

CIS 5800

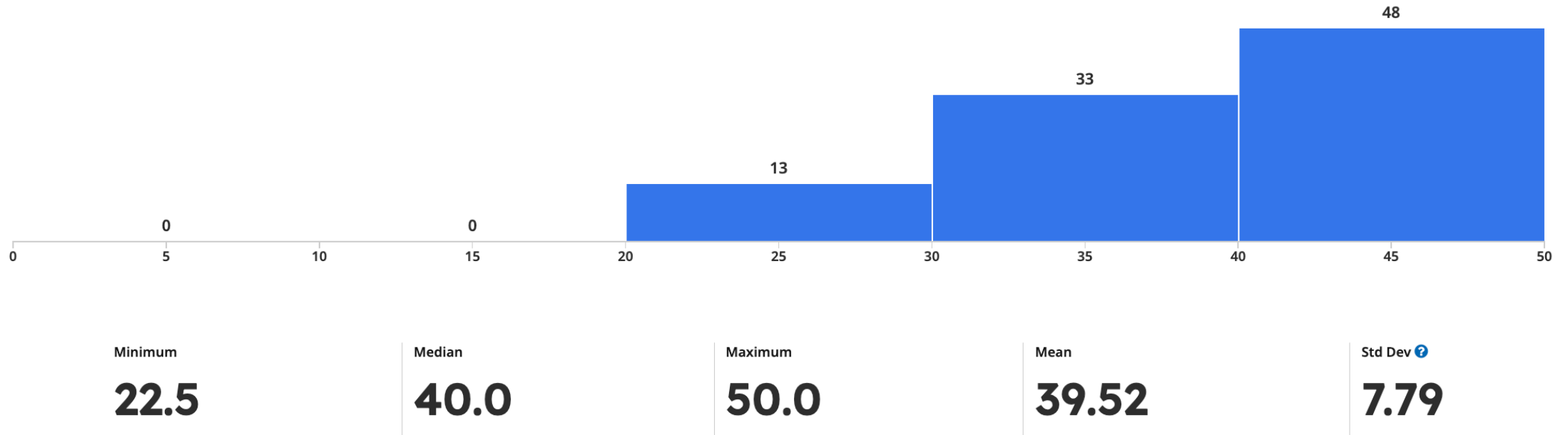
# Machine Perception

Instructor: Lingjie Liu

Lec 19: April 14, 2025

# Administrivia

Mid-term Exam (current scores).  
Q1.1 will be regraded.

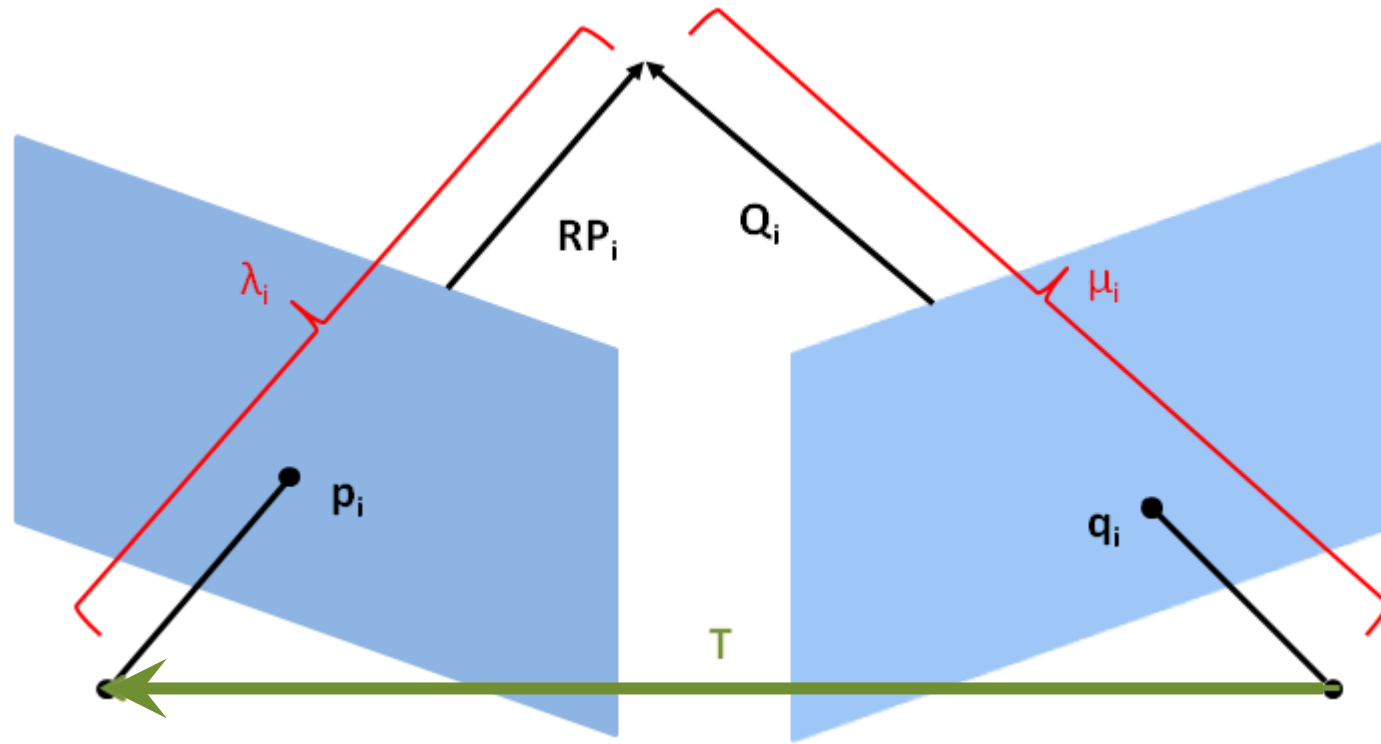


# Recap Two-view Reconstruction

With figures and text from mathworks.com

<https://www.mathworks.com/help/vision/ug/monocular-visual-simultaneous-localization-and-mapping.html>

# Two Calibrated Views of the Same 3D Scene



$$R(\lambda p) + T = \mu q$$

Given 2D correspondences  $(p, q)$

Find motion  $R, T$  and depths  $\lambda, \mu$ .

# PnP vs. 2-View Structure from Motion (SfM)

## PnP

1. Gn. “world frame” points  $P_i$  and corresponding calibrated coordinates  $K^{-1}x_i$
2. Set camera frame 3D coordinates (with scale/depth ambiguity) to  $\lambda_i K^{-1}x_i$
3. Then solve (for  $R, T, \lambda$ ):

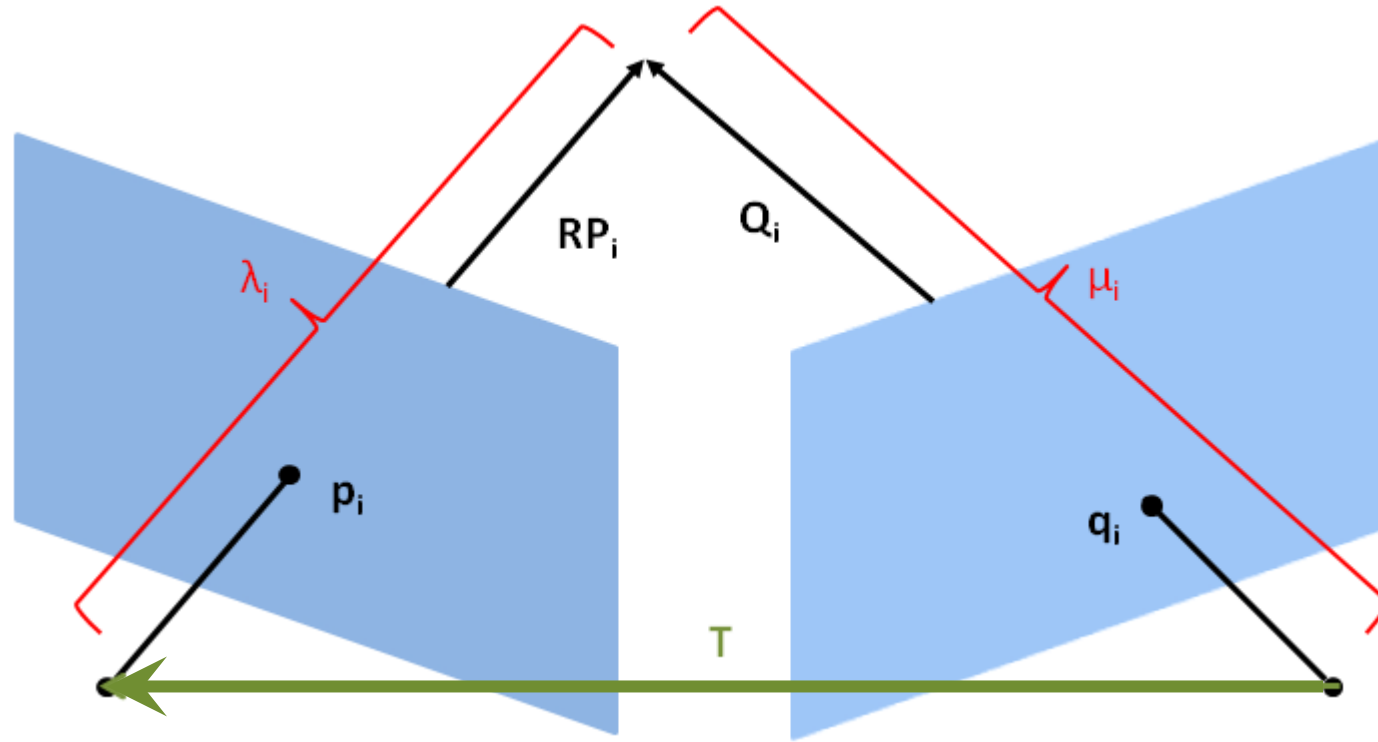
$$\lambda_i K^{-1}x_i = R P_i + T \quad \forall i$$

Where  $R, T$  denotes transformation between camera and world.

## Structure from Motion

1. No “world frame”. Instead, just calibrated image plane coordinates  $p_i = K_1^{-1}x_{i1}$  and  $q_i = K_2^{-1}x_{i2}$  **of the same 3D point  $P_i$ .**
2. So corresponding camera frame 3D coordinates in the two frames are:  $\lambda_i p_i$  and  $\mu_i q_i$ .
3. Now solve for “motion between cameras”  $R, T$  and the scales  $\lambda_i, \mu_i$  (which permit getting 3D coordinates of  $P_i$  in either camera frame)

# “Epipolar Constraints” Between Two Views of a Scene



We can eliminate the depths from  $R(\lambda p) + T = \mu q$  and obtain the epipolar constraint:

$$q_i^T (T \times R p_i) = 0$$

# The Essential Matrix $E$

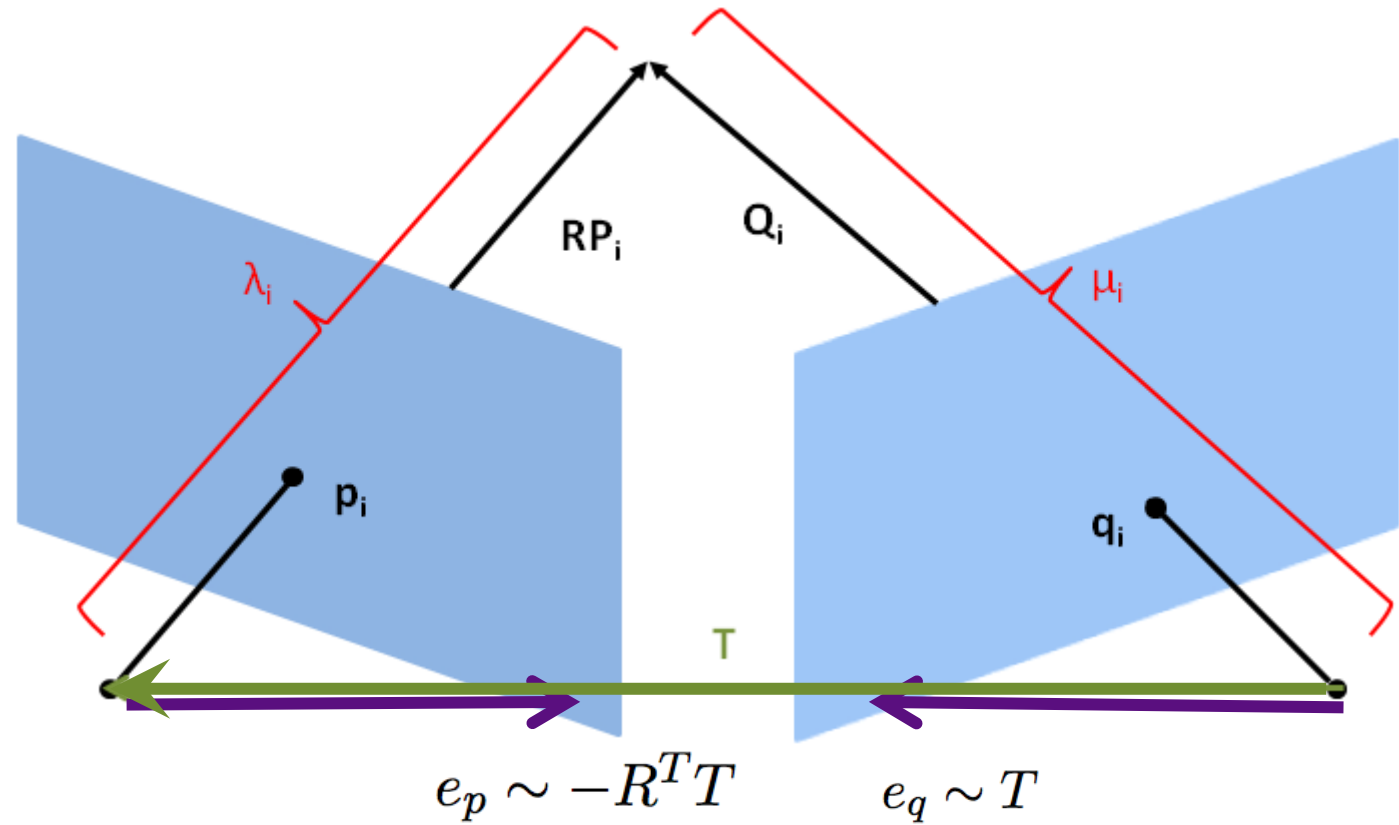
We had:  $\mathbf{q}_i^T (\mathbf{T} \times \mathbf{R} \mathbf{p}_i) = 0$

$$\Rightarrow \mathbf{q}_i^T (\hat{\mathbf{T}} \mathbf{R}) \mathbf{p}_i = 0$$

Renaming  $\mathbf{E} = (\hat{\mathbf{T}} \mathbf{R})$ :

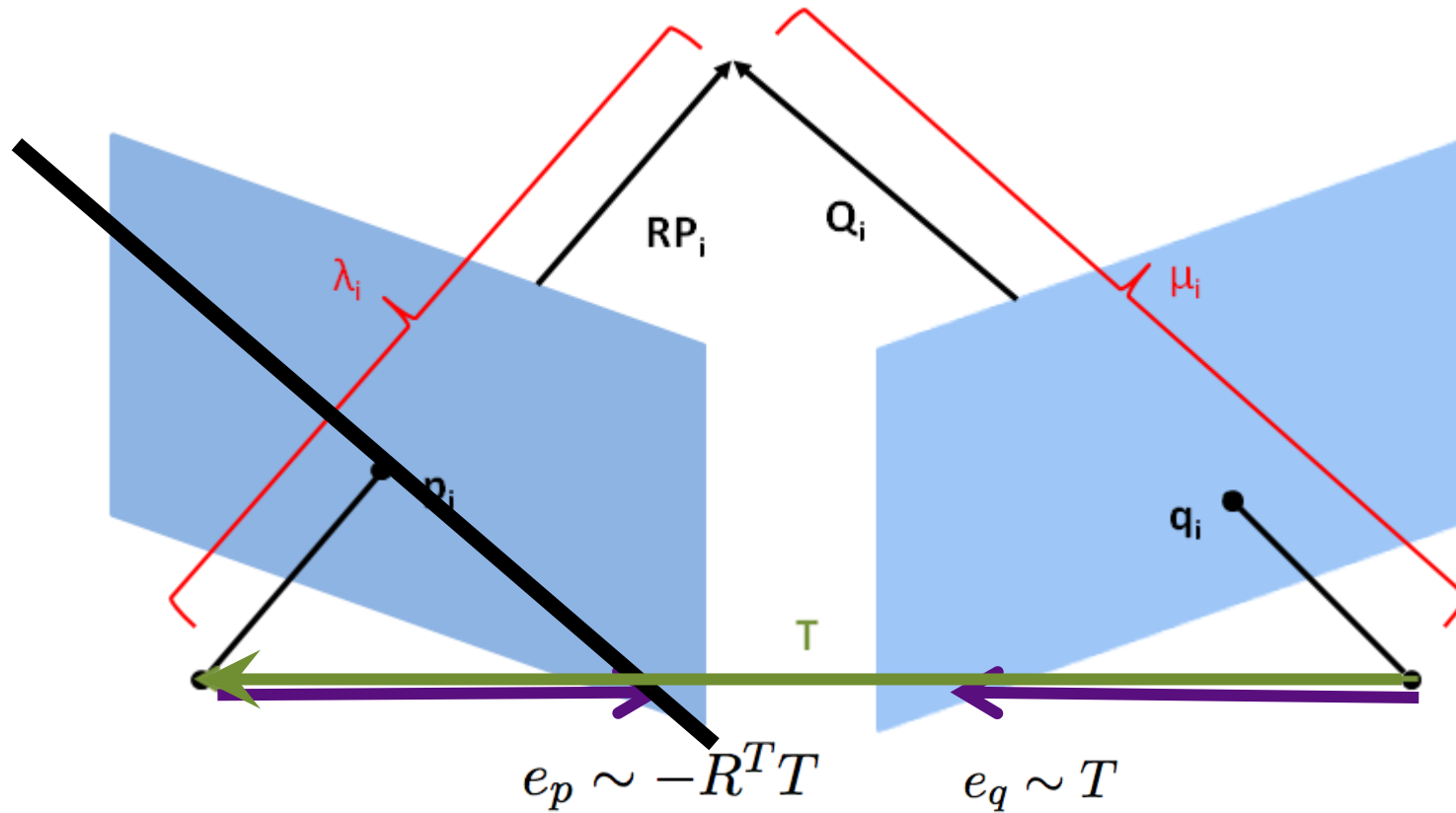
$$\mathbf{q}_i^T \mathbf{E} \mathbf{p}_i = 0$$

↓  
“Essential matrix”



Now linear in the new unknowns  $\mathbf{E}_{3 \times 3}$  ! But will need to recover  $\mathbf{T}_{3 \times 1}$ ,  $\mathbf{R}_{3 \times 3}$  later.

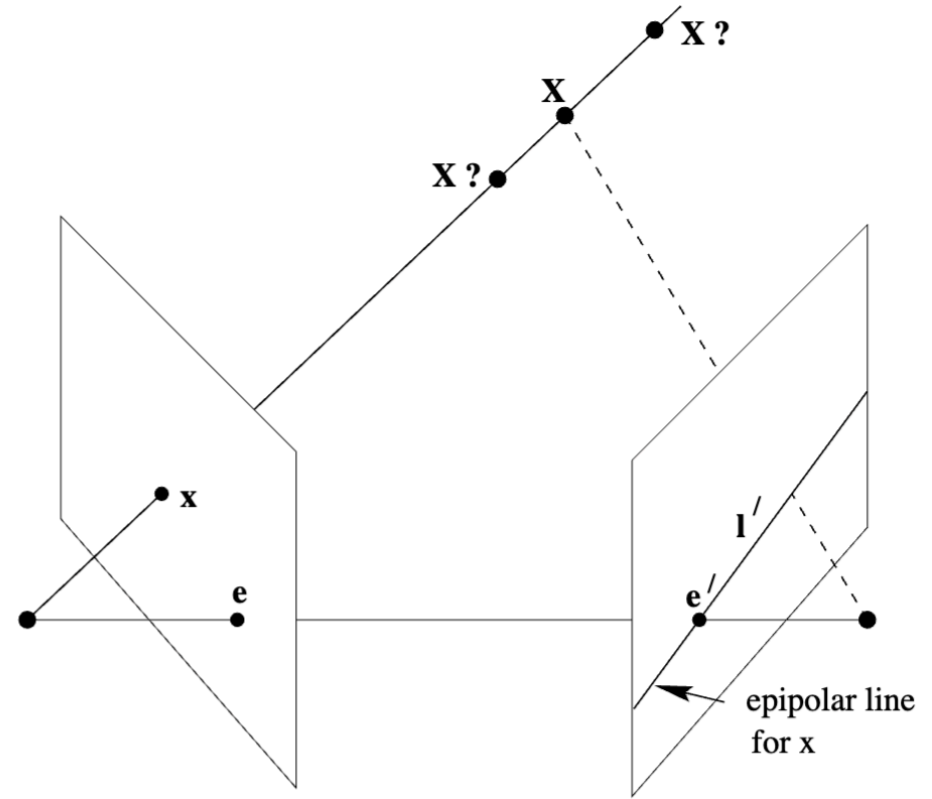
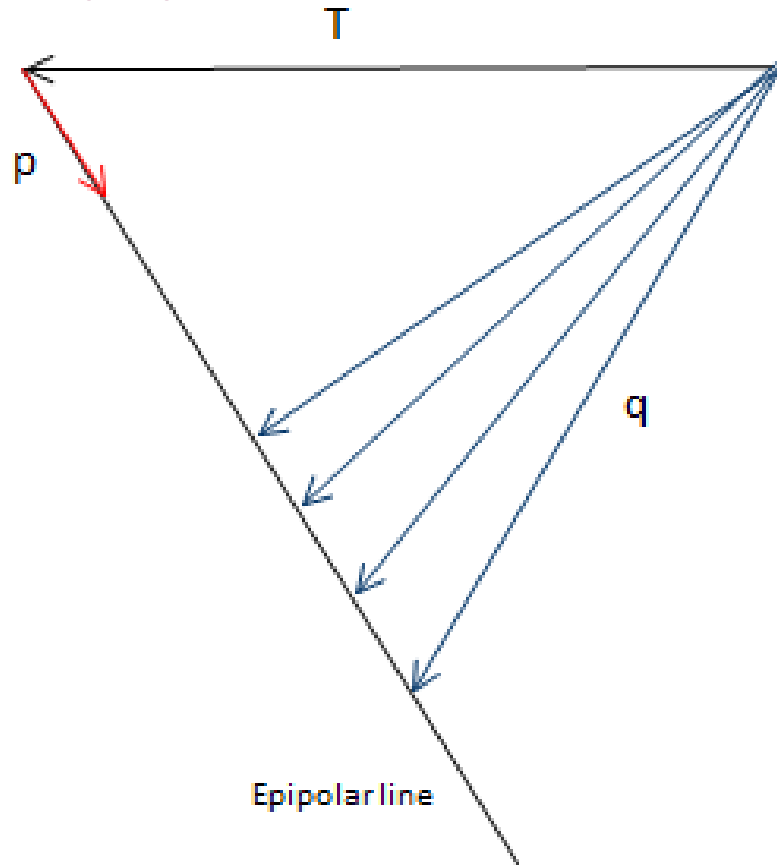
# Epipolar Lines in Essential Matrix Notation



Equation  $q^T E p = 0$  is a line equation in the  $p$ -plane with line coefficients  $E^T q$ . It is called the epipolar line in  $p$ -plane.



# Epipolar Lines Constrain Point Correspondences!



Knowledge of the  $E$ -matrix allows us to search for points  $q$  corresponding to points  $p$  along the epipolar line, reducing correspondence to 1D-search.

Position of the corresponding point  $q$  along epipolar line varies with depth of the 3D points which is still constrained to lie on the ray through  $p$ .

# 8-Point Algorithm

- Recall that each correspondence gives us one linear equation in the unknowns  $E$

Is this really linear in  $E$ ?

$q_i^T E_{3 \times 3} p_i = 0$  is a single equation that is linear in the elements of  $E$   
Can write this out explicitly as below.

If

$$E = (e_1 \ e_2 \ e_3)$$

then epipolar constraint can be rewritten as

$$q^T (e_1 \ e_2 \ e_3) \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = q^T (p_x e_1 + p_y e_2 + p_z e_3)$$

$$= \begin{pmatrix} p_x q^T & p_y q^T & p_z q^T \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = 0$$

“ $a_{1 \times 9}$ ”

This equation is linear

“ $E'_{9 \times 1}$ ”

# After solving for $E$ , not Quite Done Yet!

$E = \hat{T}R$  has fewer than 8 DOF.  $T$  has 3 DOF (+3),  $R$  has 3 DOF (+3), and  $E$  is scale invariant ( $-1$ ), so total 5 DOF. **So not any 3x3 matrix is a valid essential matrix.**

- **Problem:** Given the above, how to ensure that the estimated  $E$  is a valid essential matrix?
- **Problem:** How to decompose  $E$  into the  $\hat{T}, R$  required in SfM?

# Constructing Valid Essential Matrices and Decomposing Them

**Necessary and sufficient condition:**  $E$  is essential iff  $\sigma_1(E) = \sigma_2(E) \neq 0$  and  $\sigma_3(E) = 0$ .

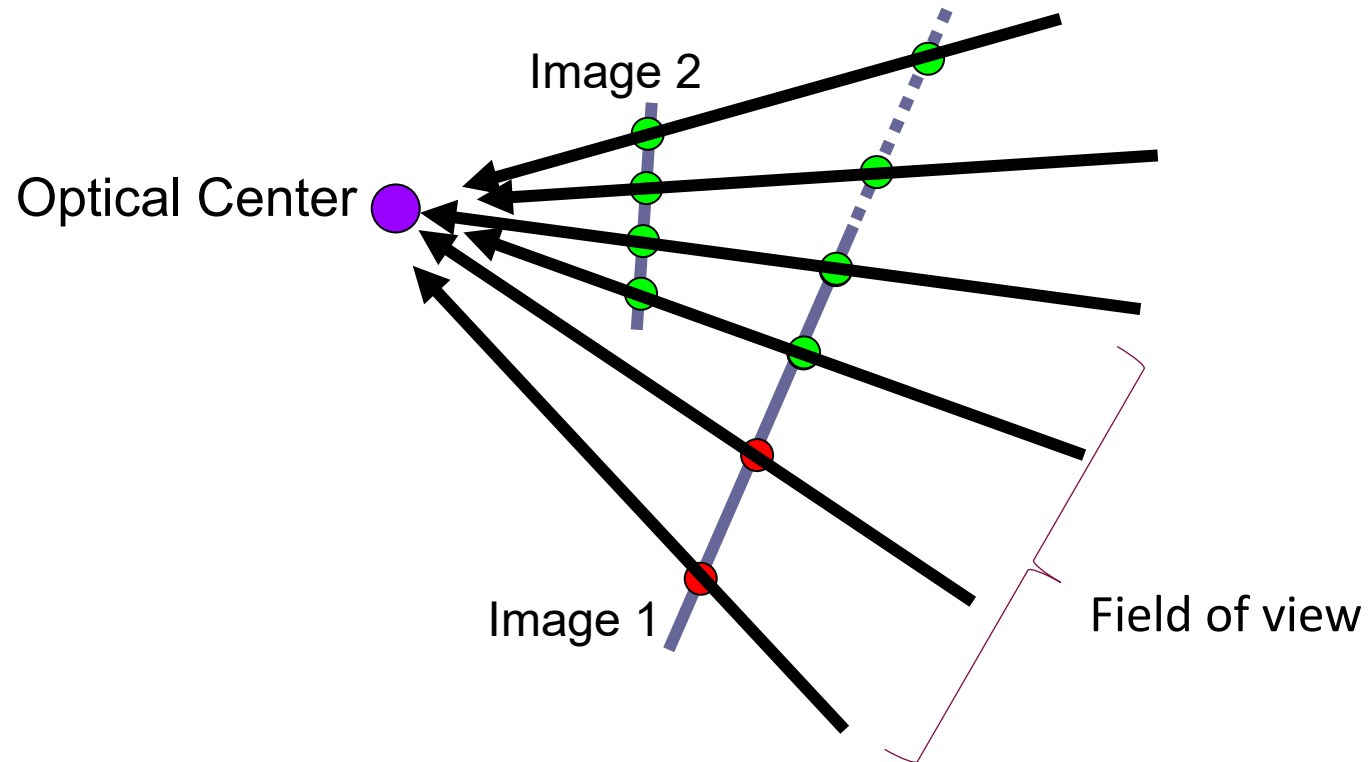
**Part 1:** Proving ‘necessary’ (“If  $E$  is essential, then ...”) will tell us about properties of essential matrices, so we can correct the  $E$  matrices from the direct method to become valid.

**Part 2:** Proving ‘sufficient’ (“If singular values ..., then ...”) will help us solve  $R, T$  from  $E$  for a particular pair of cameras.

# Cases that can't do 2-view SfM

**Case 1:**  $A$  may be **too low-rank**, i.e.,  $\text{rank}(A) < 8$ ! This can happen quite frequently in practice, e.g., smartphone moving facing a wall.

**Case 2:** No translation



# The full two-view 8-point algorithm

## Direct solution of E

- 1 Build the homogeneous linear system by stacking epipolar constraints  $q_i^T (T \times Rp_i) = 0, i = 1, \dots, 8$ :

$$\begin{bmatrix} \vdots \\ (q_i \otimes p_i)^T \\ \vdots \end{bmatrix} \begin{bmatrix} e'_1 \\ e'_2 \\ e'_3 \end{bmatrix}$$

$A \ (8 \times 9)$

- 2 Let  $\begin{bmatrix} e'_1 \\ e'_2 \\ e'_3 \end{bmatrix}$  be the nullspace of  $A$  (if  $\sigma_8 \approx 0$  give up)

# The full two-view 8-point algorithm

Decompose  
Into T, R

Make E valid

- ③  $\begin{bmatrix} e'_1 & e'_2 & e'_3 \end{bmatrix} = U \text{diag} (\sigma'_1 \ \sigma'_2 \ \sigma'_3) V^T$ . Then use the following estimate of the essential matrix:

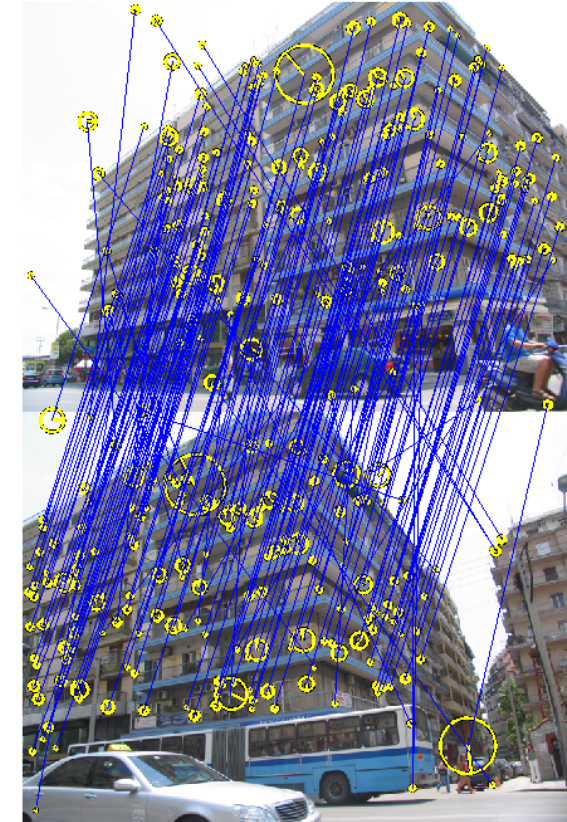
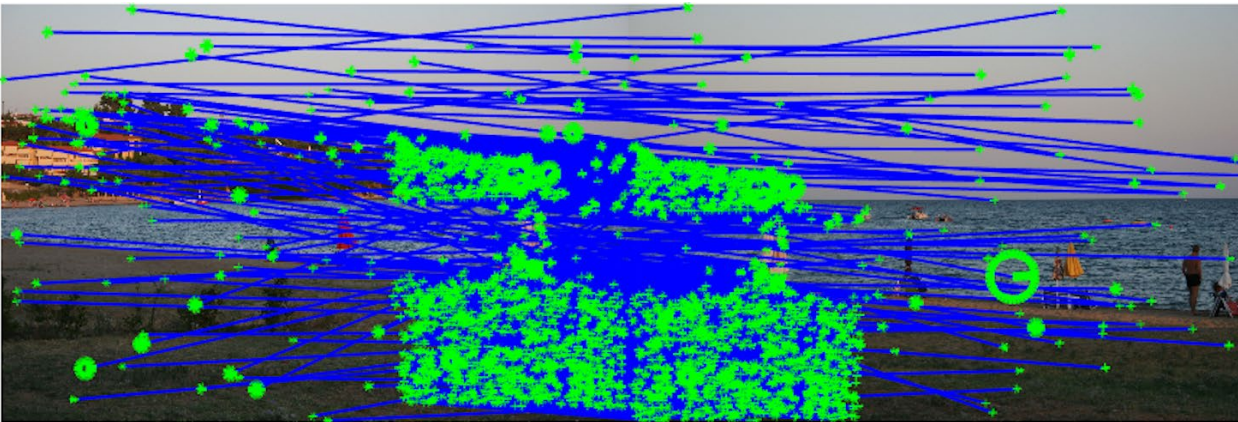
$$E = U \text{diag} \left( \frac{\sigma'_1 + \sigma'_2}{2}, \frac{\sigma'_1 + \sigma'_2}{2}, 0 \right) V^T$$

- ④  $T = \pm \hat{u}_3 \quad R = UR_{Z,\pi/2}V^T$  or  $R = UR_{Z,-\pi/2}V^T$
- ⑤ Try all four pairs  $(T, R)$  to check if reconstructed points are **in front** of the cameras  $\boxed{\lambda q = \mu Rp + T}$  give  $\lambda, \mu > 0$ .



# Correspondences are not clean!

Or we might have to group them !  
(two planes=>two homographies)

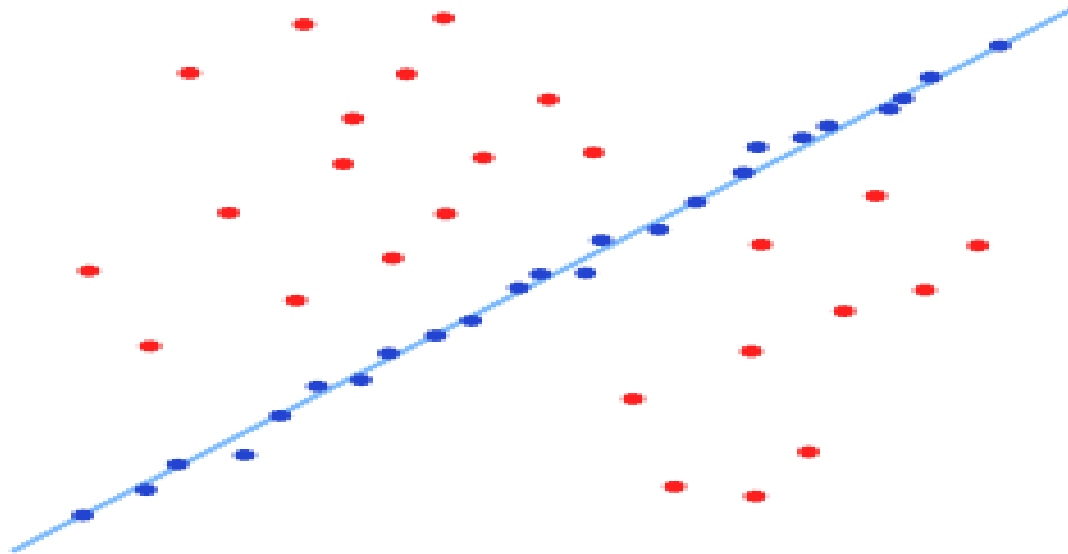




# Back to basics: a single line fitting problem

Given data  $(x_i, y_i)$  belonging to a line

$$x \cos \theta + y \sin \theta = d$$

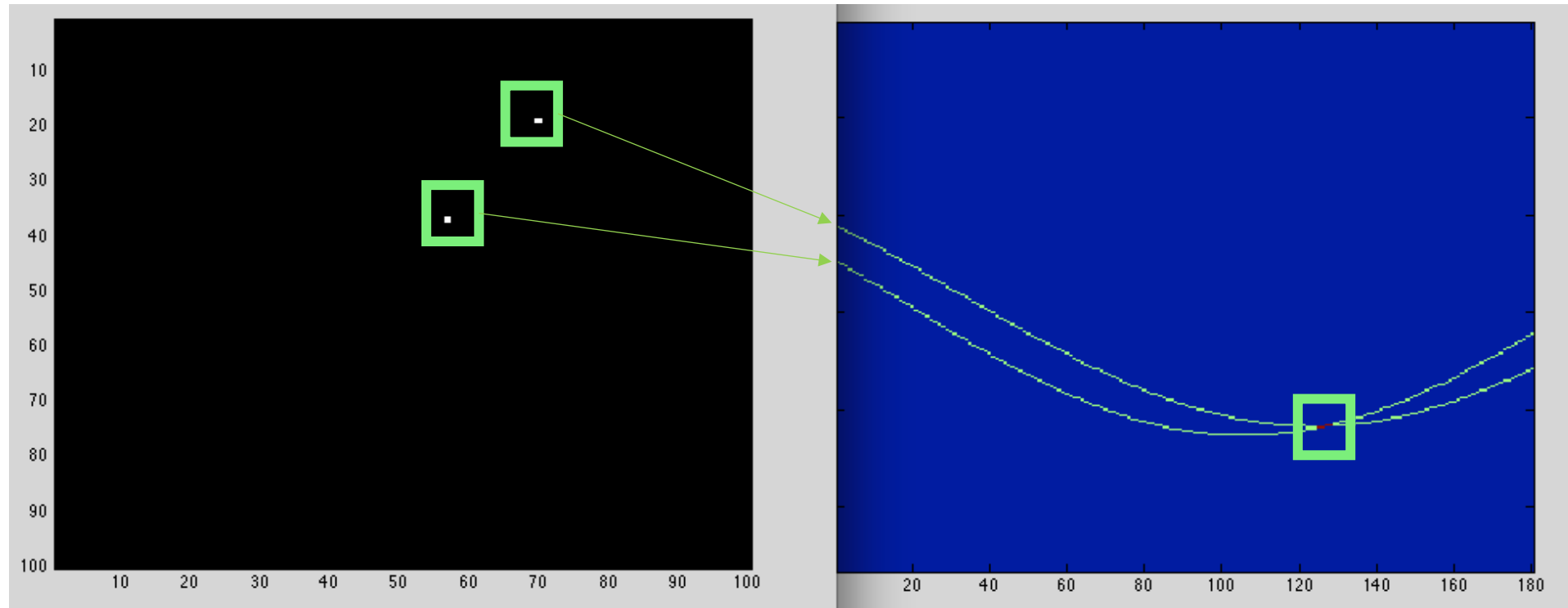


# Approach #1: Hough Transform

Line through two points  $(x_1, y_1)$  and  $(x_2, y_2)$  can be found as the intersection  $(\theta, d)$  of the two curves:

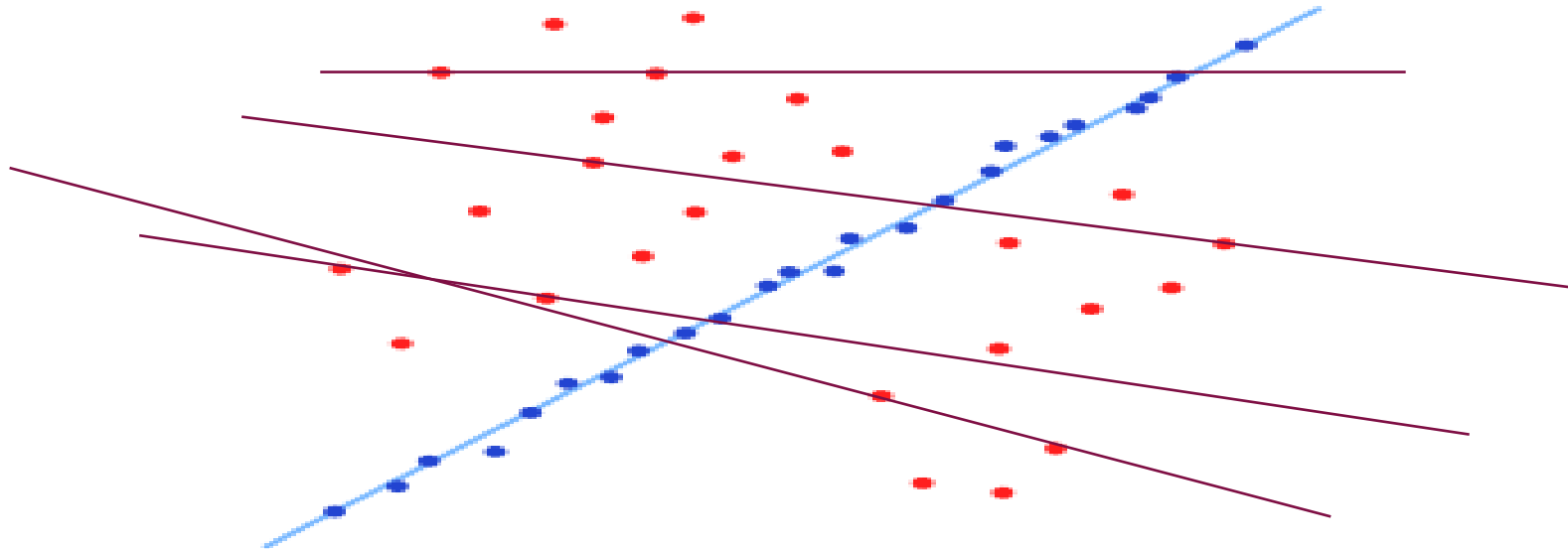
$$d = \cos \theta x_1 + \sin \theta y_1$$

$$d = \cos \theta x_2 + \sin \theta y_2$$

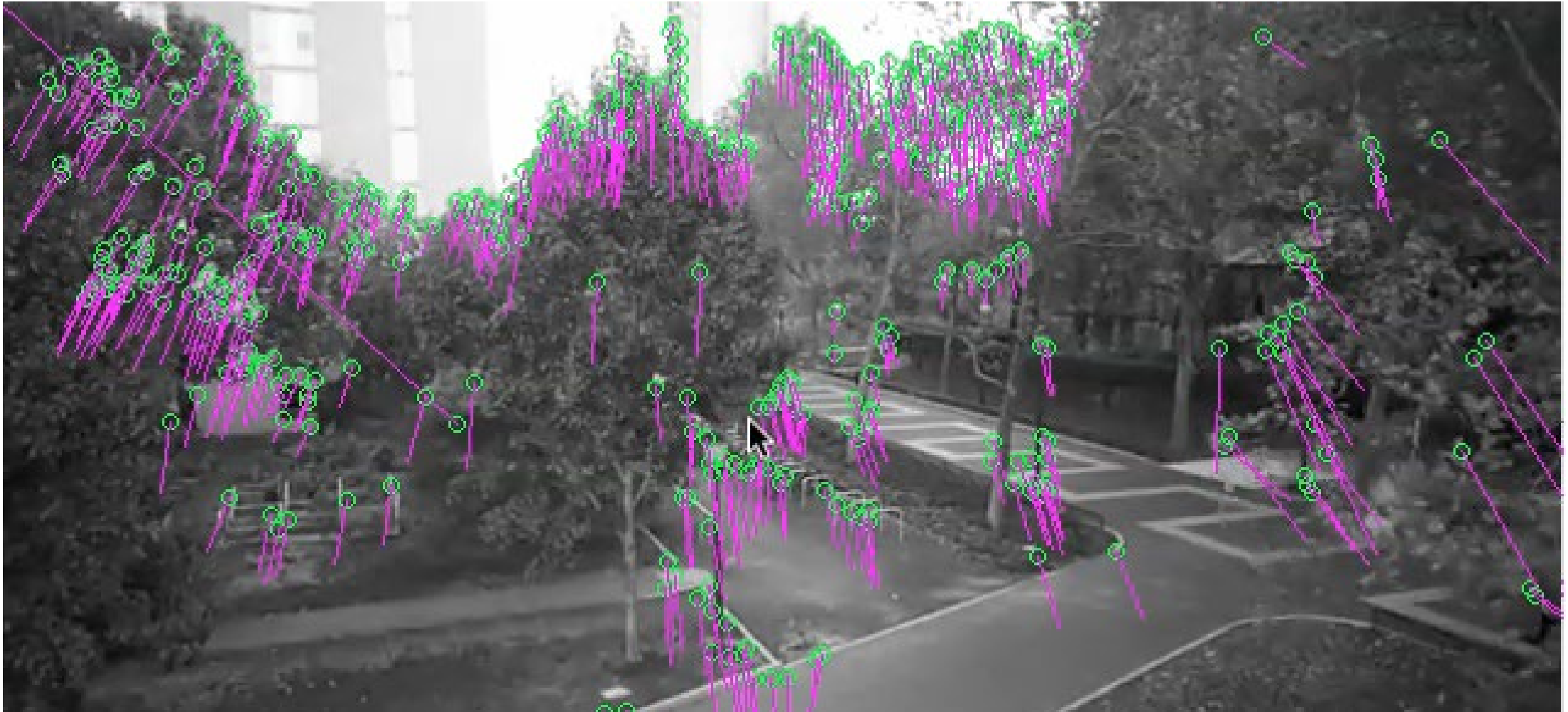


# Approach #2: RANSAC in the line fitting setting

- Set maximum inlier count  $M = 0$
- For some  $k$  randomly chosen pairs out of  $C_n^2$  pairs of points:
  - Find the corresponding line  $l$
  - Check how many other points approximately lie on this line (“inliers”).
  - If  $n_{inliers} > M$ , set  $M = n_{inliers}$  and set best candidate to  $l$



# One Approach to Get Correspondences: Optical Flow



“Tracks” attached to 3D regions in the world that show how they moved throughout a video.

# Lucas-Kanade Optical Flow

$$\left[ \nabla I_{t+1}(x, y)^T \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \Delta I_t(x, y) \right]_{x, y \in \mathcal{N}(x_0, y_0)}$$

Stacking the equations:



At point  $p_i$  in the patch

Spatial gradients  $\nabla I$  computed on second image  $I_{t+1}$

$x$  derivative

$y$  derivative

$$\begin{bmatrix} \nabla_x I \big|_{p_0} & \nabla_y I \big|_{p_0} \\ \vdots & \vdots \\ \nabla_x I \big|_{p_i} & \nabla_y I \big|_{p_i} \\ \vdots & \vdots \\ \nabla_x I \big|_{p_n} & \nabla_y I \big|_{p_n} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \begin{bmatrix} \Delta I \big|_{p_0} \\ \vdots \\ \Delta I \big|_{p_i} \\ \vdots \\ \Delta I \big|_{p_n} \end{bmatrix}$$

Pixel differences computed as  $\Delta I = I_t - I_{t+1}$

$A$

$x$

$b$

$$x^* = (A^T A)^{-1} A^T b$$

Recall that for overdetermined  $Ax = b$ , we solve  $\min_x ||Ax - b||_2^2$  using the pseudo-inverse  $(A^T A)^{-1} A^T b$

# Perhaps The Most Important Assumption: Invertibility

The diagram illustrates the linear system  $Ax = b$  used in image registration. It features several callouts and labels:

- x derivative** and **y derivative**: Labels above the first two columns of matrix  $A$ .
- At point  $p_i$  in the patch**: A callout pointing to the  $i$ -th row of matrix  $A$ .
- Spatial gradients  $\nabla I$  computed on second image  $I_{t+1}$** : A callout pointing to the entire matrix  $A$ .
- Pixel differences computed as  $\Delta I = I_t - I_{t+1}$** : A callout pointing to the vector  $b$ .
- Matrix  $A$** : A  $(n+1) \times 2$  matrix with columns for  $x$  and  $y$  derivatives. The first row corresponds to point  $p_0$ , and the last row to  $p_n$ .
- Vector  $x$** : A  $2 \times 1$  vector containing  $\delta x$  and  $\delta y$ .
- Vector  $b$** : A  $(n+1) \times 1$  vector containing pixel differences  $\Delta I$  at points  $p_0, \dots, p_n$ .
- Equation**:  $Ax = b$ .
- Solution**:  $x^* = (A^T A)^{-1} A^T b$  (highlighted in yellow).

We assumed we could invert, i.e. compute  $(A^T A)^{-1}$   
When would this fail?  $(A^T A)_{2 \times 2}$  is low-rank!

# “Interesting” Patch (Rank 2 patch)

invertible  $A^T A$ ?

$A^T A$

No

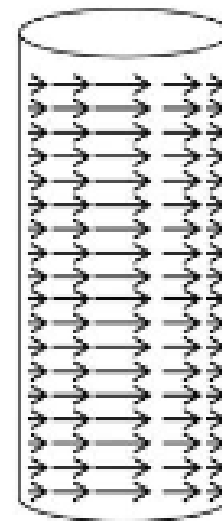
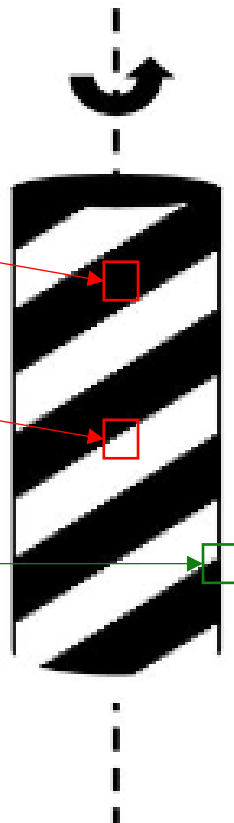
Rank 0

No

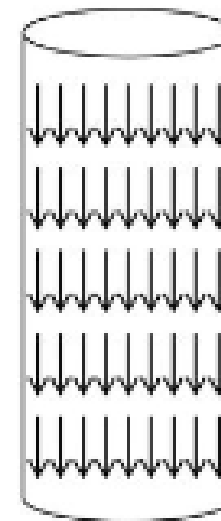
Rank 1

Yes!

Rank 2




Motion field



Optical flow

# Optical Flow: Translational Flow + Rotational Flow

$$\dot{p} = \frac{\dot{P}}{Z} - \frac{\dot{Z}}{Z}p \qquad \dot{P} = -\Omega \times P - V$$


Notation abuse warning:  $p = (x, y, 1)$ , but we will sometimes write  $\dot{p} = (\dot{x}, \dot{y})^T$

And  $V = [V_x, V_y, V_z]^T$

$$\dot{p} = \underbrace{\frac{1}{Z} \begin{bmatrix} xV_z - V_x \\ yV_z - V_y \end{bmatrix}}_{\text{translational flow}} + \underbrace{\begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \end{bmatrix} \Omega}_{\text{rotational flow independent of depth}}$$

Optical flow has two additive components: translational and rotational.



# Rotational Flow

$$\dot{\mathbf{p}} = \underbrace{\frac{1}{Z} \begin{bmatrix} xV_z - V_x \\ yV_z - V_y \end{bmatrix}}_{\text{translational flow}} + \underbrace{\begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \end{bmatrix} \Omega}_{\text{rotational flow independent of depth}}$$

$$\dot{\mathbf{p}} = \underbrace{\begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \end{bmatrix} \Omega}_{\text{rotational flow independent of depth}}$$

Not just robots, animals know  $\Omega$  through the vestibular system (inner ear)!

If we know angular velocity  $\Omega$  (e.g. from IMU gyroscope) we can:

- (1) compute optical flow  $\dot{\mathbf{p}}$  from the images (e.g. with LK)
- (2) then from  $\Omega$ , estimate rotational flow  $\dot{\mathbf{p}}_{\text{rot}}$  at each pixel independent of the scene.
- (3) then get  $\dot{\mathbf{p}}_{\text{trans}} = \dot{\mathbf{p}} - \dot{\mathbf{p}}_{\text{rot}}$

What can we do knowing the rotational and translational flows separately in this way?

Turns out, we can efficiently find the camera velocity  $V$  (up to scale)  
and also time to collision!

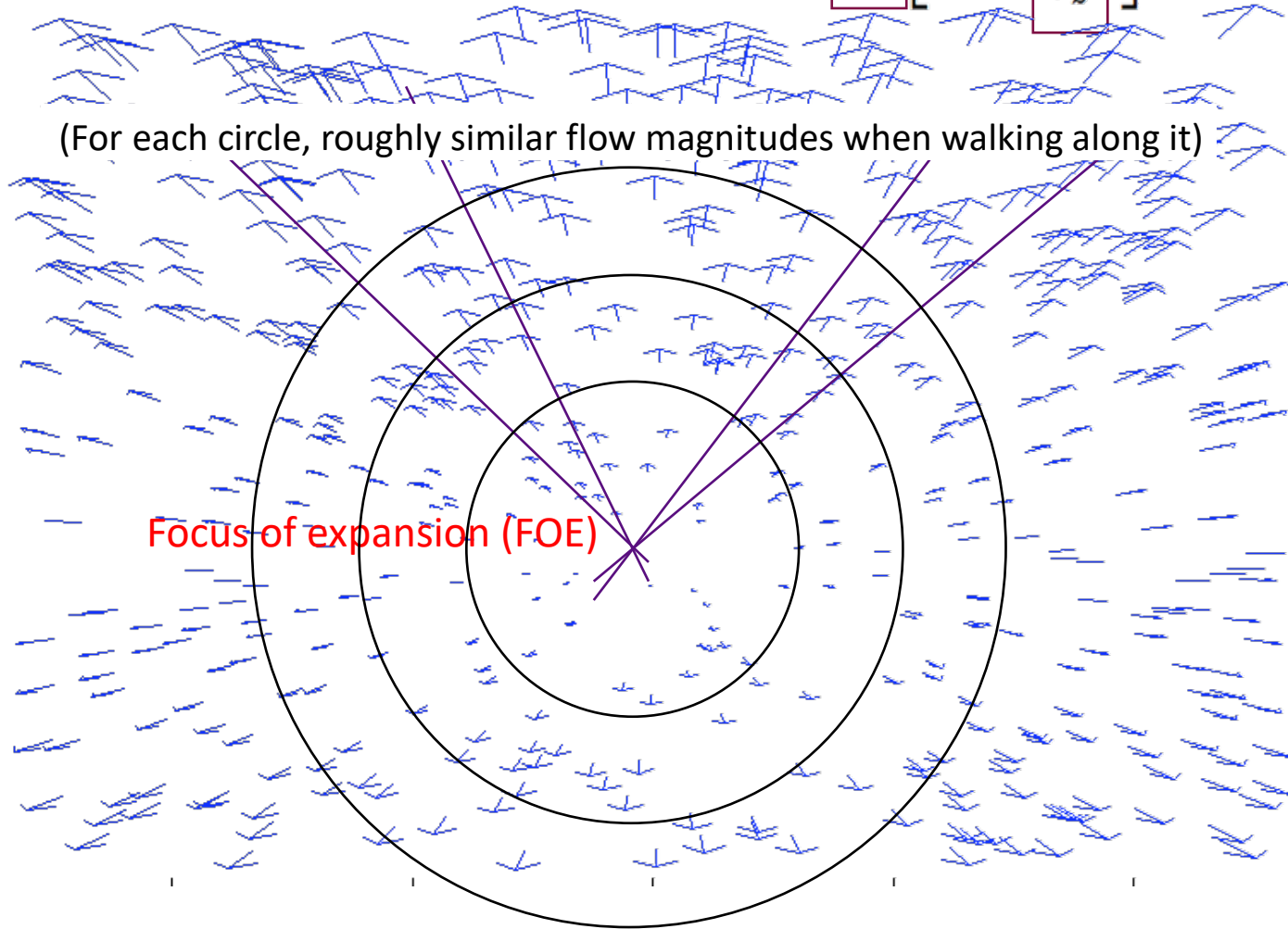
# Translational Flow

Inverse “time to collision” of object  $Z$  plane with camera

$$\dot{\mathbf{p}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{V_z}{Z} \begin{bmatrix} x - \frac{V_x}{V_z} \\ y - \frac{V_y}{V_z} \end{bmatrix}$$

Focus of expansion (FOE)

(For each circle, roughly similar flow magnitudes when walking along it)



# Finding FOE $\sim V$ upto scale ambiguity

Remember, in SfM too,  
we only computed  
translation to scale!

- We said earlier,  $\text{FOE} = \left[ \frac{V_x}{V_z}, \frac{V_y}{V_z} \right] \in \mathbb{R}^2$
- In homogeneous  $\mathbb{P}^2$  coordinates, we can write FOE as  $V \sim [V_x, V_y, V_z]$
- For point with known translational flow (we temporarily use the notation  $\dot{p}$  instead of  $\dot{p}_{\text{trans}}$ ), its “flow line” is:  $p_1 \times (p_1 + \dot{p}_1) = p_1 \times \dot{p}_1$
- FOE is the intersection of all flow lines. So,  $(p_1 \times \dot{p}_1)^T V = 0$
- Given  $n \geq 2$  points and flows,  $V$  lies on each flow line:

$$\underbrace{\begin{pmatrix} (p_1 \times \dot{p}_1)^T \\ p_2 \times \dot{p}_2)^T \\ \dots \\ p_n \times \dot{p}_n)^T \end{pmatrix}}_A V = 0$$

We know how to find null vectors!

$V \leftarrow$  the smallest right singular vector of  $A$ !

So, given camera angular velocity  $\Omega$ , we can compute camera velocity  $V$  (to scale)

# Next, Finding Time-To-Collision (TTC)

Inverse “time to collision”

$$\dot{\mathbf{p}}_{\text{trans}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{V_z}{Z} \begin{bmatrix} x - \frac{V_x}{V_z} \\ y - \frac{V_y}{V_z} \end{bmatrix}$$

Focus of expansion (FOE)

Having computed the FOE, we can compute:

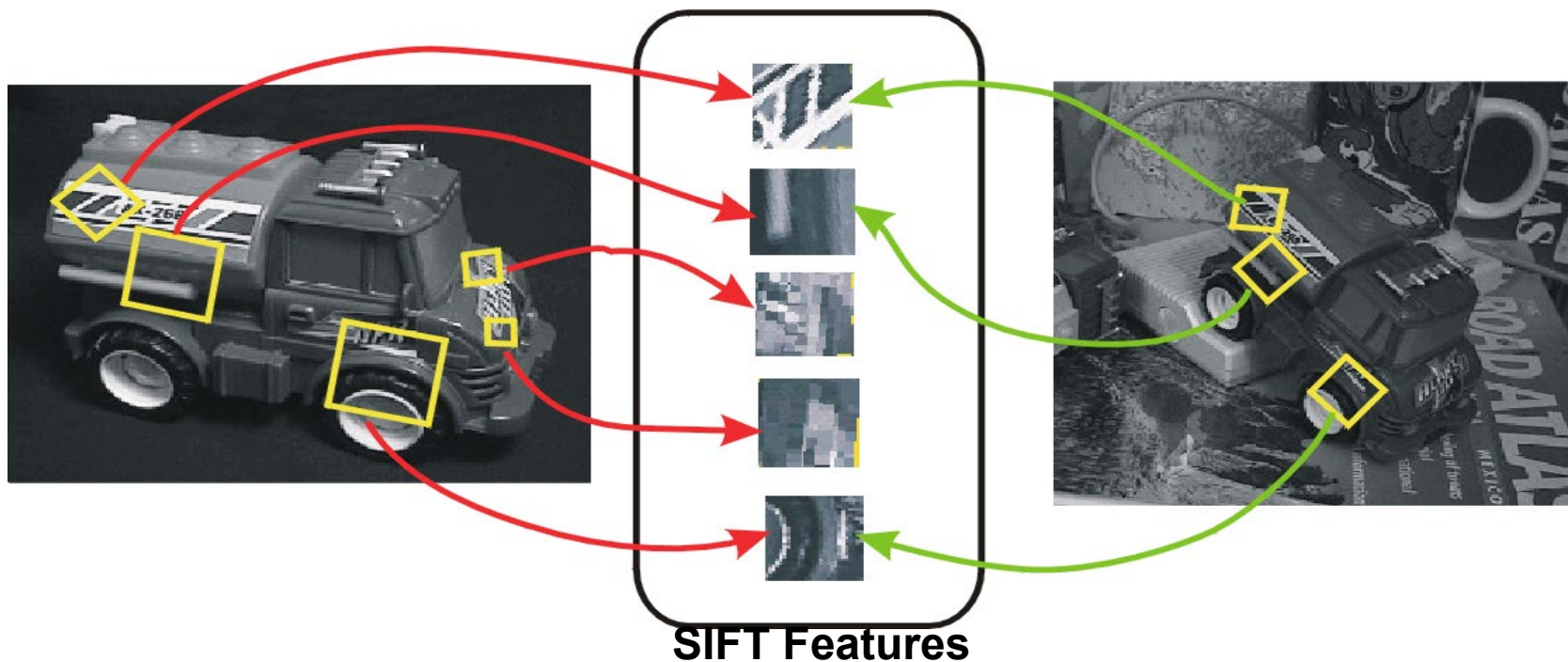
Get inverse TTC:  $\frac{V_z}{Z} = \frac{||\dot{\mathbf{p}}_{\text{trans}}||}{||\mathbf{p} - \text{FOE}||}$

# Methods for Computing Motion from Flow

- Given flow (or other) pointwise correspondences between nearby images,
  - Plus  $\Omega$ , can solve for  $FOE \sim V$  with just 2 2D-2D correspondences
  - Alone, we can solve for  $V$  and  $\Omega$  with 5 correspondences (SfM)
  - Known 3D scene, we can solve for single-frame camera pose with 3 2D-3D correspondences (PnP)

# Motivation of SIFT

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



# What is SIFT (Scale Invariant Feature Transform)

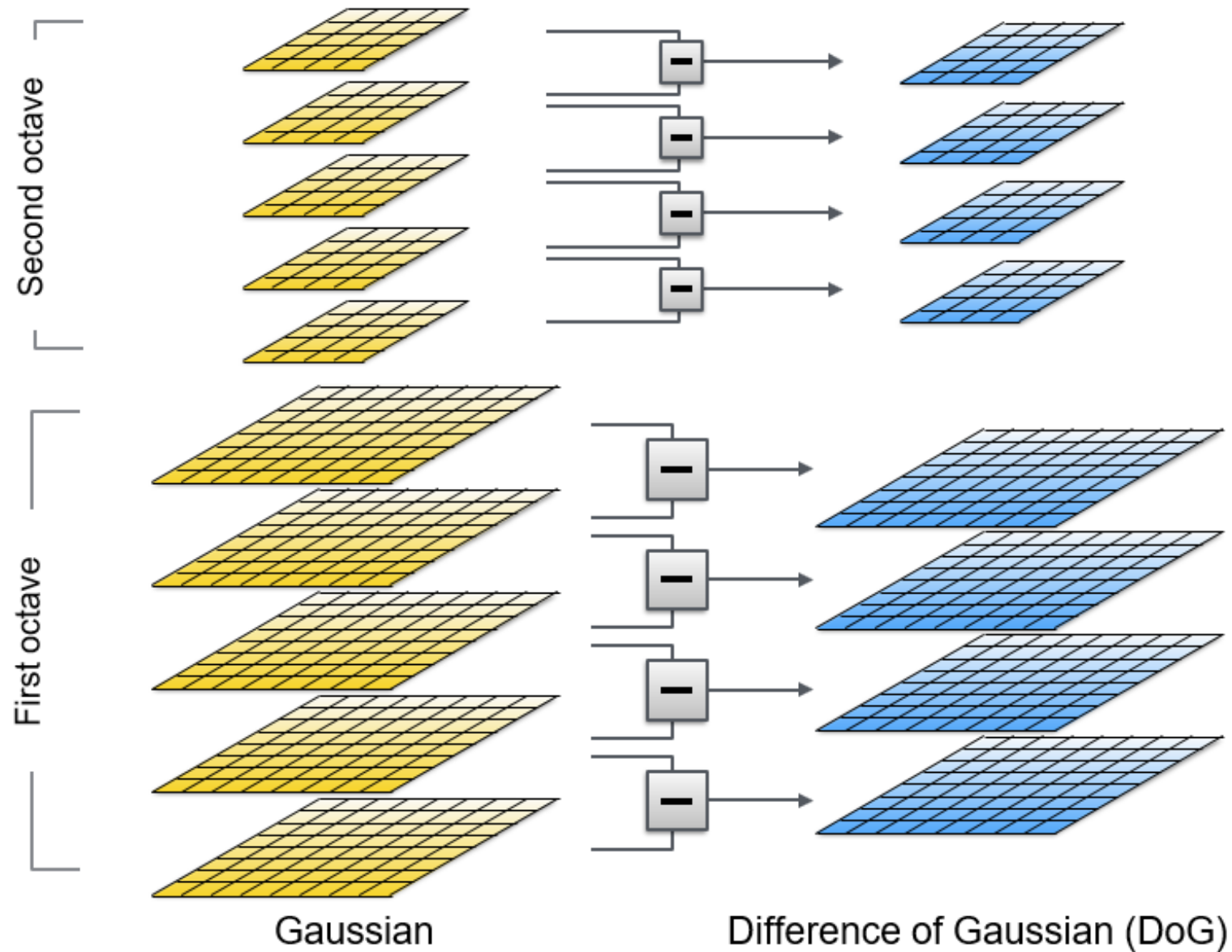


- SIFT describes both a **detector** and **descriptor**

1. Multi-scale extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor



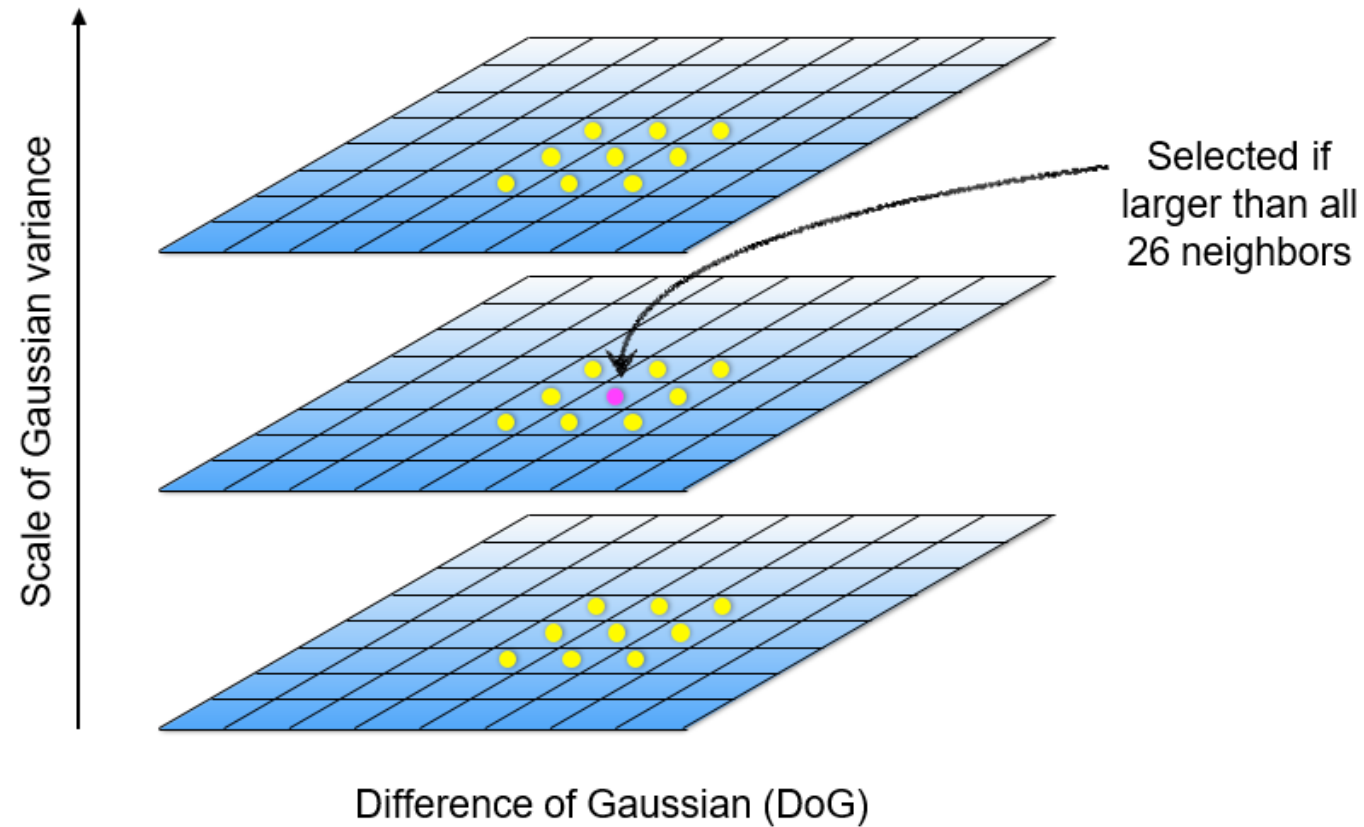
# 1. Multi-scale extrema detection





# 1. Multi-scale extrema detection

## Scale-space extrema



## 2. Keypoint Localization

2nd order Taylor series approximation of DoG scale-space

$$f(\mathbf{x}) = f + \frac{\partial f^T}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 f}{\partial \mathbf{x}^2} \mathbf{x}$$

$$\mathbf{x} = \{x, y, o\}$$

Take the derivative and solve for extrema

$$\mathbf{x}_m = - \frac{\partial^2 f^{-1}}{\partial \mathbf{x}^2} \frac{\partial f}{\partial \mathbf{x}}$$

Additional tests to retain only strong features

### 3. Orientation assignment

For a keypoint, **L** is the **Gaussian-smoothed** image with the closest scale,

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

x-derivativey-derivative

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$

Detection process returns

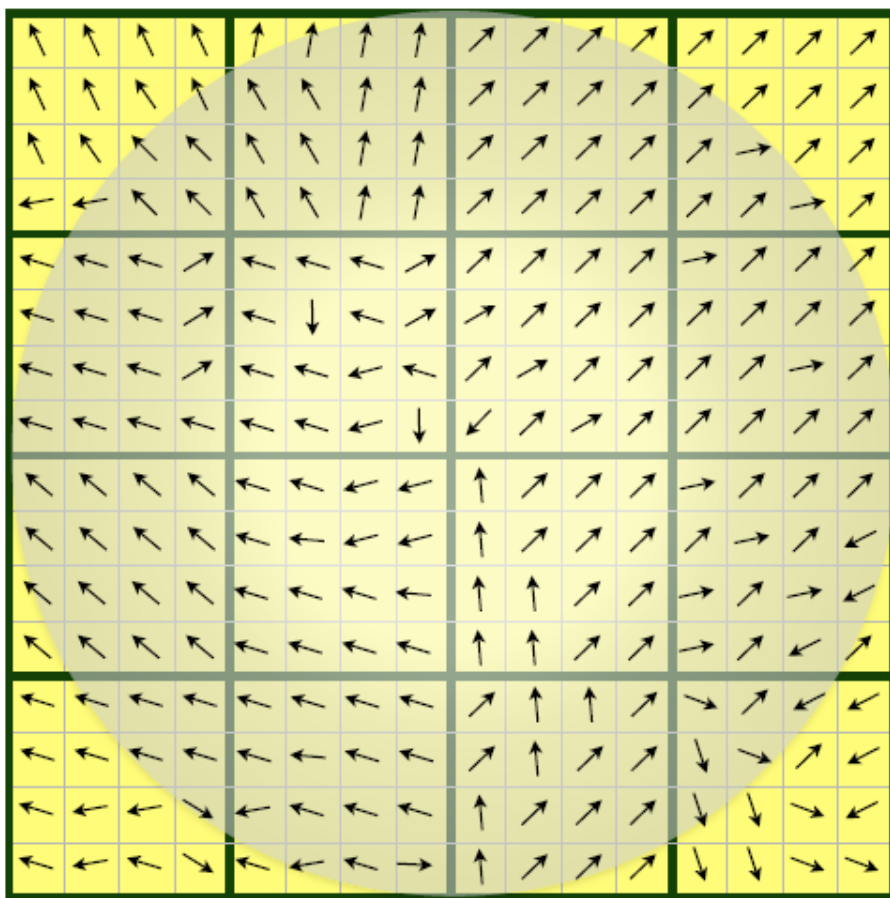
$$\{x, y, \sigma, \theta\}$$

location   scale   orientation

# 4. Keypoint Descriptor

## Image Gradients

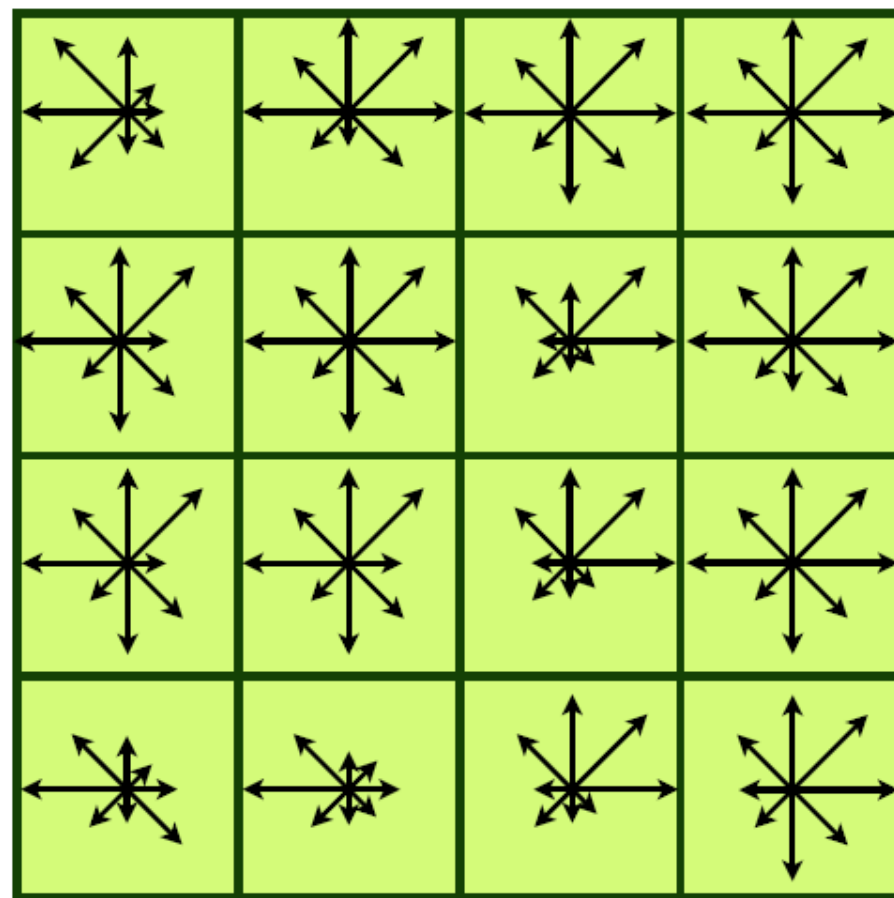
(4 x 4 pixel per cell, 4 x 4 cells)



Gaussian weighting  
(sigma = half width)

## SIFT descriptor

(16 cells x 8 directions = 128 dims)



# Towards Multi-view Reconstruction

- Coming up next, extending SfM to  $> 2$  views:
  - The Incremental Approach, through SLAM / odometry. ORB-SLAM
  - The Global Approach i.e. Bundle Adjustment

# Visual Odometry / SLAM

With figures and text from mathworks.com

<https://www.mathworks.com/help/vision/ug/monocular-visual-simultaneous-localization-and-mapping.html>

# “Visual Odometry”

What is **Odometry** ?

- Measuring how far you go by counting wheel rotations or steps.
- Known as “**path integration**” in biological perception.
- More general, integration of velocity or acceleration measurements: **inertial odometry**.

What is **Visual Odometry** ?

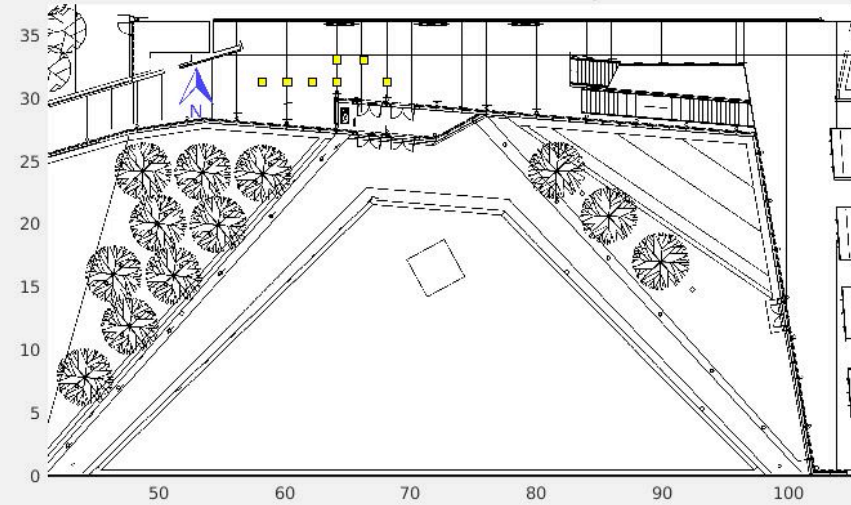
- The process of **incrementally** estimating your position and orientation with respect to an initial reference frame by tracking visual features, in an unknown environment

# Visual Odometry

Current Center GoPro (C2) frame

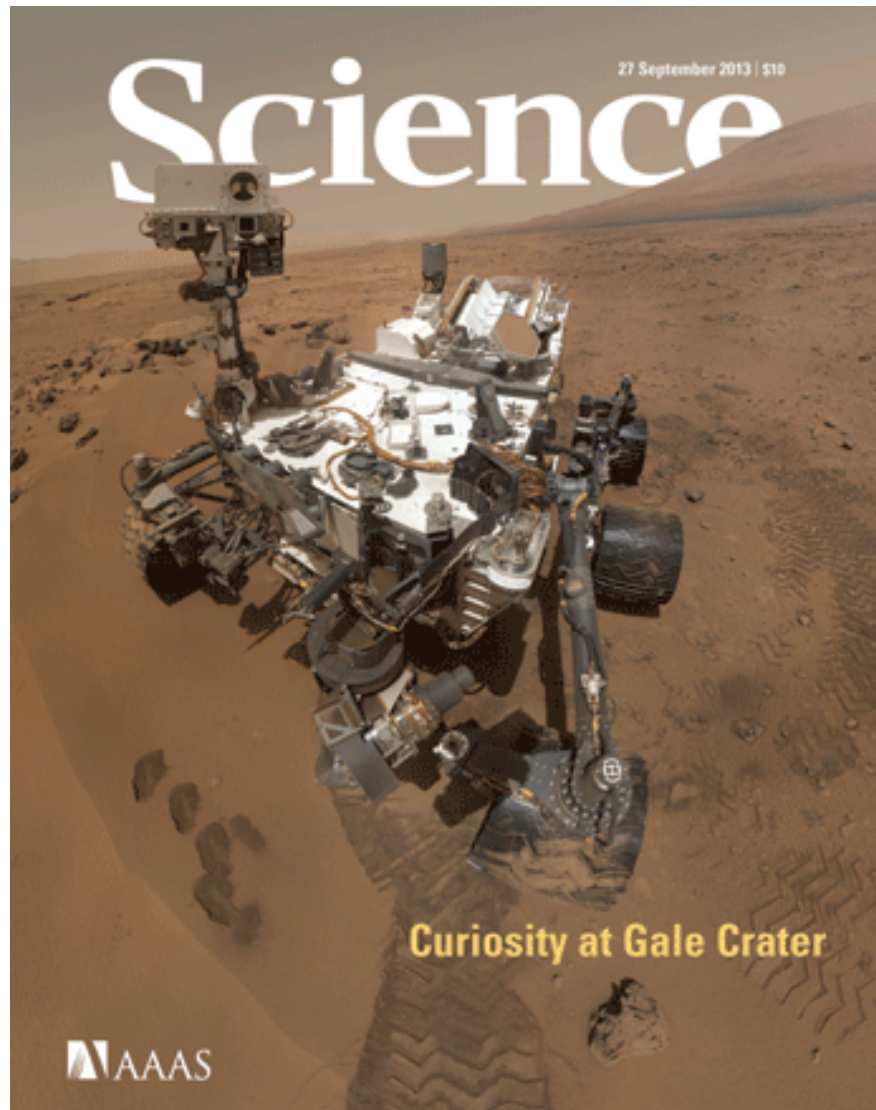


Ground Truth Path Tracking





# Visual odometry on Mars!



## Two Years of Visual Odometry on the Mars Exploration Rovers

Mark Maimone, Yang Cheng, and Larry Matthies  
Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, CA USA  
mark.maimone@jpl.nasa.gov

### Abstract

NASA's two Mars Exploration Rovers (MER) have successfully demonstrated a robotic Visual Odometry capability on another world for the first time. This provides each rover with accurate knowledge of its position, which allows it to autonomously detect and compensate for any unforeseen slip encountered during a drive. It has enabled the rovers to drive safely and more effectively in highly-sloped and sandy terrains, and has resulted in increased mission science return by reducing the number of days required to drive into interesting areas. The MER Visual Odometry system comprises onboard software for comparing stereo pairs taken by the pointable mast-mounted 45 degree FOV Navigation cameras (NAVCAMs). The system computes an update to the 6 Degree Of Freedom rover pose ( $x, y, z$ , roll, pitch, yaw) by tracking the motion of autonomously-selected terrain features between two pairs of 256x256 stereo images. It has demonstrated good performance with high rates of successful convergence (97% on Spirit, 95% on Opportunity), successfully detected slip ratios as high as 125%, and measured changes as small as 2 mm, even while driving on slopes as high as 31 degrees.

During the first two years of operations, Visual Odometry evolved from an "extra credit" capability into a critical vehicle safety system. In this paper we describe our Visual Odometry algorithm, discuss several driving strategies that rely on it (including Slip Checks, Keep-out Zones, and Wheel Dragging), and summarize its results from the first two years of operations on Mars.

### 1 Background

Keeping track of a vehicle's location is one of the most challenging aspects of planetary rover operations. NASA's Mars Exploration Rovers (MERs) typically have been commanded only once per Martian solar day (or "sol") using a pre-scheduled sequence of precise metrically specified commands (e.g., "drive forward 2.34 meters, turn in place 0.3567 radians to the right, drive to location  $X, Y$ , take color pictures of the terrain at location  $X, Y, Z$ " (Biesiadecki et al., 2005)), so having an accurate position estimate onboard during the execution of *all* terrain-based commands has been of critical importance.

# Why is *Vision* the Right Tool for Odometry?

- Why not just simple wheel odometry like on a car?
  - On rugged terrain with slopes, rocks, slip etc. simple wheel odometry like on a car doesn't cut it.
  - Unmanned Aerial Vehicles (UAVs) don't have wheels.
- Why not inertial measurement units (IMU)?
  - Drift because of double integration over acceleration inputs.
- Why not GPS?
  - Not accurate enough (up to a few meters off),
  - Missing in many places (like on Mars, or in the deep ocean).

# Back to Mars

The design goal for MER was to maintain a position estimate that drifted no more than 10% during a 100 meter drive. MER rover onboard position and attitude estimates were updated at 8 Hz nearly every time the wheels or rover arm (Instrument Deployment Device, or IDD) were actuated. Changes in attitude (roll, pitch, yaw) were measured using a Litton LN-200 Inertial Measurement Unit (IMU) that has 3-axis accelerometers and 3-axis angular rate sensors, and changes in position were estimated by combining attitude measurements with encoder readings of how much the wheels turned (wheel odometry). Position estimates derived solely from those sensors easily achieved the desired accuracy in benign terrains (Li et al., 2005), but not on steep slopes or sandy terrain.

After moving a small amount on a slippery surface, the rovers were often commanded to use camera-based *Visual Odometry* to correct any errors in the initial wheel odometry-based estimate that occur when the wheels lose traction on large rocks and steep slopes. Our

Two Years of Visual Odometry on the Mars  
Exploration Rovers

---

Mark Maimone, Yang Cheng, and Larry Matthies  
Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, CA USA  
mark.maimone@jpl.nasa.gov

# Do We Need New Methods For Visual Odometry?

Q: Didn't we just discuss many methods for computing camera motion from 2 frames? Can we not just keep repeating this for every new pair of frames and integrating the motions?

A: Indeed, a naïve system could look like this, but this would lead to ***drift***, i.e., accumulating errors over time.

Can avoid drift by maintaining some kind of *consistency* over longer durations, rather than make independent measurements over pairs of consecutive frames alone.

Emphasis on *avoiding drift*