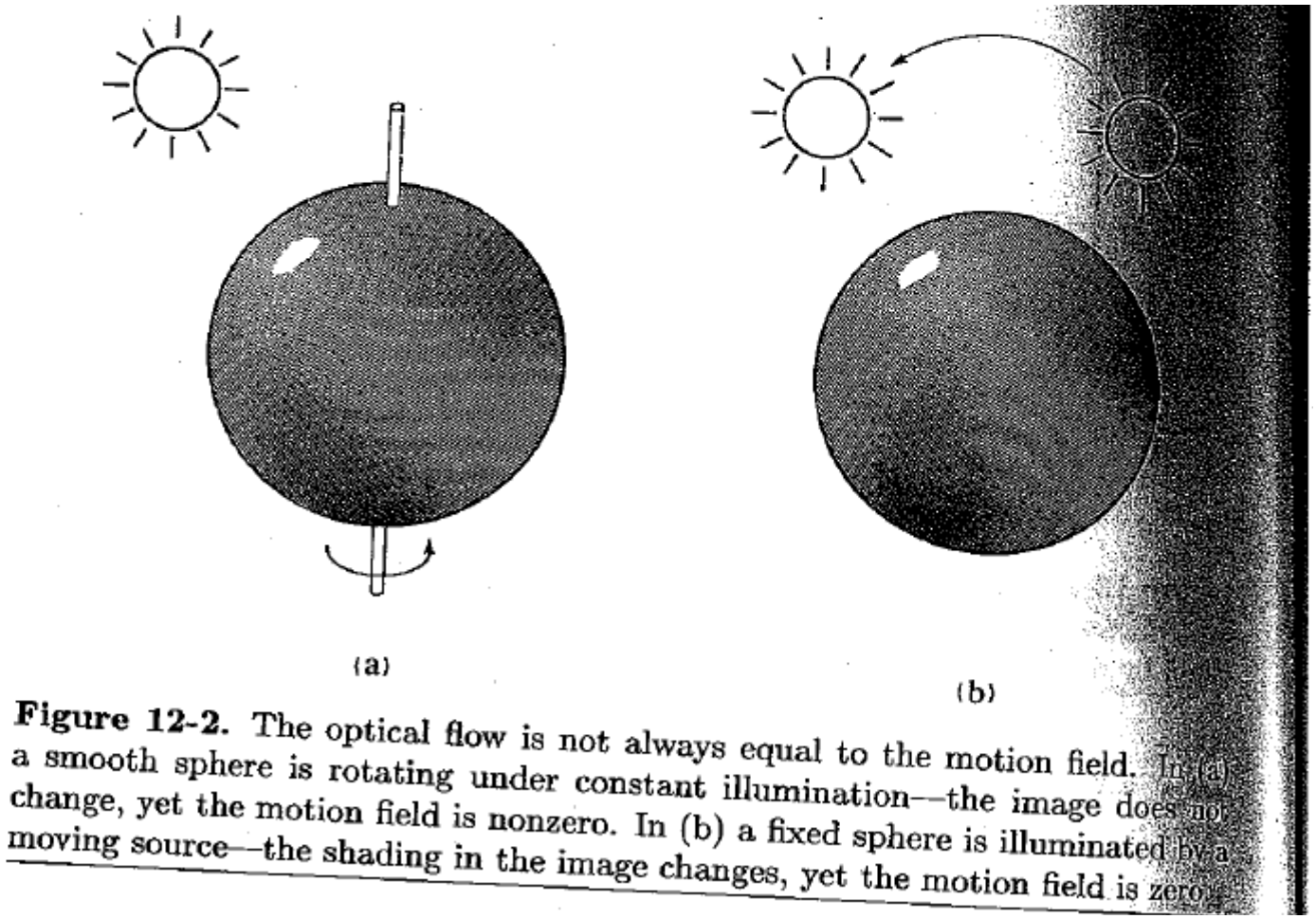
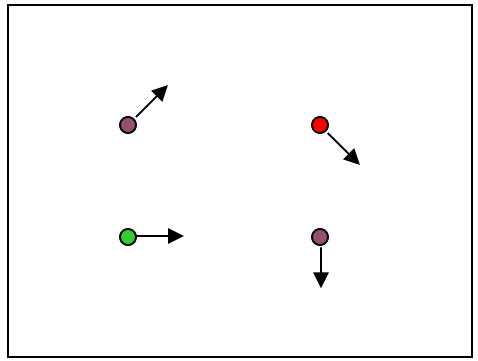


Figure 12-2. The optical flow is not always equal to the motion field. In (a) a smooth sphere is rotating under constant illumination—the image does not change, yet the motion field is nonzero. In (b) a fixed sphere is illuminated by a moving source—the shading in the image changes, yet the motion field is zero.

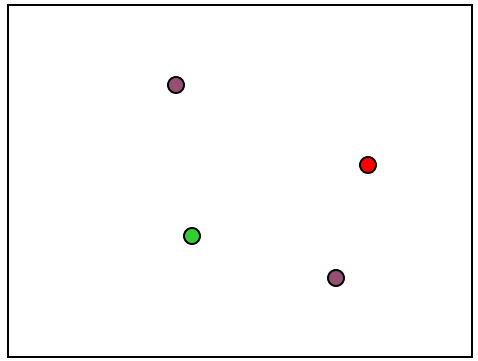
Figure from Horn book

Credit: K. Grauman

Problem definition: optical flow



$H(x, y)$



$I(x, y)$

- How to estimate pixel motion from image H to image I?
  - Solve pixel correspondence problem
    - given a pixel in H, look for nearby pixels of the same color in I

Key assumptions

- **color constancy:** a point in H looks the same in I
  - For grayscale images, this is **brightness constancy**
- **small motion**: points do not move very far

This is called the **optical flow** problem

Credit: K. Grauman

Color/brightness constancy

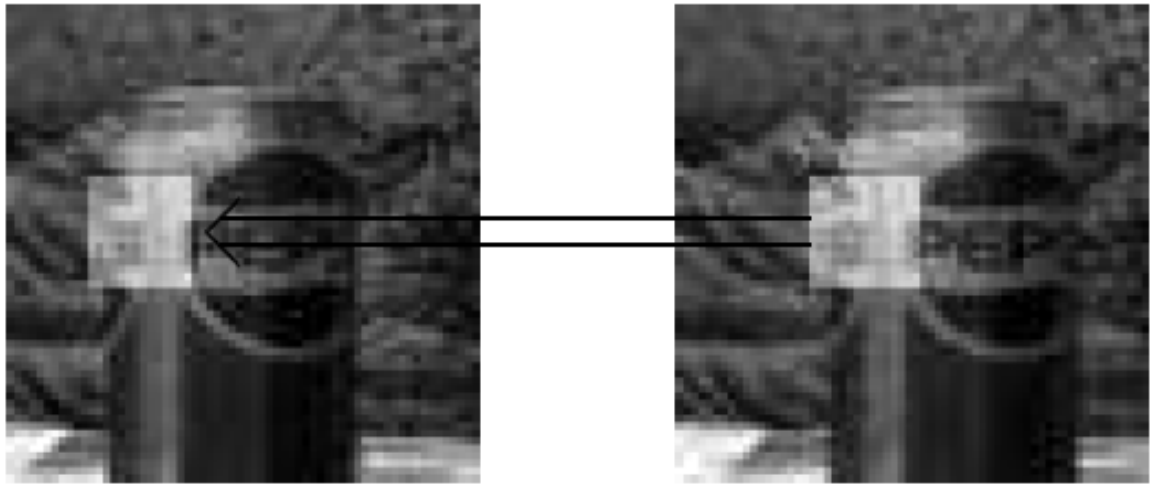
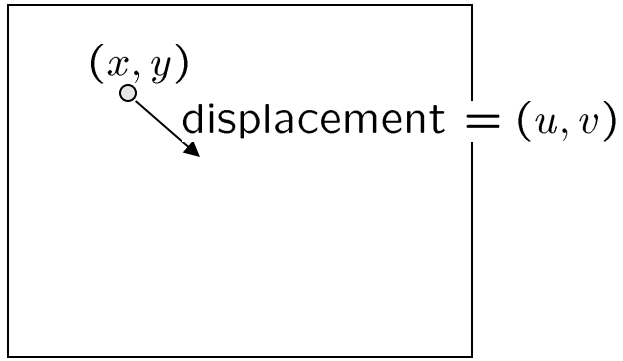


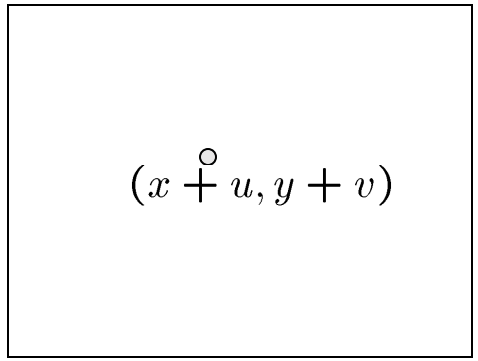
Figure 1.5: Data conservation assumption. The highlighted region in the right image looks roughly the same as the region in the left image, despite the fact that it has moved.

Figure by Michael Black

Optical flow constraints



$H(x, y)$



$I(x, y)$

- Let's look at these constraints more closely
  - brightness constancy: Q: what's the equation?

$$H(x, y) = I(x + u, y + v)$$

  - small motion:

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

$$\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

Optical flow equation

shorthand: $I_x = \frac{\partial I}{\partial x}$

- Combining these two equations

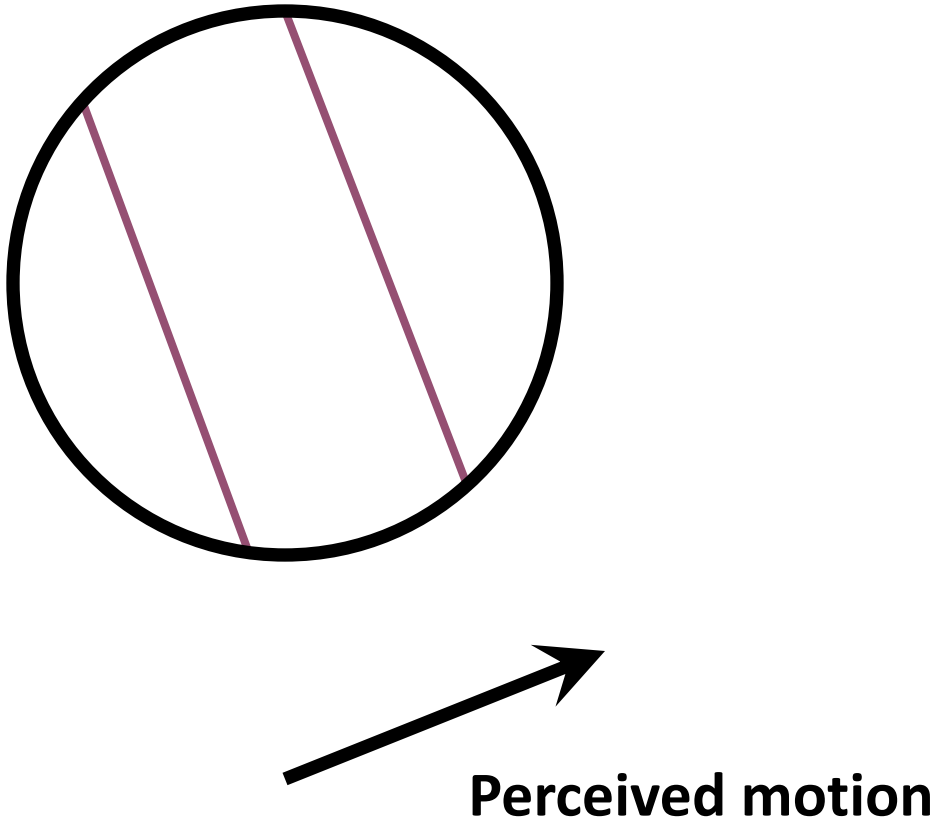
$$
\begin{aligned}
0 &= I(x+u, y+v) - H(x,y) \\
&\approx I(x,y) + I_x u + I_y v - H(x,y) \\
&\approx (I(x,y) - H(x,y)) + I_x u + I_y v \\
&\approx I_t + I_x u + I_y v \\
&\approx I_t + \nabla I \cdot [u \ v]
\end{aligned}
$$

Optical flow equation

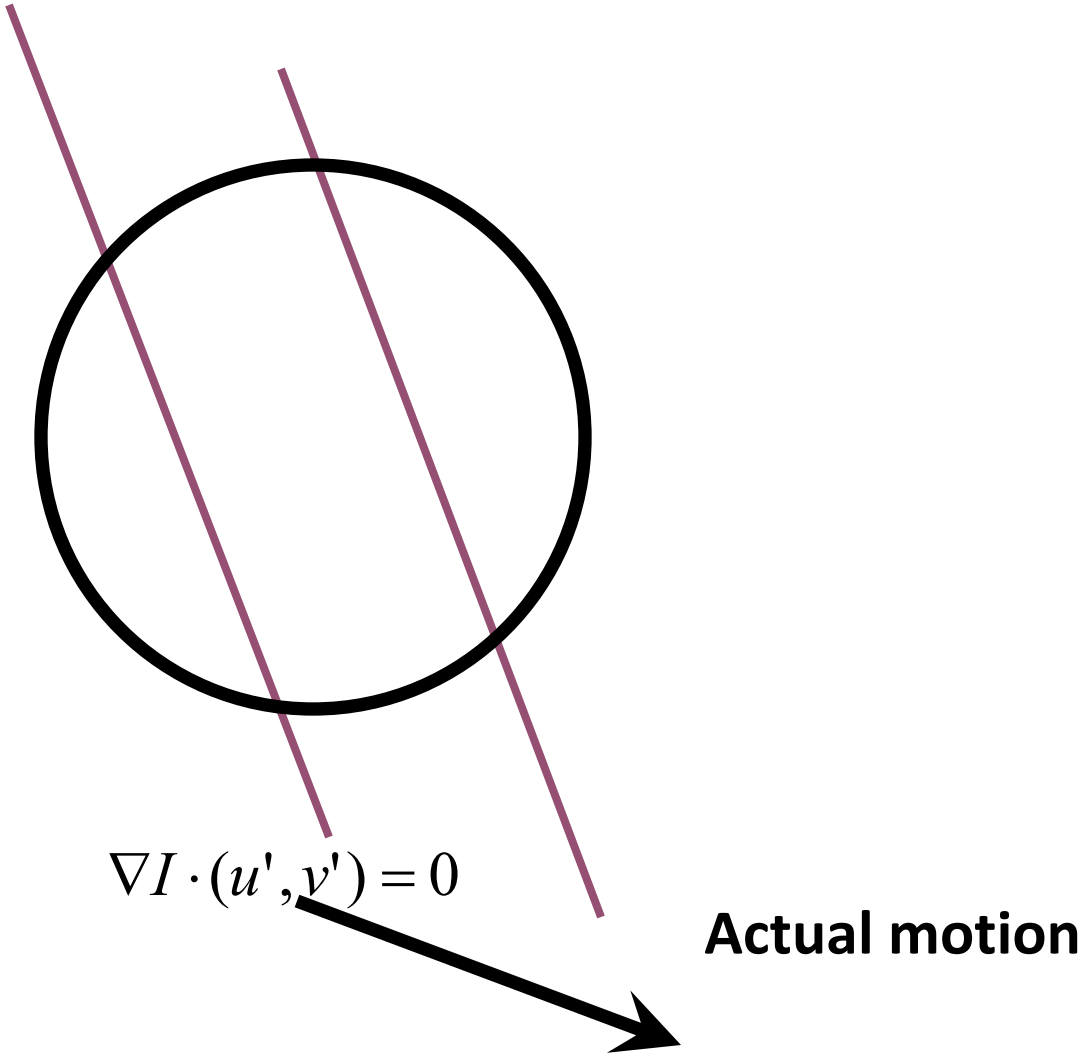
$$0 = I_t + \nabla I \cdot [u \ v]$$

- Q: how many unknowns and equations per pixel?

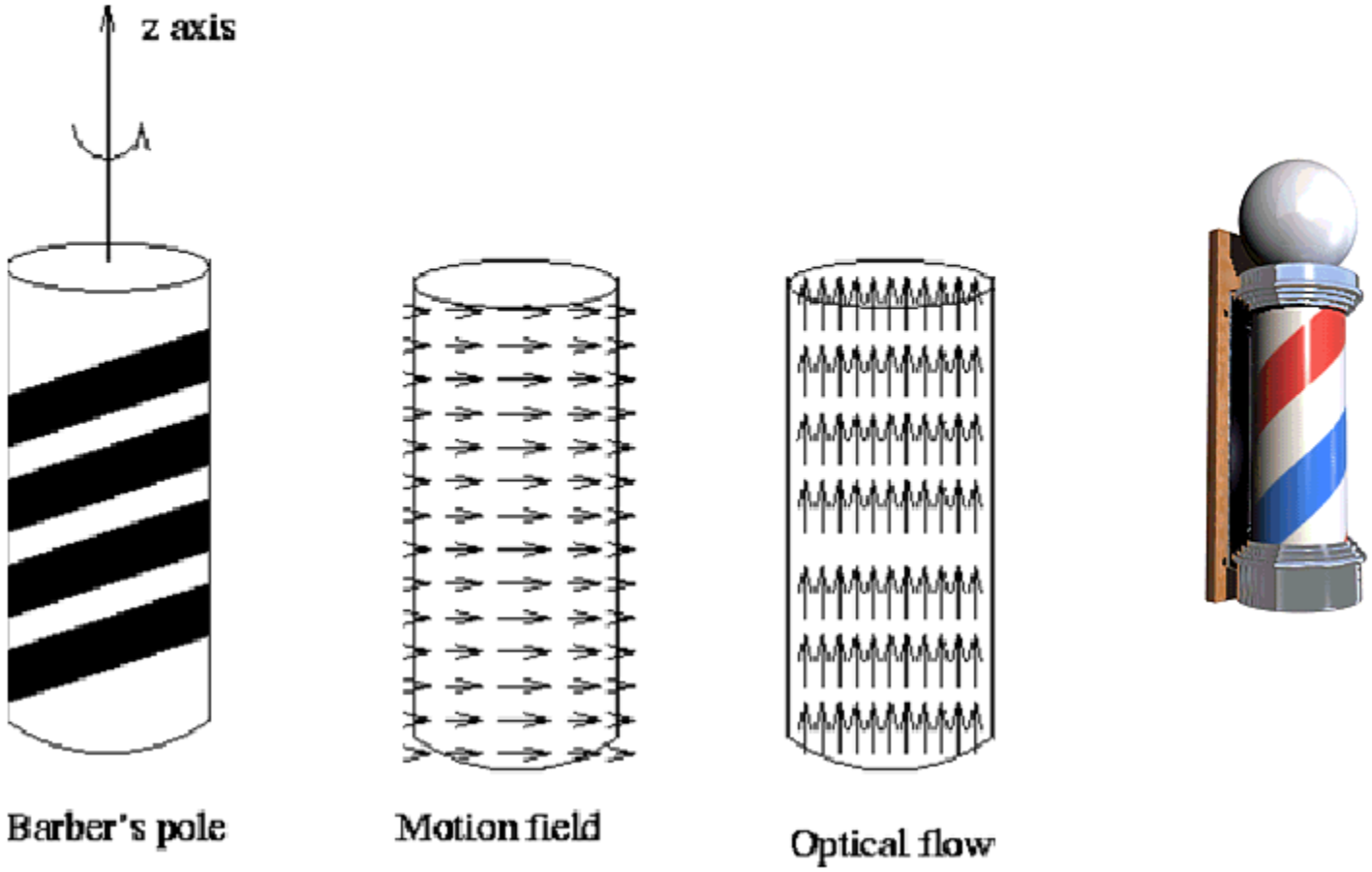
The aperture problem



The aperture problem



The barber pole illusion



http://en.wikipedia.org/wiki/Barberpole_illusion

Solving the aperture problem

- How to get more equations for a pixel?
- **Spatial coherence constraint:** pretend the pixel's neighbors have the same (u,v)

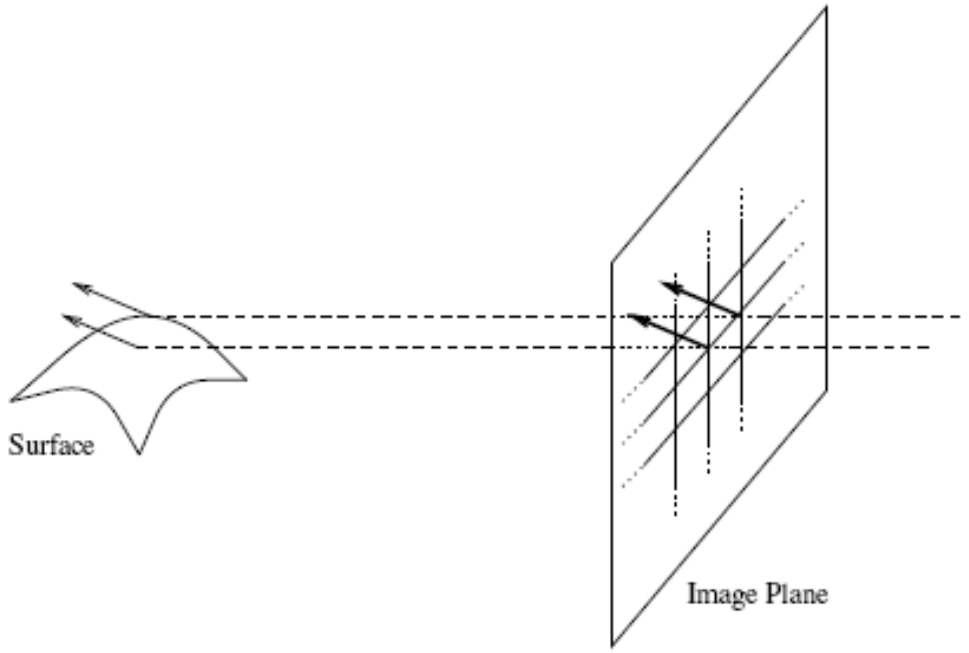


Figure 1.7: Spatial coherence assumption. Neighboring points in the image are assumed to belong to the same surface in the scene.

Figure by Michael Black

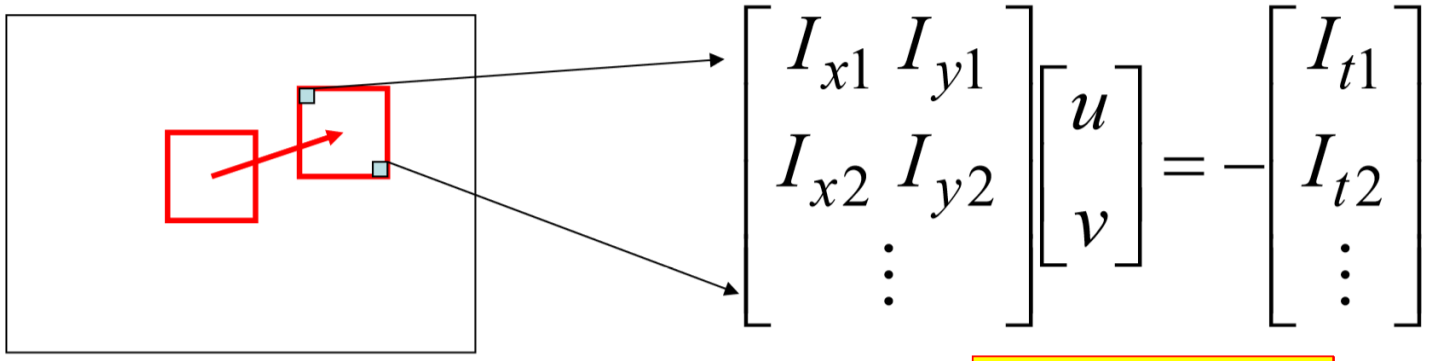
Solving the aperture problem

- How to get more equations for a pixel?
- **Spatial coherence constraint:** pretend the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel $0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \ v]$

Local Patch Constant Motion: Lucas–Kanade optical flow algorithm

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

Assume constant motion (u,v) in small neighborhood



$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \end{bmatrix}$$

$$A\vec{u} = b$$

5x5的矩阵（Ix1 Iy1 … Ix5 Iy5），
视为共用相同的变换 (u, v)。

Lucas–Kanade Optical Flow Algorithm

Goal: Minimize $\left\| A\vec{\mathrm{u}} - b \right\|^2$

Method: Least-Squares

$$A\vec{\mathrm{u}} = b$$

$$\Downarrow$$

$$\underbrace{A^T A}_{2\mathrm{x}2}\,\underbrace{\vec{\mathrm{u}}}_{2\mathrm{x}1} = \underbrace{A^T b}_{2\mathrm{x}1}$$

$$\Downarrow$$

$$\vec{\mathrm{u}} = \left(A^T A\right)^{-1} A^T b$$

Solving the aperture problem

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

- The summations are over all pixels in the K x K window
- This technique was first proposed by Lucas & Kanade (1981)

Slide credit: Steve Seitz

Conditions for solvability

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

$A^T A$ $\qquad$ $A^T b$

When is this solvable?

- $\mathbf{A^T A}$ should be invertible
- $\mathbf{A^T A}$ should not be very small
  - eigenvalues λ_1 and λ_2 of $\mathbf{A^T A}$ should not be very small
- $\mathbf{A^T A}$ should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Slide by Steve Seitz, UW

Edge

– gradients very large or very small

– large λ_1, small λ_2

Low-texture region

– gradients have small magnitude

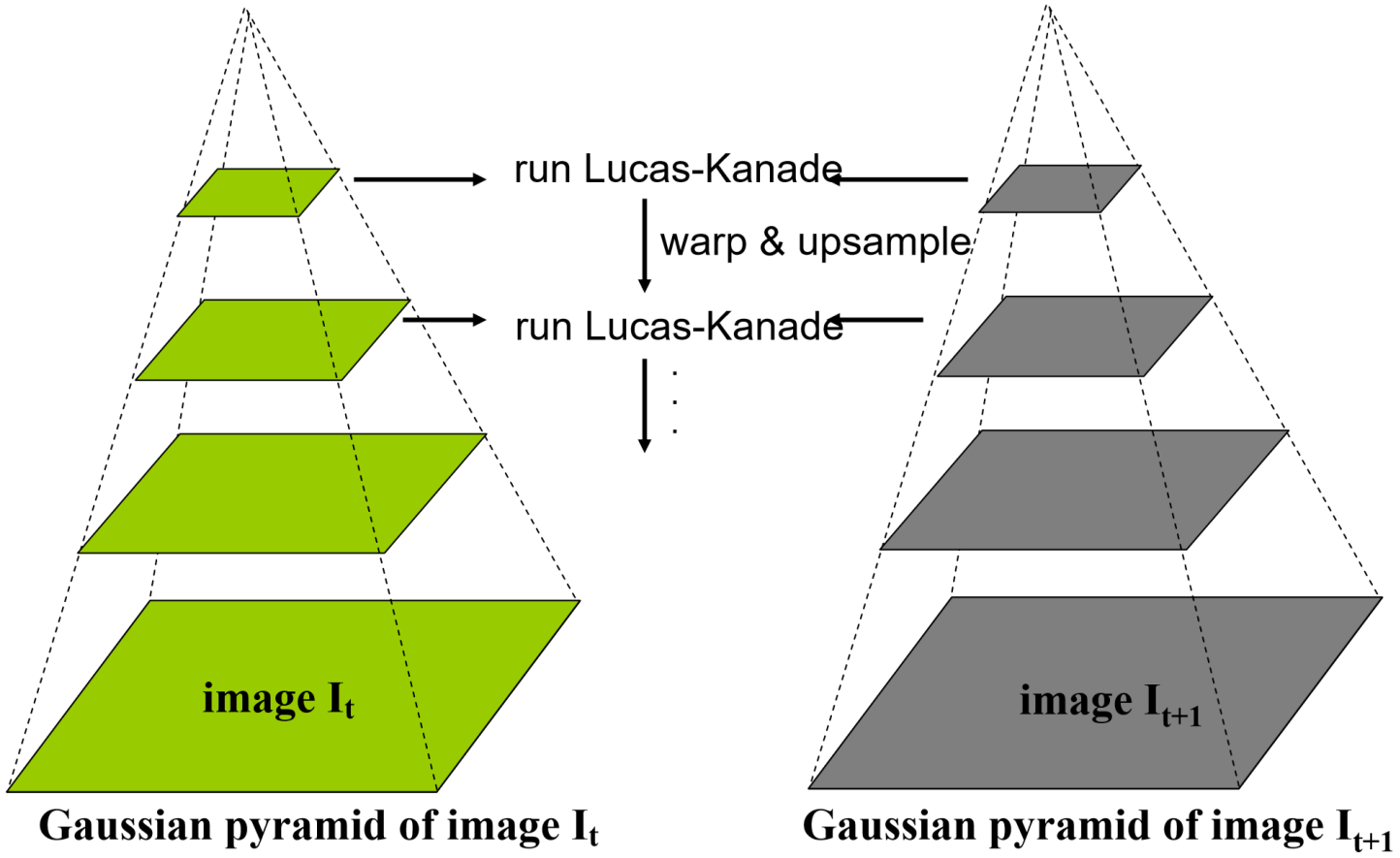
– small λ_1, small λ_2

High-texture region

– gradients are different, large magnitudes

– large λ_1, large λ_2

Multi-Scale Pyramid Estimation



Horn–Schunck Optical Flow Algorithm (1981)

- Regularisation: use a global smoothness term

Smoothness error: $$E_s = \iint_D \left(u_x^2 + u_y^2\right) + \left(v_x^2 + v_y^2\right) dx\, dy$$

Error in brightness constancy equation: $$E_c = \iint_D \left(I_x u + I_y v + I_t\right)^2 dx\, dy$$

Minimize: $E_c + \lambda E_s$

Solve by calculus of variations

Variational Minimization

The Euler-Lagrange equations :

$$F_u - \frac{\partial}{\partial x} F_{u_x} - \frac{\partial}{\partial y} F_{u_y} = 0$$

$$F_v - \frac{\partial}{\partial x} F_{v_x} - \frac{\partial}{\partial y} F_{v_y} = 0$$

In our case ,

$$F = (u_x^2 + u_y^2) + (v_x^2 + v_y^2) + \lambda (I_x u + I_y v + I_t)^2,$$

so the Euler-Lagrange equations are

$$\Delta u = \lambda (I_x u + I_y v + I_t) I_x,$$

$$\Delta v = \lambda (I_x u + I_y v + I_t) I_y,$$

$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplacian operator

Horn–Schunck Optical Flow Algorithm

1. Coupled PDEs solved using iterative methods and finite differences

$$\frac{\partial u}{\partial t} = \Delta u - \lambda(I_x u + I_y v + I_t)I_x,$$

$$\frac{\partial v}{\partial t} = \Delta v - \lambda(I_x u + I_y v + I_t)I_y,$$

2. More than two frames allow a better estimation of I_t

3. Information spreads from corner-type patterns

Optical Flow: Summary, readings

- Optical flow definition

- Lucas–Kanade optical flow algorithm (KLT tracker)

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

- Horn–Schunck optical flow algorithm

B.K.P. Horn and B.G. Schunck, "Determining optical flow." *Artificial Intelligence, 1981*

- Computer Vision: Algorithms and Applications (2nd edition), Chapter 9.3

Next week

- Shape from Shading
- Photometric Stereo