

# COMP417

## Introduction to Robotics and Intelligent Systems

### Lecture 21: Visual SLAM and Visual Odometry

Florian Shkurti

Computer Science Ph.D. student

[florian@cim.mcgill.ca](mailto:florian@cim.mcgill.ca)

- **Visual SLAM**

- Localization and mapping with measurements usually coming from tracking image features:
  - keypoints/corners
  - edges
  - image intensity patches
- Can use one or more cameras

- **Visual Odometry**

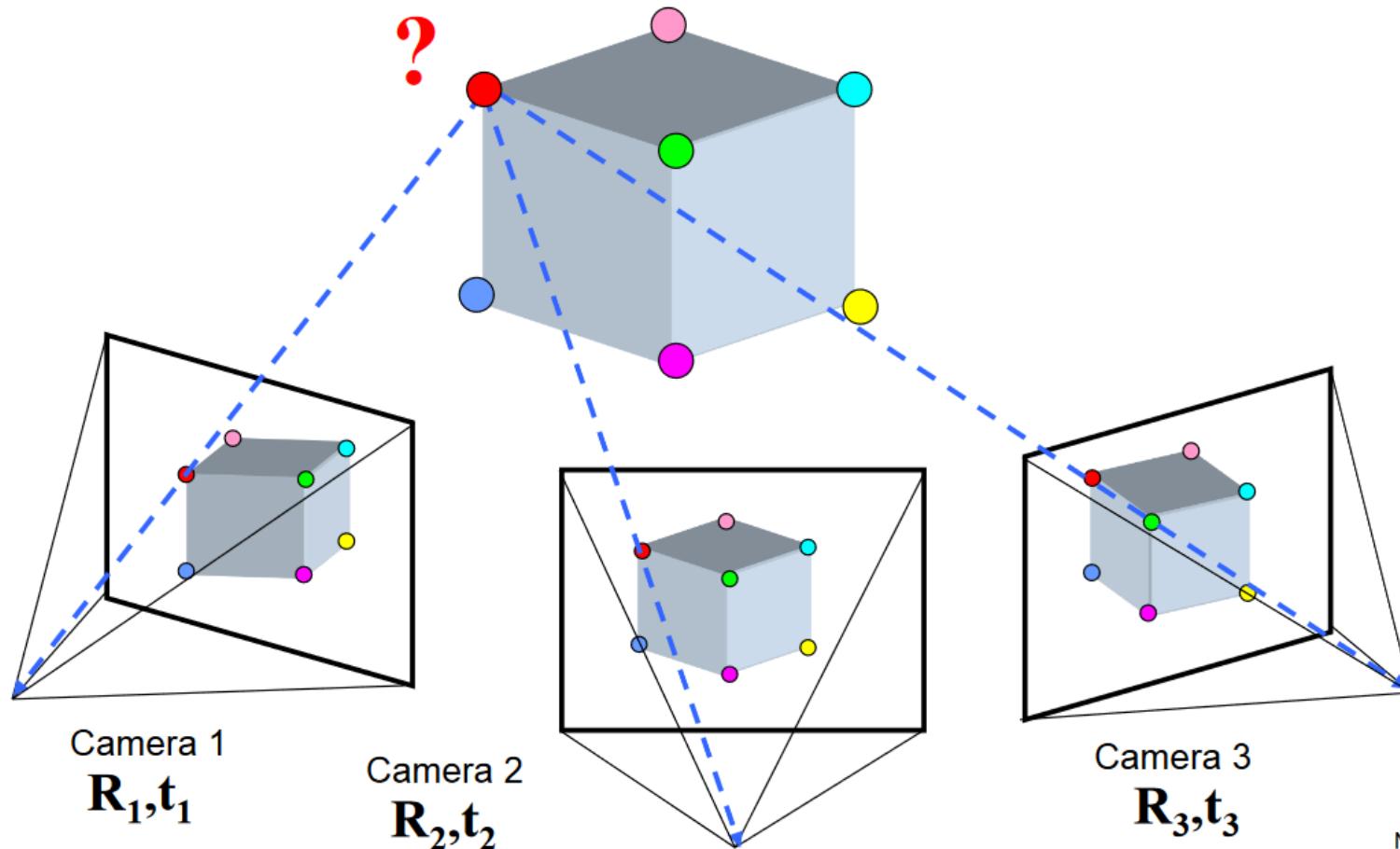
- Real-time localization with measurements usually coming from tracking image features:
  - keypoints/corners
  - edges
  - image intensity patches
- Can use one or more cameras

# Multi-view geometry problems

A.k.a. mapping

- **Structure:** Given projections of the same 3D point in two or more images, compute the 3D coordinates of that point

3D point coordinates  
are unknown and to  
be estimated



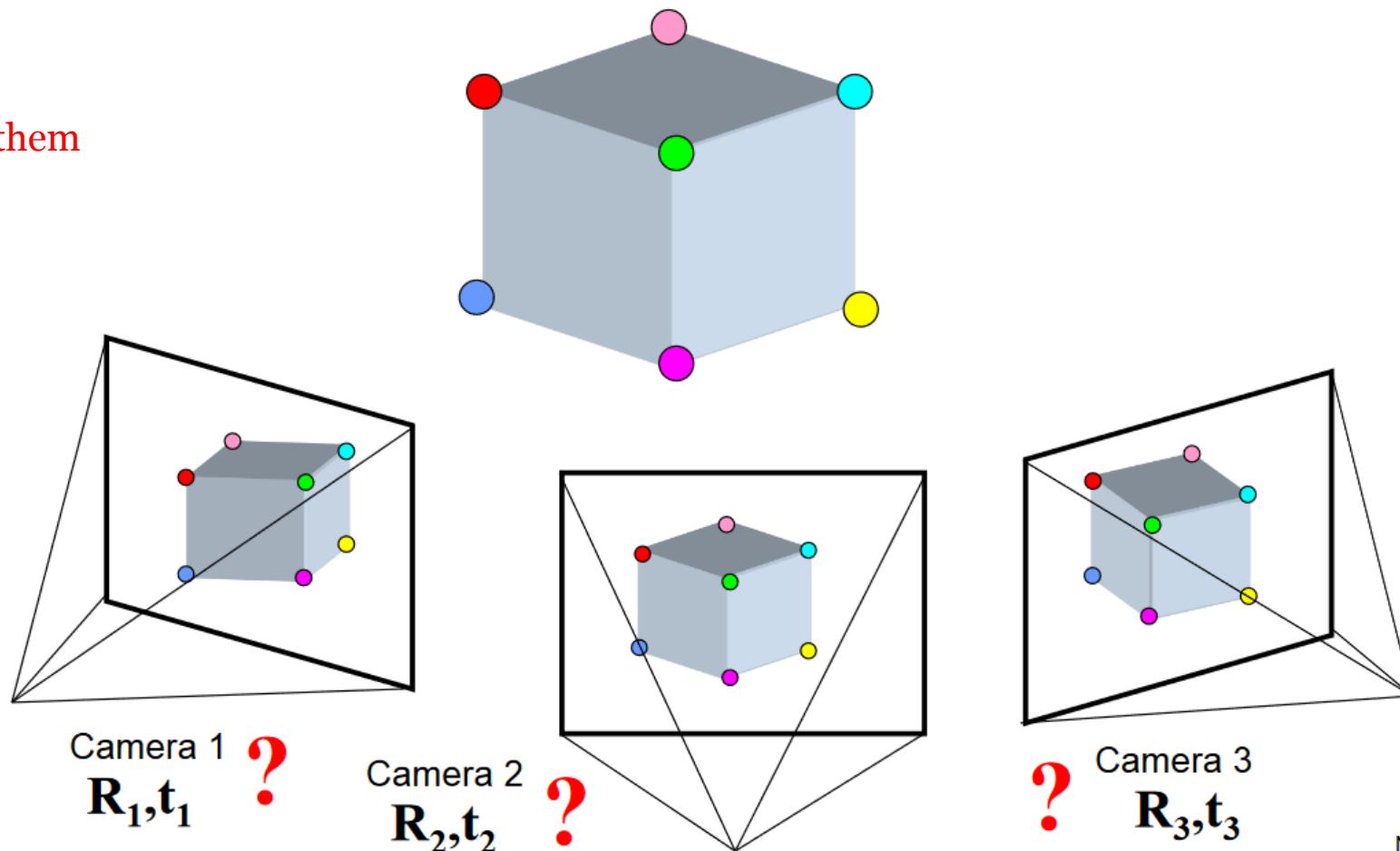
Camera frame  
transformations  
are known

Slide credit:  
Noah Snavely

# Multi-view geometry problems

- **Motion:** Given a set of corresponding points in two or more images, compute the camera parameters

3D point coordinates  
are unknown, but we  
won't try to estimate them



Camera frame  
transformations  
are unknown and to  
be estimated

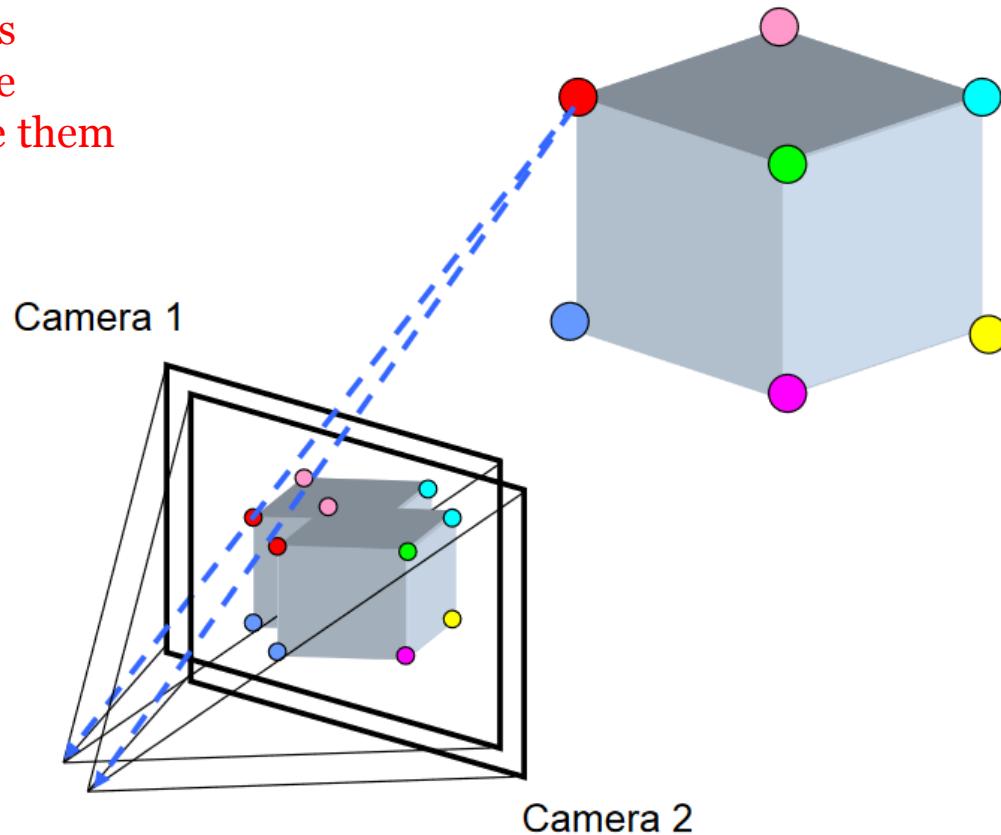
Slide credit:  
Noah Snavely

# Multi-view geometry problems

---

- **Optical flow:** Given two images, find the location of a world point in a second close-by image with no camera info.

3D point coordinates  
are unknown, but we  
won't try to estimate them



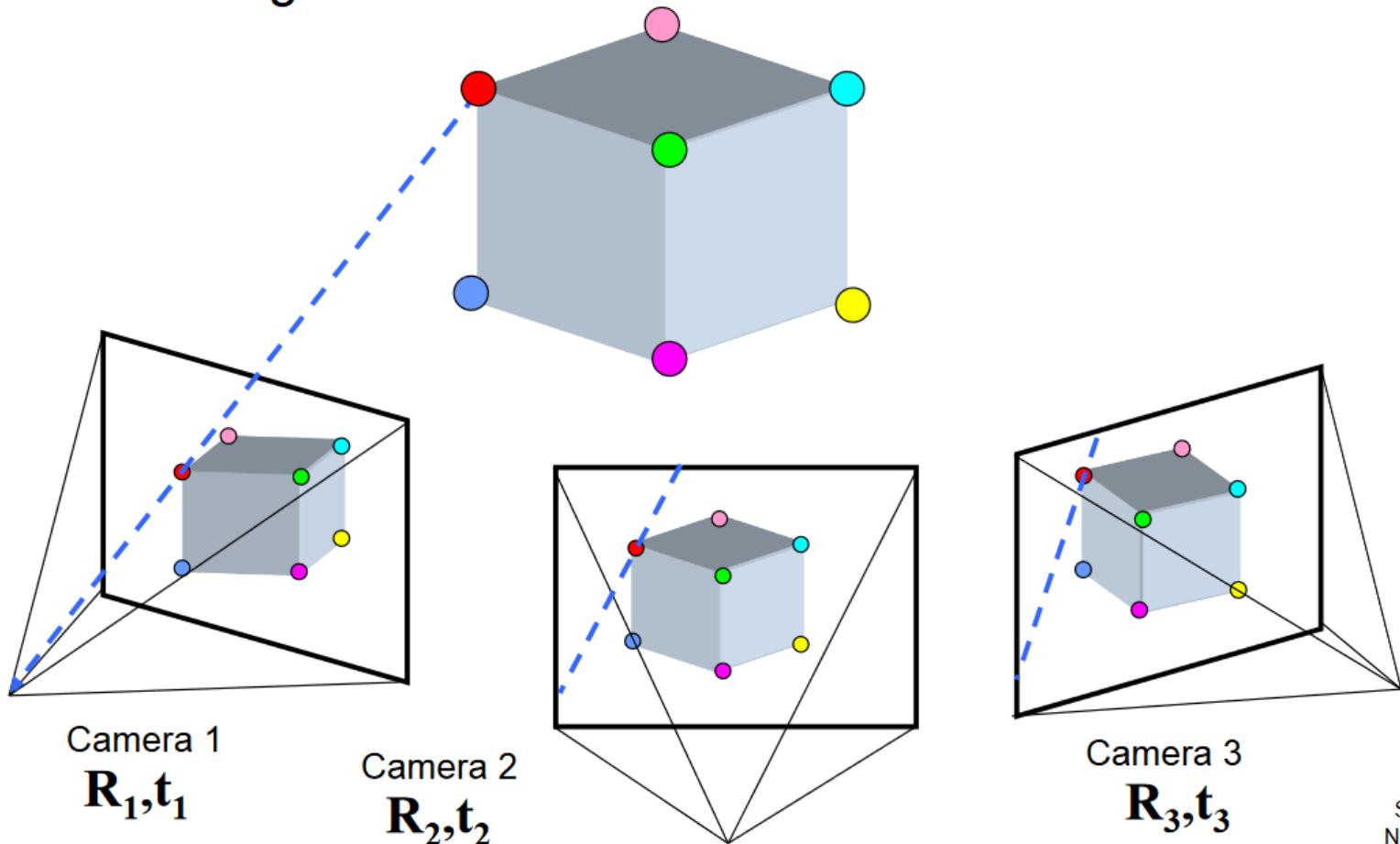
Camera frame  
transformations  
are unknown, but we  
won't try to estimate them

We are estimating pixel displacement  
from one image to the next

# Multi-view geometry problems

---

- **Stereo correspondence:** Given a point in one of the images, where could its corresponding points be in the other images?



Slide credit:  
Noah Snavely

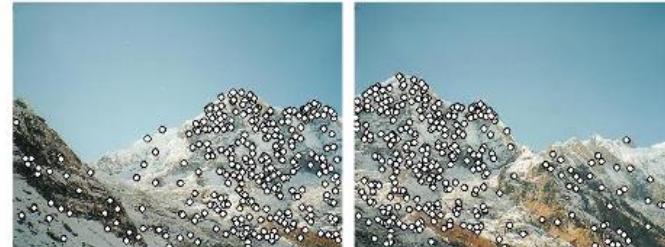
Basic underlying component in many of these problems:  
keypoint detection and  
matching across images

# Local features: main components

---

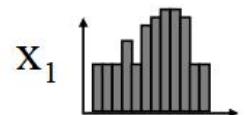
## 1) Detection:

Find a set of distinctive key points.

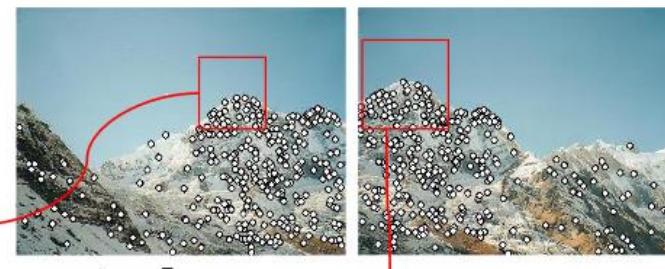


## 2) Description:

Extract feature descriptor around each interest point as vector.



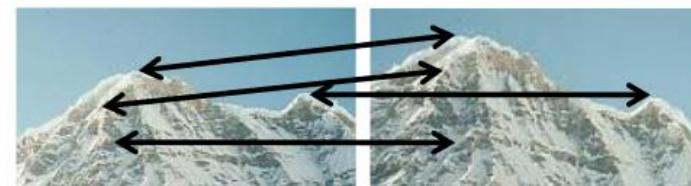
$$\mathbf{x}_1 = [x_1^{(1)}, \dots, x_d^{(1)}]$$



## 3) Matching:

Compute distance between feature vectors to find correspondence.

$$d(\mathbf{x}_1, \mathbf{x}_2) < T$$

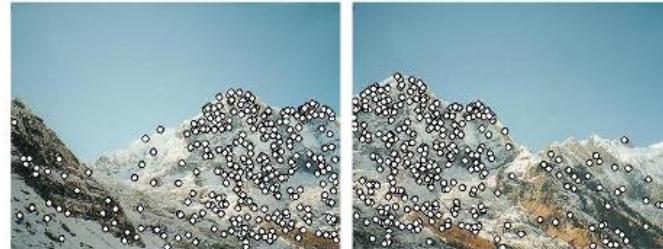


# Local features: main components

---

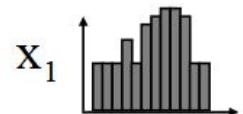
## 1) Detection:

Find a set of distinctive key points.

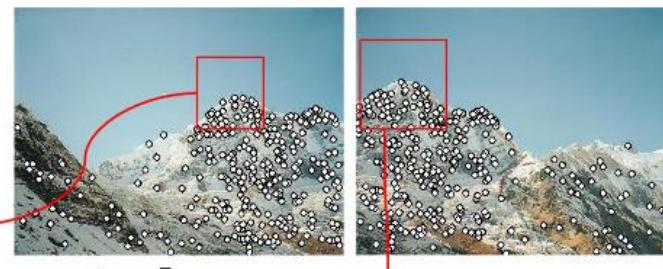


## 2) Description:

Extract feature descriptor around each interest point as vector.



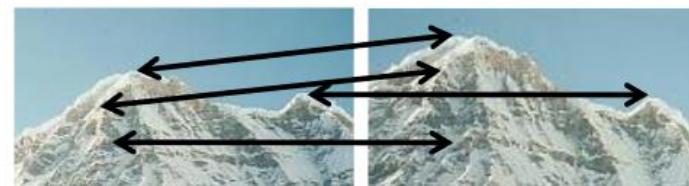
$$\mathbf{x}_1 = [x_1^{(1)}, \dots, x_d^{(1)}]$$



## 3) Matching:

Compute distance between feature vectors to find correspondence.

$$d(\mathbf{x}_1, \mathbf{x}_2) < T$$



Ideally, we want the descriptor to be invariant (i.e. little to no change) when there are

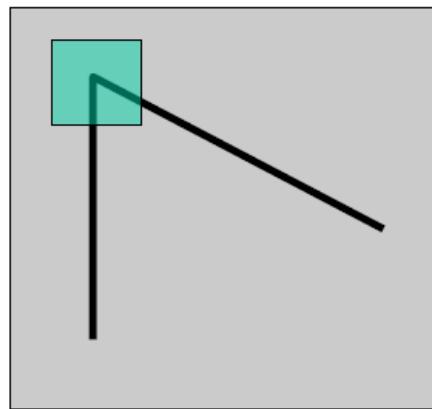
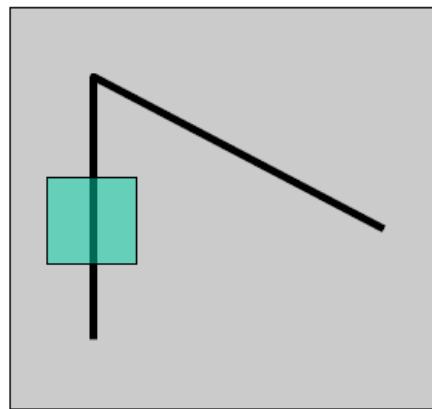
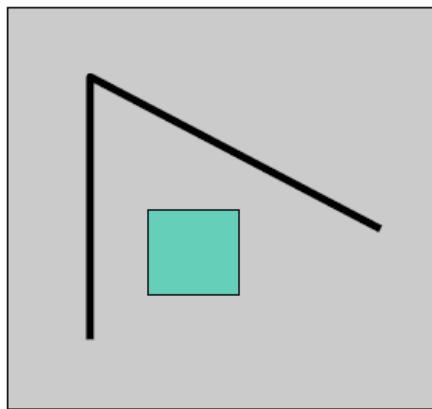
- viewpoint changes  
(small rotation or translation of the camera)
- scale-changes
- illumination changes

# Local measures of uniqueness

---

Suppose we only consider a small window of pixels

- What defines whether a feature is a good or bad candidate?

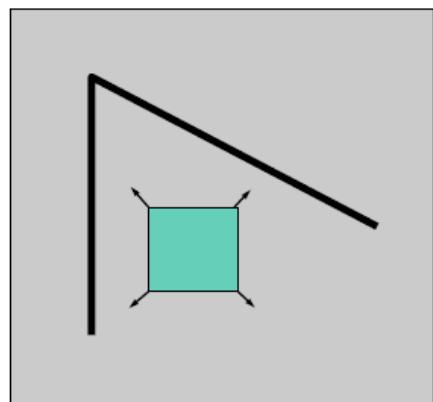


# Feature detection

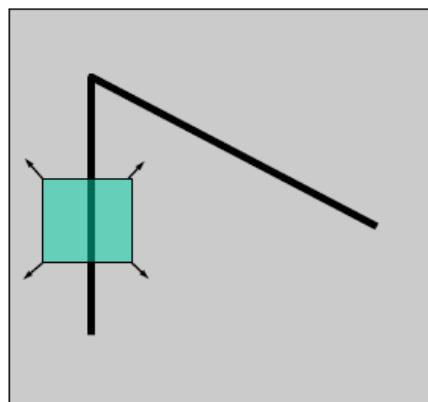
---

Local measure of feature uniqueness

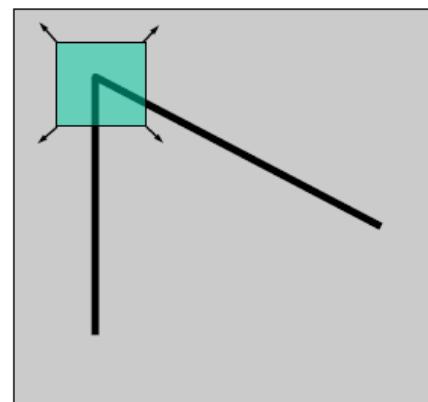
- How does the window change when you shift by a *small amount*?



“flat” region:  
no change in all  
directions



“edge”:  
no change along the  
edge direction

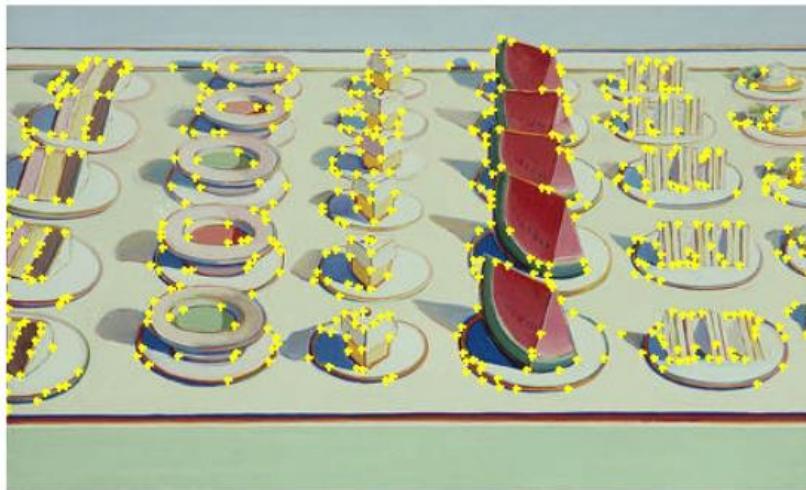


“corner”:  
significant change in  
all directions

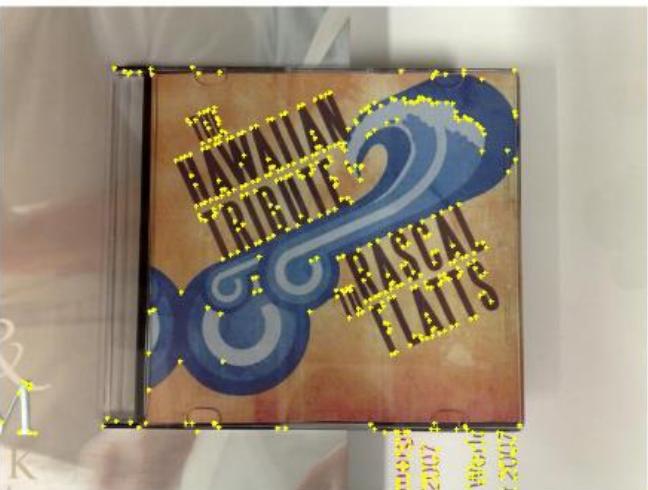
# Corner detectors

- Harris
- FAST
- Laplacian of Gaussian detector
- SUSAN
- Forstner

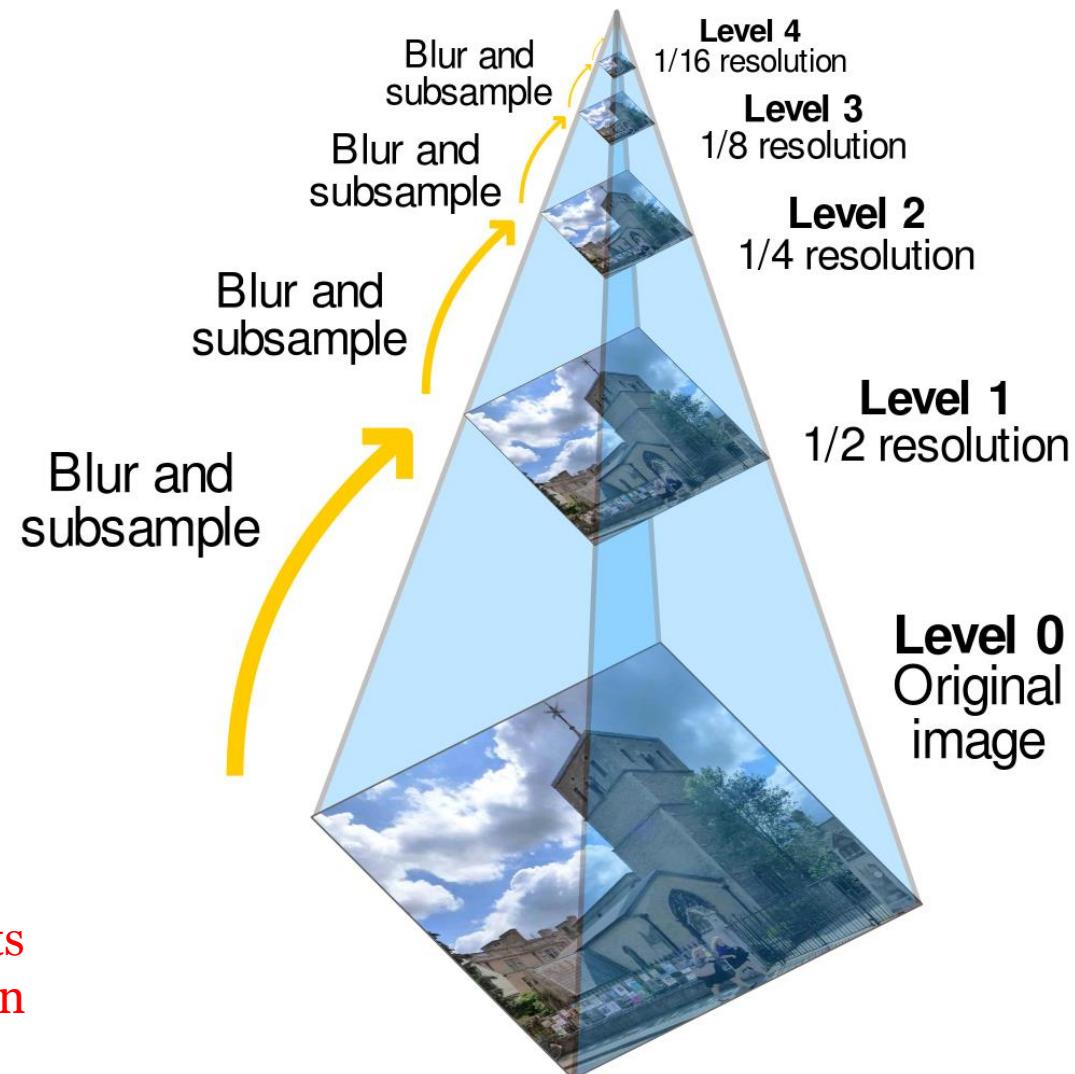
# Superimposed Harris keypoints



500 strongest  
keypoints



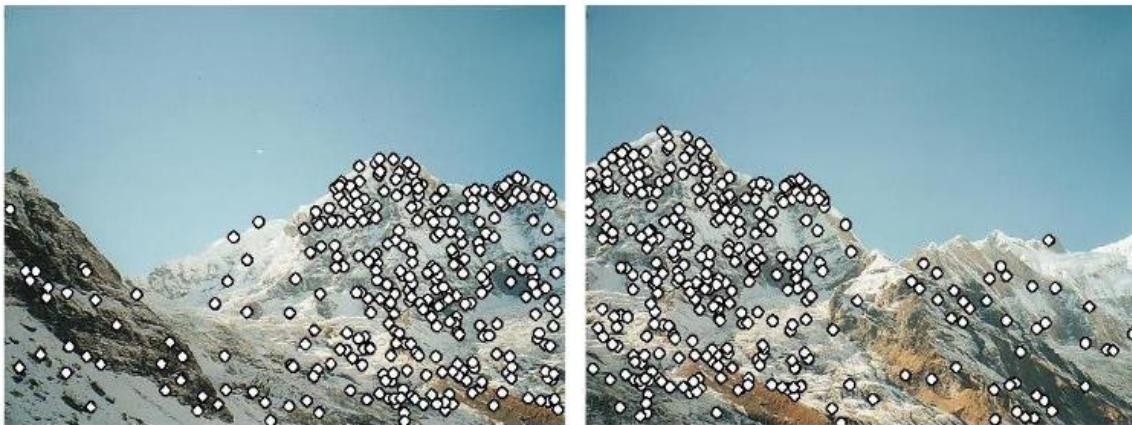
# Scale-space representation



Feature detection:

search for “corners”/keypoints  
across many scales, and return  
a list of (x, y, scale) keypoints

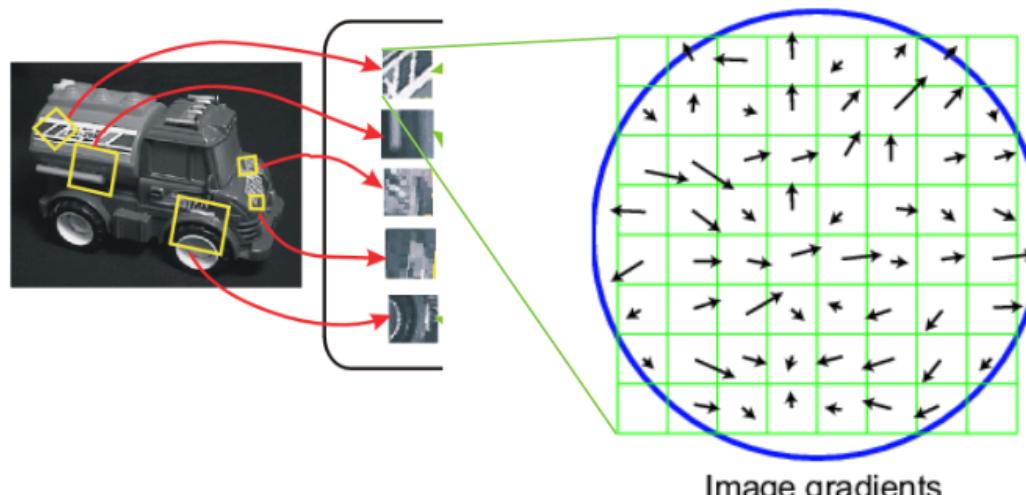
# Characteristics of good features



- **Repeatability**
  - The same feature can be found in several images despite geometric and photometric transformations
- **Saliency**
  - Each feature is distinctive
- **Compactness and efficiency**
  - Many fewer features than image pixels
- **Locality**
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

# SIFT descriptor formation

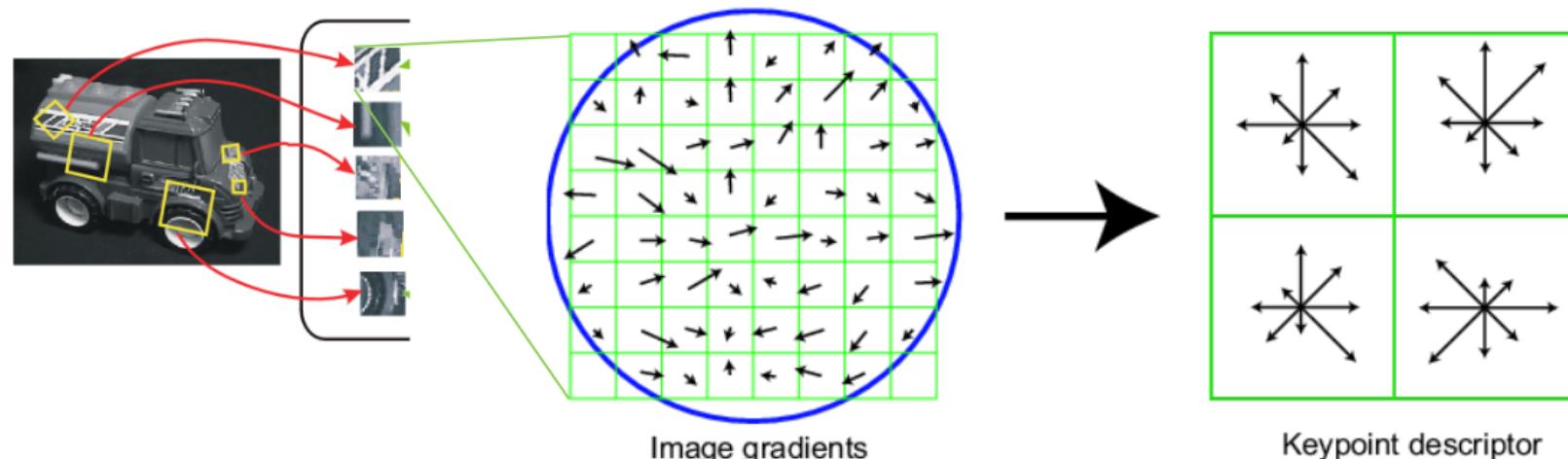
- Compute on local  $16 \times 16$  window around detection.
- Rotate and scale window according to discovered orientation  $\Theta$  and scale  $\sigma$  (gain invariance).
- Compute gradients weighted by a Gaussian of variance half the window (for smooth falloff).



Actually  $16 \times 16$ , only showing  $8 \times 8$

# SIFT vector formation

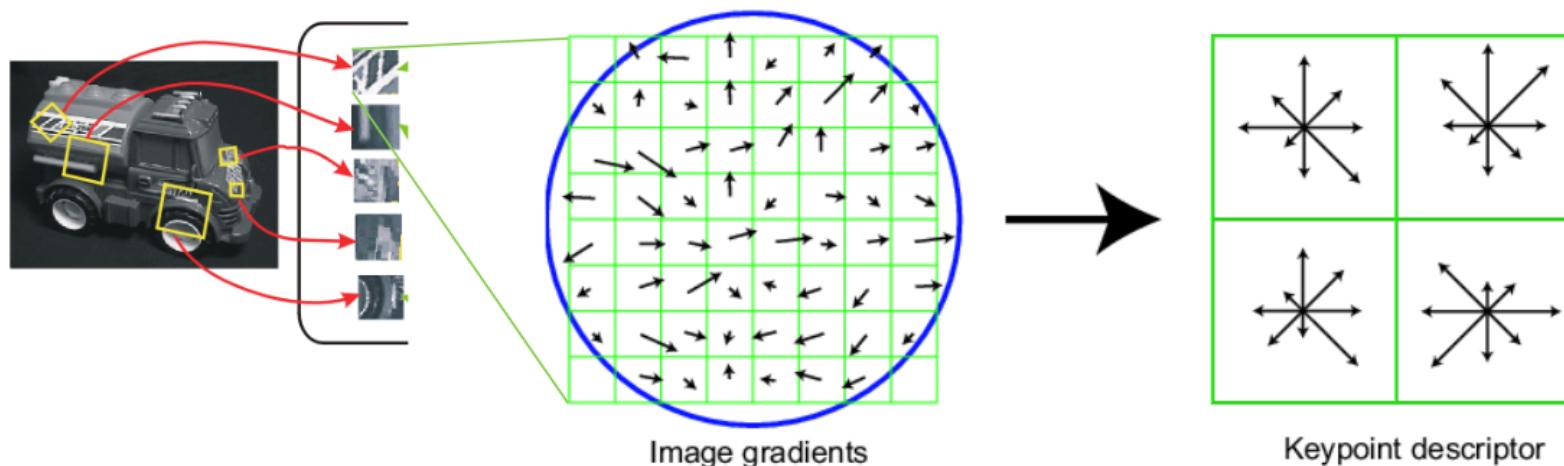
- 4x4 array of gradient orientation histograms weighted by gradient magnitude.
- Bin into 8 orientations x 4x4 array = 128 dimensions.



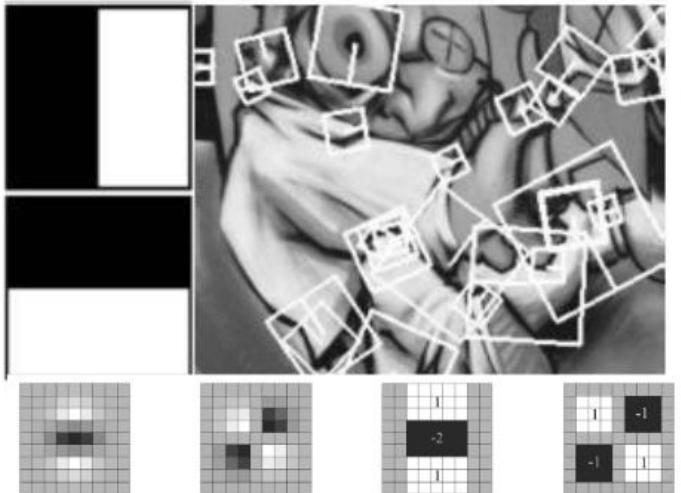
Showing only 2x2 here but is 4x4  
James Hays

# Reduce effect of illumination

- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - After normalization, clamp gradients  $> 0.2$
  - Renormalize



# Local Descriptors: SURF



## Fast approximation of SIFT idea

Efficient computation by 2D box filters & integral images  
⇒ 6 times faster than SIFT  
Equivalent quality for object identification

## GPU implementation available

Feature extraction @ 200Hz  
(detector + descriptor, 640×480 img)  
<http://www.vision.ee.ethz.ch/~surf>

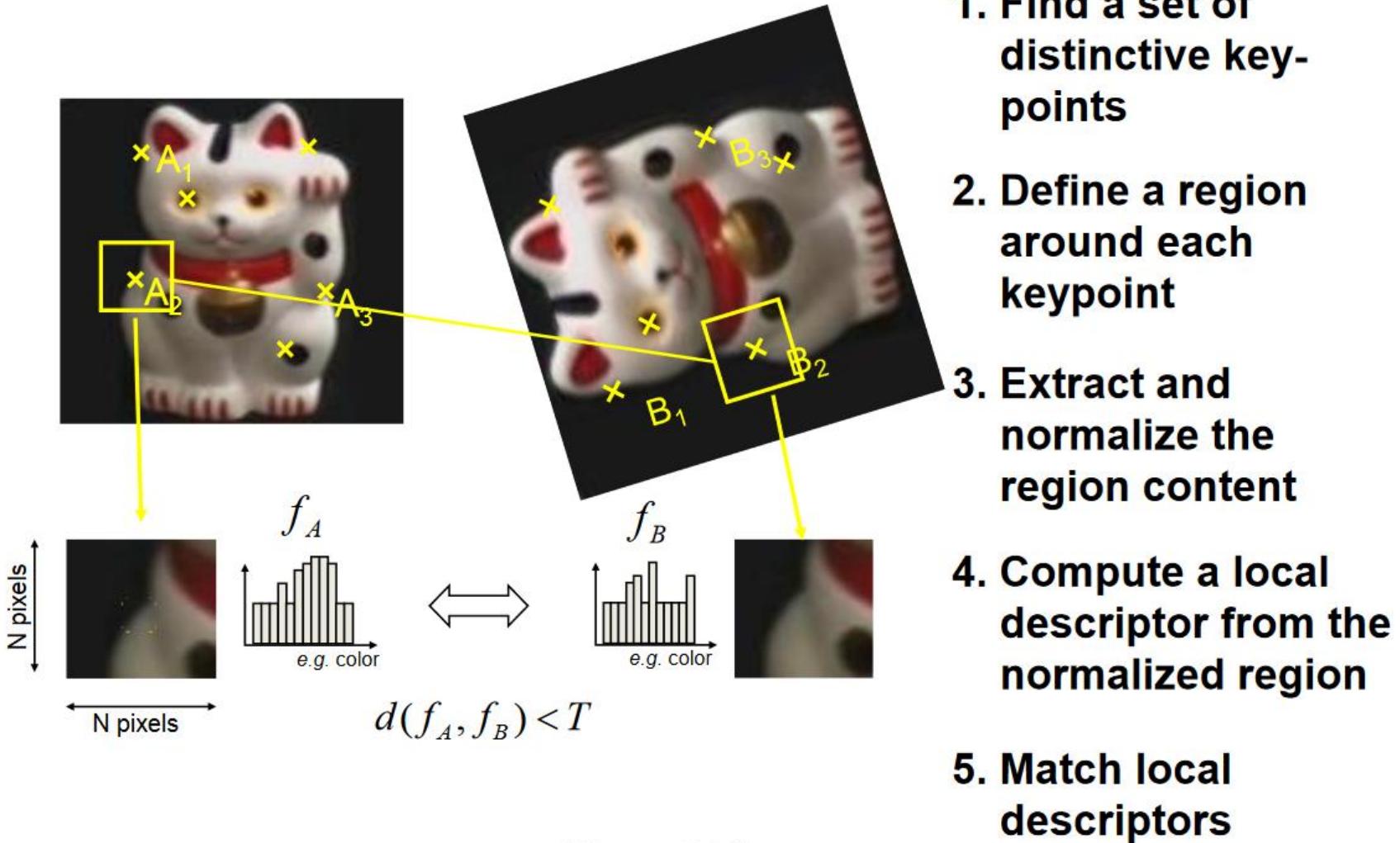
# Many other local descriptors

- ORB
- BRIEF
- FREAK
- RootSIFT-PCA

# Feature matching



# Overview of Keypoint Matching



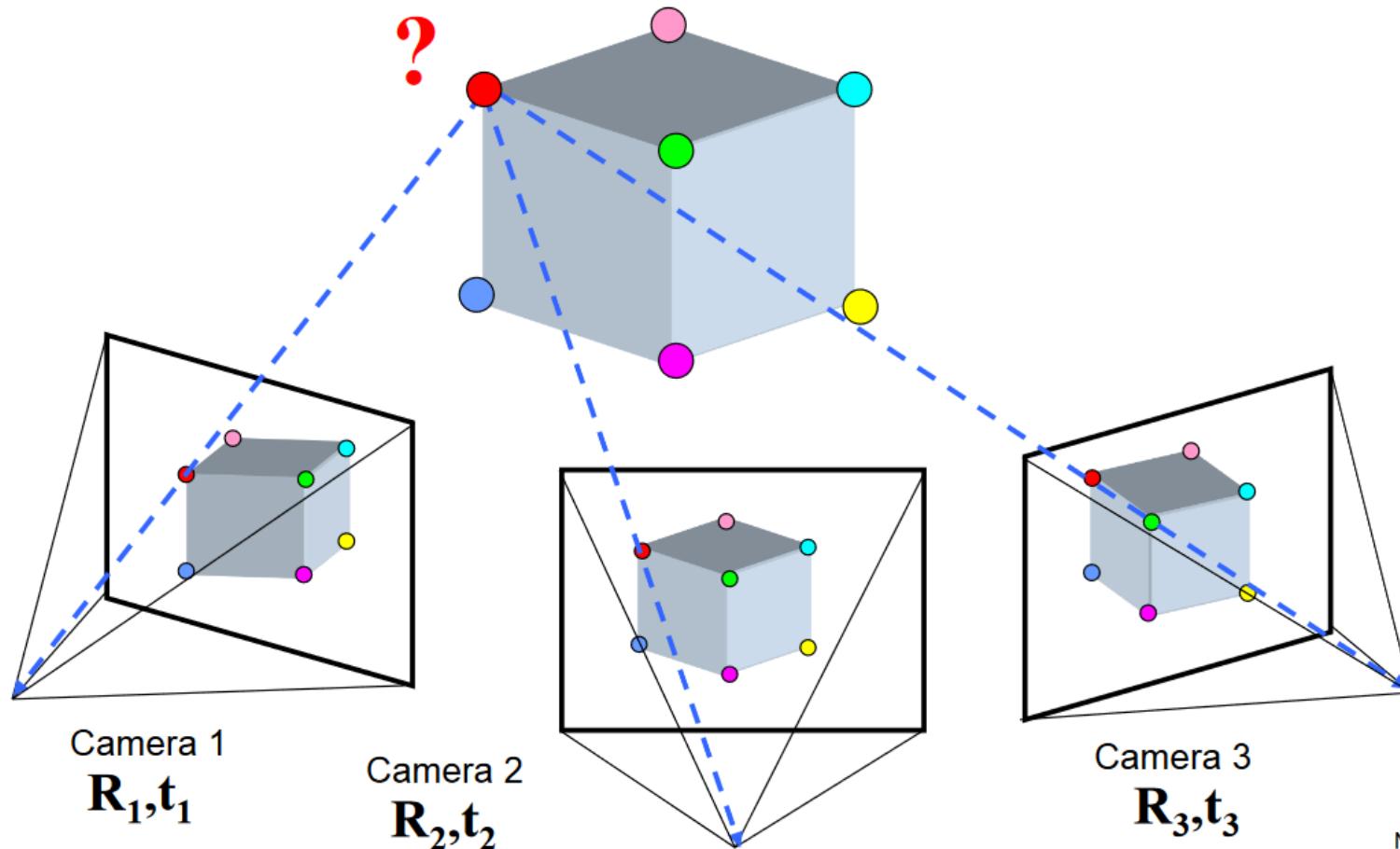
Problem #1: landmark triangulation

# Multi-view geometry problems

A.k.a. mapping

- **Structure:** Given projections of the same 3D point in two or more images, compute the 3D coordinates of that point

3D point coordinates  
are unknown and to  
be estimated



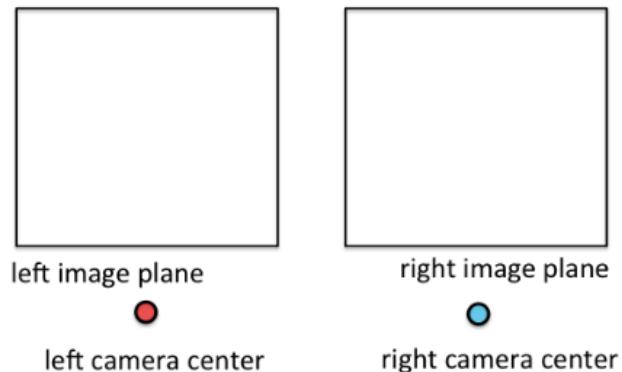
Camera frame  
transformations  
are known

Slide credit:  
Noah Snavely

## Epipolar geometry

- Case with two cameras with parallel optical axes ← **First this**
- General case

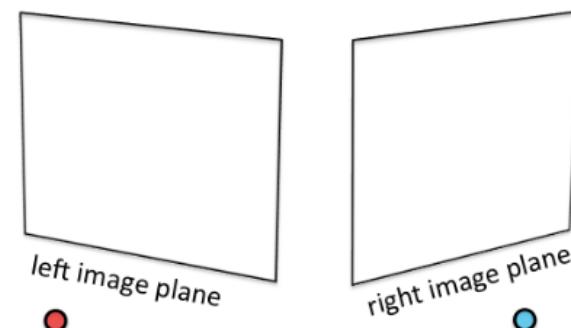
Parallel stereo cameras:



$$\mathbf{t} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix}$$

An orange arrow points from the left camera center towards the right camera center, indicating the translation vector  $\mathbf{t}$ .

General stereo cameras:

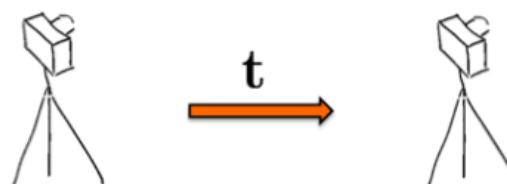
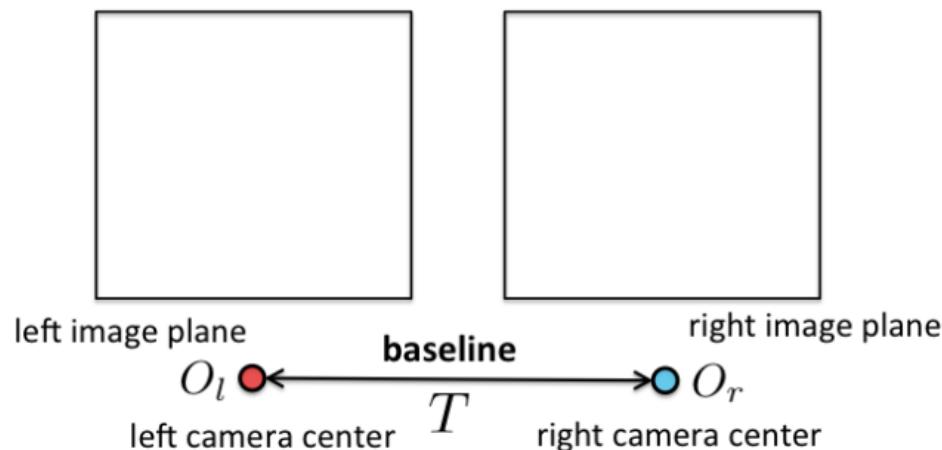


$$[R \mid \mathbf{t}]$$

An orange arrow points from the left camera center towards the right camera center, indicating the transformation matrix  $[R \mid \mathbf{t}]$ .

# Stereo: Parallel Calibrated Cameras

- We assume that the two calibrated cameras (we know intrinsics and extrinsics) are parallel, i.e. the right camera is just some distance to the right of left camera. We assume we know this distance. We call it the **baseline**.

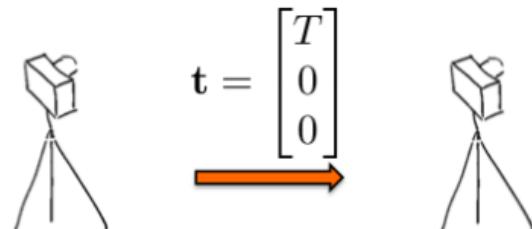
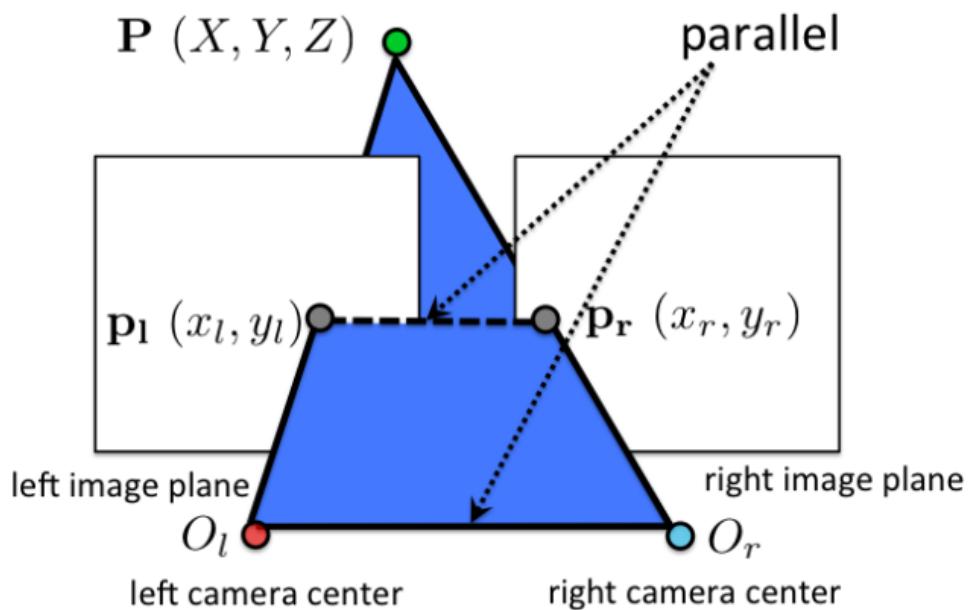


$$t = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix}$$

The right camera is shifted to the right in X direction

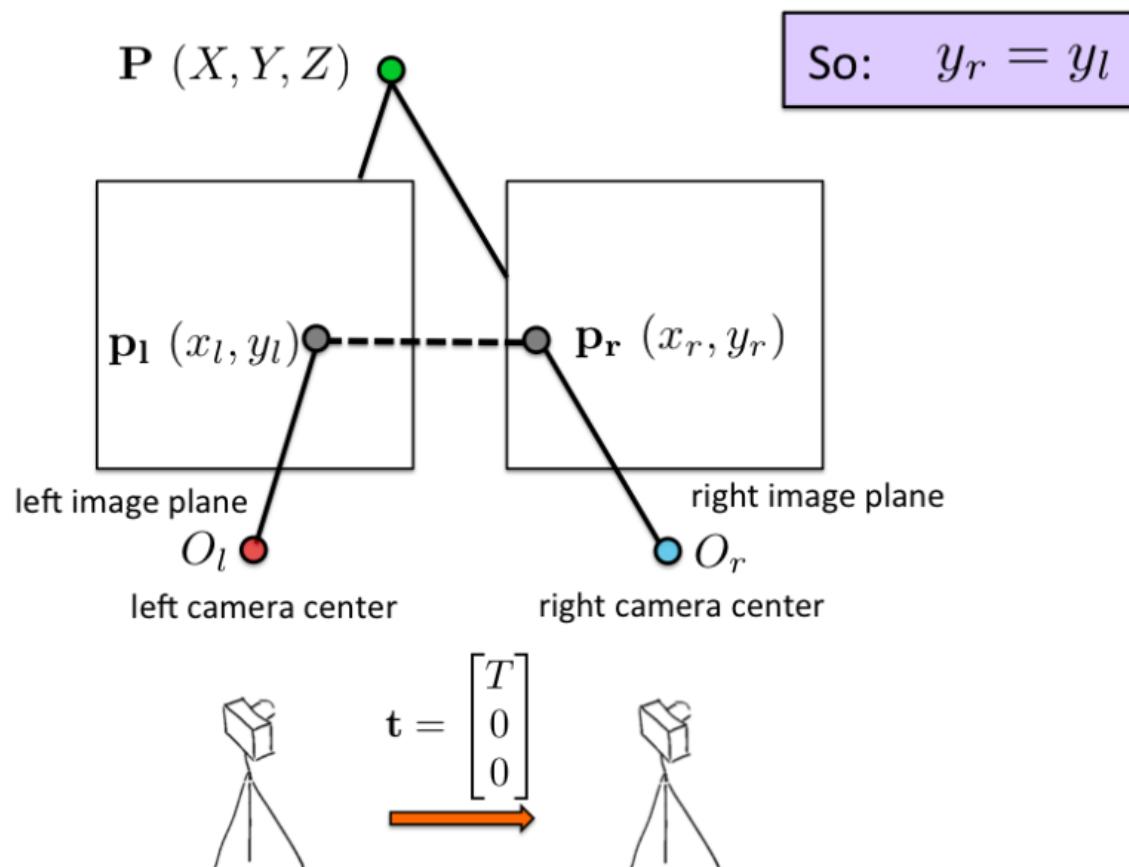
# Stereo: Parallel Calibrated Cameras

- Points  $O_l$ ,  $O_r$  and  $P$  (and  $p_l$  and  $p_r$ ) lie on a plane. Since two image planes lie on the same plane (distance  $f$  from each camera), the lines  $O_lO_r$  and  $p_lp_r$  are parallel.



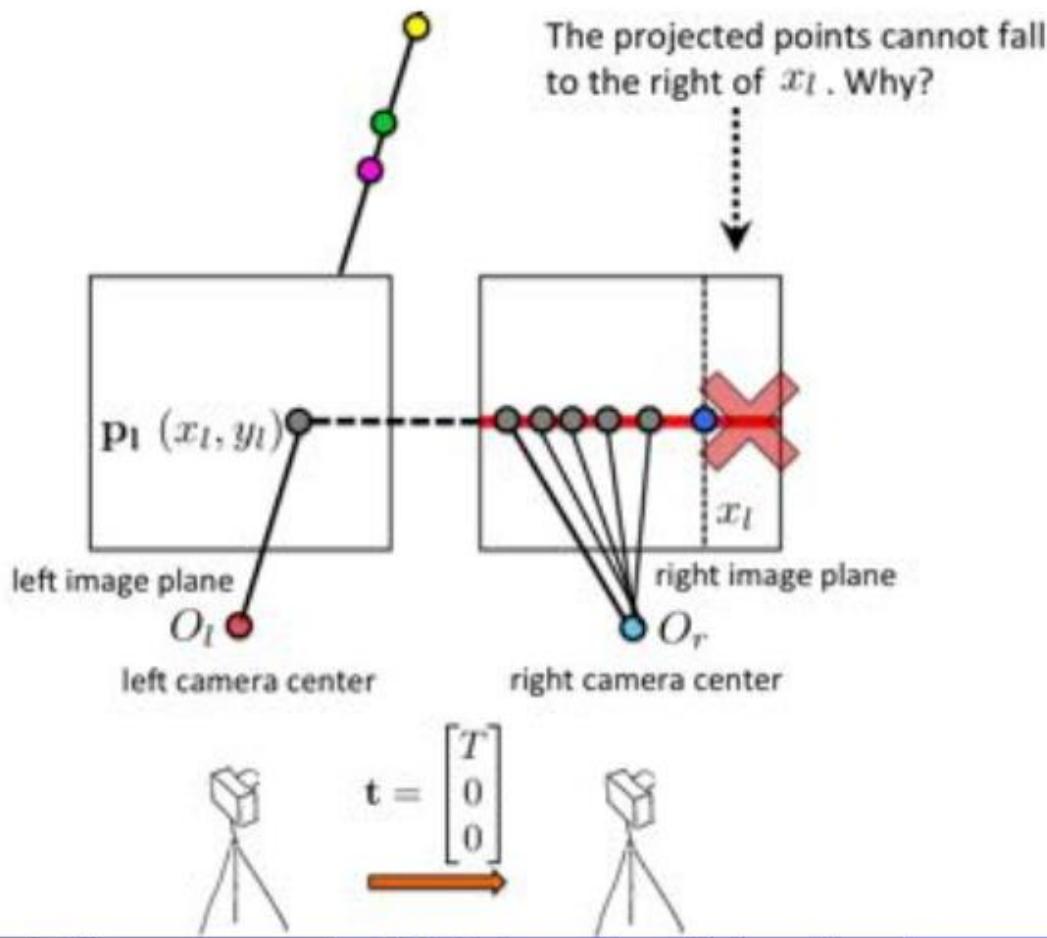
# Stereo: Parallel Calibrated Cameras

- Since lines  $O_l O_r$  and  $p_l p_r$  are parallel, and  $O_l$  and  $O_r$  have the same  $y$ , then also  $p_l$  and  $p_r$  have the same  $y$ :  $y_r = y_l$ !



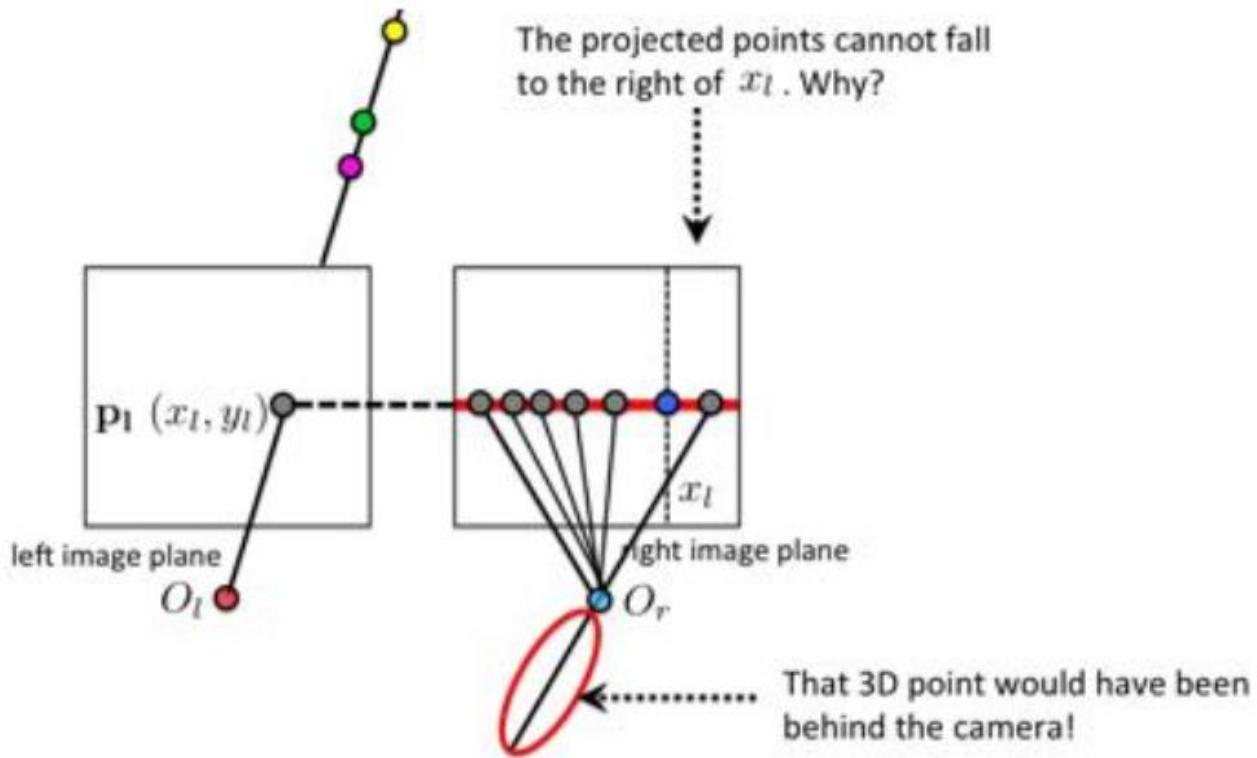
## Stereo: Parallel Calibrated Cameras

- Another observation: No point from  $O_l p_l$  can project to the right of  $x_l$  in the right image. Why?



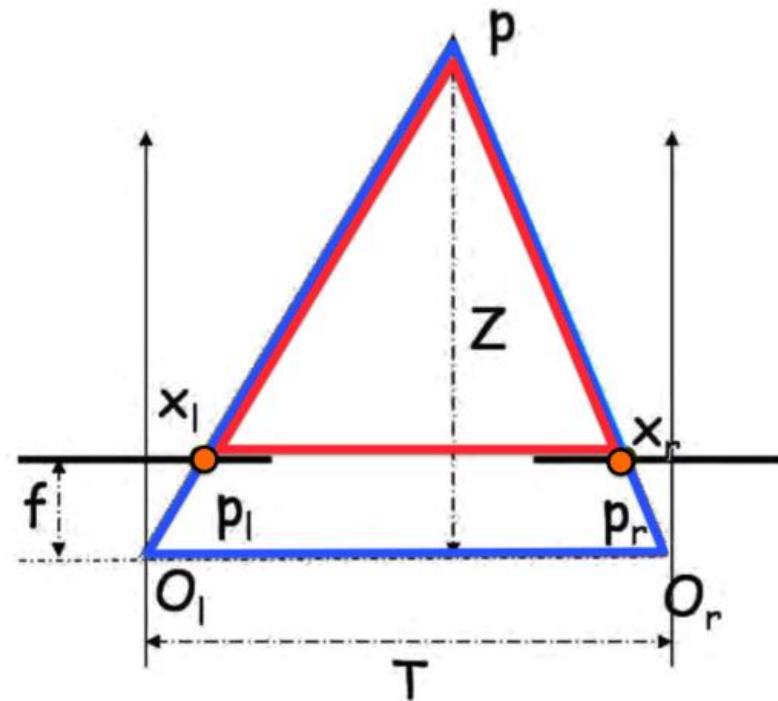
## Stereo: Parallel Calibrated Cameras

- Because that would mean our image can see behind the camera...



# Stereo: Parallel Calibrated Cameras

- We can then use similar triangles to compute the depth of the point  $P$



Similar triangles:

$$\frac{T}{Z} = \frac{T + x_r - x_l}{Z - f}$$

$$Z = \frac{f \cdot T}{x_l - x_r}$$

baseline  
focal length      disparity

In metric, not in pixel coordinates. To convert to pixel coordinates need to use elements of the camera calibration matrix.

[Adopted from: J. Hays]

Conclusion: if you have a well-calibrated and rectified (parallel) stereo camera you do not need to do least squares triangulation.

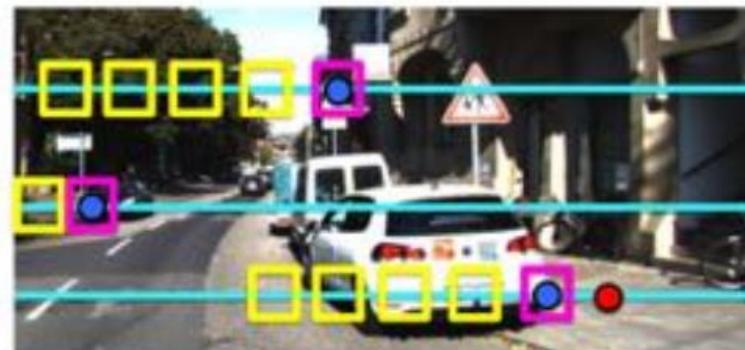
You can estimate depth via the disparity map.

## Stereo: Parallel Calibrated Cameras

- For each point  $\mathbf{p}_l = (x_l, y_l)$ , how do I get  $\mathbf{p}_r = (x_r, y_r)$ ? By matching. Patch around  $(x_r, y_r)$  should look similar to the patch around  $(x_l, y_l)$ .



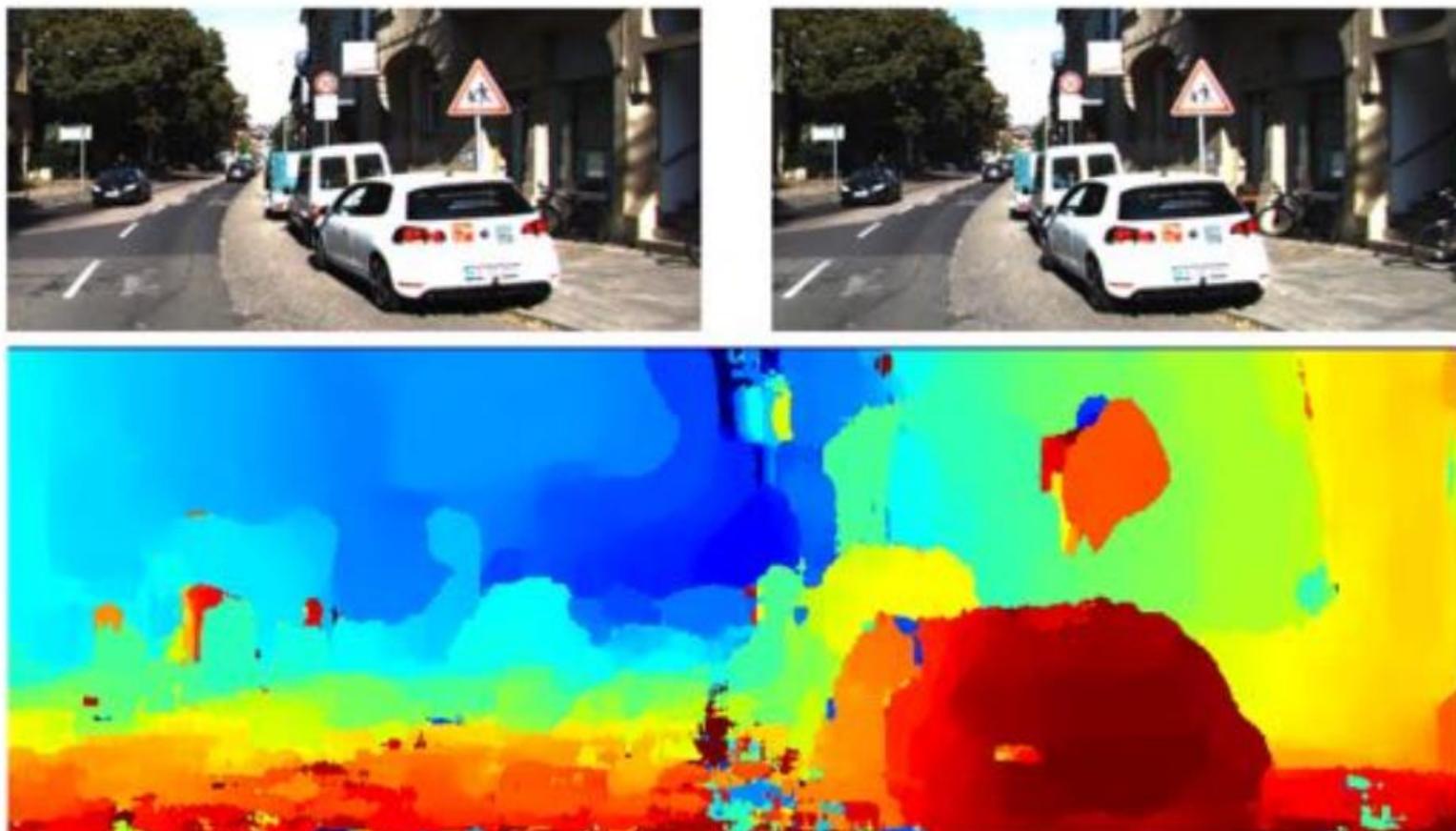
left image



Do this for all the points in the left image!

## Stereo: Parallel Calibrated Cameras

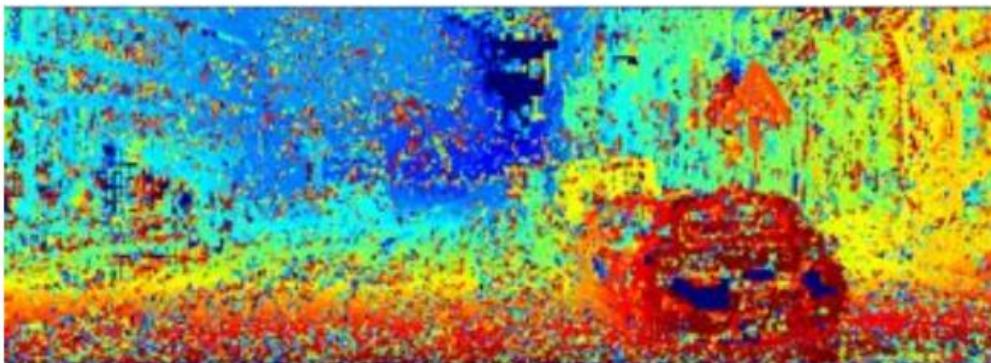
- We get a disparity map as a result



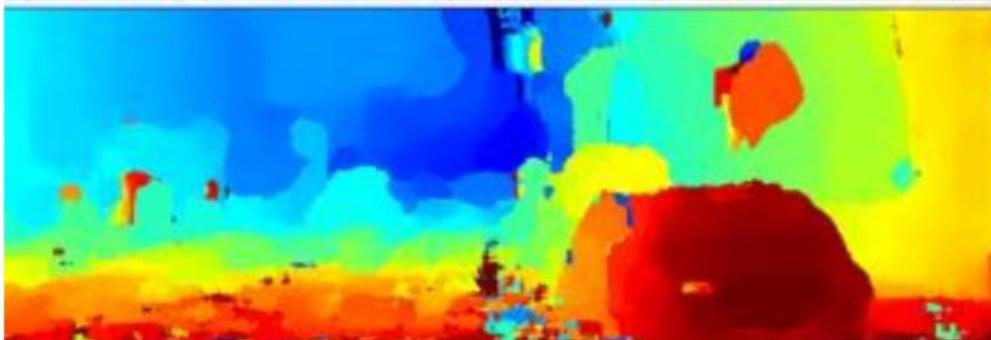
**Result: Disparity map**  
(red values large disp., blue small disp.)

# Stereo: Parallel Calibrated Cameras

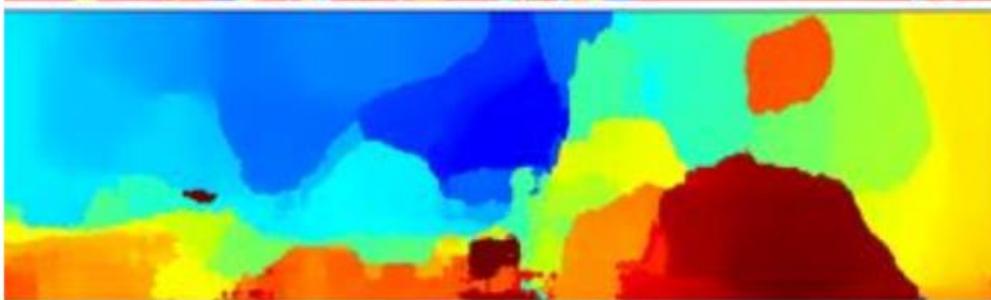
- Smaller patches: more detail, but noisy. Bigger: less detail, but smooth



patch size = 5



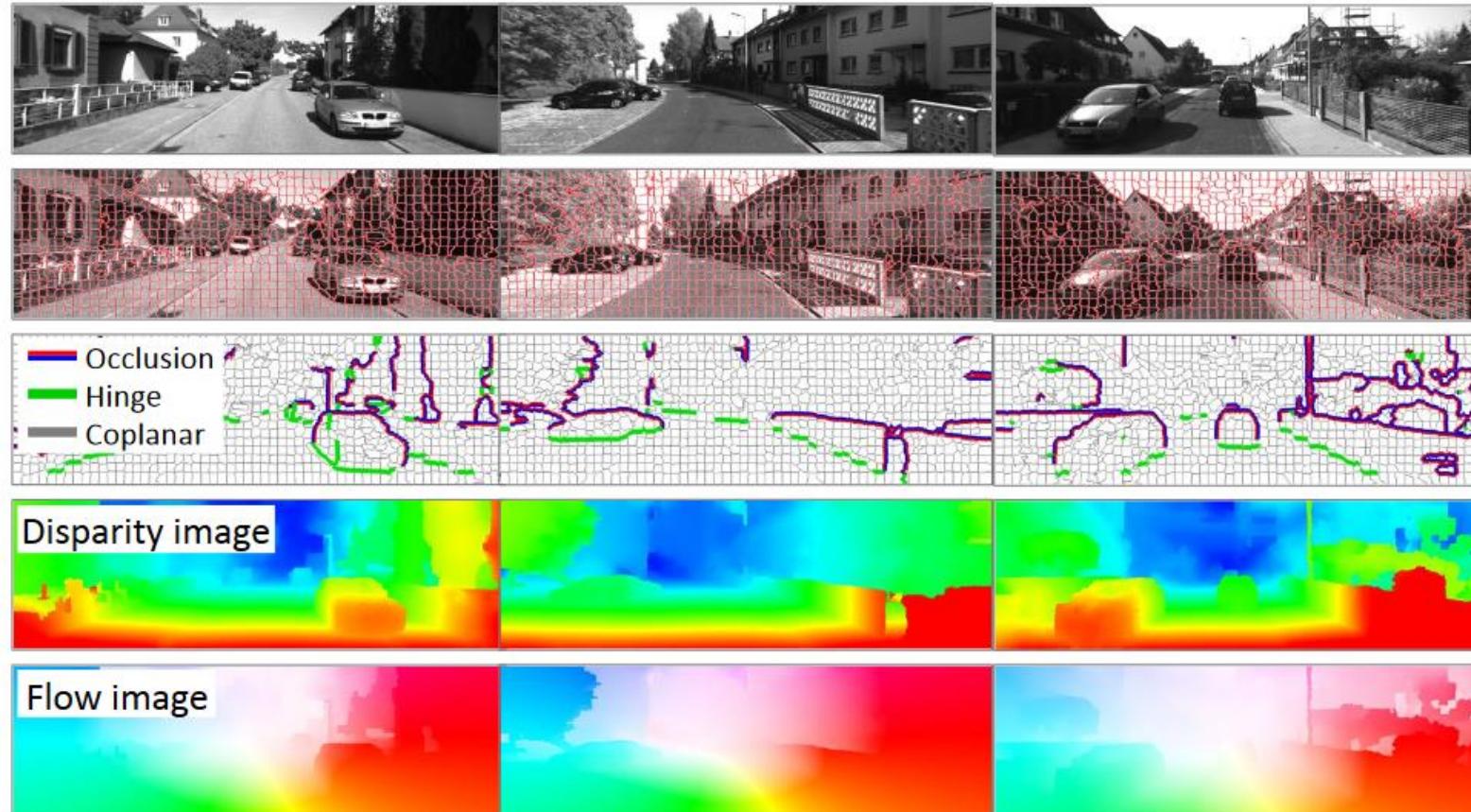
patch size = 35



patch size = 85

# You Can Do It Much Better...

[K. Yamaguchi, D. McAllester and R. Urtasun, ECCV 2014]



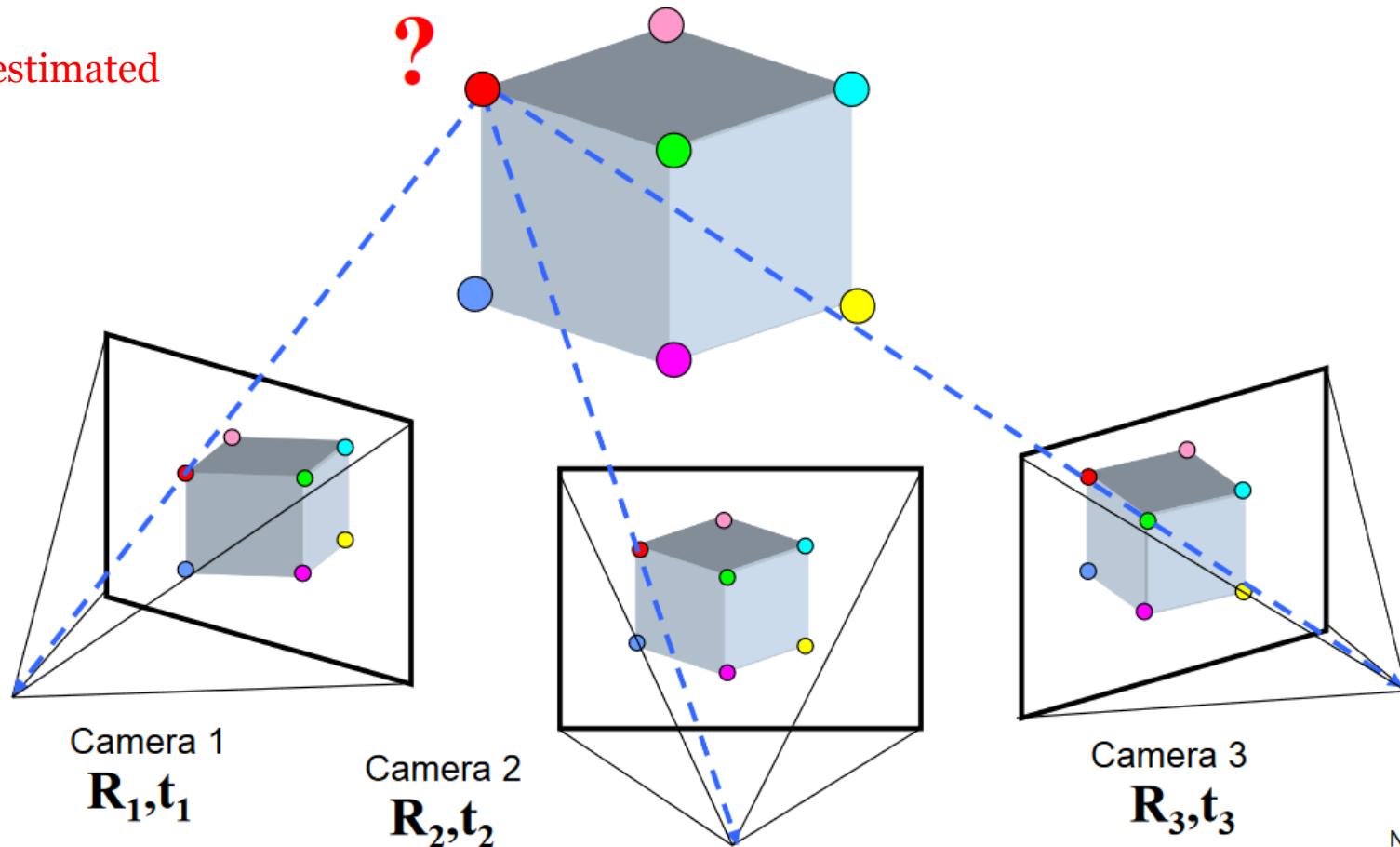
# Multi-view geometry problems

- **Structure:** Given projections of the same 3D point in two or more images, compute the 3D coordinates of that point

$3 \times 7 = 21$  variables to be estimated

7 pixel observations in each camera, so 21 pixel observations across all cameras

→ 42 constraints in total



# Triangulation as a least squares problem

$${}^w X^*, {}^w Y^*, {}^w Z^* = \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z]}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k R^w p + {}^k t_{kw}) ] \|^2$$

Actual pixel observation of a  
keypoint by camera frame  $k$

Expected pixel observation of  
3D point  ${}^w p$  by camera frame  $k$

$$h_{\text{pinhole}}(X, Y, Z) = \begin{bmatrix} \frac{f m_x X}{Z} + c_x \\ \frac{f m_y Y}{Z} + c_y \end{bmatrix} + n, \quad n \sim \mathcal{N}(0, R) \quad \text{noise (in pixels)}$$

# Triangulation as a least squares problem

$${}^w X^*, {}^w Y^*, {}^w Z^* = \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z]}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k R {}^w p + {}^k t_k) ] \|^2$$

Enumerate all cameras that observed the keypoint.

The only term to be optimized. The rest are known.

$$h_{\text{pinhole}}(X, Y, Z) = \begin{bmatrix} \frac{f m_x X}{Z} + c_x \\ \frac{f m_y Y}{Z} + c_y \end{bmatrix} + n, \quad n \sim \mathcal{N}(0, R) \quad \text{noise (in pixels)}$$

# Triangulation as a least squares problem

$${}^w X^*, {}^w Y^*, {}^w Z^* = \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z]}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k_w R^w p + {}^k t_{kw}) ] \|^2$$



3D point expressed in the  
frame of camera k

$${}^k p = {}^k_w R^w p + {}^k t_{kw}$$

$$h_{\text{pinhole}}(X, Y, Z) = \begin{bmatrix} \frac{f m_x X}{Z} + c_x \\ \frac{f m_y Y}{Z} + c_y \end{bmatrix} + n, \quad n \sim \mathcal{N}(0, R) \quad \text{noise (in pixels)}$$

# Triangulation as a least squares problem

$${}^w X^*, {}^w Y^*, {}^w Z^* = \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z]}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k R^w p + {}^k t_{kw}) ] \|^2$$



Reprojection error of point

$${}^k p = {}^k R^w p + {}^k t_{kw}$$

into camera k's frame

$$h_{\text{pinhole}}(X, Y, Z) = \begin{bmatrix} \frac{f m_x X}{Z} + c_x \\ \frac{f m_y Y}{Z} + c_y \end{bmatrix} + n, \quad n \sim \mathcal{N}(0, R) \quad \text{noise (in pixels)}$$

# Triangulation as a least squares problem

$$\begin{aligned} {}^w X^*, {}^w Y^*, {}^w Z^* &= \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z]}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k R^w p + {}^k t_{kw}) ] \|^2 \\ &= \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z], {}^w p \rightarrow {}^k p}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k p) ] \|^2 \end{aligned}$$

$$h_{\text{pinhole}}(X, Y, Z) = \begin{bmatrix} \frac{f m_x X}{Z} + c_x \\ \frac{f m_y Y}{Z} + c_y \end{bmatrix} + n, \quad n \sim \mathcal{N}(0, R) \quad \text{noise (in pixels)}$$

# Triangulation as a least squares problem

$$\begin{aligned}
{}^w X^*, {}^w Y^*, {}^w Z^* &= \operatorname{argmin}_{{}^w p = [{}^w X, {}^w Y, {}^w Z]} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k R^w p + {}^k t_{kw}) ] \|^2 \\
&= \operatorname{argmin}_{{}^w p = [{}^w X, {}^w Y, {}^w Z]} \sum_{k=1, {}^w p \rightarrow {}^k p}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k p) ] \|^2 \\
&= \operatorname{argmin}_{{}^w p = [{}^w X, {}^w Y, {}^w Z]} \sum_{k=1, {}^w p \rightarrow {}^k p}^K \| \begin{bmatrix} \bar{u}^{(k)} \\ \bar{v}^{(k)} \end{bmatrix} - \begin{bmatrix} \frac{f m_x {}^k X}{Z} + c_x \\ \frac{f m_y {}^k Y}{Z} + c_y \end{bmatrix} \|^2
\end{aligned}$$

$$h_{\text{pinhole}}(X, Y, Z) = \begin{bmatrix} \frac{f m_x X}{Z} + c_x \\ \frac{f m_y Y}{Z} + c_y \end{bmatrix} + n, \quad n \sim \mathcal{N}(0, R) \quad \text{noise (in pixels)}$$

# Triangulation as a least squares problem

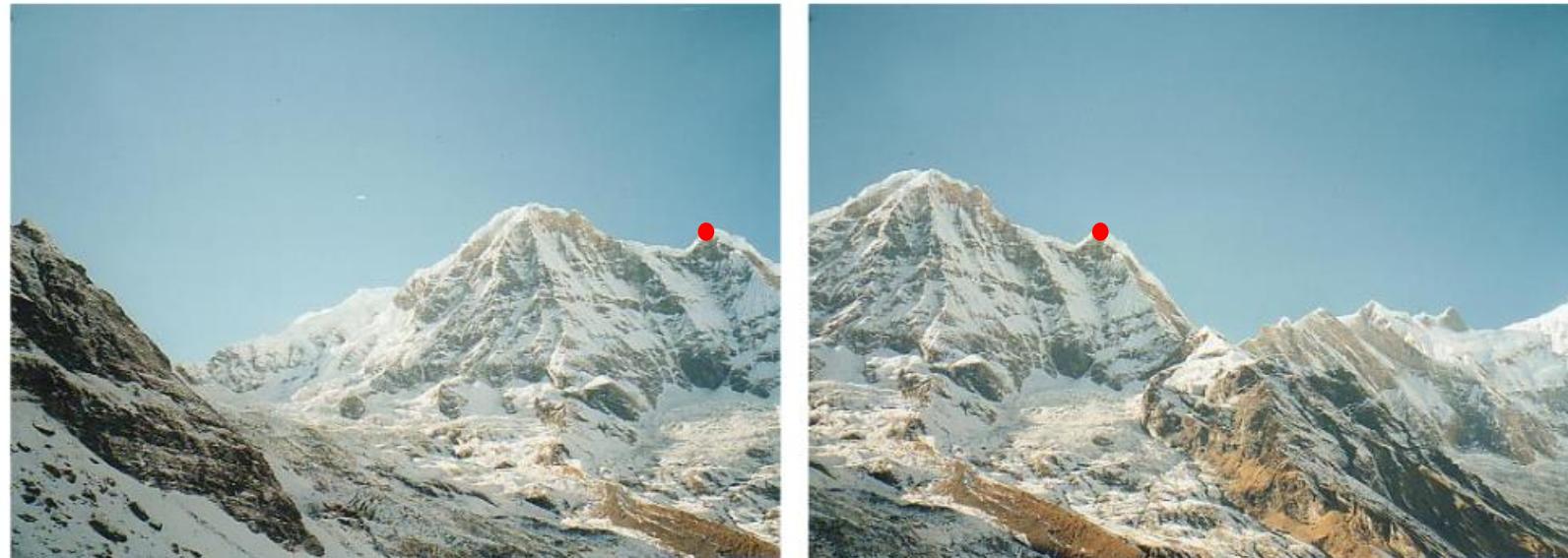
$$\begin{aligned}
 {}^w X^*, {}^w Y^*, {}^w Z^* &= \operatorname{argmin}_{{}^w p = [{}^w X, {}^w Y, {}^w Z]} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k R^w p + {}^k t_{kw}) ] \|^2 \\
 &= \operatorname{argmin}_{{}^w p = [{}^w X, {}^w Y, {}^w Z]} \sum_{k=1, {}^w p \rightarrow {}^k p}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k p) ] \|^2 \\
 &= \operatorname{argmin}_{{}^w p = [{}^w X, {}^w Y, {}^w Z]} \sum_{k=1, {}^w p \rightarrow {}^k p}^K \| \begin{bmatrix} \bar{u}^{(k)} \\ \bar{v}^{(k)} \end{bmatrix} - \begin{bmatrix} \frac{f m_x {}^k X}{{}^k Z} + c_x \\ \frac{f m_y {}^k Y}{{}^k Z} + c_y \end{bmatrix} \|^2
 \end{aligned}$$

Note: unconstrained optimization does not guarantee that the solution will be in the camera's field of view. For example, it could happen that it returns  ${}^k Z < 0$  which is an invalid solution (i.e. behind the camera)

$$h_{\text{pinhole}}(X, Y, Z) = \begin{bmatrix} \frac{f m_x X}{Z} + c_x \\ \frac{f m_y Y}{Z} + c_y \end{bmatrix} + n, \quad n \sim \mathcal{N}(0, R) \quad \text{noise (in pixels)}$$

# Potential pitfalls with triangulation: near parallel rays

“point at infinity”



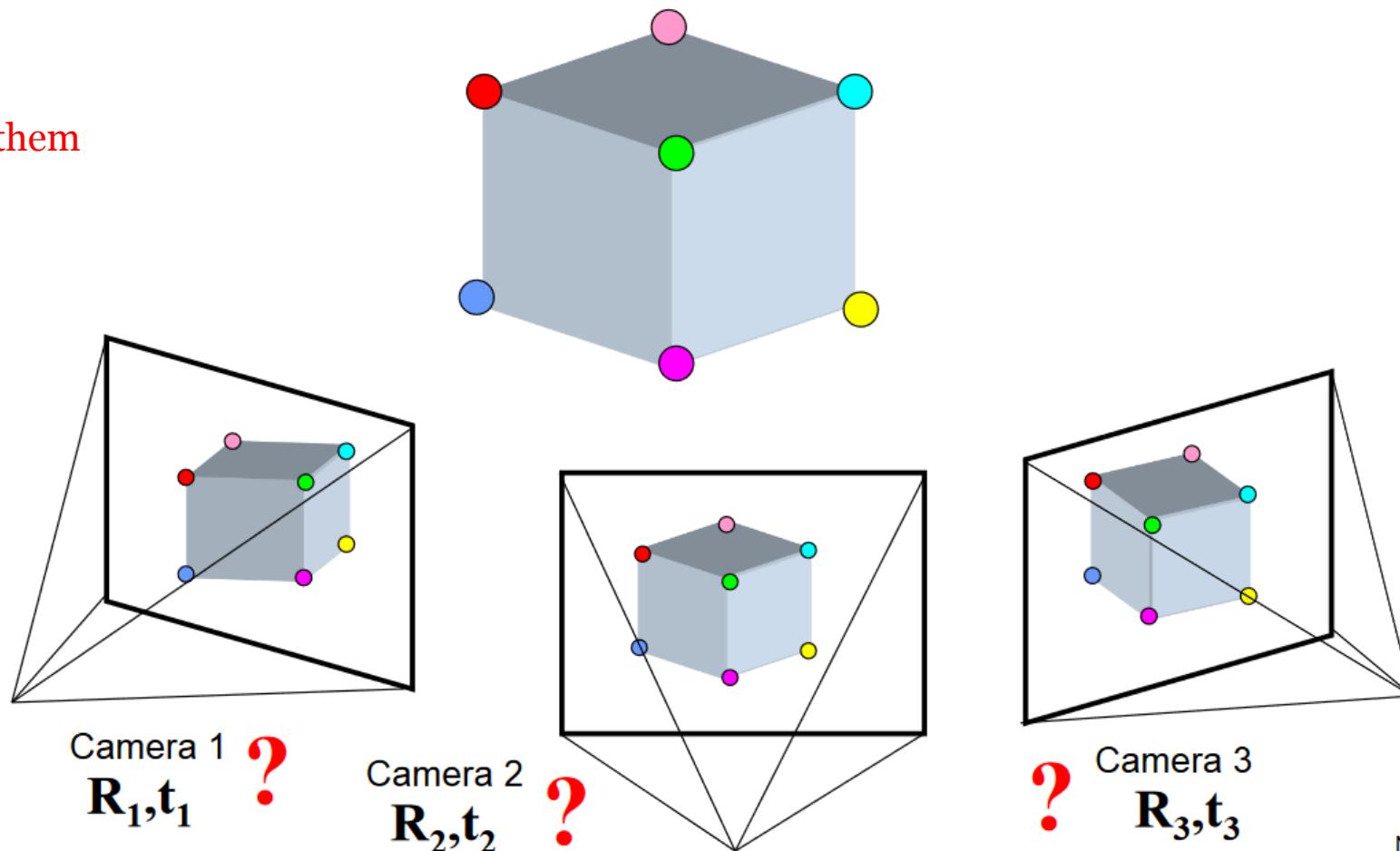
Intersection point is too far away, dominated by noise and insufficient image resolution.  
Triangulating these points is typically impossible without sufficient baseline between camera frames.

# Problem #2: camera localization/visual odometry

# Multi-view geometry problems

- **Motion:** Given a set of corresponding points in two or more images, compute the camera parameters

3D point coordinates  
are unknown, but we  
won't try to estimate them



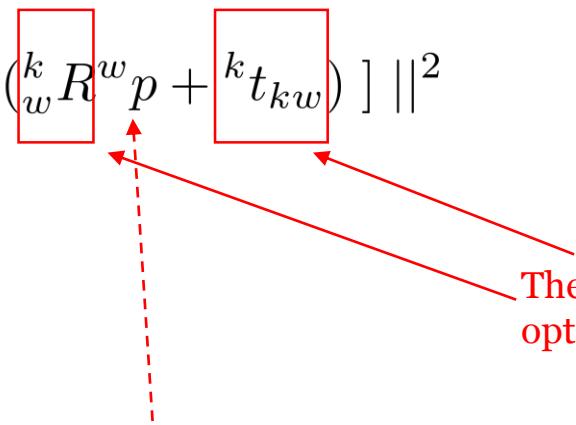
Camera frame  
transformations  
are unknown and to  
be estimated

Slide credit:  
Noah Snavely

# Camera localization as a least squares problem?

$${}^k_w R^*, {}^k t_{kw}^* = \underset{{}^k_w R, {}^k t_{kw}}{\operatorname{argmin}}$$

$$\sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k_w R^w p + {}^k t_{kw}) ] \|^2$$

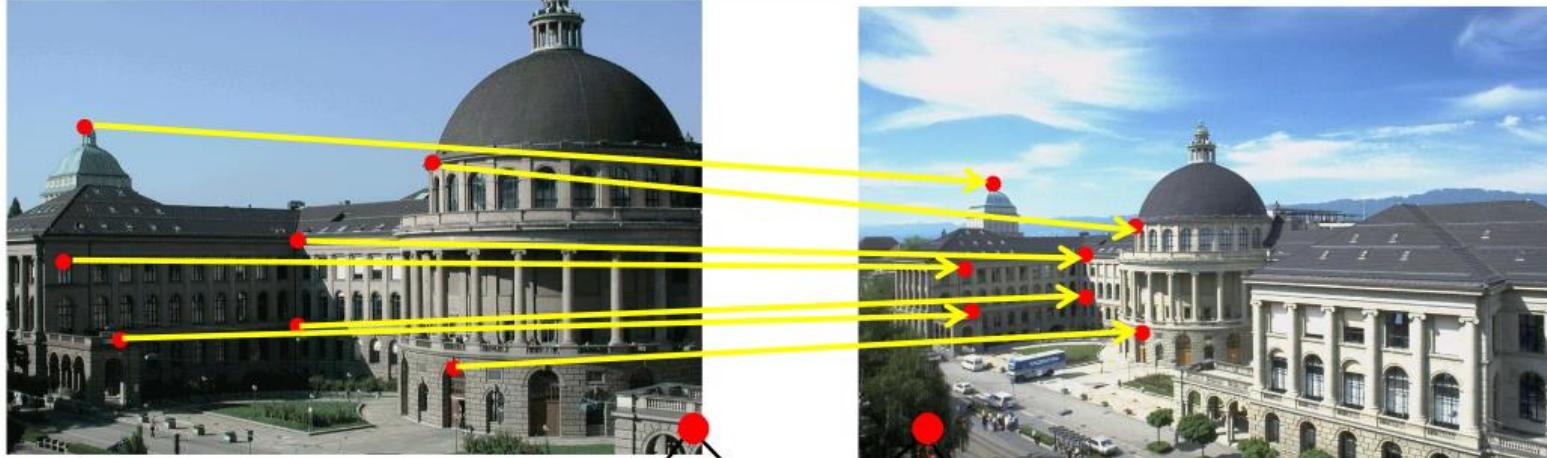


But, 3D position is unknown!

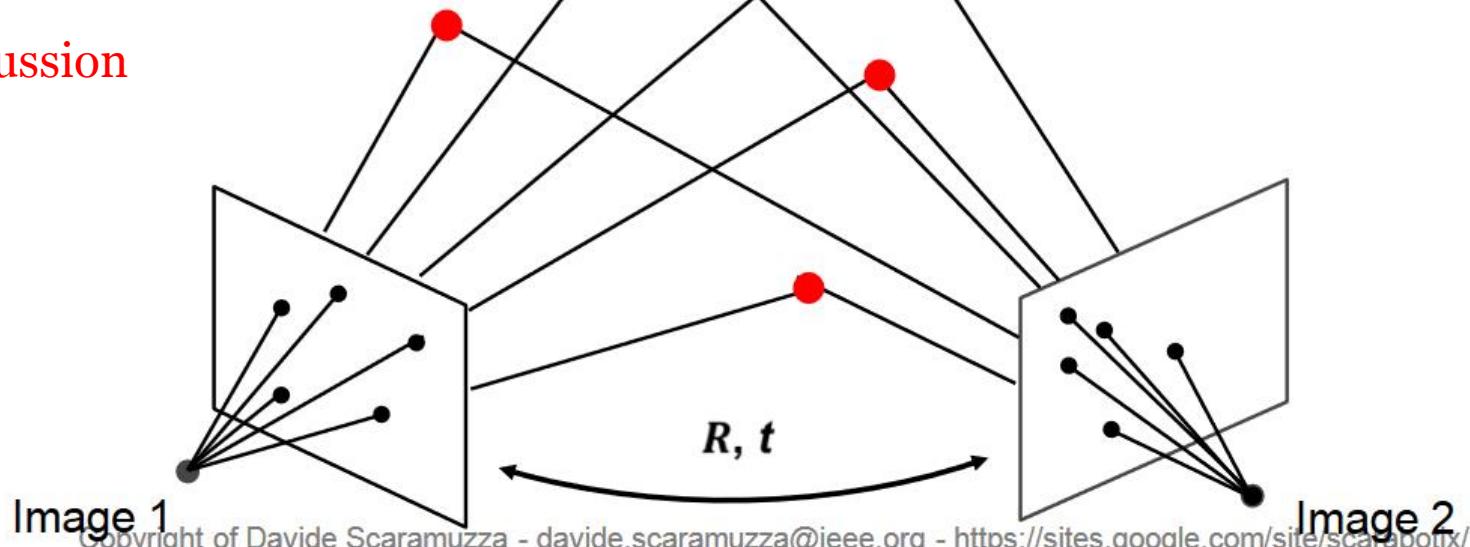
So, we cannot solve the problem using the reprojection error unless we know the 3D position corresponding to the keypoint.

$$h_{\text{pinhole}}(X, Y, Z) = \begin{bmatrix} \frac{f m_x X}{Z} + c_x \\ \frac{f m_y Y}{Z} + c_y \end{bmatrix} + n, \quad n \sim \mathcal{N}(0, R) \quad \text{noise (in pixels)}$$

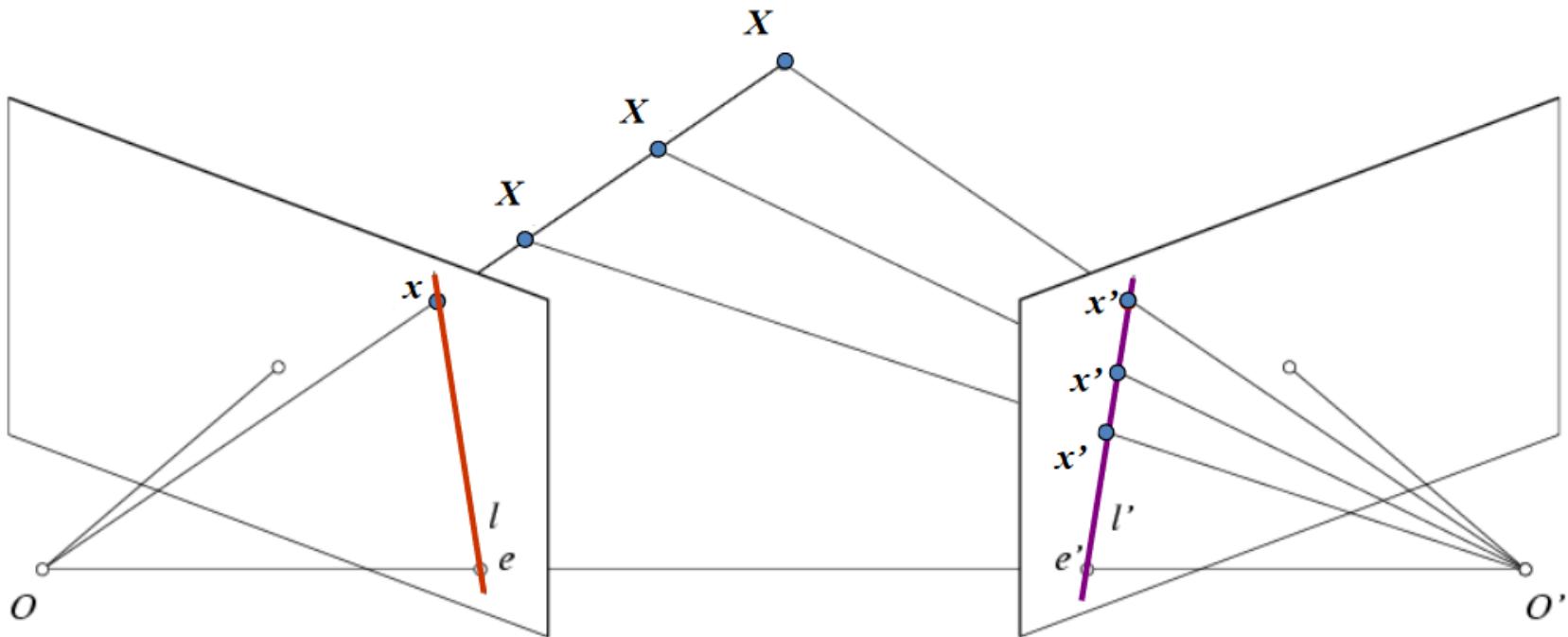
# Working Principle



Let's restrict the discussion  
to two cameras only



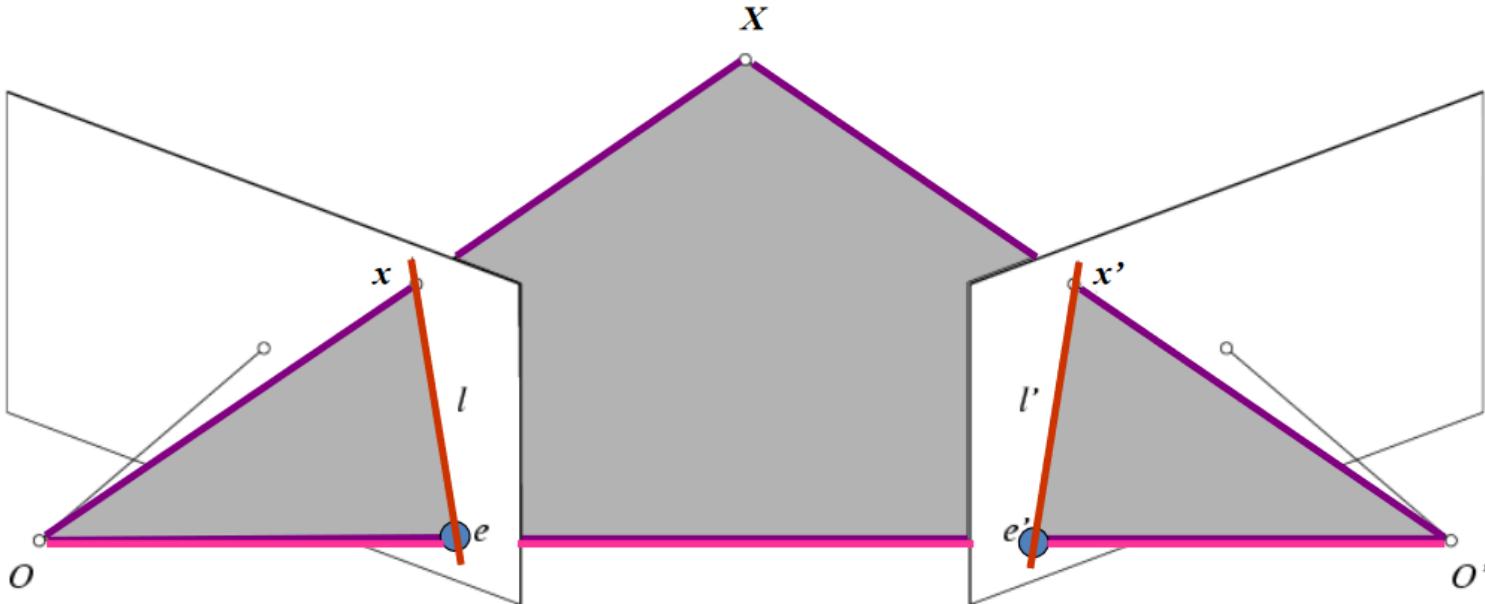
# Key idea: Epipolar constraint



Potential matches for  $x'$  have to lie on the corresponding line  $l$ .

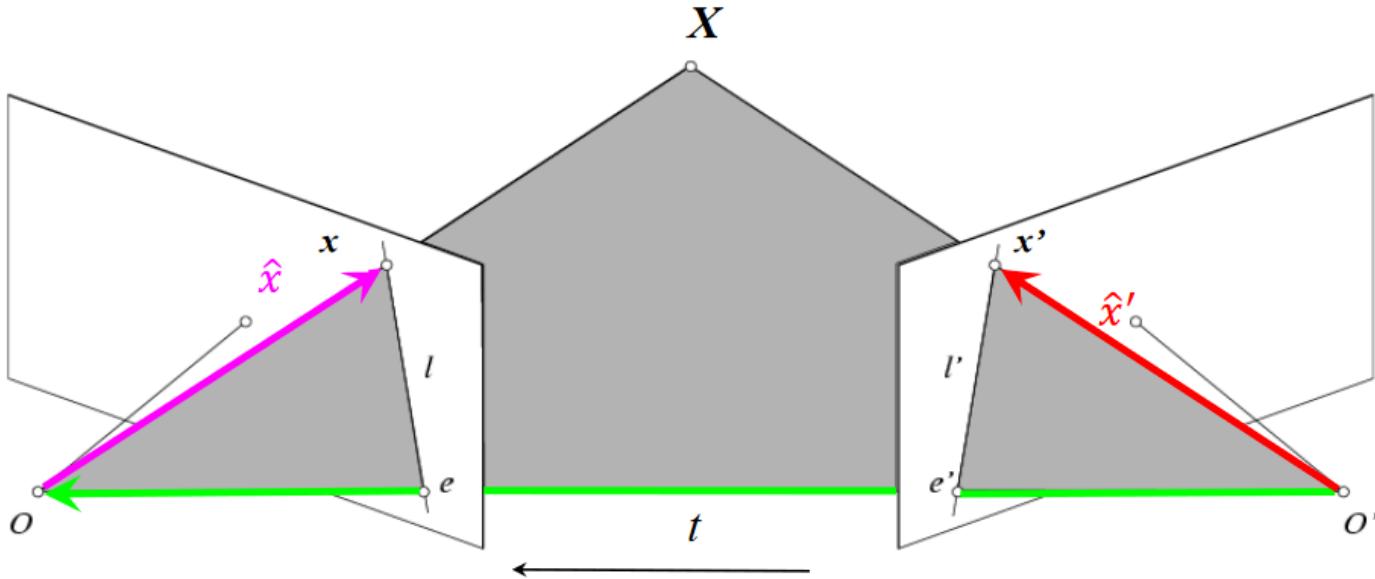
Potential matches for  $x$  have to lie on the corresponding line  $l'$ .

# Epipolar geometry: notation



- **Baseline** – line connecting the two camera centers
- **Epipoles**
  - = intersections of baseline with image planes
  - = projections of the other camera center
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipolar Lines** - intersections of epipolar plane with image planes (always come in corresponding pairs)

# Epipolar constraint: Calibrated case



$$\hat{x} = K^{-1}x = X \quad \hat{x}' = K'^{-1}x' = X'$$

Homogeneous 2d point  
(3D ray towards  $X$ )

2D pixel coordinate  
(homogeneous)

3D scene point

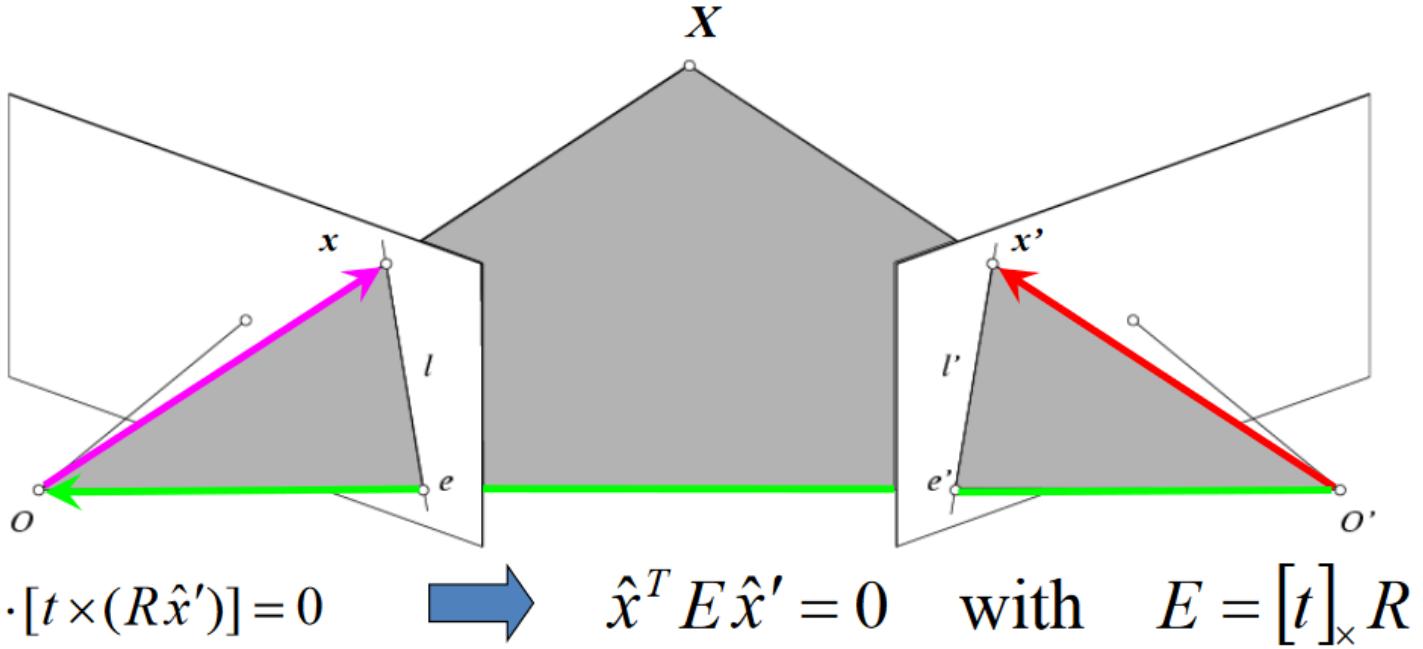
3D scene point in 2<sup>nd</sup>  
camera's 3D coordinates

$$\hat{x} \cdot [t \times (R\hat{x}')] = 0$$

(because  $\hat{x}, R\hat{x}'$ , and  $t$  are co-planar)

# Essential matrix

“5-point algorithm” by David Nister computes essential matrix and then decomposes it into rotation and translation.



$E$  is a  $3 \times 3$  matrix which relates corresponding pairs of normalized homogeneous image points across pairs of images – for  $K$  calibrated cameras.

*Estimates relative position/orientation.*

**Essential Matrix**  
(Longuet-Higgins, 1981)

Note:  $[t]_x$  is matrix representation of cross product

$$[t]_x = \begin{bmatrix} 0 & -t_2 & t_1 \\ t_2 & 0 & -t_0 \\ -t_1 & t_0 & 0 \end{bmatrix}$$

After estimating the essential matrix, we extract  $t$ ,  $R$ .

However, the translation  $t$ , is only estimated up to a multiplicative scale.

→ Translation is not fully observable with a single camera.

→ To make it observable we need stereo

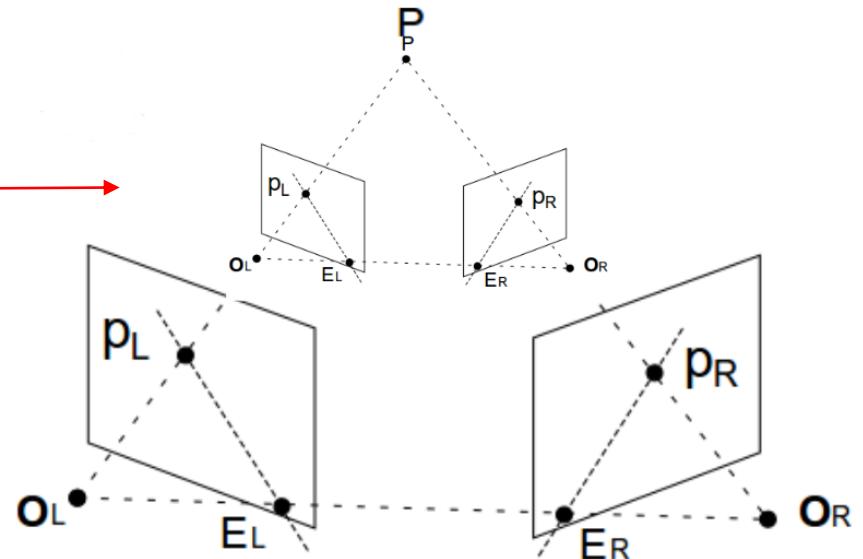
# Visual odometry with a single camera: translation is recovered only up to a scale

- Scale = relationship between real-world metric distance units and estimated map distance units

Camera placements (1) and (2) generate the same observation of  $P$ . In fact, infinitely many possible placements of the two camera frames along their projection rays could have generated the same measurement.

(1)

(2)

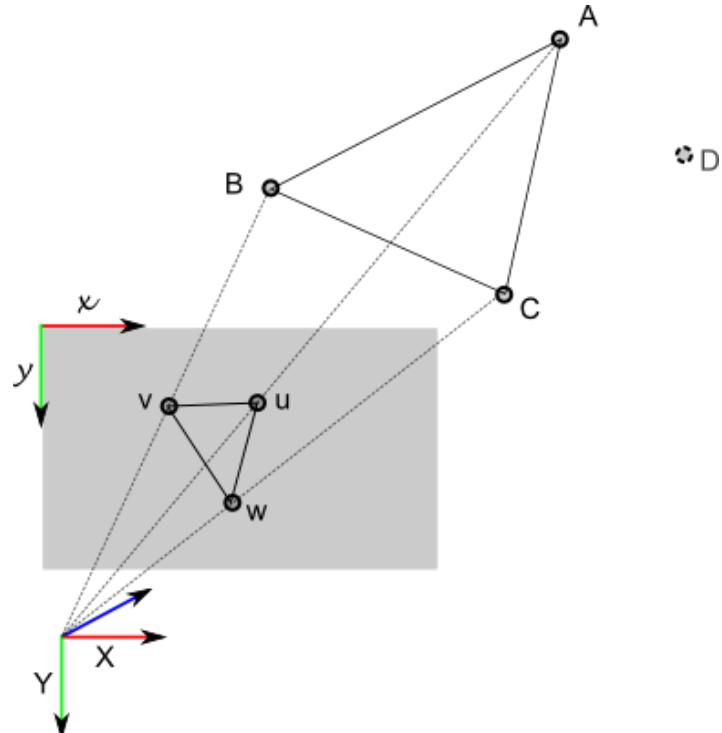


# Visual odometry with a single camera: translation is recovered only up to a scale

- Scale = relationship between real-world metric distance units and estimated map distance units

Q: Is there a way to obtain true metric distances only with a single camera?

A: The only way is to have an object of known metric dimensions in the observed scene. For example if you know distances AB, BC, CA then you can recover true translation. This is commonly referred to as the Perspective-3-Point (P3P), or in General, the Perspective-n-Point (PnP) problem.

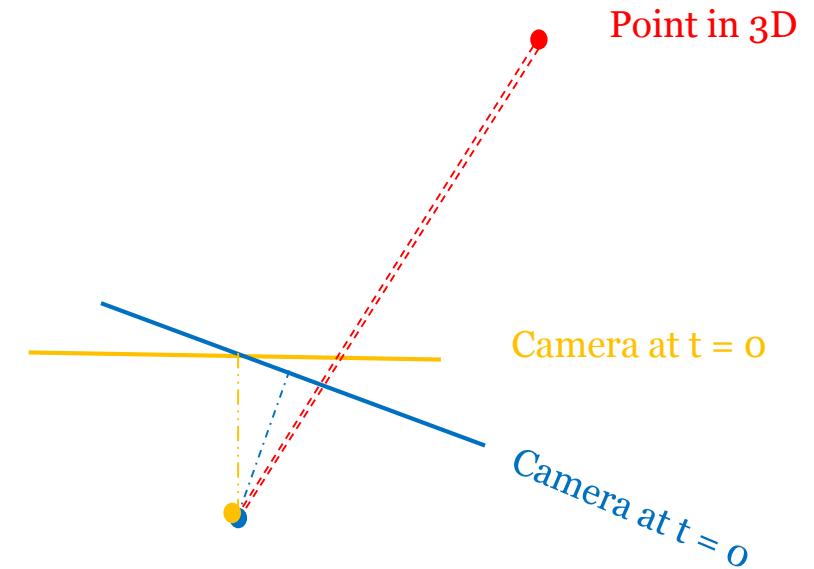


# Visual odometry with a single camera: translation is recovered only up to a scale

- Scale = relationship between real-world metric distance units and estimated map distance units

Q: Does scale remain constant throughout the trajectory of a single camera?

A: No, there is **scale drift**, which is most apparent during in-place rotations (i.e. pure rotation, no translation), because depth estimation for 3D points is unconstrained, so it is easily misestimated.



# 2D-to-2D Algorithm

Motion estimation		
2D-2D	3D-3D	3D-2D

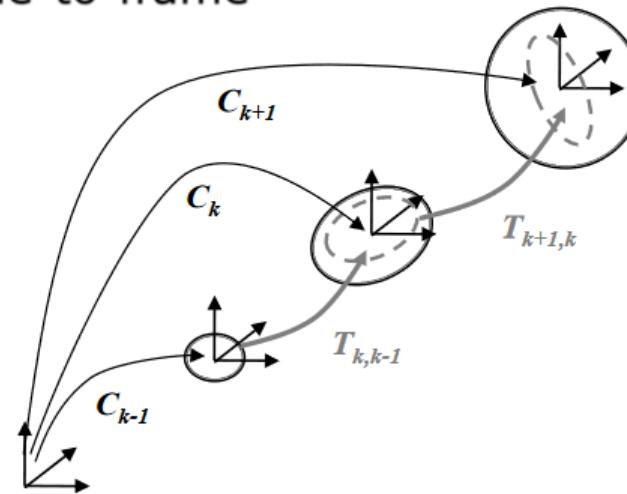
---

## Algorithm 1: VO from 2D-to-2D correspondences

- 1 Capture new frame  $I_k$
- 2 Extract and match features between  $I_{k-1}$  and  $I_k$ ,
- 3 Compute essential matrix for image pair  $I_{k-1}, I_k$
- 4 Decompose essential matrix into  $R_k$  and  $t_k$ , and form  $T_k$
- 5 Compute relative scale and rescale  $t_k$  accordingly
- 6 Concatenate transformation by computing  $C_k = C_{k-1}T_k$
- 7 Repeat from 1

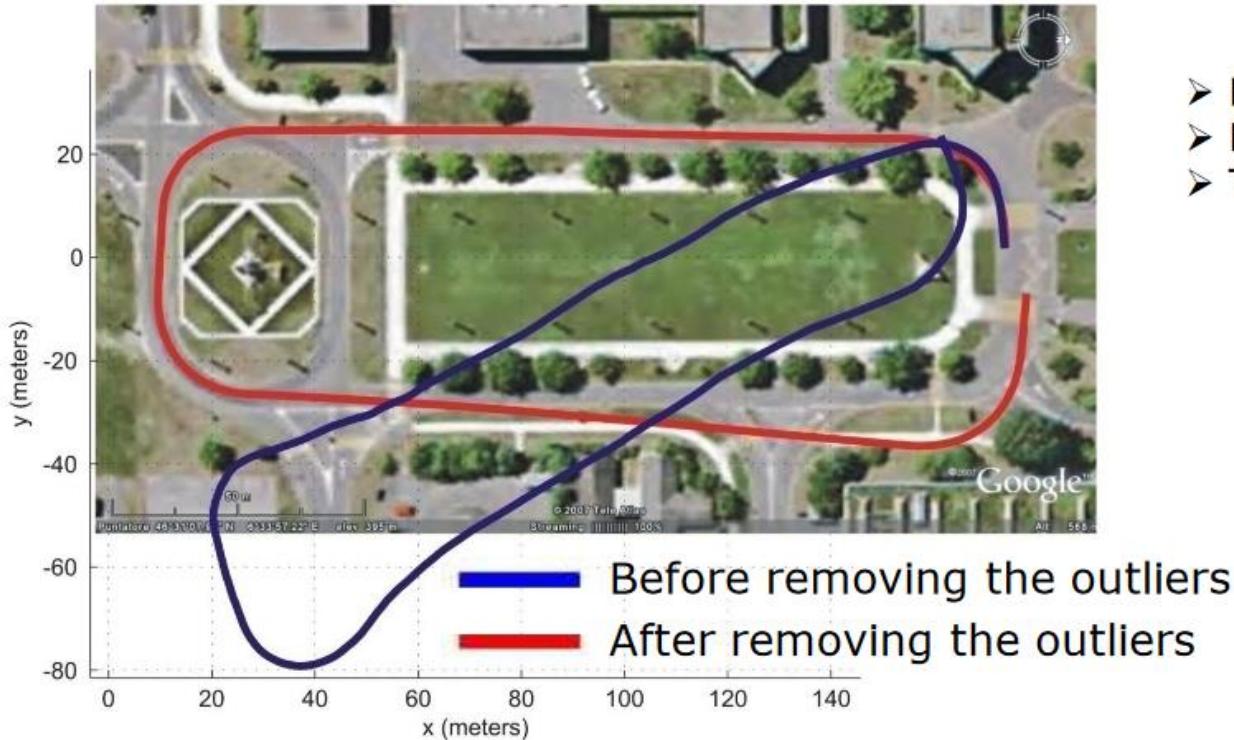
# VO Drift

- The errors introduced by each new frame-to-frame motion accumulate over time
- This generates a drift of the estimated trajectory from the real path



The uncertainty of the camera pose at  $C_k$  is a combination of the uncertainty at  $C_{k-1}$  (black solid ellipse) and the uncertainty of the transformation  $T_{k,k-1}$  (gray dashed ellipse)

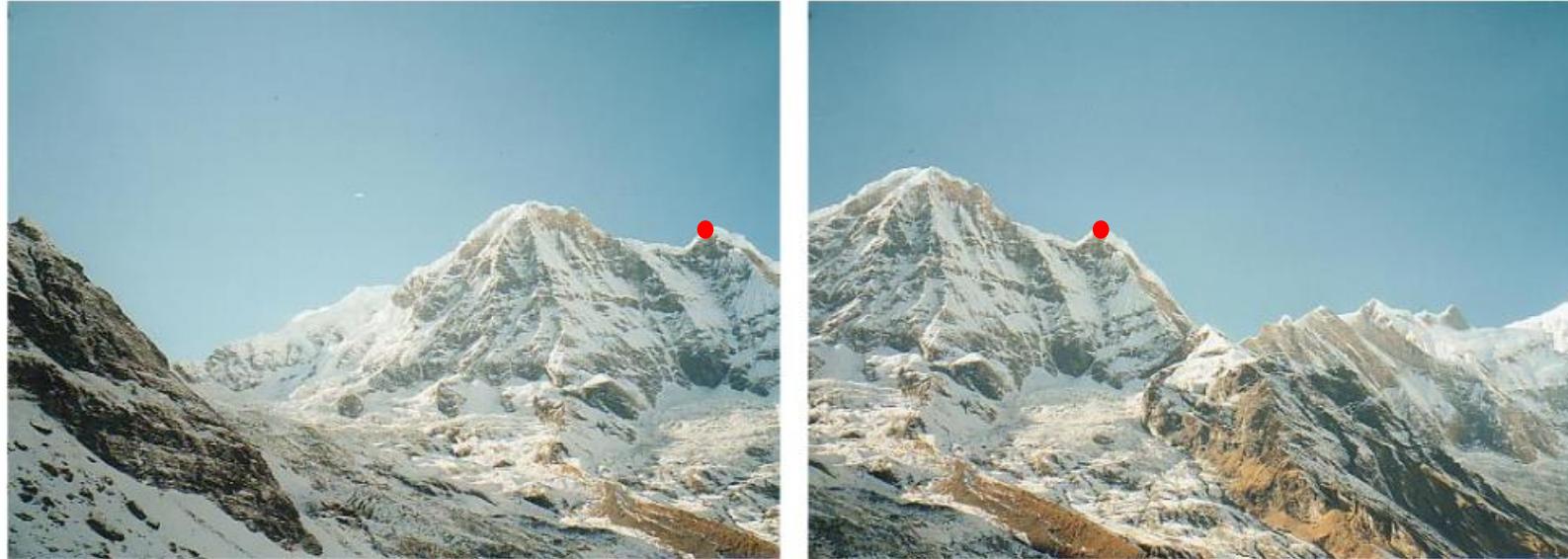
# Influence of Outliers on Motion Estimation



- Error at the loop closure: 6.5 m
- Error in orientation: 5 deg
- Trajectory length: 400 m

# Are points-at-infinity useful for localization?

Points-at-infinity can help estimate the camera's rotation, similarly to how we use stars for navigation, without estimating how far they are.



For estimating translation, most likely no. For estimating rotation, yes.  
Look up “Inverse Depth Parameterization for Monocular SLAM” for more info.

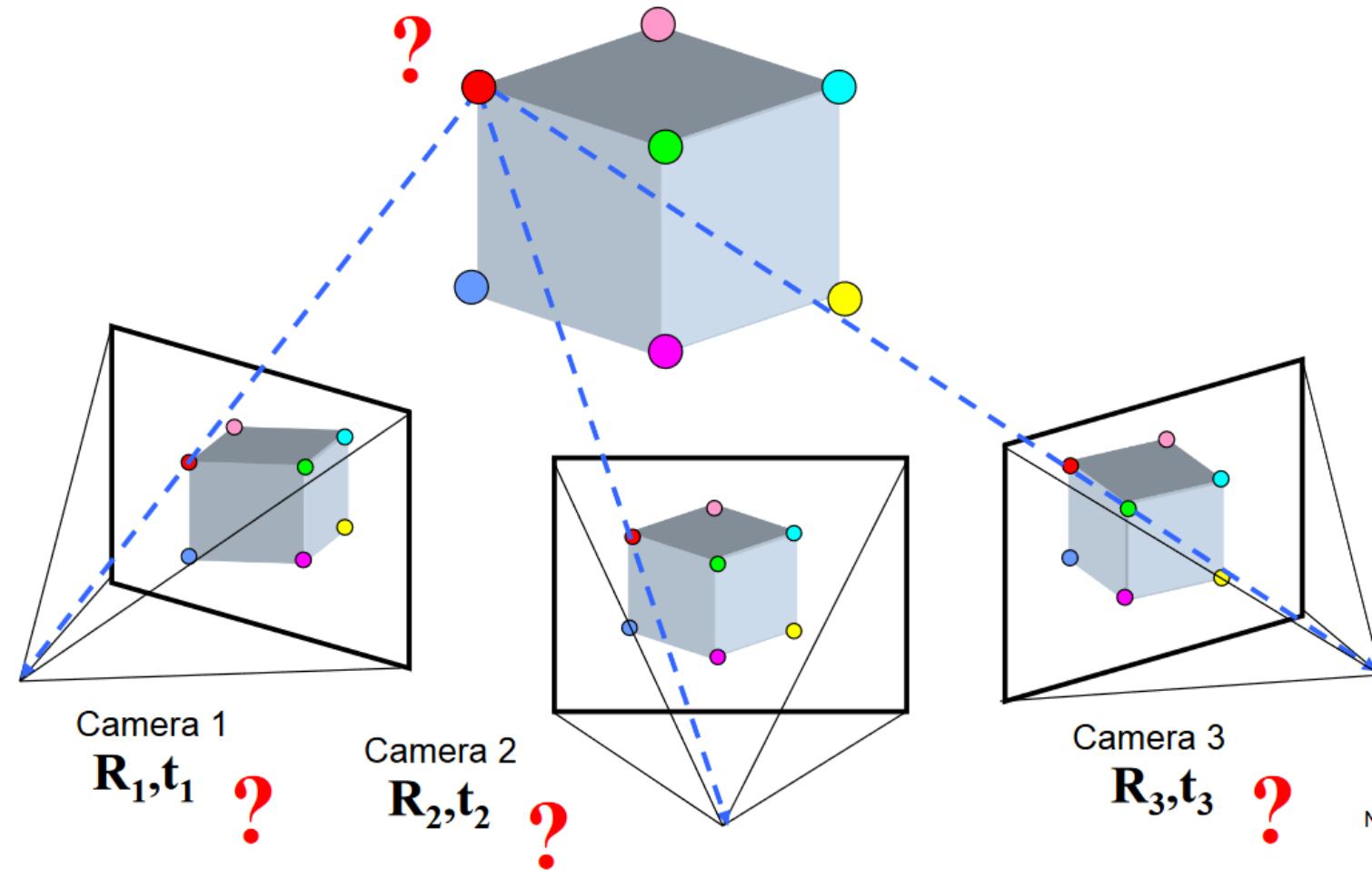
# Problem #3: Visual SLAM

# Structure from Motion

How can we estimate both 3D point positions and the relative camera transformations?

Sometimes also  
called bundle  
adjustment

Q: Why is it different than  
SLAM?



Slide credit:  
Noah Snavely

# Structure from Motion

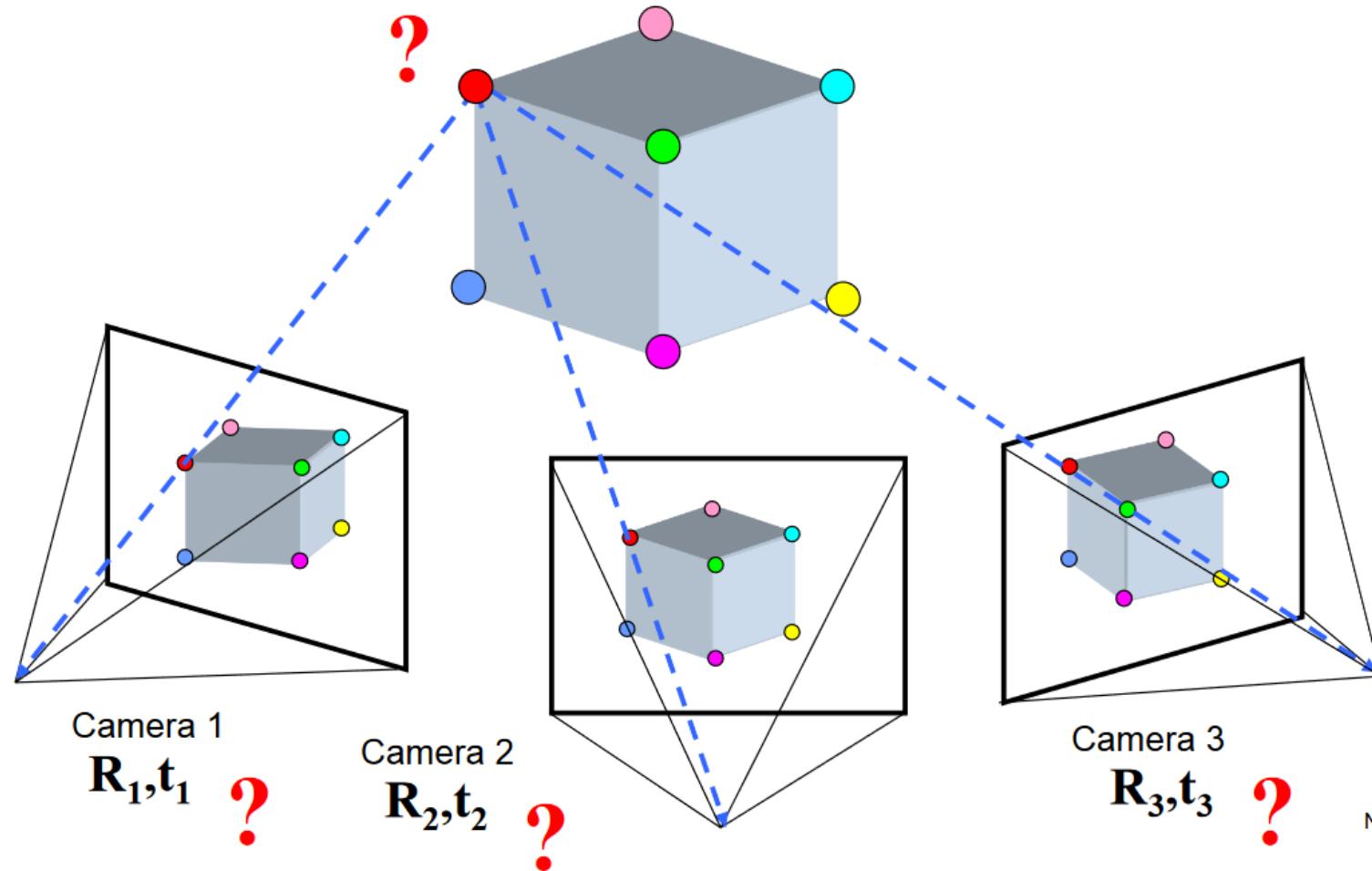
How can we estimate both 3D point positions and the relative camera transformations?

Sometimes also  
called bundle  
adjustment

Q: Why is it different than  
SLAM?

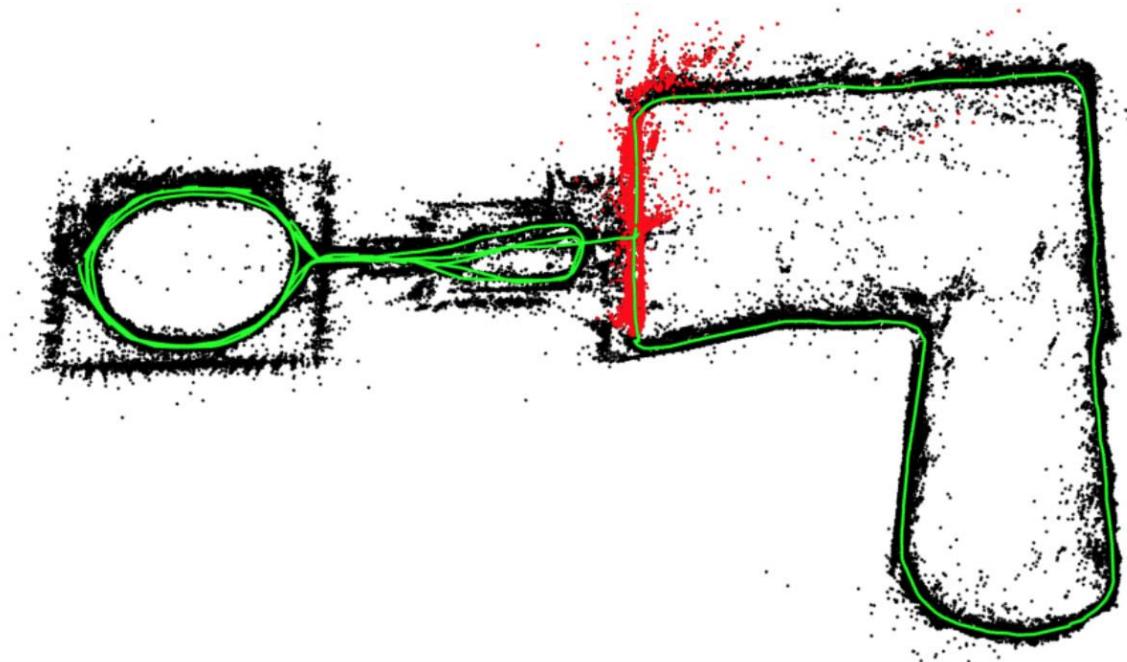
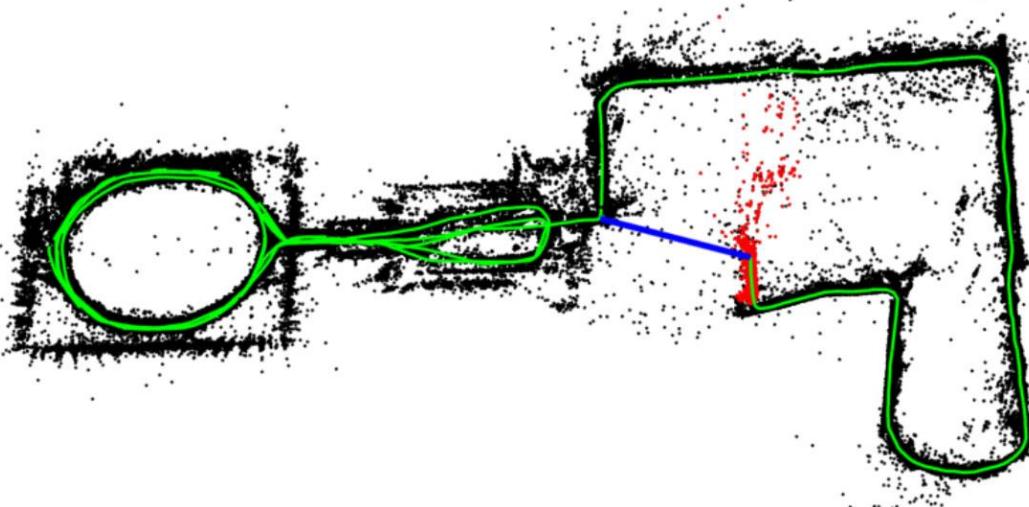
A: SLAM potentially  
includes

- loop closure
- dynamics constraints
- velocities, accelerations



Slide credit:  
Noah Snavely

# Loop Closure in Visual SLAM



Bundler (bundle adjustment/structure from motion)



# Structure from Motion as Least Squares

$$\begin{bmatrix} {}^k_w R^*, {}^k_w t_{kw}^* \end{bmatrix}, \begin{bmatrix} {}^w X^*, {}^w Y^*, {}^w Z^* \end{bmatrix} = \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z], {}^k_w R, {}^k_w t_{kw}}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k_w R {}^w p + {}^k_w t_{kw})] \|^2$$

Indicates the frame of the k-th camera.

$$h_{\text{pinhole}}(X, Y, Z) = \begin{bmatrix} \frac{f m_x X}{Z} + c_x \\ \frac{f m_y Y}{Z} + c_y \end{bmatrix} + n, \quad n \sim \mathcal{N}(0, R) \quad \text{noise (in pixels)}$$

# Structure from Motion as Least Squares

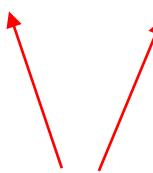
$${}^k_w R^*, {}^k_w t_{kw}^*, {}^w X^*, {}^w Y^*, {}^w Z^* = \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z], {}^k_w R, {}^k_w t_{kw}}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k_w R {}^w p + {}^k_w t_{kw}) ] \|^2$$

Actual pixel measurement of  
3D point  ${}^w p$  from camera k

Expected pixel projection of  
3D point  ${}^w p$  onto camera k

$$h_{\text{pinhole}}(X, Y, Z) = \begin{bmatrix} \frac{f m_x X}{Z} + c_x \\ \frac{f m_y Y}{Z} + c_y \end{bmatrix} + n, \quad n \sim \mathcal{N}(0, R) \quad \text{noise (in pixels)}$$

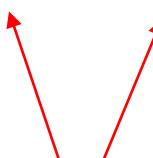
# Structure from Motion as Least Squares

$${}^k_w R^*, {}^k t_{kw}^*, [{}^w X^*, {}^w Y^*, {}^w Z^*] = \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z], {}^k w R, {}^k t_{kw}}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k_w R^{} w p + {}^k t_{kw}) ] \|^2$$


Q: Is the scale of these two estimates accurate/unambiguous when measurements are done from a monocular (single) camera in motion? I.e. is it observable?

Note: **scale** = relationship between real-world metric distances and estimated map distances. I.e. relationship between distance units.

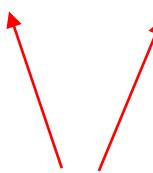
# Structure from Motion as Least Squares

$${}^k_w R^*, {}^k_w t_{kw}^*, [{}^w X^*, {}^w Y^*, {}^w Z^*] = \operatorname{argmin}_{{}^w p = [{}^w X, {}^w Y, {}^w Z], {}^k_w R, {}^k_w t_{kw}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[h_{\text{pinhole}}({}^k_w R {}^w p + {}^k_w t_{kw})] \|^2$$


Q: Is the scale of these two estimates accurate/unambiguous when measurements are done from a monocular (single) camera in motion? I.e. is it observable?

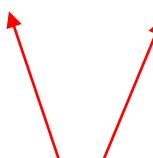
A: No, regardless of how many common keypoints are matched in between camera frames. Without external reference distance, e.g. stereo baseline, or real size of observed object, the scale is ambiguous and unobservable, just as it was in Visual Odometry.

# Structure from Motion as Least Squares

$${}^k_w R^*, {}^k_w t_{kw}^*, [{}^w X^*, {}^w Y^*, {}^w Z^*] = \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z], {}^k_w R, {}^k_w t_{kw}}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k_w R {}^w p + {}^k_w t_{kw}) ] \|^2$$


Q: Is the scale of these two estimates constant during the entire experiment, if we use a monocular camera?

# Structure from Motion as Least Squares

$${}^k_w R^*, {}^k_w t_{kw}^*, [{}^w X^*, {}^w Y^*, {}^w Z^*] = \underset{{}^w p = [{}^w X, {}^w Y, {}^w Z], {}^k_w R, {}^k_w t_{kw}}{\operatorname{argmin}} \sum_{k=1}^K \| \bar{z}^{(k)} - \mathbb{E}[ h_{\text{pinhole}}({}^k_w R {}^w p + {}^k_w t_{kw}) ] \|^2$$


Q: Is the scale of these two estimates constant during the entire experiment, if we use a monocular camera?

A: No. During in-place rotations there is not enough baseline between camera frames to triangulate new points. So, error in structure and in motion accumulates → **scale drift**

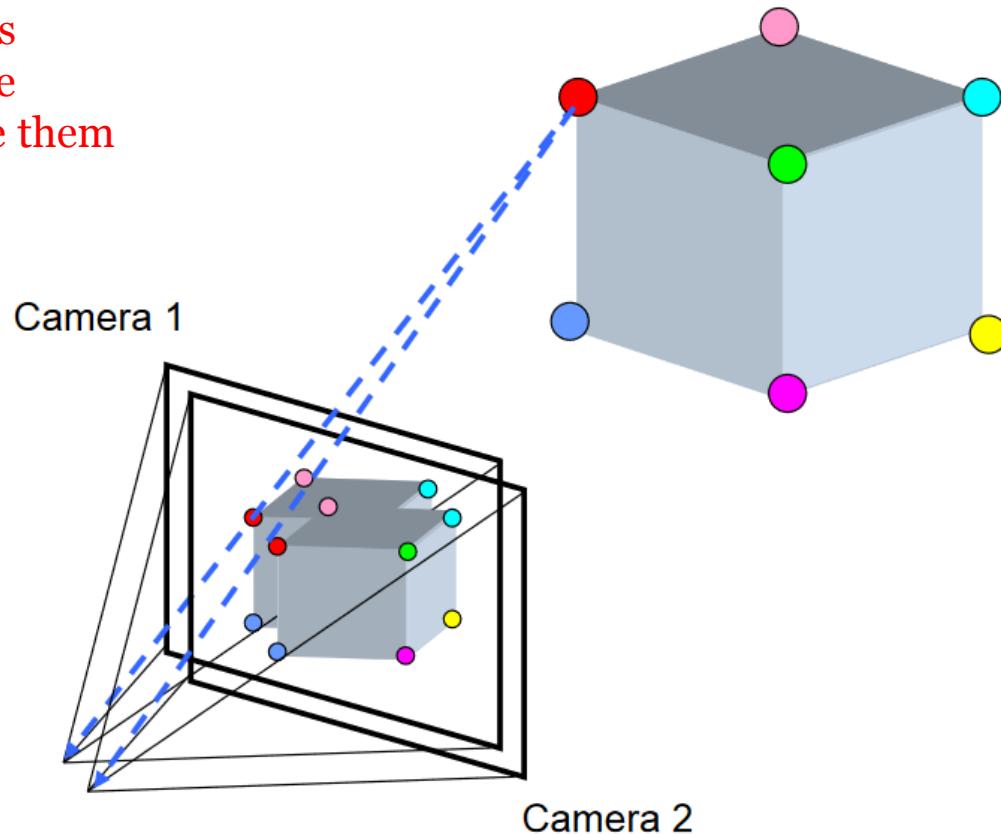
Similarly to visual odometry with a single camera.

# Problem #4: optical flow

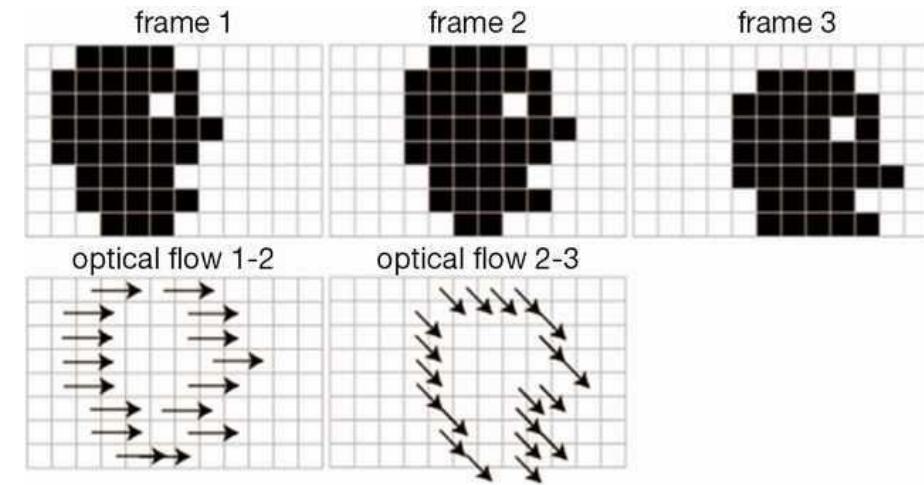
# Multi-view geometry problems

- **Optical flow:** Given two images, find the location of a world point in a second close-by image with no camera info.

3D point coordinates  
are unknown, but we  
won't try to estimate them



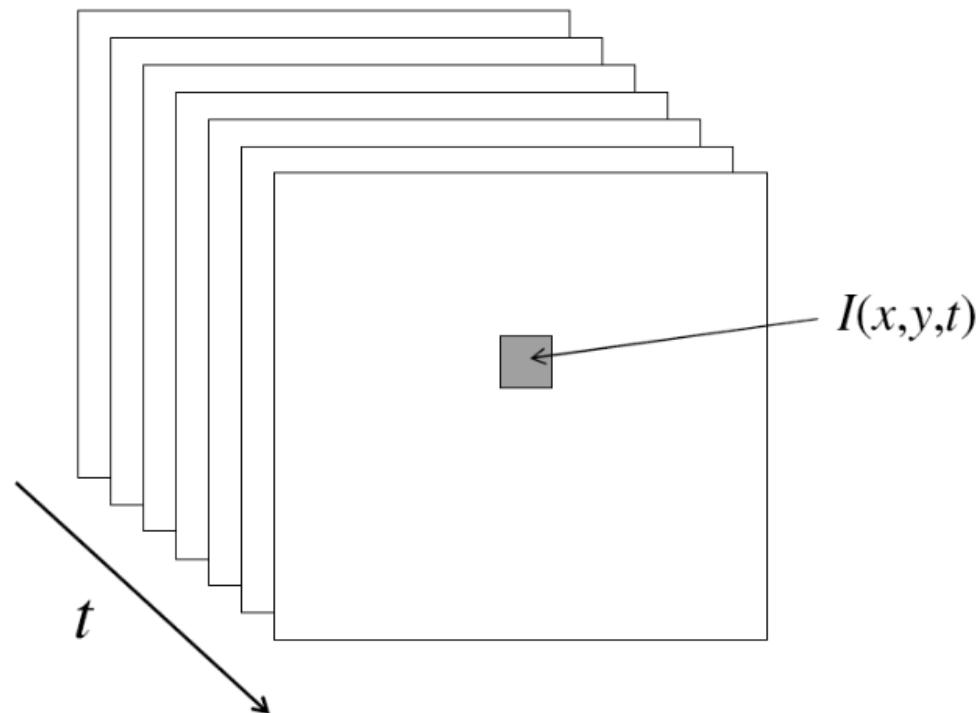
We are estimating pixel displacement  
from one image to the next



Camera frame  
transformations  
are unknown, but we  
won't try to estimate them

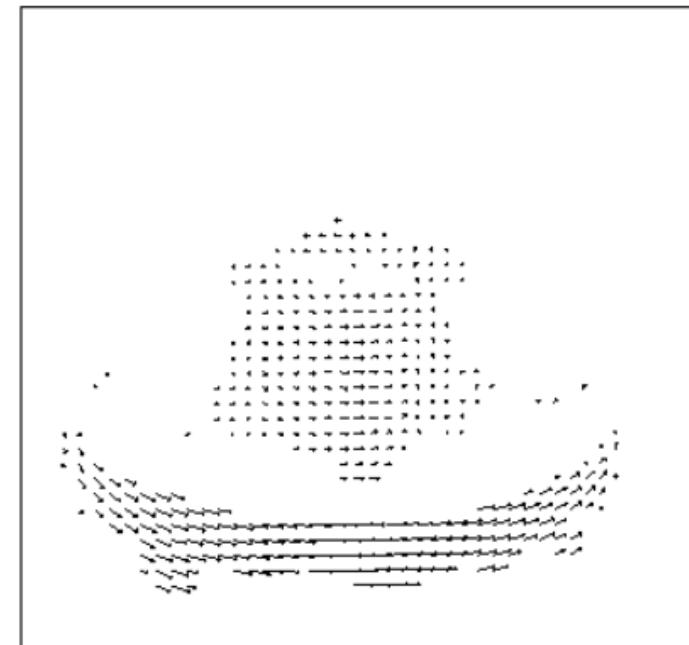
# Video

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



# Motion estimation: Optical flow

*Optic flow* is the **apparent** motion of objects or surfaces



Will start by estimating motion of each pixel separately  
Then will consider motion of entire image

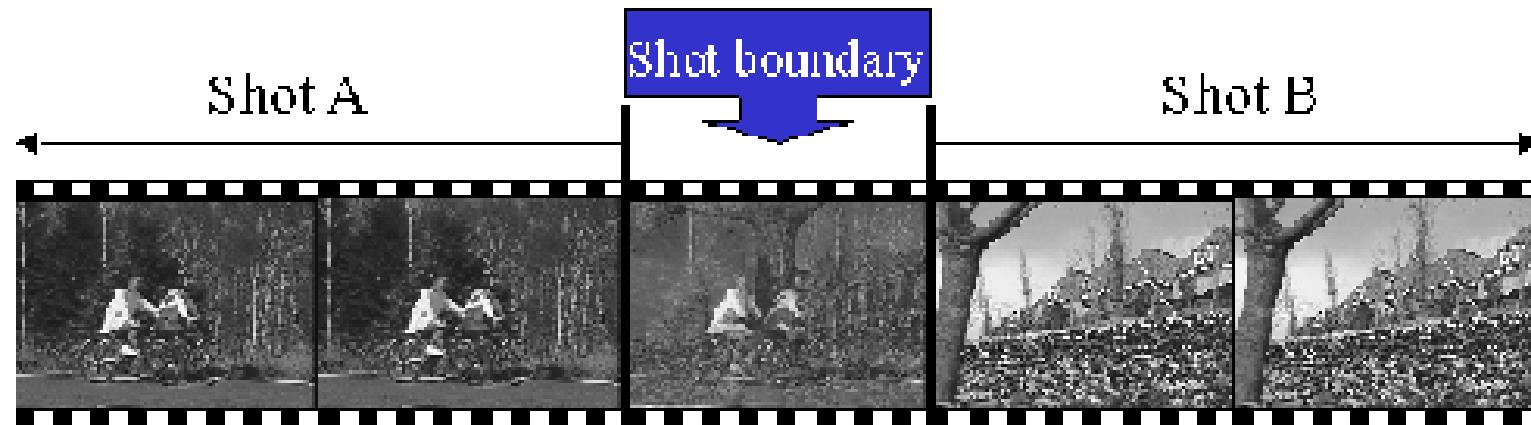
# Motion Applications: Segmentation of video

- Background subtraction
  - A static camera is observing a scene
  - Goal: separate the static *background* from the moving *foreground*



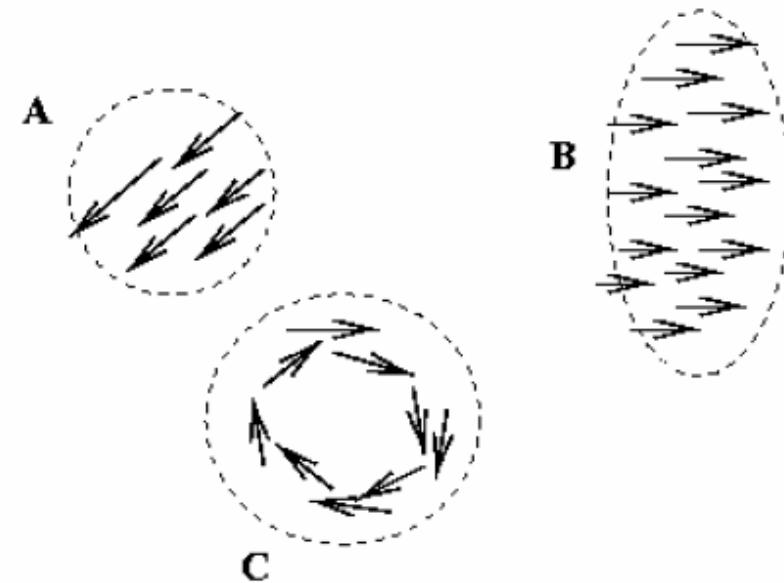
# Motion Applications: Segmentation of video

- Shot boundary detection in edited video
  - Edited video is usually composed of *shots* or sequences showing the same objects or scene
  - Goal: segment video into shots for summarization and browsing (each shot can be represented by a single keyframe in a user interface)
  - Difference from background subtraction: the camera is not necessarily stationary



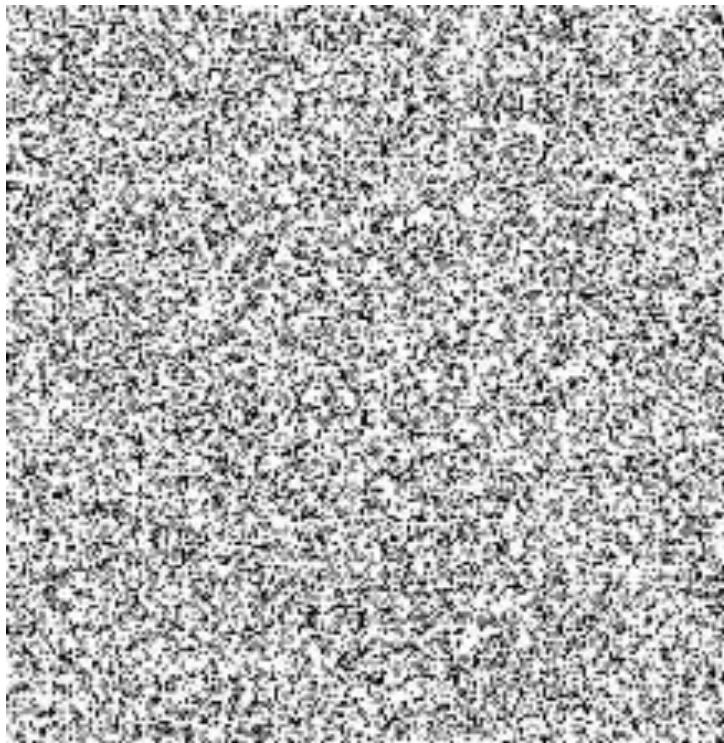
# Motion Applications: Segmentation of video

- Background subtraction
- Shot boundary detection
- Motion segmentation
  - Segment the video into multiple coherently moving objects



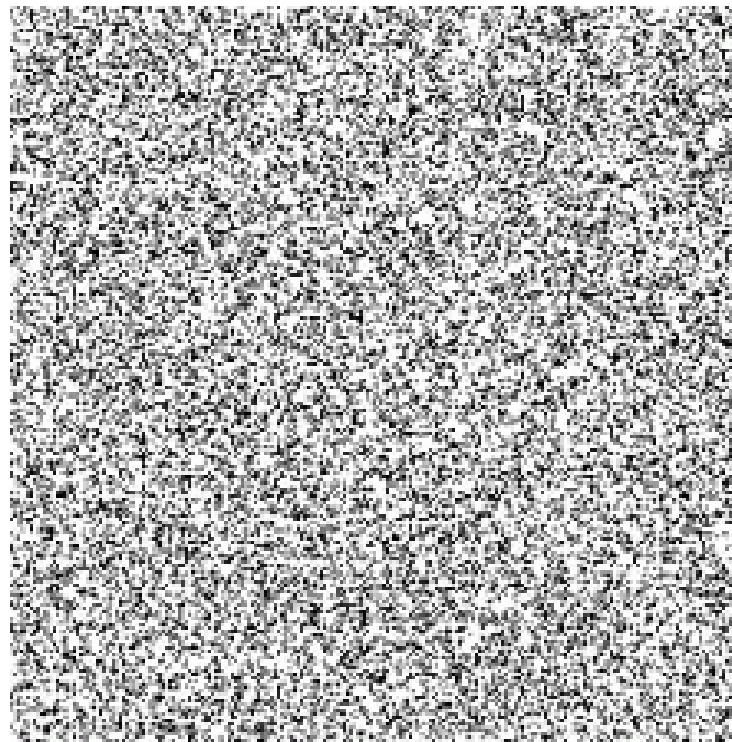
# Motion and perceptual organization

- Sometimes, motion is the only cue



# Motion and perceptual organization

- Sometimes, motion is the only cue

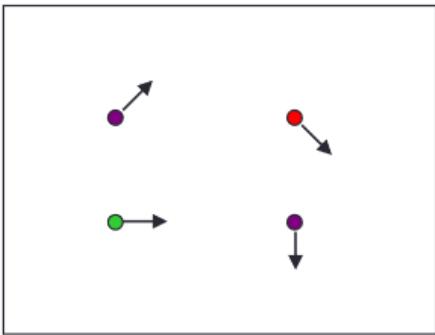
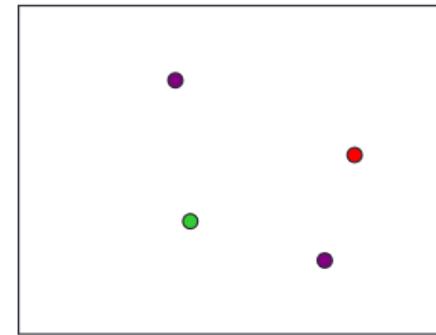


# Motion and perceptual organization



Experimental study of apparent behavior.  
Fritz Heider & Marianne Simmel. 1944

# Problem definition: optical flow

 $I(x, y, t)$  $I(x, y, t + 1)$ 

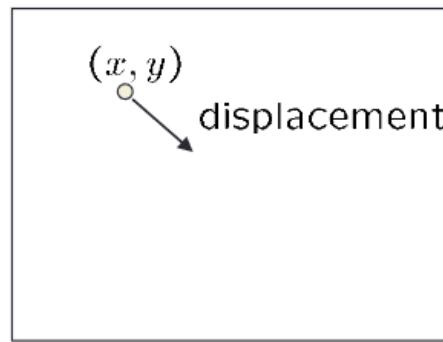
How to estimate pixel motion from image  $I(x, y, t)$  to  $I(x, y, t+1)$  ?

- Solve pixel correspondence problem
  - Given a pixel in  $I(x, y, t)$ , look for **nearby** pixels of the **same color** in  $I(x, y, t+1)$

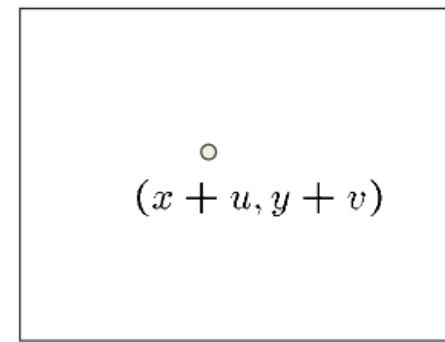
Key assumptions

- **Small motion**: Points do not move very far
- **Color constancy**: A point in  $I(x, y, t)$  looks the same in  $I(x, y, t+1)$ 
  - For grayscale images, this is brightness constancy

# Optical flow constraints (grayscale images)



$$I(x, y, t)$$



$$I(x, y, t + 1)$$

- Let's look at these constraints more closely
  - Brightness constancy constraint (equation)
$$I(x, y, t) = I(x + u, y + v, t + 1)$$
  - Small motion: ( $u$  and  $v$  are less than 1 pixel, or smooth)

From Taylor expansion of  $I$

$$I(x + u, y + v, t + 1) = I(x, y, t + 1) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

# Optical flow equation

- Combining these two equations

$$\begin{aligned} 0 &= I(x+u, y+v, t+1) - I(x, y, t) \\ &\approx I(x, y, t+1) + I_x u + I_y v - I(x, y, t) \end{aligned}$$

(Short hand:  $I_x = \frac{\partial I}{\partial x}$   
for  $t$  or  $t+1$ )

## Optical flow equation

- Combining these two equations

$$\begin{aligned} 0 &= I(x+u, y+v, t+1) - I(x, y, t) \\ &\approx I(x, y, t+1) + I_x u + I_y v - I(x, y, t) \\ &\approx [I(x, y, t+1) - I(x, y, t)] + I_x u + I_y v \\ &\approx I_t + I_x u + I_y v \end{aligned}$$

(Short hand:  $I_x = \frac{\partial I}{\partial x}$   
for  $t$  or  $t+1$ )

## Optical flow equation

- Combining these two equations

$$0 = I(x+u, y+v, t+1) - I(x, y, t)$$

$$\approx I(x, y, t+1) + I_x u + I_y v - I(x, y, t)$$

$$\approx [I(x, y, t+1) - I(x, y, t)] + I_x u + I_y v$$

$$\approx I_t + I_x u + I_y v$$

(Short hand:  $I_x = \frac{\partial I}{\partial x}$   
for  $t$  or  $t+1$ )

In the limit as  $u$  and  $v$  go to zero, this becomes exact

*Brightness constancy constraint equation*

$$I_x u + I_y v + I_t = 0$$

# The brightness constancy constraint

Can we use this equation to recover image motion  $(u, v)$  at each pixel?

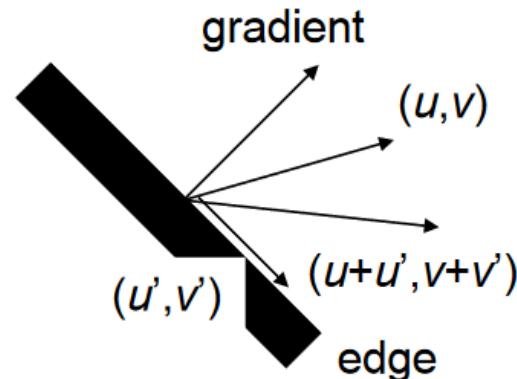
$$0 = I_t + \nabla I \cdot \langle u, v \rangle \quad \text{or} \quad I_x u + I_y v + I_t = 0$$

- How many equations and unknowns per pixel?
  - One equation (this is a scalar equation!), two unknowns  $(u, v)$

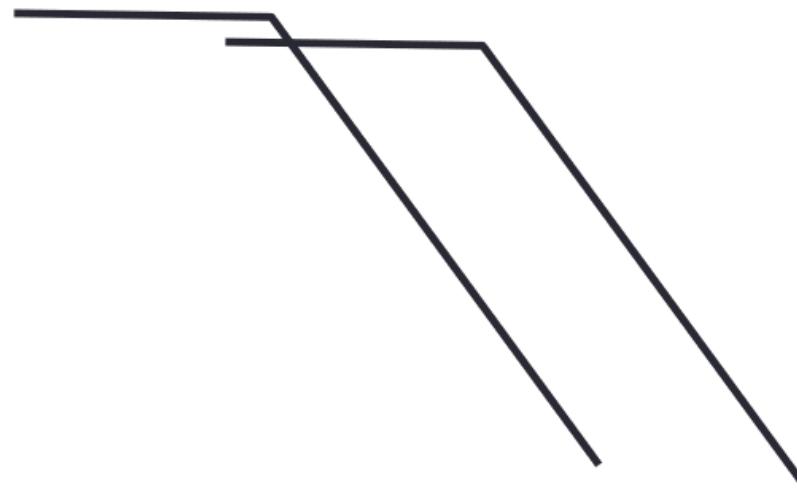
The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If  $(u, v)$  satisfies the equation,  
so does  $(u+u', v+v')$  if

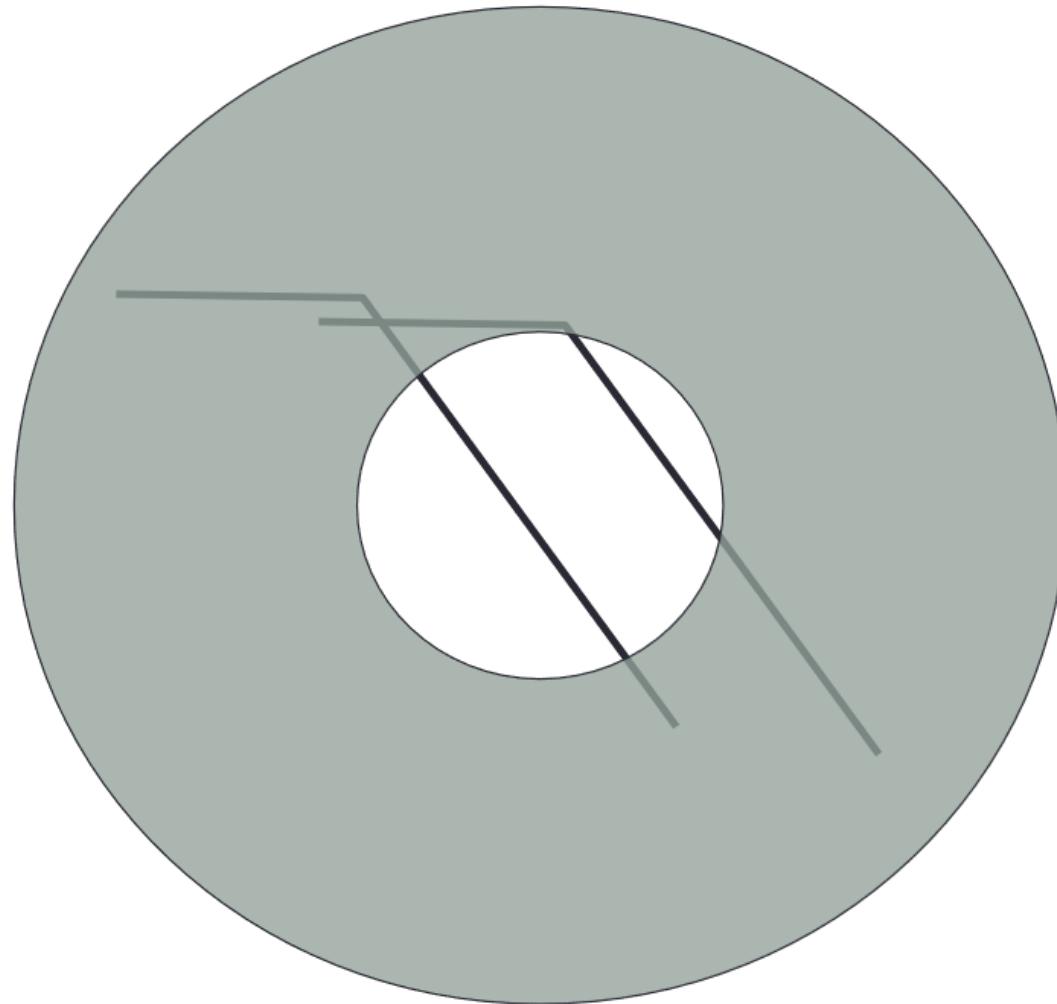
$$\nabla I \cdot [u' \ v']^T = 0$$



# Aperture problem



# Aperture problem



# Solving the ambiguity...

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.

- How to get more equations for a pixel?
- **Spatial coherence constraint**
- Assume the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

$$0 = I_t(\mathbf{p}_i) + \nabla I(\mathbf{p}_i) \cdot [u \ v]$$

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

## Solving the ambiguity...

- Least squares problem:

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix} \quad \begin{matrix} A & d = b \\ 25 \times 2 & 2 \times 1 & 25 \times 1 \end{matrix}$$