# Pose Tracking: Structure from Motion and SLAM

CS 6334 Virtual Reality

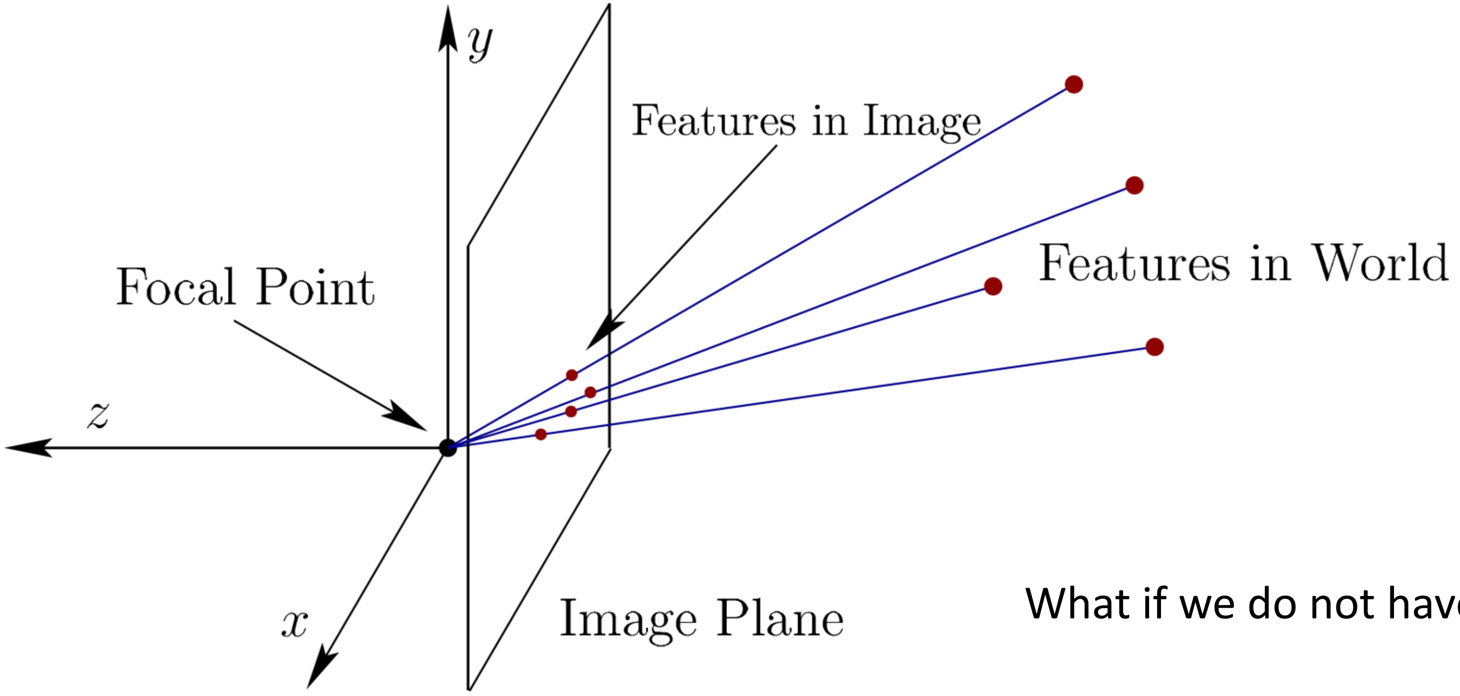
Professor Yu Xiang

The University of Texas at Dallas

# Tracking in VR

- Tracking the user's sense organs
  - E.g., Head and eye
  - Render stimulus accordingly
- Tracking user's other body parts
  - E.g., human body and hands
  - Locomotion and manipulation
- Tracking the rest of the environment
  - Augmented reality
  - Obstacle avoidance in the real world

# Feature-based Tracking




The PnP problem

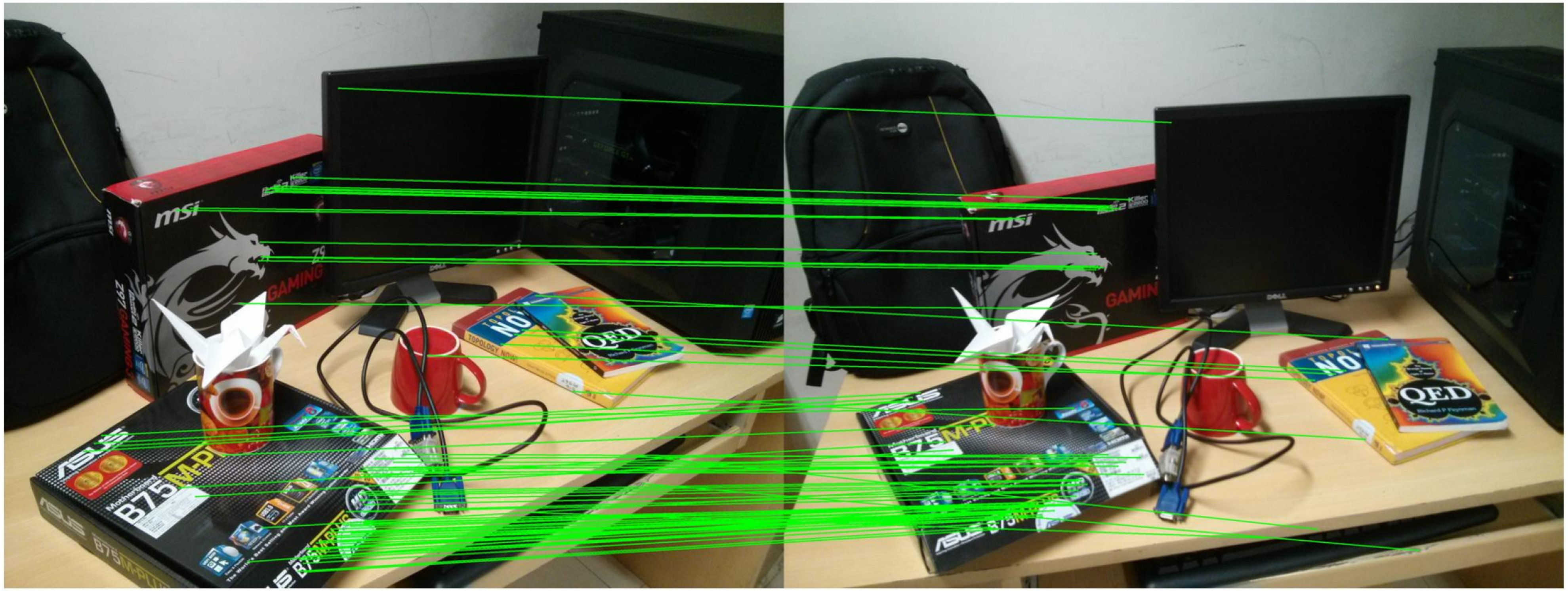
- Known: 3D locations, 2D locations, camera intrinsics
- Unknown:

  6D pose of the camera

What if we do not have the 3D locations of these feature points?

# Feature-based Tracking

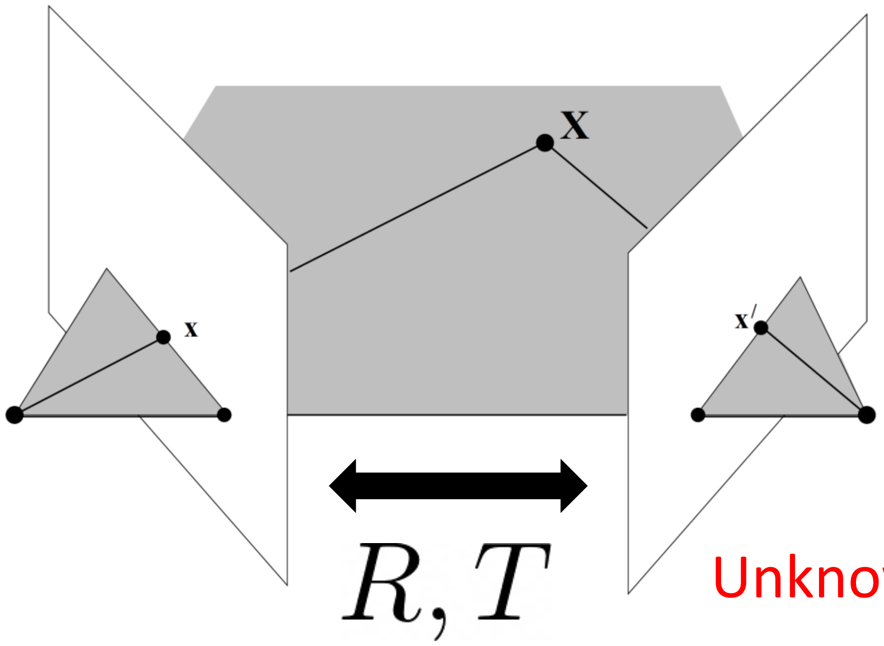
- Idea: using images from different views and feature matching



Geometry-aware Feature Matching for Structure from Motion Applications. Shah et al, WACV'15

# Feature-based Tracking

- Idea: using images from different views and feature matching
- Triangulation from pixel correspondences to compute 3D location




Unknow

Intersection of two backprojected lines

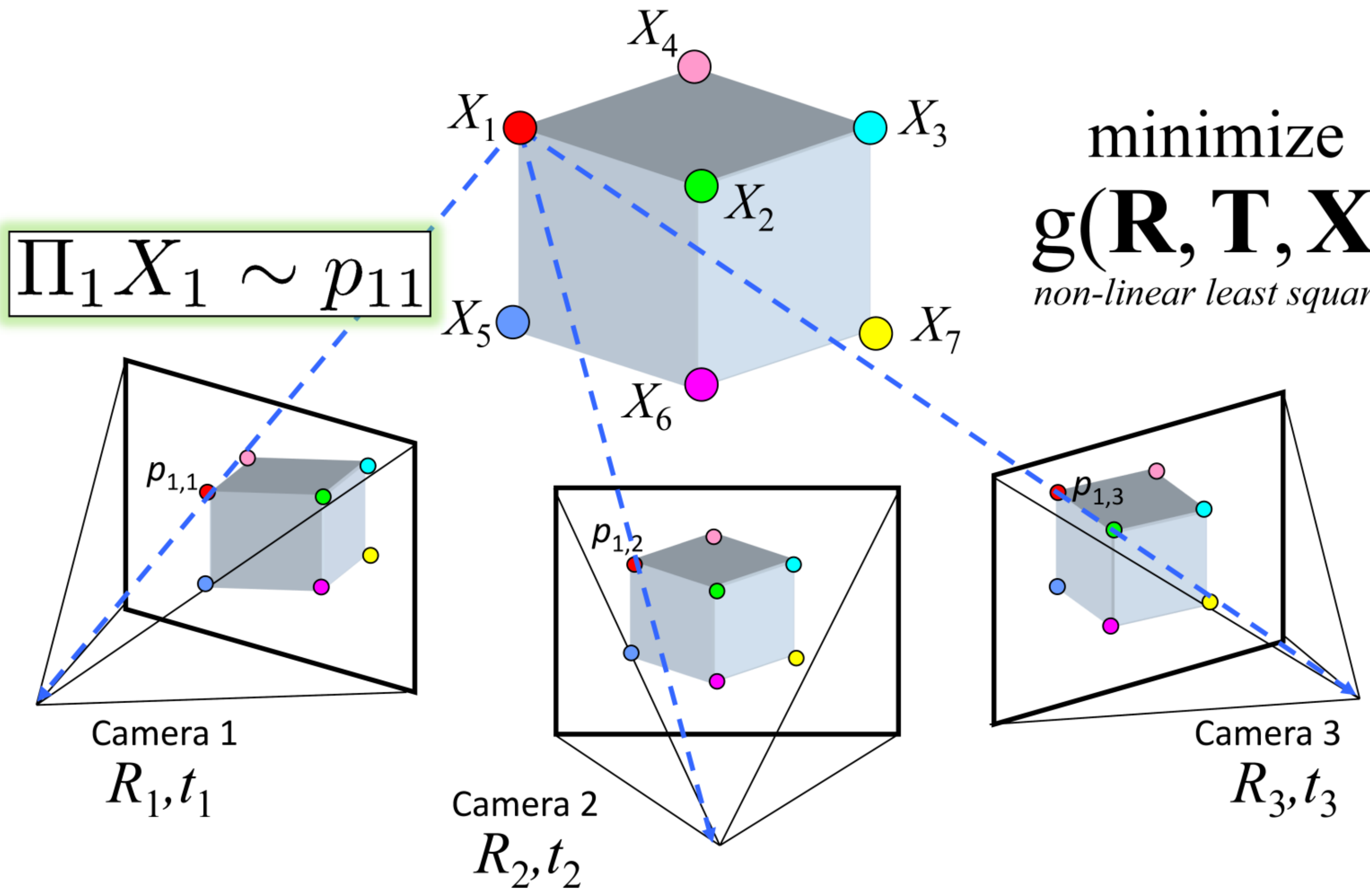
$$\mathbf{X} = \mathbf{l} \times \mathbf{l}'$$

# Structure from Motion

- Input
  - A set of images from different views

- Output
  - 3D Locations of all feature points in a world frame
  - Camera poses of the images

# Structure from motion




minimize

$$\mathbf{g}(\mathbf{R}, \mathbf{T}, \mathbf{X})$$

*non-linear least squares*

# Structure from Motion

- Minimize sum of squared reprojection errors

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

*predicted* image location

*observed* image location

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

m points, n images

A non-linear least squares problem

- E.g. Levenberg-Marquardt

# The Levenberg-Marquardt Algorithm

- Nonlinear least squares $\hat{\boldsymbol{\beta}} \in \operatorname{argmin}_{\boldsymbol{\beta}} S(\boldsymbol{\beta}) \equiv \operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{m} \left[y_i - f(x_i, \boldsymbol{\beta})\right]^2$

- An iterative algorithm
  - Start with an initial guess $\beta_0$
  - For each iteration $\beta \leftarrow \beta + \delta$

- How to get $\delta$?
  - Linear approximation $f(x_i, \boldsymbol{\beta} + \boldsymbol{\delta}) \approx f(x_i, \boldsymbol{\beta}) + \mathbf{J}_i \boldsymbol{\delta}$ $\quad \mathbf{J}_i = \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$
  - Find to $\delta$ minimize the objective $S(\boldsymbol{\beta} + \boldsymbol{\delta}) \approx \sum_{i=1}^{m} \left[y_i - f(x_i, \boldsymbol{\beta}) - \mathbf{J}_i \boldsymbol{\delta}\right]^2$

Wikipedia

# The Levenberg-Marquardt Algorithm

- Vector notation for $S(\boldsymbol{\beta}+\boldsymbol{\delta}) \approx \sum_{i=1}^{m} \left[y_i - f(x_i, \boldsymbol{\beta}) - \mathbf{J}_i \boldsymbol{\delta}\right]^2$

$$
\begin{aligned}
S(\boldsymbol{\beta}+\boldsymbol{\delta}) &\approx \|\mathbf{y} - \mathbf{f}(\boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}\|^2 \\
&= [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}]^{\mathrm{T}} [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}] \\
&= [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]^{\mathrm{T}} [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})] - [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]^{\mathrm{T}} \mathbf{J}\boldsymbol{\delta} - (\mathbf{J}\boldsymbol{\delta})^{\mathrm{T}} [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})] + \boldsymbol{\delta}^{\mathrm{T}} \mathbf{J}^{\mathrm{T}} \mathbf{J}\boldsymbol{\delta} \\
&= [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]^{\mathrm{T}} [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})] - 2[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]^{\mathrm{T}} \mathbf{J}\boldsymbol{\delta} + \boldsymbol{\delta}^{\mathrm{T}} \mathbf{J}^{\mathrm{T}} \mathbf{J}\boldsymbol{\delta}.
\end{aligned}
$$

Take derivation with respect to $\delta$ and set to zero $\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}\right)\boldsymbol{\delta} = \mathbf{J}^{\mathrm{T}}[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]$

Levenberg's contribution $\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{I}\right)\boldsymbol{\delta} = \mathbf{J}^{\mathrm{T}}[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]$ damped version

$$\beta \leftarrow \beta + \delta$$

Wikipedia

# Structure from Motion

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

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

*predicted* image location

*observed* image location

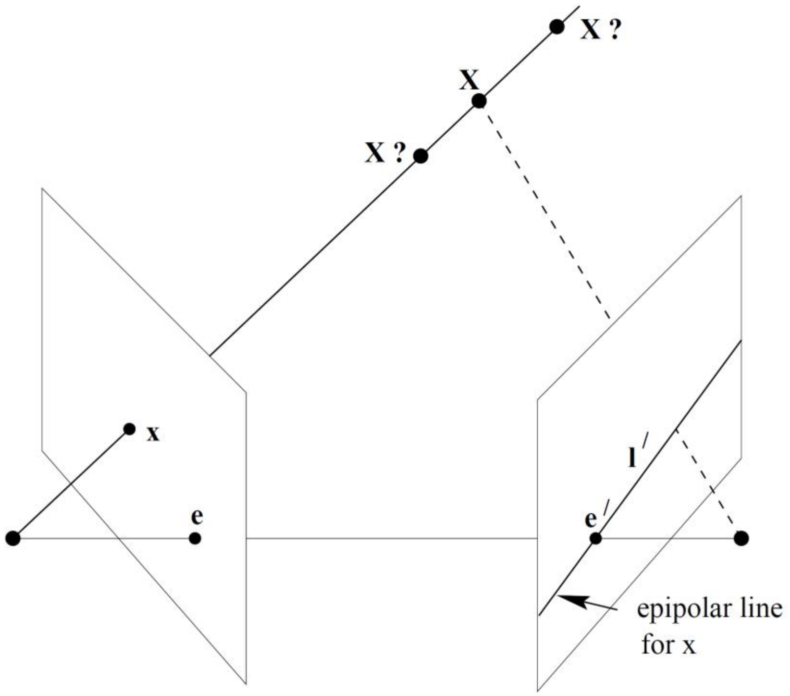
$$\beta = (\mathbf{X}, \mathbf{R}, \mathbf{T})$$

How to get the initial estimation $\beta_0$ ?

Random guess is not a good idea.

# Matching Two Views

- Fundamental matrix




$\mathbf{x}'$ is on the epiploar line $\mathbf{l}' = F\mathbf{x}$

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

$$\begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

$$x_i x_i' f_{11} + x_i y_i' f_{21} + x_i f_{31} + y_i x_i' f_{12} + y_i y_i' f_{22} + y_i f_{32} + x_i' f_{13} + y_i' f_{23} + f_{33} = 0$$

$$\begin{bmatrix} x_1 x_1' & x_1 y_1' & x_1 & y_1 x_1' & y_1 y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_m x_m' & x_m y_m' & x_m & y_m x_m' & y_m y_m' & y_m & x_m' & y_m' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

We need 8 points to solve this system.

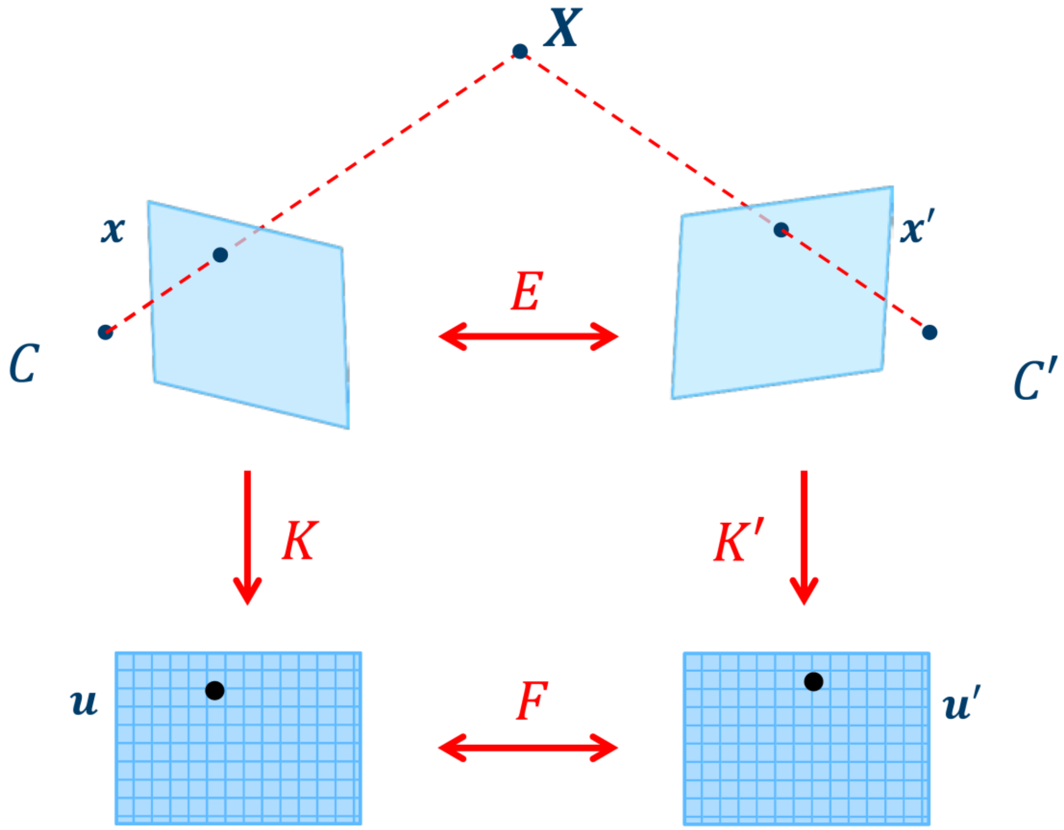
# Matching Two Views

- Essential matrix E

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

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

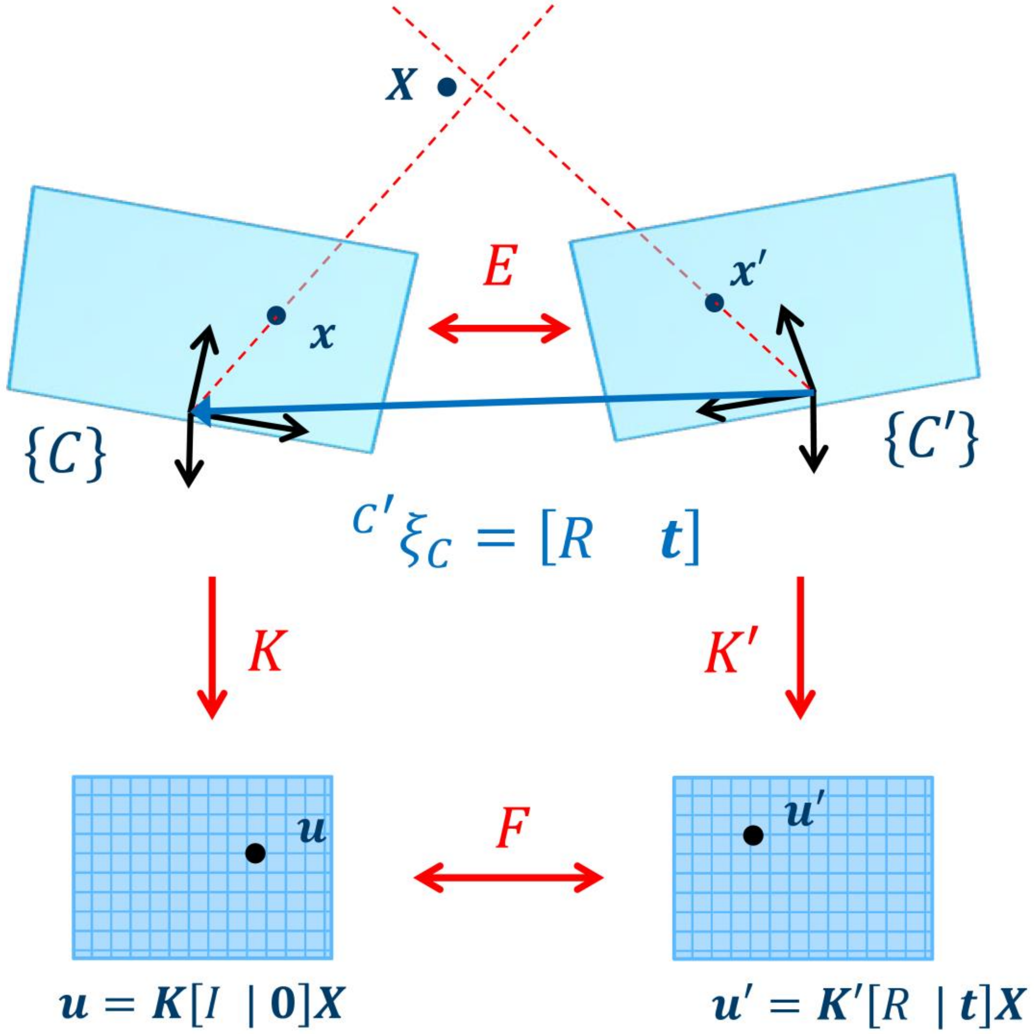
$$F = K'^{-T} E K^{-1}$$




Credit: Thomas Opsahl

# Matching Two Views

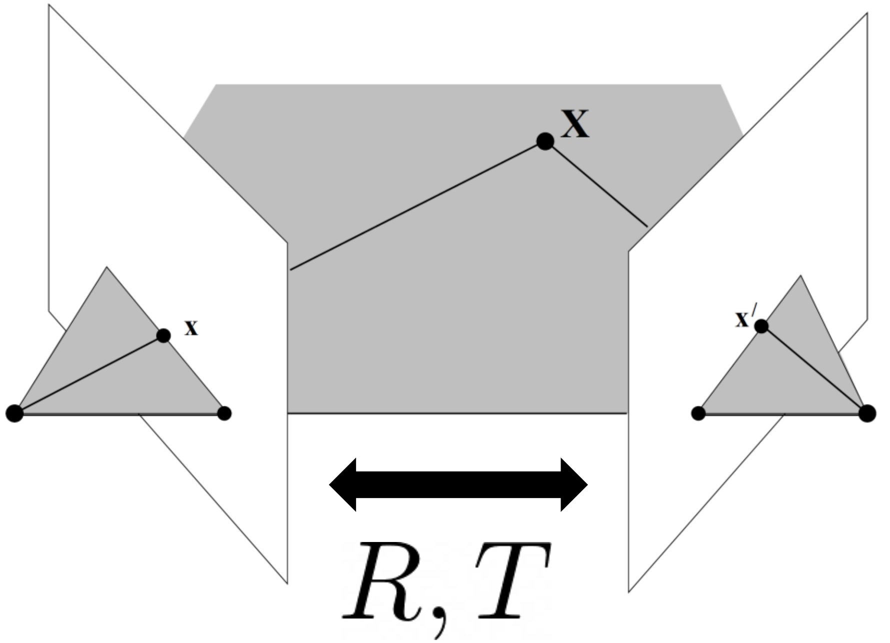
- In 1981 H. C Longuet-Higgins proved that one could recover the relative pose $R$ and $\boldsymbol{t}$ from the essential matrix E up to the scale of $\boldsymbol{t}$




Credit: Thomas Opsahl

H. C Longuet-Higgins, *A computer algorithm for reconstructing a scene from two projections*, 1981

# Triangulation




Estimated from essential matrix E

Intersection of two backprojected lines

$$\mathbf{X} = \mathbf{l} \times \mathbf{l}'$$

How to get the initial estimation $\beta_0$ ?

$$\beta = (\mathbf{X}, \mathbf{R}, \mathbf{T})$$

# Structure from Motion

- Bundle adjustment
  - Iteratively refinement of structure (3D points) and motion (camera poses)

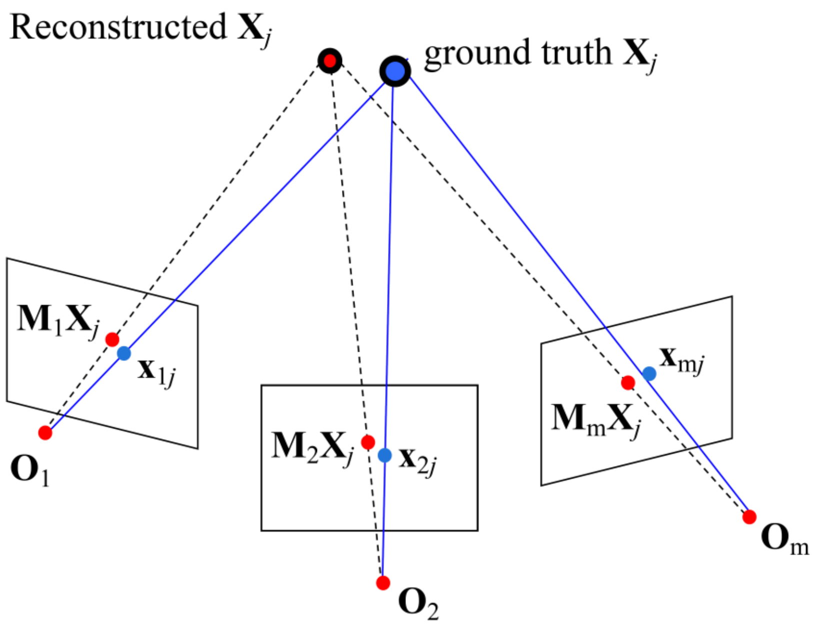
  - Levenberg-Marquardt algorithm

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

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

*predicted* image location

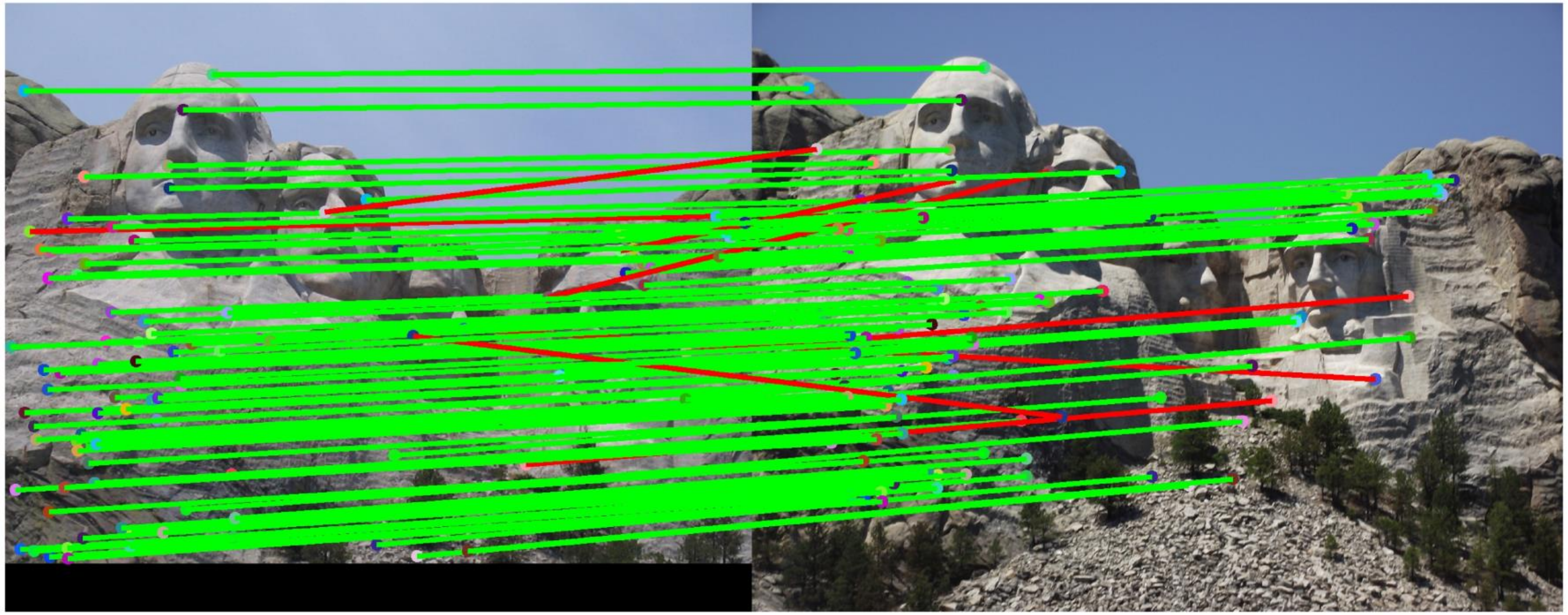
*observed* image location




Examples: [http://vision.soic.indiana.edu/projects/disco/](http://vision.soic.indiana.edu/projects/disco/)

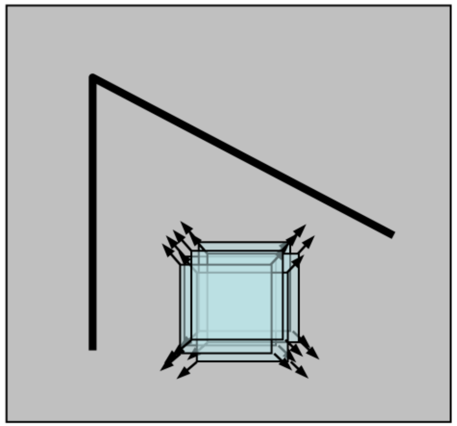
# Basics

- Image feature matching

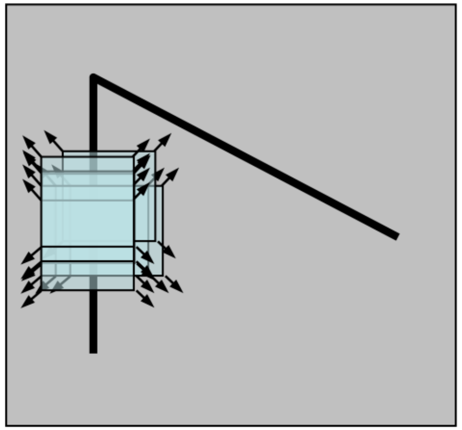


# Harris Corner Detector

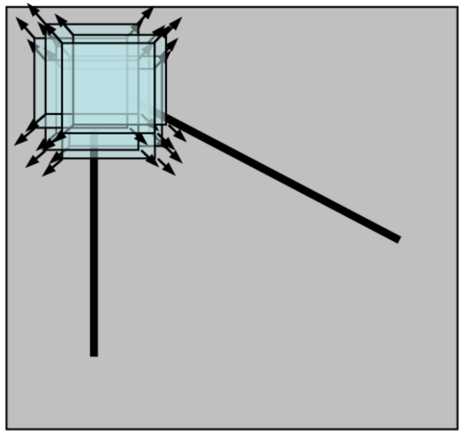
- Corners are regions with large variation in intensity in all directions



"flat" region:
no change in
all directions



"edge":
no change
along the edge
direction



"corner":
significant
change in all
directions

# Harris Corner Detector

$$f(\Delta x, \Delta y) = \sum_{(x_k, y_k) \in W} (I(x_k, y_k) - I(x_k + \Delta x, y_k + \Delta y))^2$$

Taylor expansion

$$I(x + \Delta x, y + \Delta y) \approx I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y$$

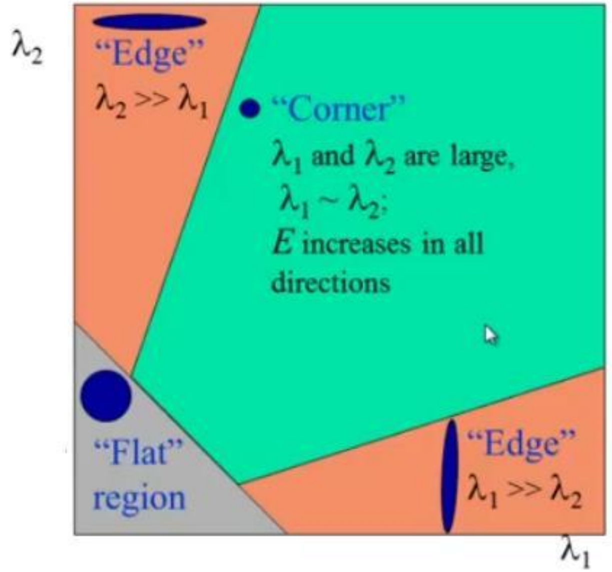
$$f(\Delta x, \Delta y) \approx \sum_{(x,y) \in W} (I_x(x, y)\Delta x + I_y(x, y)\Delta y)^2$$

$$f(\Delta x, \Delta y) \approx \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} M \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

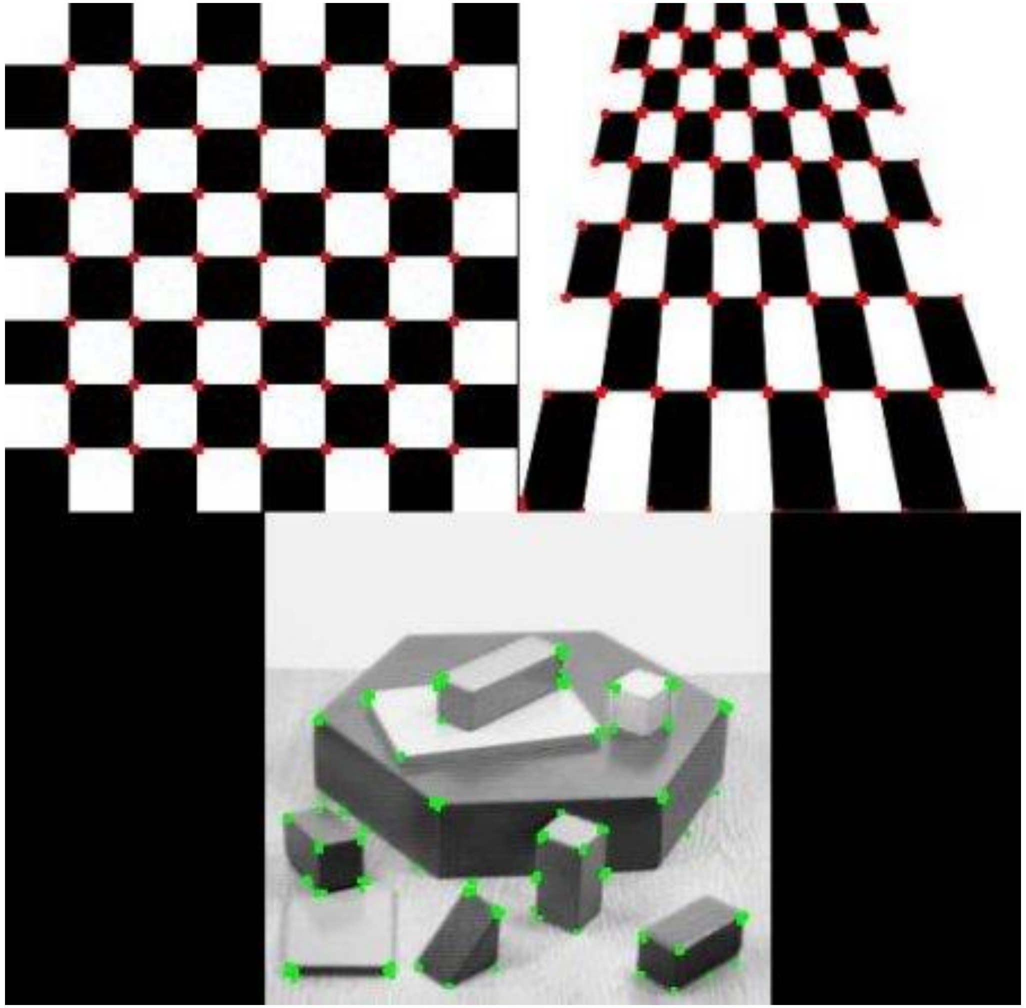
$$M = \sum_{(x,y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_{(x,y) \in W} I_x^2 & \sum_{(x,y) \in W} I_x I_y \\ \sum_{(x,y) \in W} I_x I_y & \sum_{(x,y) \in W} I_y^2 \end{bmatrix}$$

$$R = \det(M) - k(\mathrm{trace}(M))^2$$

- $\det(M) = \lambda_1 \lambda_2$
- $\mathrm{trace}(M) = \lambda_1 + \lambda_2$
- $\lambda_1$ and $\lambda_2$ are the eigenvalues of $M$



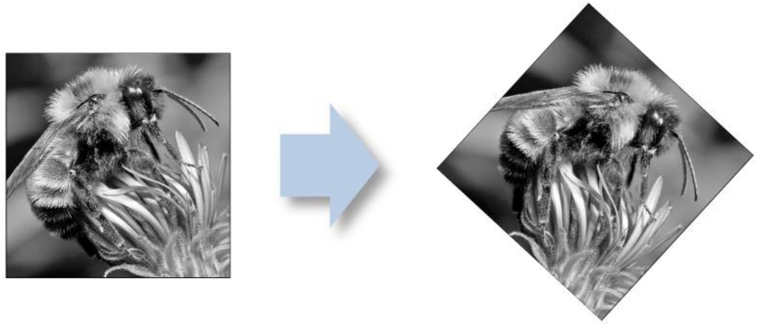
# Harris Corner Detector



https://docs.opencv.org/master/dc/d0d/tutorial_py_features_harris.html

# Invariance

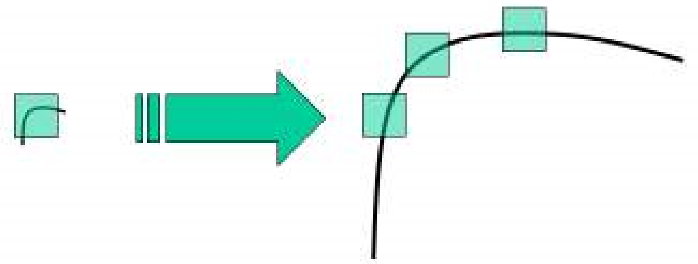
- Can the same feature point be detected after some transformation?
  - Translation invariance
  - 2D rotation invariance

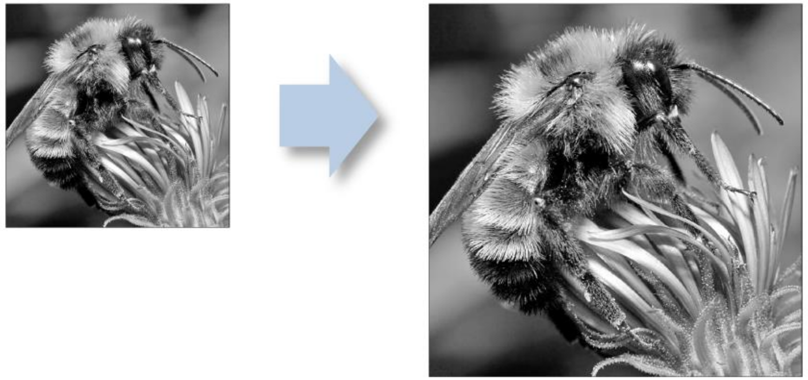


  - Scale invariance

Are Harris corners scale invariance?

No



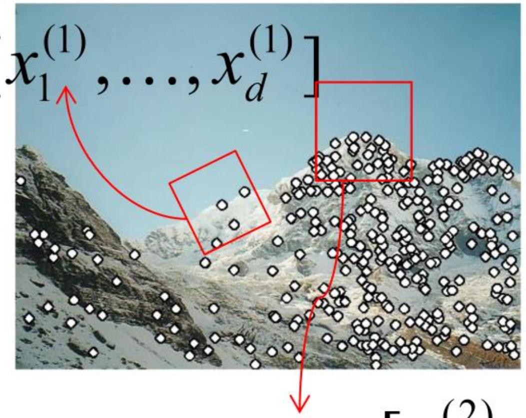


# Scale Invariance Feature Transform (SIFT)

- Keypoint detection

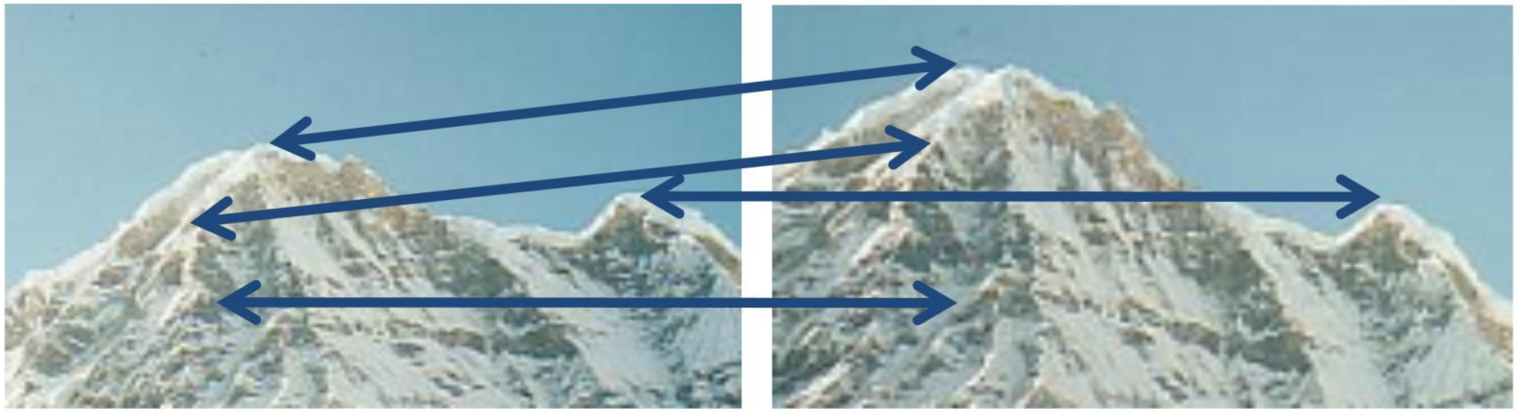
- Compute descriptors

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



$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

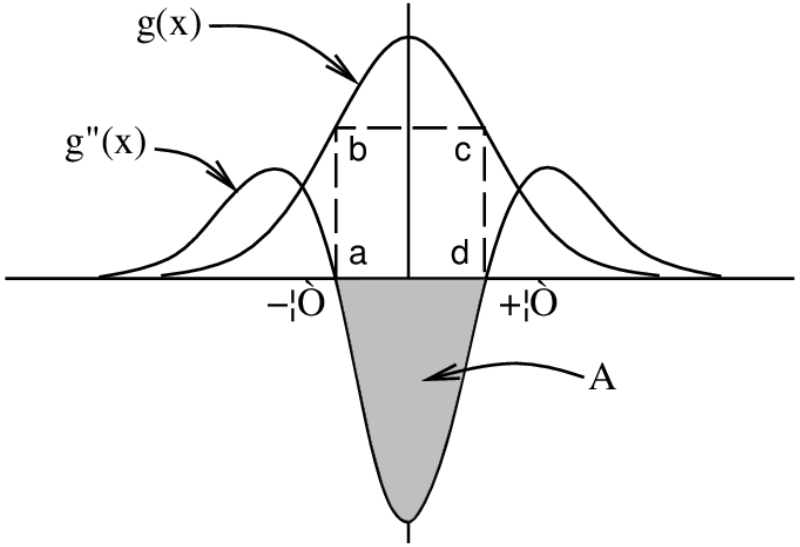
- Matching descriptors

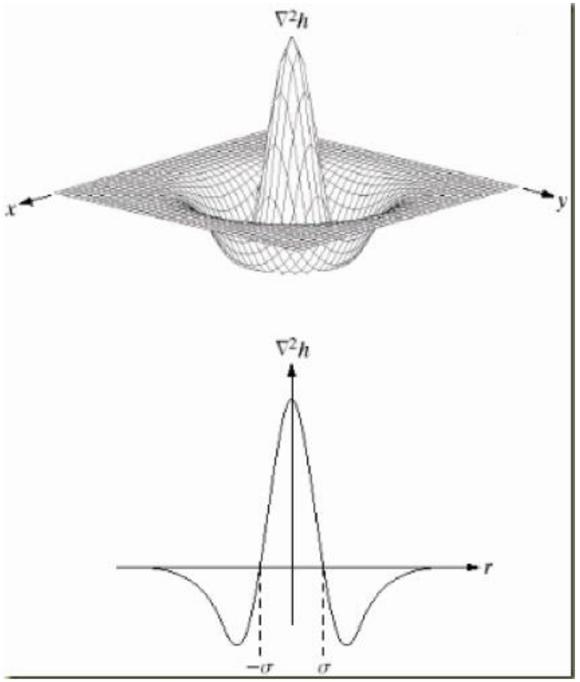


# SIFT: Scale-space Extrema Detection

- How to detect keypoints?
  - E.g., applying a second derivative of Gaussian kernel to an image (Laplacian of Gaussian)

Gaussian $$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

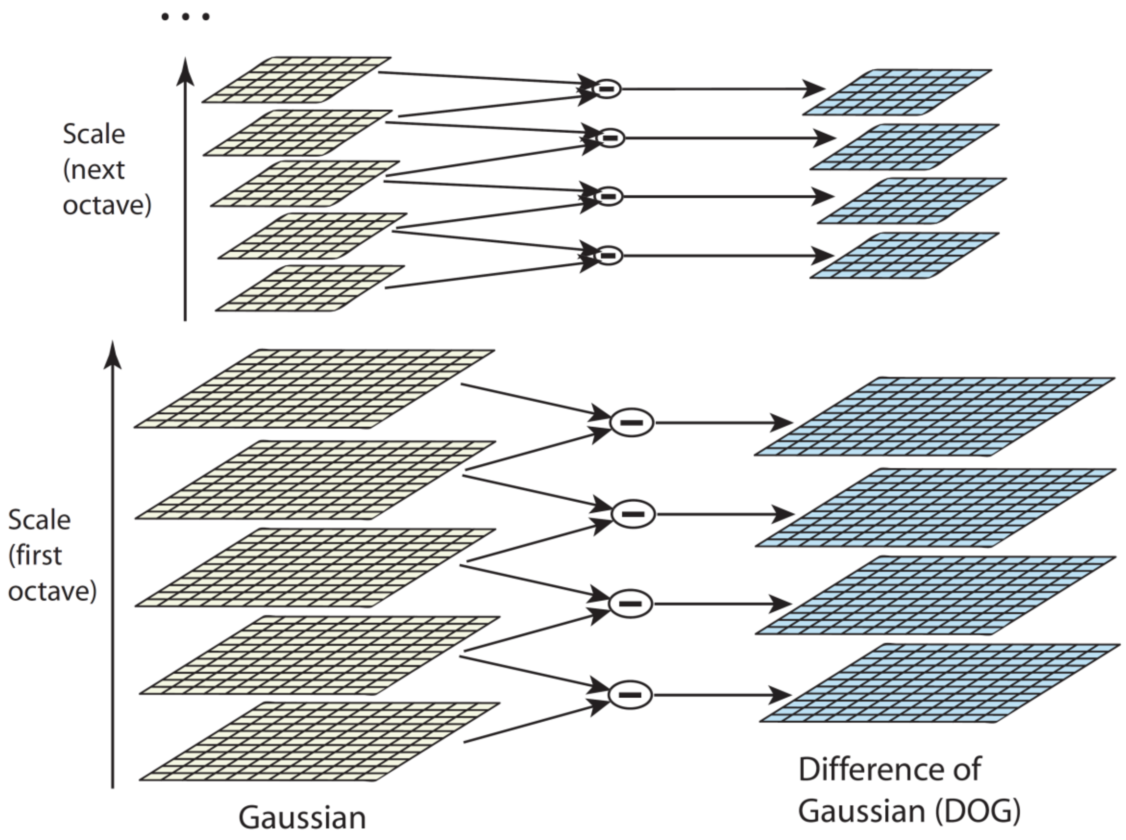


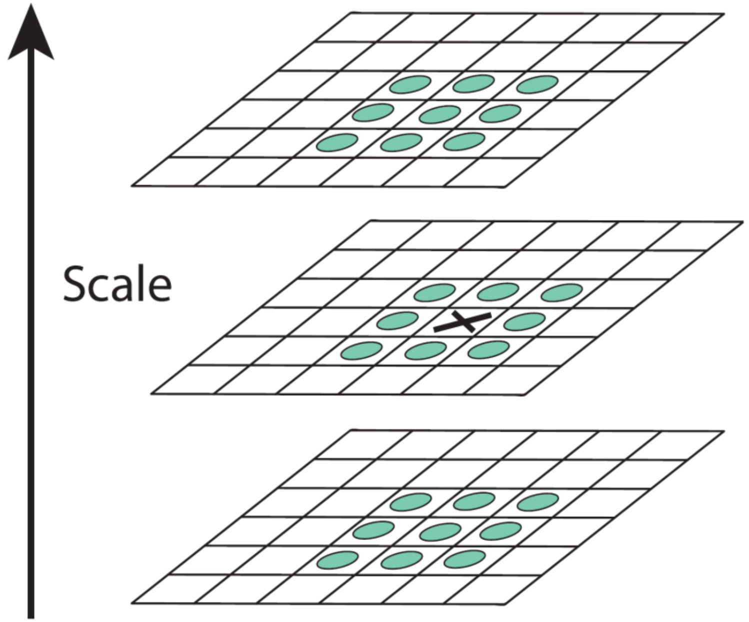



Scale $\sigma$

In pixels, radius of the kernel

# SIFT: Scale-space Extrema Detection




Maxima and minima of DOG images

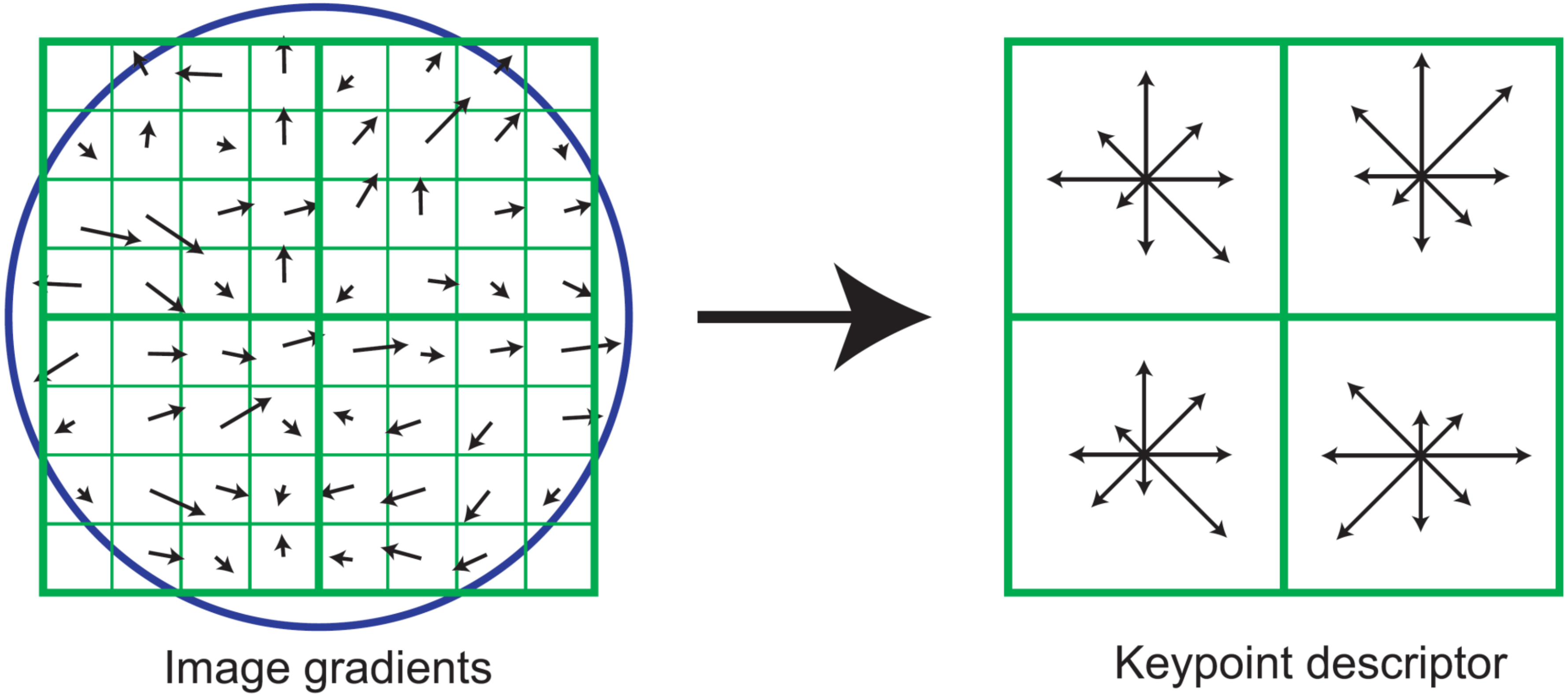
$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

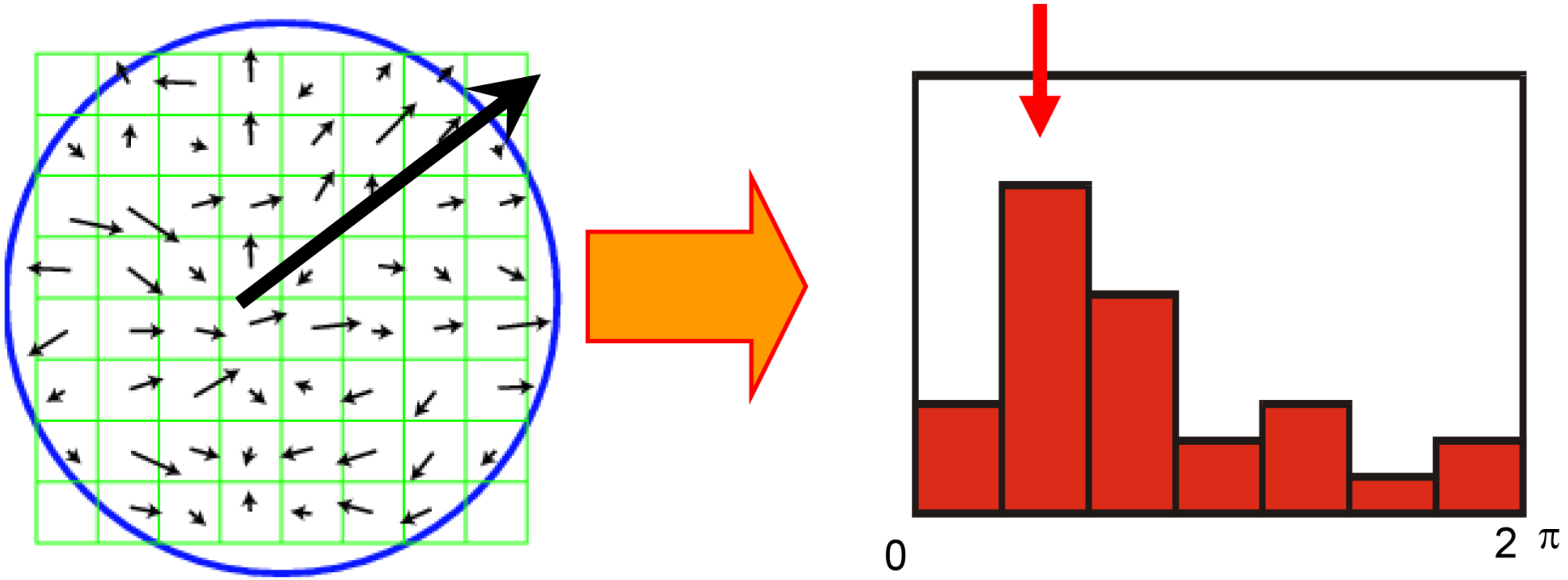
# SIFT Descriptor

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

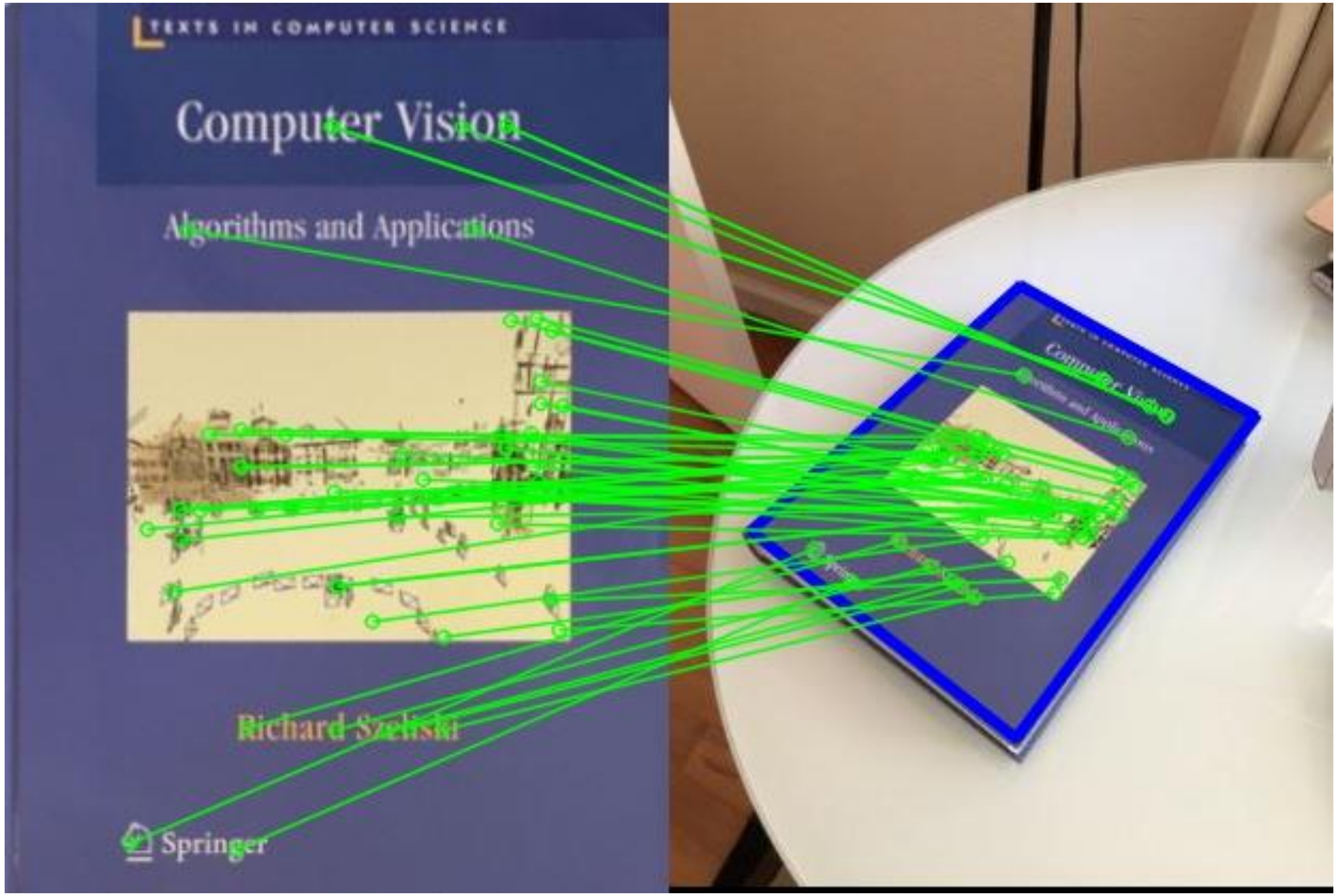


# SIFT: Rotation Invariance

- Rotate all orientations by the dominant orientation

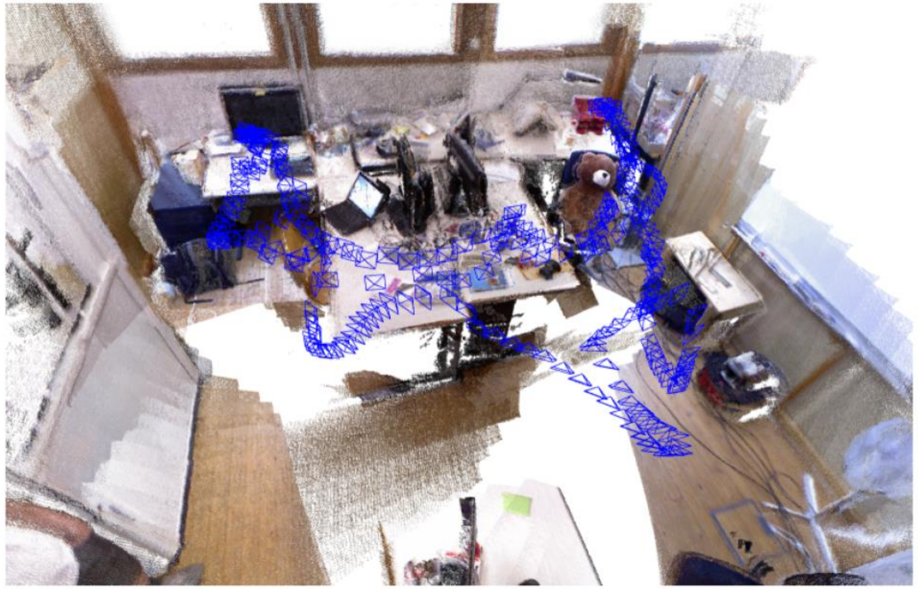


# SIFT Matching Example



# Simultaneous Localization and Mapping (SLAM)

- Localization: camera pose tracking
- Mapping: building a 2D or 3D representation of the environment
- The goal here is the same as structure from motion, usually with video input



ORB-SLAM2
- Point cloud and camera poses

# ORB-SLAM

- Oriented FAST and Rotated BRIEF (ORB)
- Tracking camera poses
  - Motion only Bundle Adjustment (BA)
- Mapping
  - Local BA around camera pose
- Loop closing
  - Loop detection

https://webdiis.unizar.es/~raulmur/orbslam/

# 3D Scanning

- Using laser to create "point clouds"


(a) (b)

Figure 9.26: (a) The Afinia ES360 scanner, which produces a 3D model of an object while it spins on a turntable. (b) The Focus3D X 330 Laser Scanner, from FARO Technologies, is an outward-facing scanner for building accurate 3D models of large environments; it includes a GPS receiver to help fuse individual scans into a coherent map.

# 3D Scanning


https://matterport.com/

# Further Reading

- Section 9.5, Virtual Reality, Steven LaValle
- SIFT: Distinctive Image Features from Scale-Invariant Keypoints, David Lowe, IJCV'04
- ORB-SLAM: ORB-SLAM: a Versatile and Accurate Monocular SLAM System, Mur-Artal et al., T-RO'15