Principles of Robot Autonomy I

PnP, SfM, RANSAC

Stanford University

MSL

IPRL

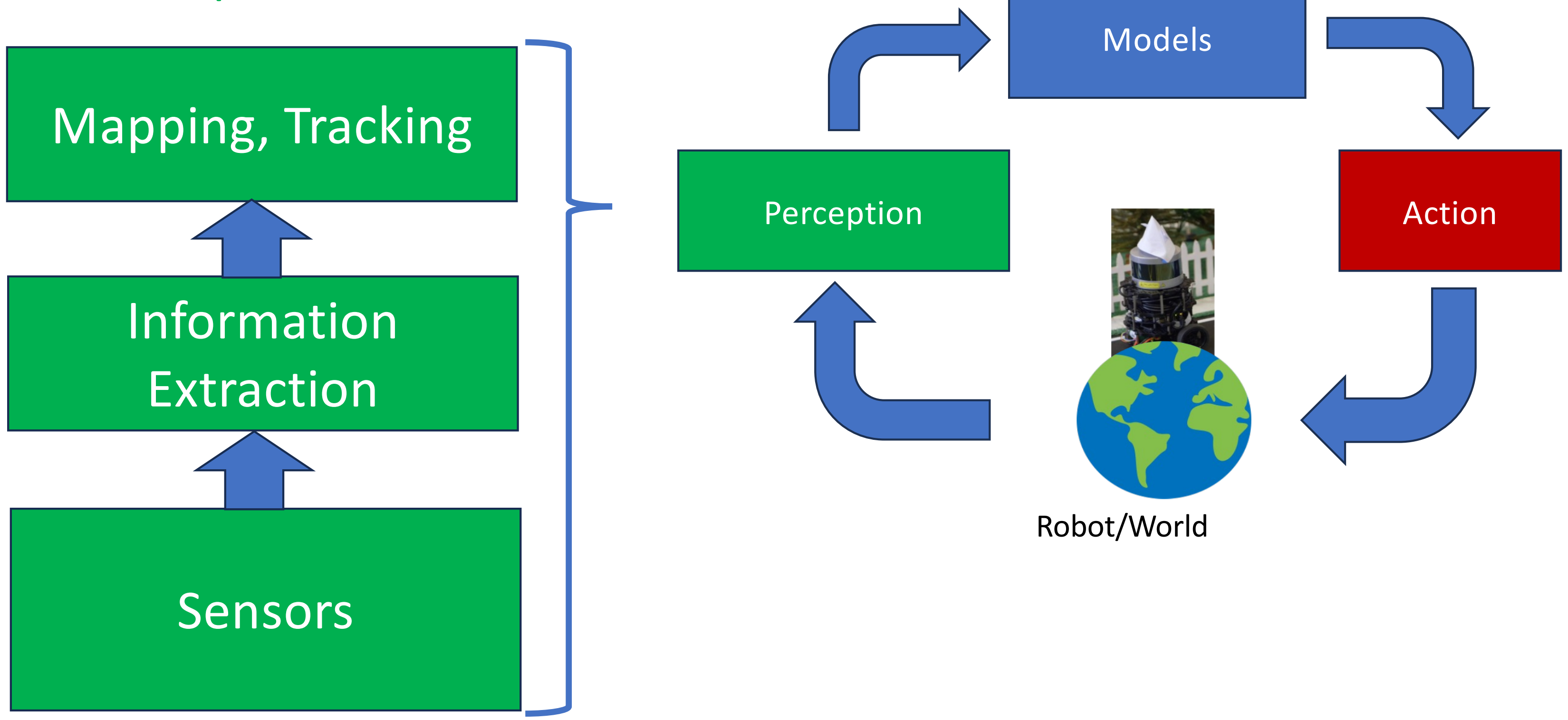
ASL Autonomous Systems Lab

Logistics

- Homework 3: due Tues, Oct 28
- Homework 4: out Tues Nov 4 (After the Midterm)
- Midterm window: Wed, Oct 29, 5pm – Sat, Nov 1
  - Take home, 72 hour window
  - Check out exam on gradescope
  - You will have personal 5 hour time slot
  - Open notes, book, HW solutions
  - No internet, no GenAI, no working with others
- Lecture 9:
  - N point perspective problem (PnP)
  - Structure from Motion (SfM)
  - RANSAC and feature correspondences

Robot Perception

See: Perception Stack



Perception Stack: Computer vision, filtering, SLAM

Use sensor data to update the models

Localization, Mapping, Tracking:

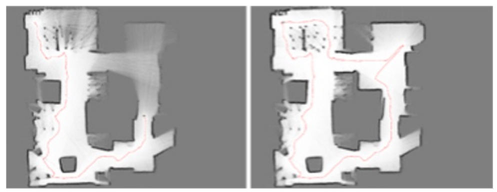
- EKF/Monte Carlo localization
- Occupancy grid mapping
- Pose graph optimization
- Tracking (EKF and Particle Filter)
- **AA273: Filtering (Schwager)**
- **AA275: Navigation (Gao)**

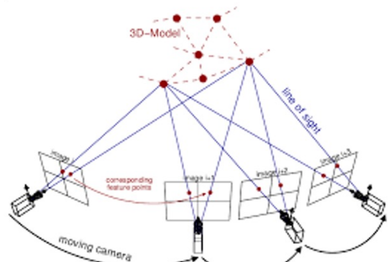
Information extraction

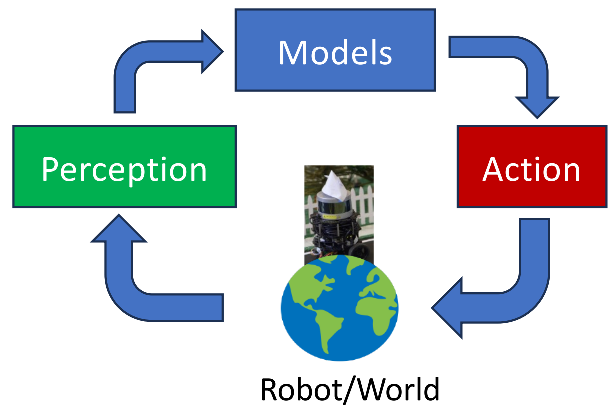
- Computer vision: features, correspondences, Structure from Motion (SfM), depth
- Lidar scan matching, ICP
- **CS231A: Comp Vision**

Sensors:

- RGB Cameras, RGB-D/stereo cameras, Lidar
- IMU, GPS, wheel encoders



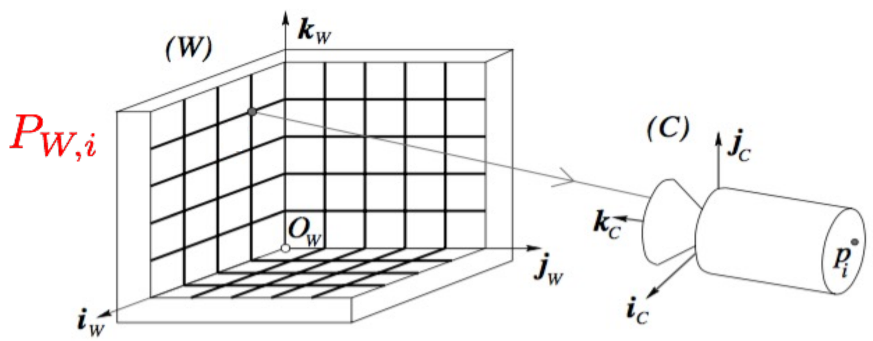




Recall: Camera calibration

Goal: find the intrinsic and extrinsic parameters of the camera

- Known 3D world and 2D images coordinates
- Known correspondences between them



Credit: FP Chapter 1

$$\overbrace{\boxed{p^h = K[R \quad t]P_W^h}}^{\text{Projection matrix } M}$$

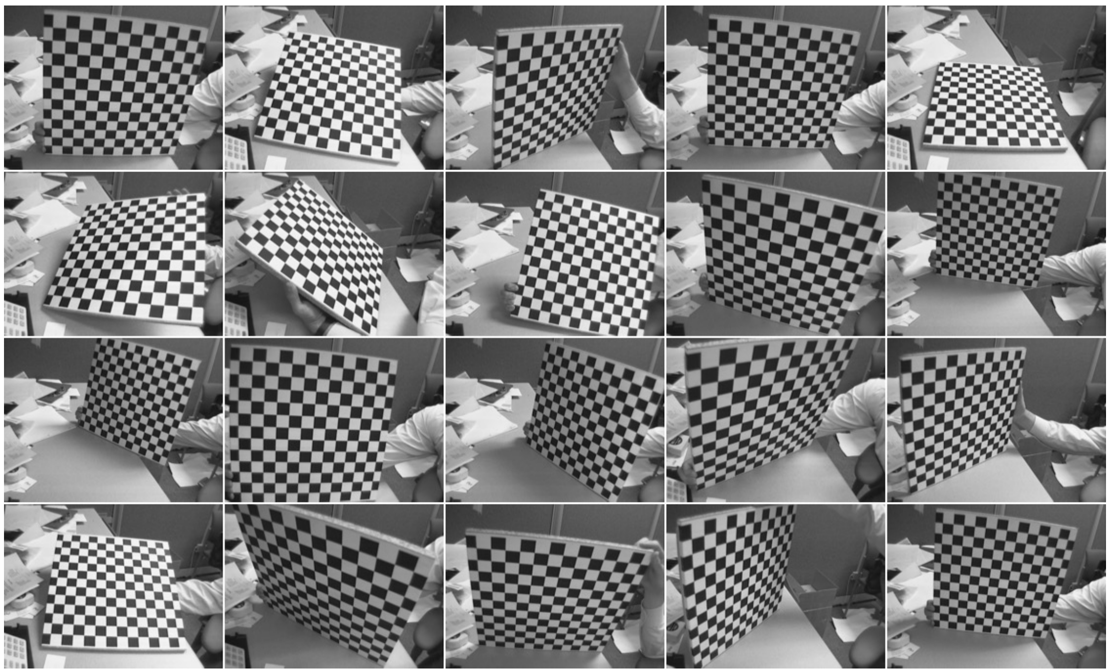
From world frame to camera frame (inverse cam pose)

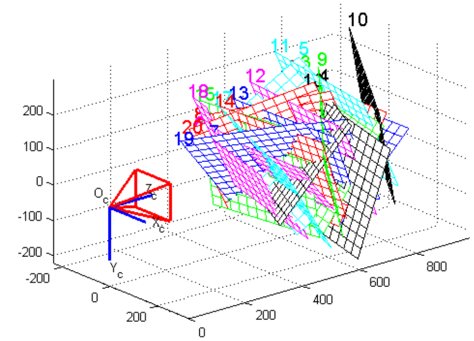
$$\min_{m \in R^{12}} \quad \|\tilde{P}m\|^2$$

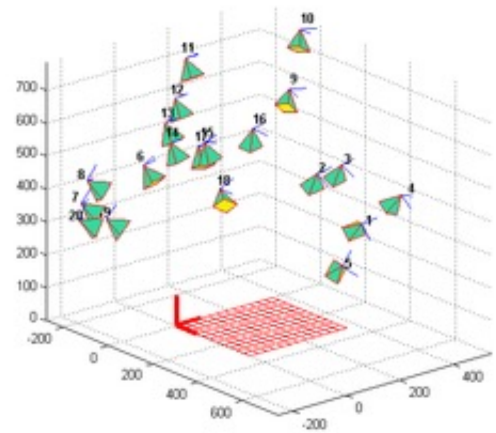
$$\text{subject to} \quad \|m\|^2 = 1$$

Solution: take SVD and find singular vector with smallest singular value.

Examples for Calibration Images







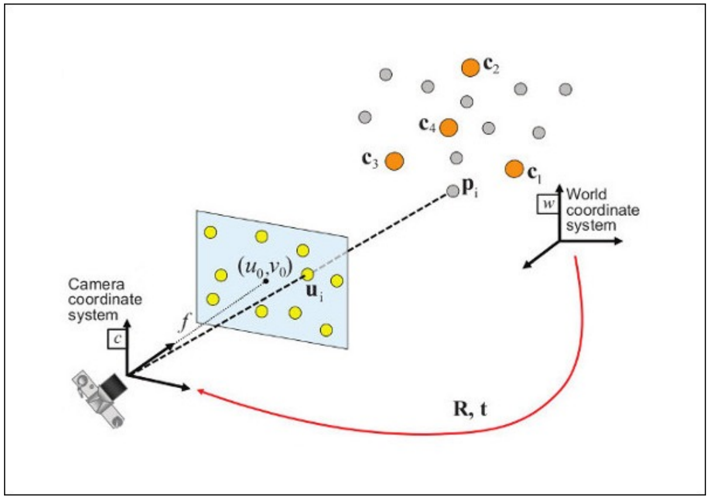
- Due to co-planarity of cal board, need multiple images from different poses
- Origin of world frame defined by board lower right corner

Source: Wikipedia

Perspective-n-Point (PnP) Problem (e.g. for object pose tracking)

Goal: find 3D object pose with respect to camera

- Known 3D coords of object points in its body frame, 2D image coordinates, and known correspondences
- Known camera calibration matrix K
- Same solution technique as camera calibration!



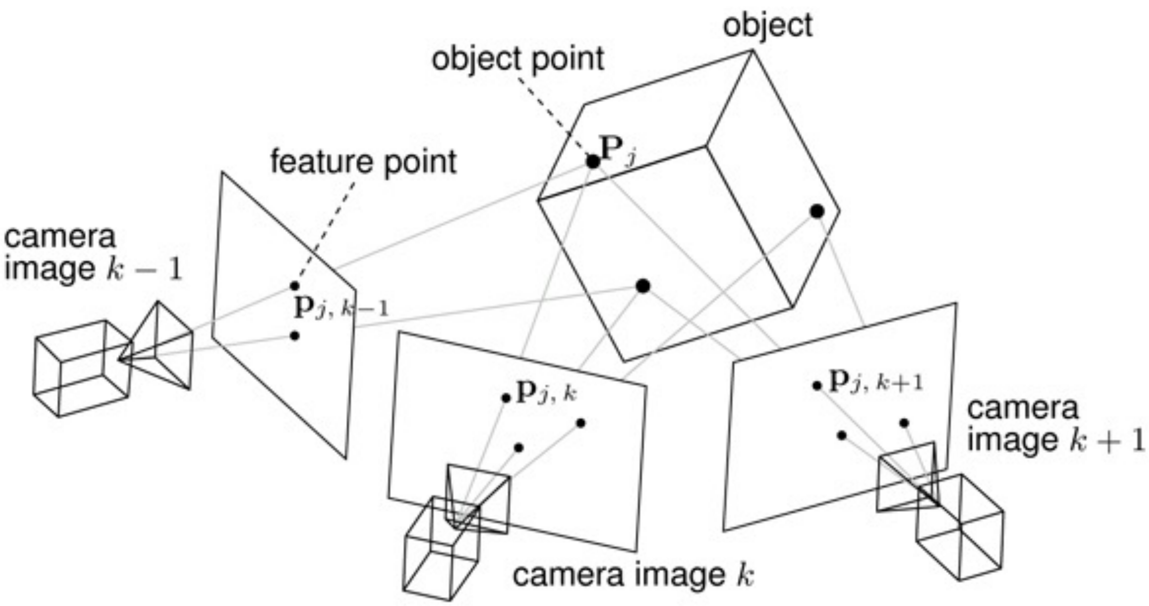
$$\boxed{K^{-1}p^h = \underbrace{[R \quad t]}_{\textit{Pose of object wrt camera}} P_W^h}$$

$$\min_{m \in R^{12}} \quad \|\tilde{P}m\|^2$$
$$\text{subject to} \quad \|m\|^2 = 1$$

- $\tilde{P}$ and m defined with K^{-1} on LHS.

Solution: take SVD and find singular vector with smallest singular value.

Structure from motion (SFM)



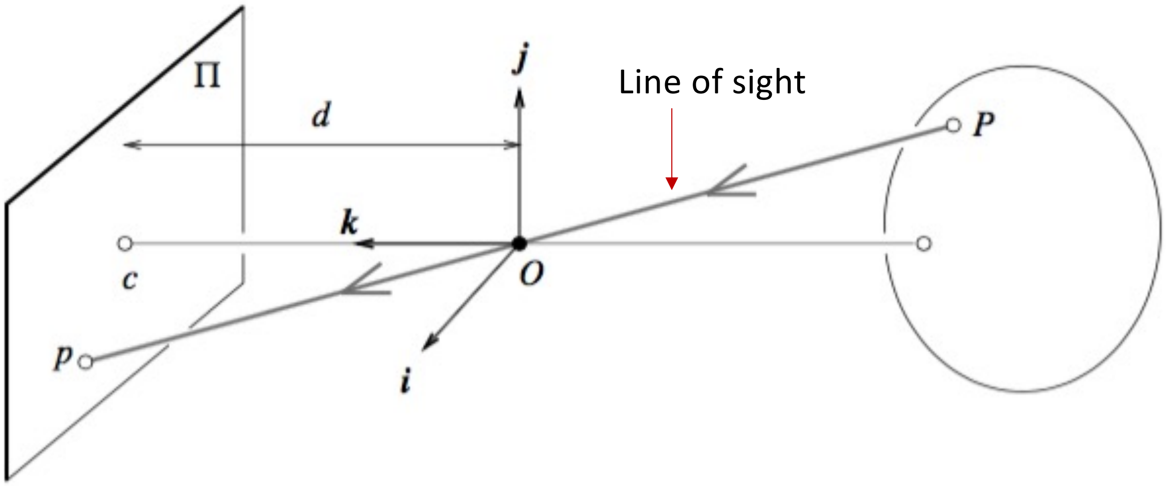
Given *m* images of *n* fixed 3D points

$$p_{j,k}^h = M_k P_j^h$$

Find:

- *m* projection matrices M_k (motion)
- *n* 3D points P_j (structure)

Measuring depth



$$p^h = K[R \quad t]P_W^h$$

Homogeneous coordinates

- Once the camera is calibrated, can we measure the location of a point *P* in 3D given its known observation *p*?
- No: one can only say that *P* is located *somewhere* along the line joining *p* and *O*!

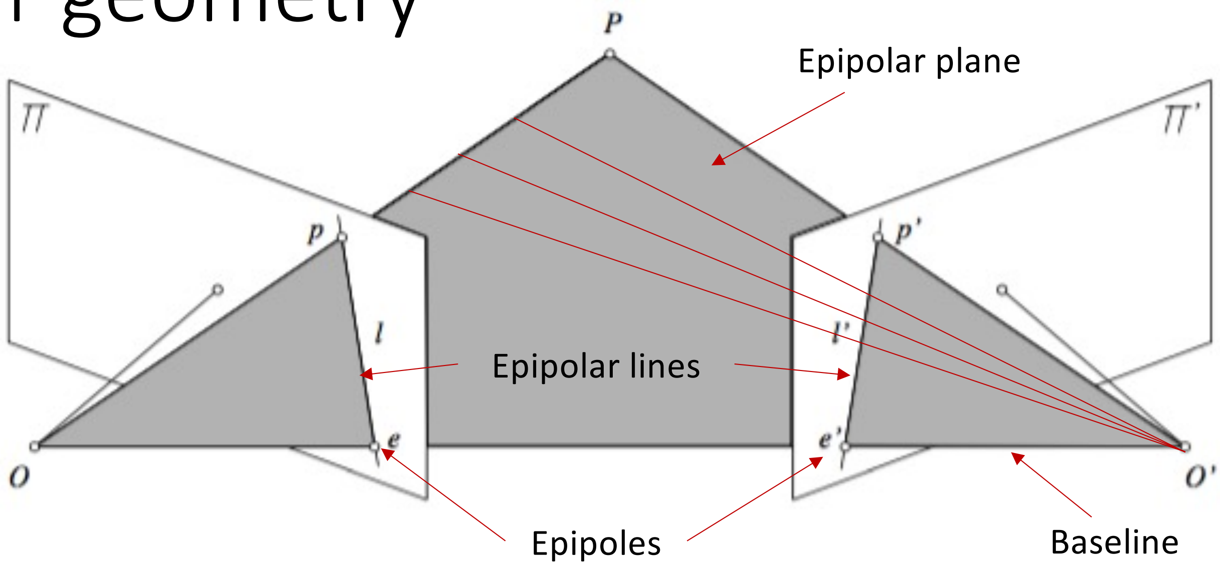
Issues with recovering structure (3D point locations in world frame)

Recovering structure

- Structure: 3D point locations to be reconstructed by having access to only 2D images

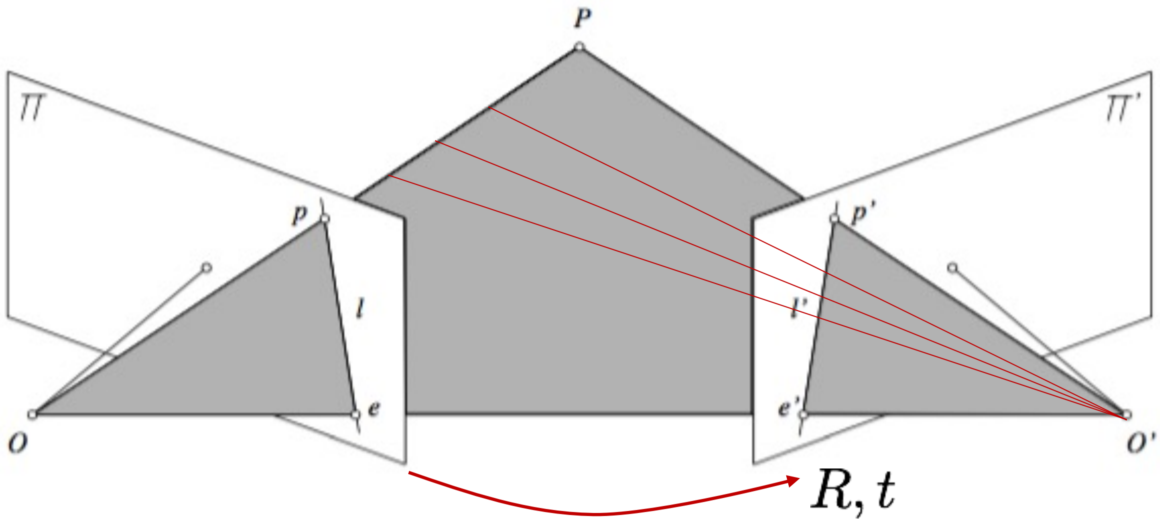
- Common methods
  1. Structure from motion (SfM): processes two images taken with the same or different cameras at *different times* and from different *unknown* positions
  2. Stereo vision: processes two distinct images taken at the *same time* and assumes that the relative pose between the two cameras is *known*
  3. Monocular depth estimation: Deep learned from context using a history of depth images (learned scale of objects, lighting queues, function queues, ect)
  4. Depth from focus: determines distance to one point by taking multiple images with better and better focus

Epipolar geometry



- Consider image coords p and p' of a point P in world frame observed by two cameras centered at O, O'
- These five points all belong to the *epipolar plane* defined by p, O, O', or equivalently, p', O, O'
- Epipolar constraint: potential matches for p must lie on epipolar line l' (and vice-versa)

Epipolar constraint derivation



- Epipolar constraint: $\overline{O'p'}$, $\overline{Op}$, and $\overline{O'O}$ (when expressed in that same frame) must be co-planar, or

$$\overline{O'p'} \cdot [\overline{O'O} \times \overline{Op}] = 0$$

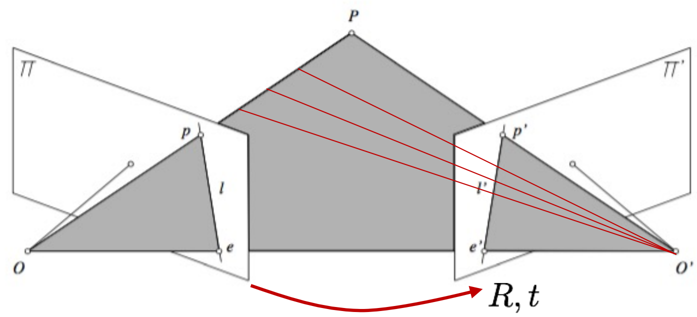
Aside: matrix notation for cross product

- Cross product can be expressed as the product of a skew-symmetric matrix and a vector

$$a \times b = \underbrace{\begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}}_{:= [a]_\times} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = [a]_\times b$$

Epipolar constraint derivation

(more details: CS231a)



- Translate into vector expressions, all in frame O'

$$\overline{O'p'} \cdot [\overline{O'O} \times \overline{Op}] = 0$$

$$(K'^{-1}p')^T [t]_\times (RK^{-1}p) = 0$$

$$p'^T F p = 0 \qquad \text{where} \qquad F = K'^{-T}[t]_\times R K^{-1}$$

Fundamental matrix

Key facts

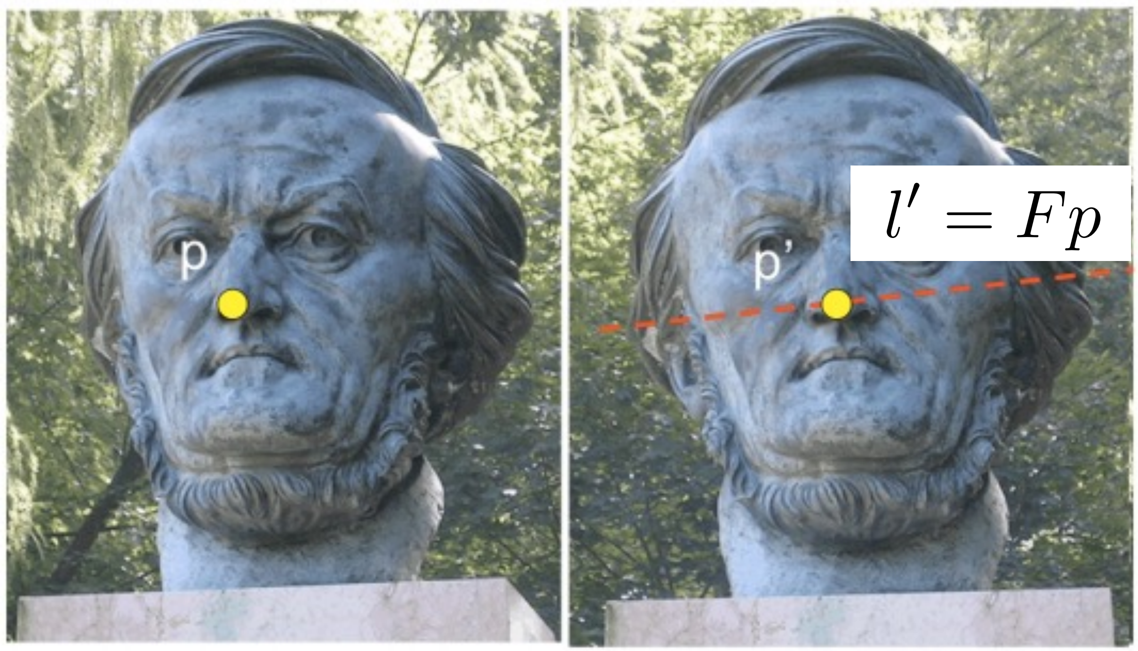
- F is referred to as the fundamental matrix

- $l' = Fp$ (resp. $l = F^T p'$) represents the epipolar line corresponding to the point p (resp. p') in the second (resp. first) image. This exploits the homogenous notation for lines.

- $Fe = F^T e' = 0$ → F is also singular (as t is parallel to the coordinate vectors of the epipoles)

- F has 7 DoF (9 elements – common scaling – det(F)=0)

Usefulness of fundamental matrix



- Assume *F* is given
- Given a point in image 1, one can compute the corresponding epipolar line in image 2 without any additional information needed!

Estimating the fundamental matrix

- 8-point algorithm

$$p = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad p' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \qquad \Rightarrow \begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix} u'u & v'u & u & u'v & v'v & v & u' & v' & 1 \end{bmatrix} \underbrace{\begin{bmatrix} F_{11} \\ F_{21} \\ F_{31} \\ F_{12} \\ F_{22} \\ F_{32} \\ F_{13} \\ F_{23} \\ F_{33} \end{bmatrix}}_{f} = 0 \qquad \Rightarrow \underbrace{W}_{n\text{x}9 \text{ matrix of known coefficients}} f = 0$$

- Given $n \geq 8$ correspondences, one then solves

$$\begin{aligned} \min_{f \in R^9} \quad & \|Wf\|^2 \\ \text{subject to} \quad & \|f\|^2 = 1 \end{aligned} \Rightarrow \tilde{F}$$

Enforcing the rank constraint

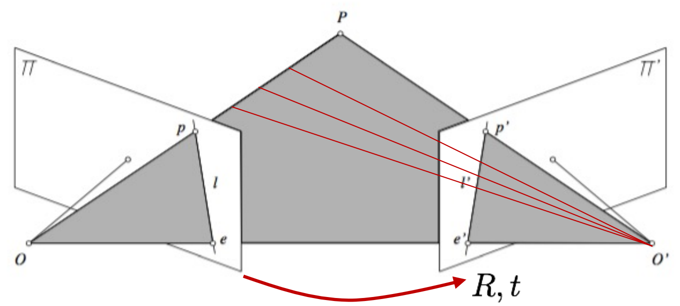
- $\tilde{F}$ satisfies the epipolar constraints, but is not necessarily singular (hence, is not necessarily a proper fundamental matrix)
- Enforce rank constraint (again, via SVD decomposition)

$$\begin{aligned} \text{Find } F \text{ that minimizes} \quad & \|F - \tilde{F}\|^2 \\ \text{subject to} \quad & \det(F) = 0 \end{aligned}$$

Frobenius norm

- 8-point algorithm
  1. Use least norms to compute $\tilde{F}$ by taking SVD of W, and finding right singular vector with smallest singular value.
  2. Enforce rank constraint on $\tilde{F}$ by taking SVD and setting smallest two singular values to zero, then re-multiply $\tilde{F} = U\Sigma V^T$

Obtaining relative camera pose



- Recall:

$$F = K'^{-T}[t]_\times RK^{-1}$$

- To get (R, t) we compute

$$[t]_\times R = K'^T FK$$

- Use RQ decomposition to get

$$[t]_\times R \to ([t]_\times, R)$$

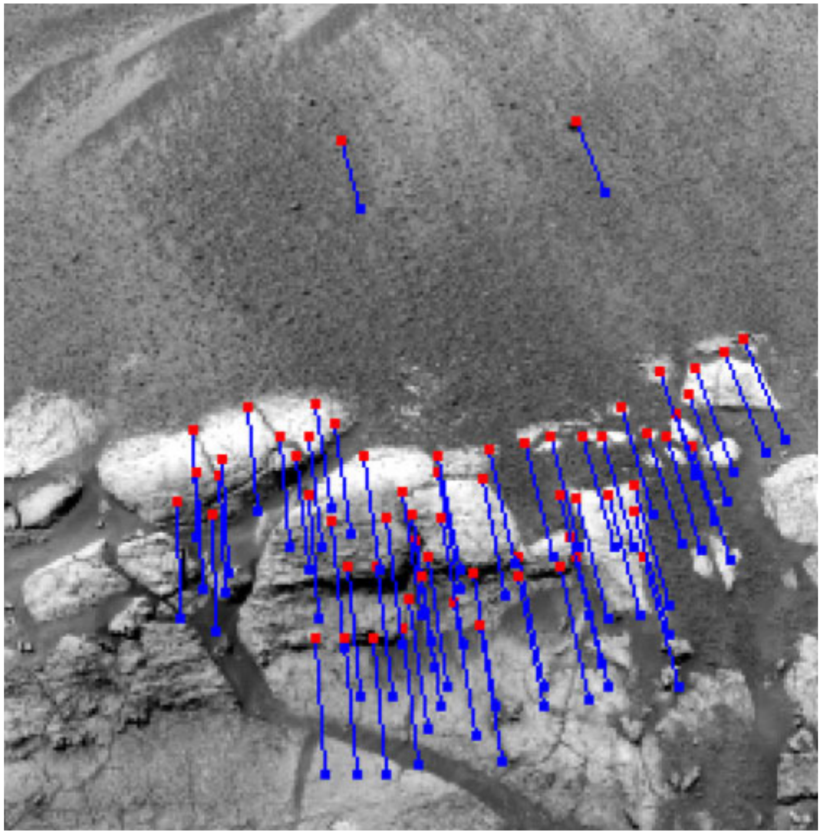
$$[t]_\times \to t$$

Solution to SFM problem (high-level)

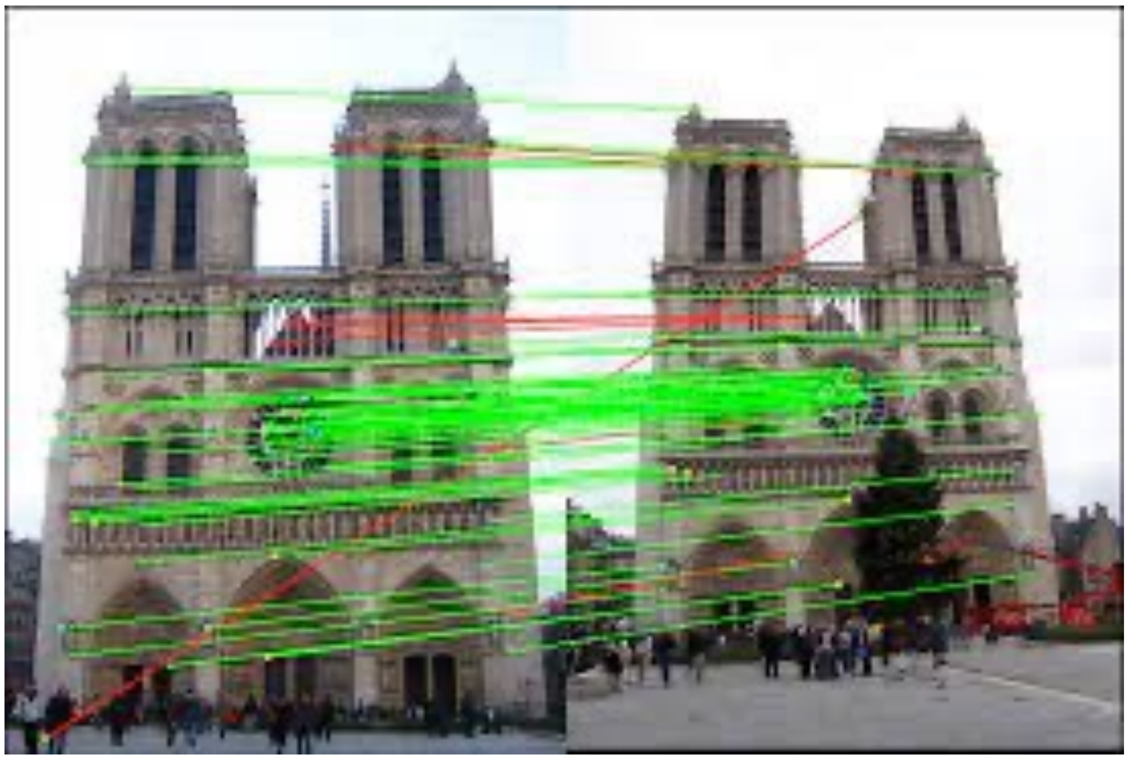
- Several approaches available:
  - Algebraic approach (by fundamental matrix)
  - Bundle adjustment (large-scale non-convex optimization problem to find [R t] camera poses for each image, and 3D points (i.e., structure) in common world frame
- Algebraic approach (2-views) – gives relative poses between two successive views, and 3D points relative to first view
  1. Compute fundamental matrix F (e.g., via 8-point algorithm)
  2. Use F to estimate camera projection matrices
  3. Use camera projection matrices for triangulation
- Used as ”front end” in visual SLAM, to obtain pose graph for PGO back end

Application of SFM: visual odometry

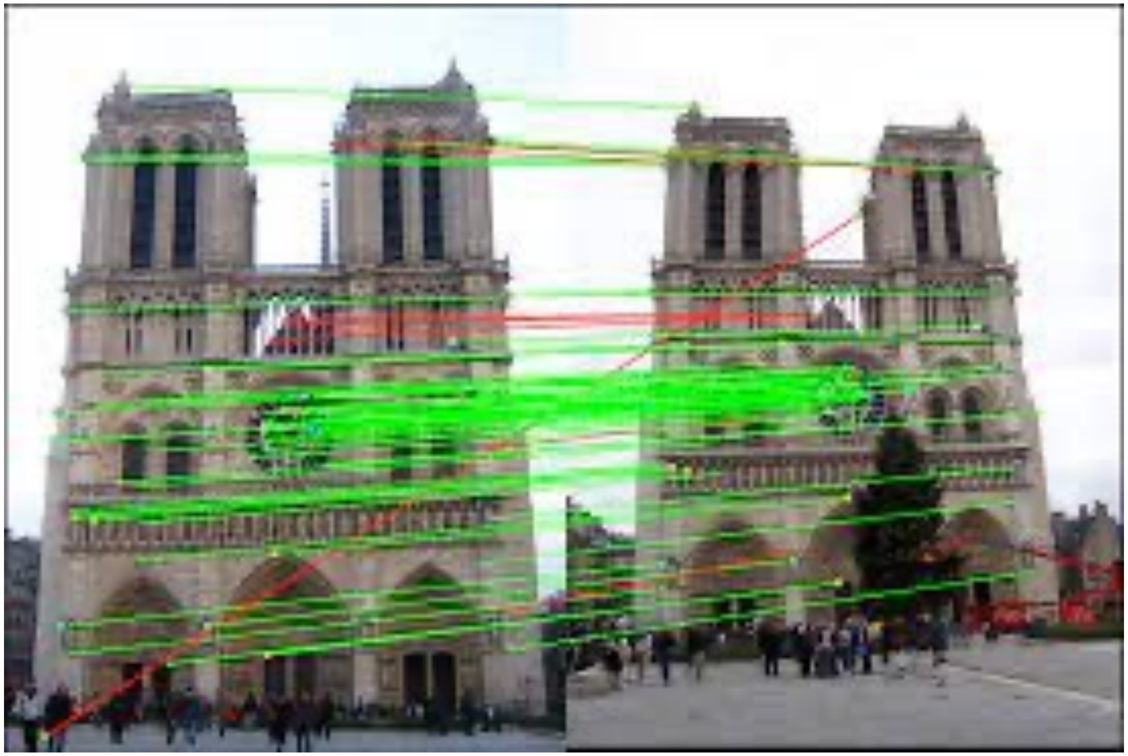
- SLAM/Visual Odometry: estimate the motion of the robot by using visual input (and possibly additional information)
  - Single camera: absolute scale must be estimated in other ways
  - Stereo camera: measurements are directly provided in absolute scale



Elephant in the room: how to find correspondences for PnP, stereo, or SfM?

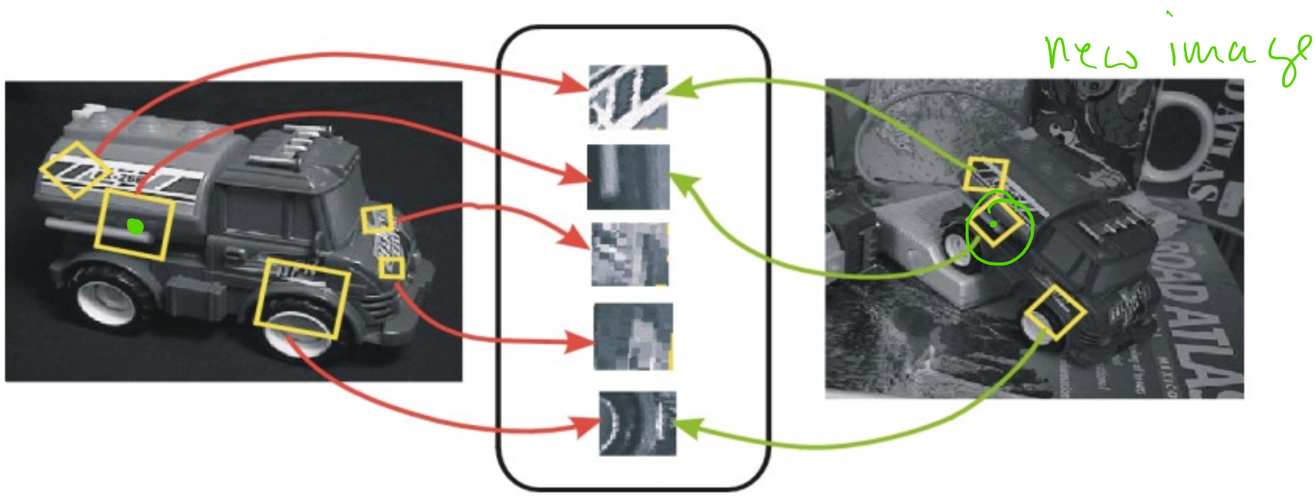


Features, feature descriptors, feature matching, RANSAC



Keypoint Features and their Descriptors

- Goal: *describe* keypoints so that we can compare them across images or use them for object detection or matching
- Desired properties:
  - Invariance with respect to pose, scale, illumination, etc.
  - Distinctiviness

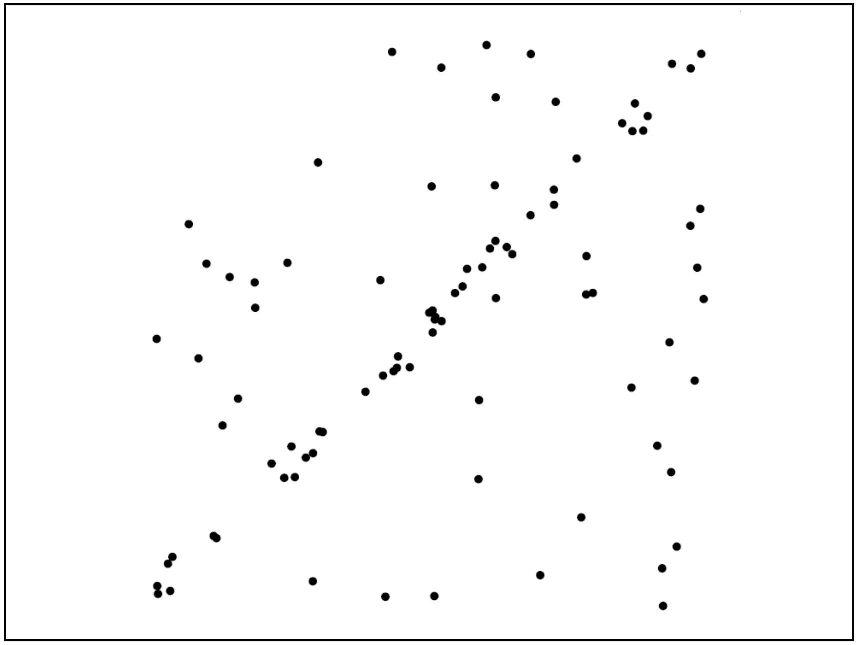


RANSAC iterations

- In principle, one would need to check all possible combinations of 2 points in dataset
- If $|S| = N$, number of combinations is $\frac{N(N-1)}{2}$ → too many
- However, if we have a rough estimate of the percentage of inliers, we do not need to check all combinations…

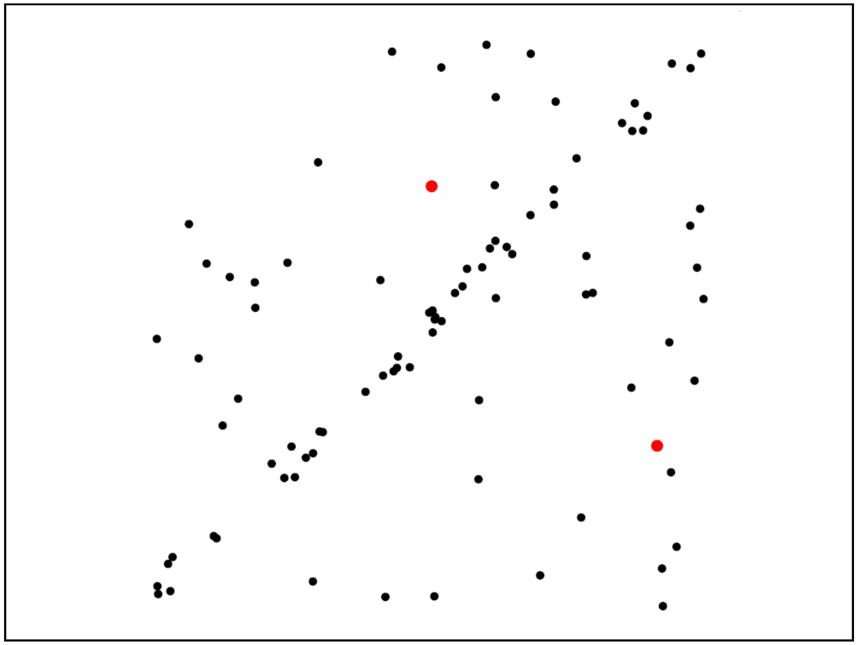
E.g., RANSAC for line fitting (for simplicity)

Data: Set S consisting of all N points
Result: Set with maximum number of inliers
(and corresponding fitting line)
while $i \leq k$ **do**
randomly select 2 points from S;
fit line l_i through the 2 points;
compute distance of all other points to line l_i ;
construct *inlier* set, i.e., count number of
points with distance to the line less than γ;
store line l_i and associated set of inliers;
$i \leftarrow i + 1$
end
Choose set with maximum number of inliers



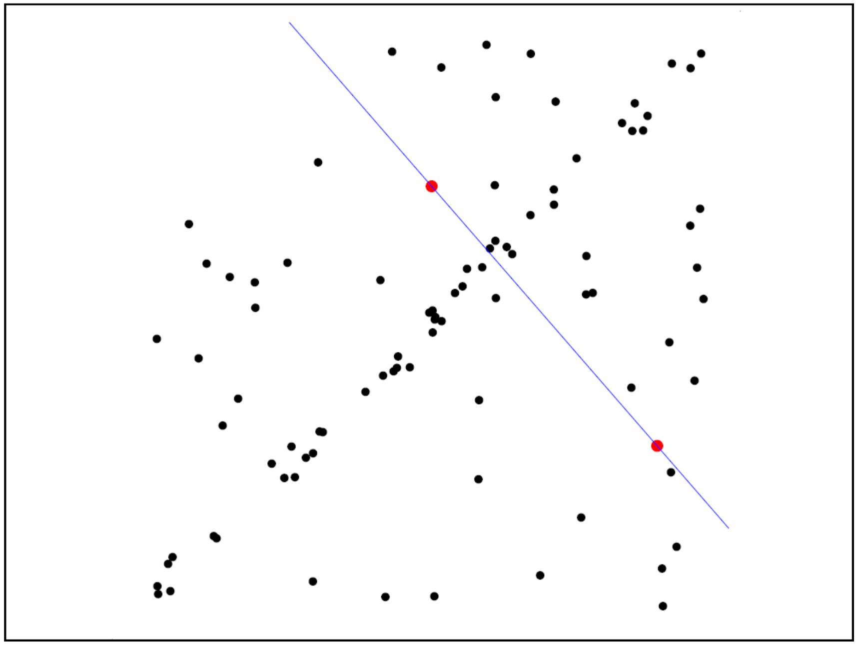
RANSAC

Data: Set S consisting of all N points
Result: Set with maximum number of inliers
(and corresponding fitting line)
while $i \leq k$ **do**
randomly select 2 points from S;
fit line l_i through the 2 points;
compute distance of all other points to line l_i ;
construct *inlier* set, i.e., count number of
points with distance to the line less than γ;
store line l_i and associated set of inliers;
$i \leftarrow i + 1$
end
Choose set with maximum number of inliers



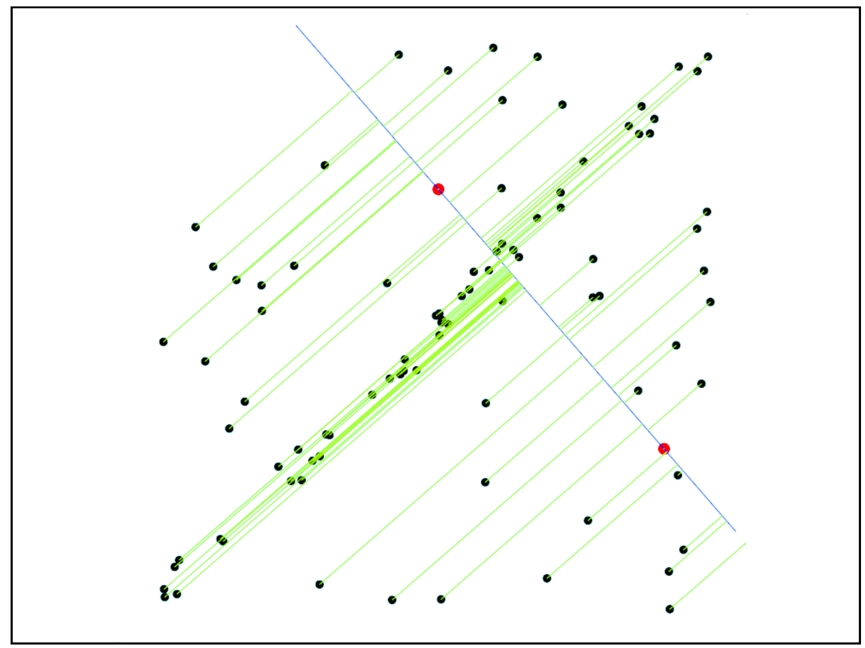
RANSAC

Data: Set S consisting of all N points
Result: Set with maximum number of inliers
  (and corresponding fitting line)
while $i \leq k$ **do**
  randomly select 2 points from S;
  fit line l_i through the 2 points;
  compute distance of all other points to line l_i ;
  construct *inlier* set, i.e., count number of
    points with distance to the line less than γ;
  store line l_i and associated set of inliers;
  $i \leftarrow i + 1$
end
Choose set with maximum number of inliers



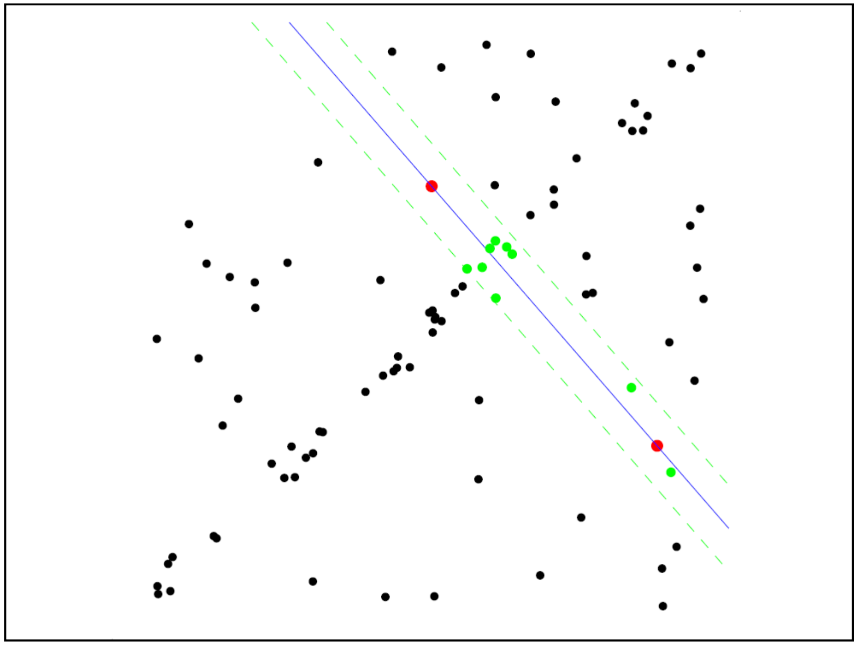
RANSAC

Data: Set S consisting of all N points
Result: Set with maximum number of inliers
  (and corresponding fitting line)
while $i \leq k$ **do**
  randomly select 2 points from S;
  fit line l_i through the 2 points;
  compute distance of all other points to line l_i ;
  construct *inlier* set, i.e., count number of
    points with distance to the line less than γ;
  store line l_i and associated set of inliers;
  $i \leftarrow i + 1$
end
Choose set with maximum number of inliers



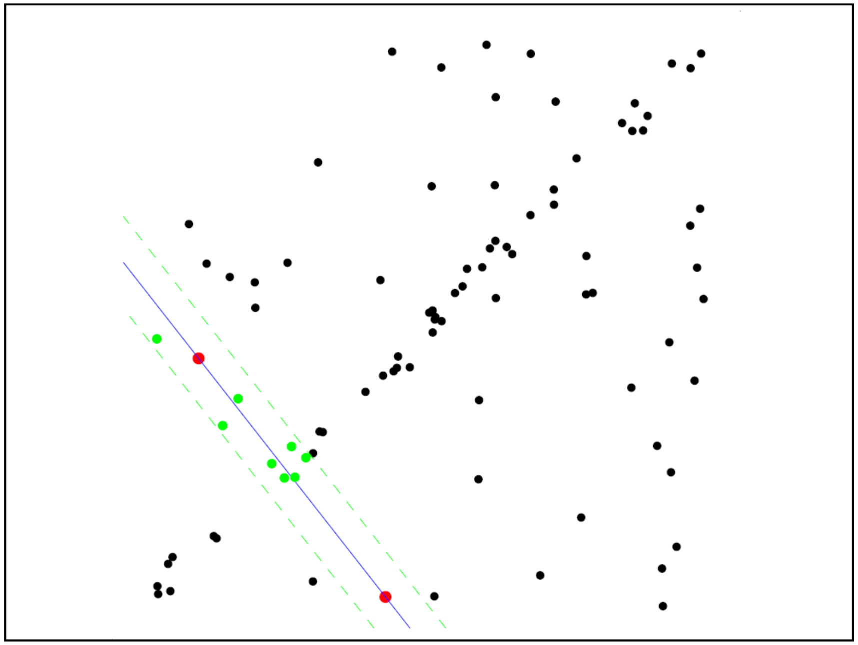
RANSAC

Data: Set S consisting of all N points
Result: Set with maximum number of inliers
(and corresponding fitting line)
while $i \leq k$ **do**
randomly select 2 points from S;
fit line l_i through the 2 points;
compute distance of all other points to line l_i ;
construct *inlier* set, i.e., count number of
points with distance to the line less than γ;
store line l_i and associated set of inliers;
$i \leftarrow i + 1$
end
Choose set with maximum number of inliers



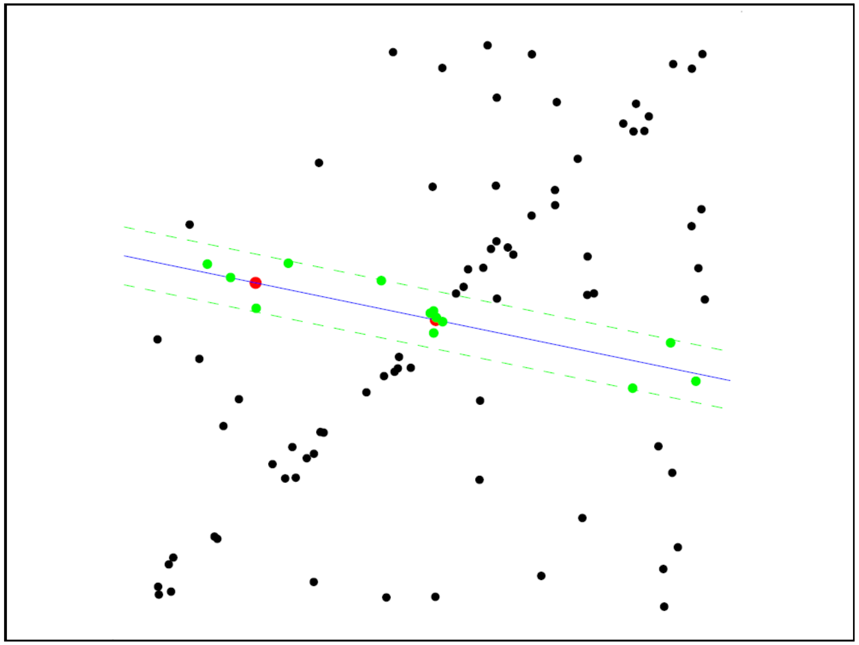
RANSAC

Data: Set S consisting of all N points
Result: Set with maximum number of inliers
(and corresponding fitting line)
while $i \leq k$ **do**
randomly select 2 points from S;
fit line l_i through the 2 points;
compute distance of all other points to line l_i ;
construct *inlier* set, i.e., count number of
points with distance to the line less than γ;
store line l_i and associated set of inliers;
$i \leftarrow i + 1$
end
Choose set with maximum number of inliers



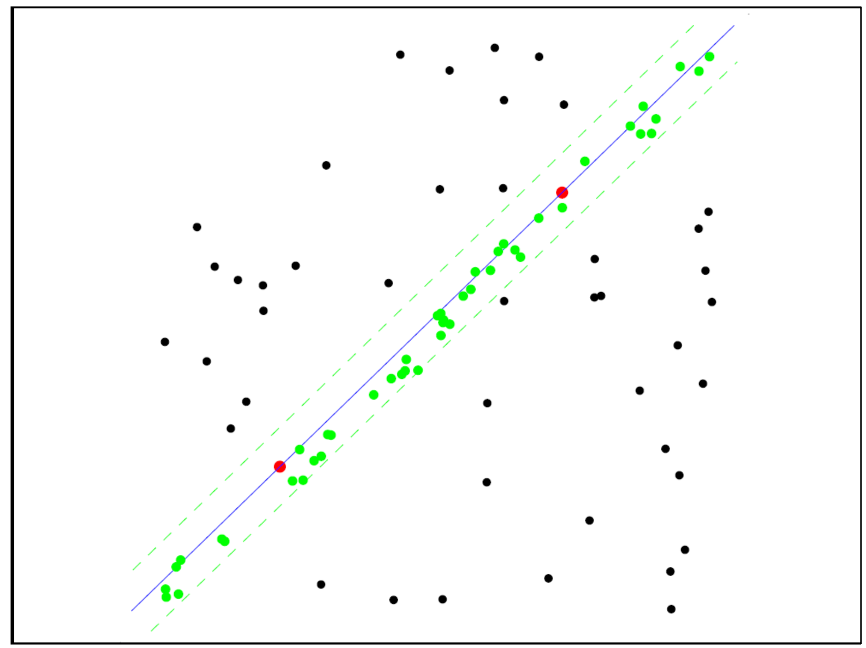
RANSAC

Data: Set S consisting of all N points
Result: Set with maximum number of inliers
(and corresponding fitting line)
while $i \leq k$ **do**
  randomly select 2 points from S;
  fit line l_i through the 2 points;
  compute distance of all other points to line l_i ;
  construct *inlier* set, i.e., count number of
    points with distance to the line less than γ;
  store line l_i and associated set of inliers;
  $i \leftarrow i + 1$
end
Choose set with maximum number of inliers



RANSAC

Data: Set S consisting of all N points
Result: Set with maximum number of inliers
(and corresponding fitting line)
while $i \leq k$ **do**
randomly select 2 points from S;
fit line l_i through the 2 points;
compute distance of all other points to line l_i ;
construct *inlier* set, i.e., count number of
points with distance to the line less than γ;
store line l_i and associated set of inliers;
$i \leftarrow i + 1$
end
Choose set with maximum number of inliers



Core “feature based” CV pipeline

- Input two images (stereo or SfM), or image and 3D model (PnP)
- Feature detection, feature matching -> N corespondences
- Problem: in practice, we get loads of bad feature matches
- Use Random Sample Consensus (RANSAC) to reject outliers
- In RANSAC, we
  - Sample a small subset n of matched features
  - Solve geometric CV problem with these correspondences (PnP, stereo, SfM)
  - Check solution fit with remaining (N-n) feature matches – record fit score
  - Repeat
  - Return the fit with the highest score