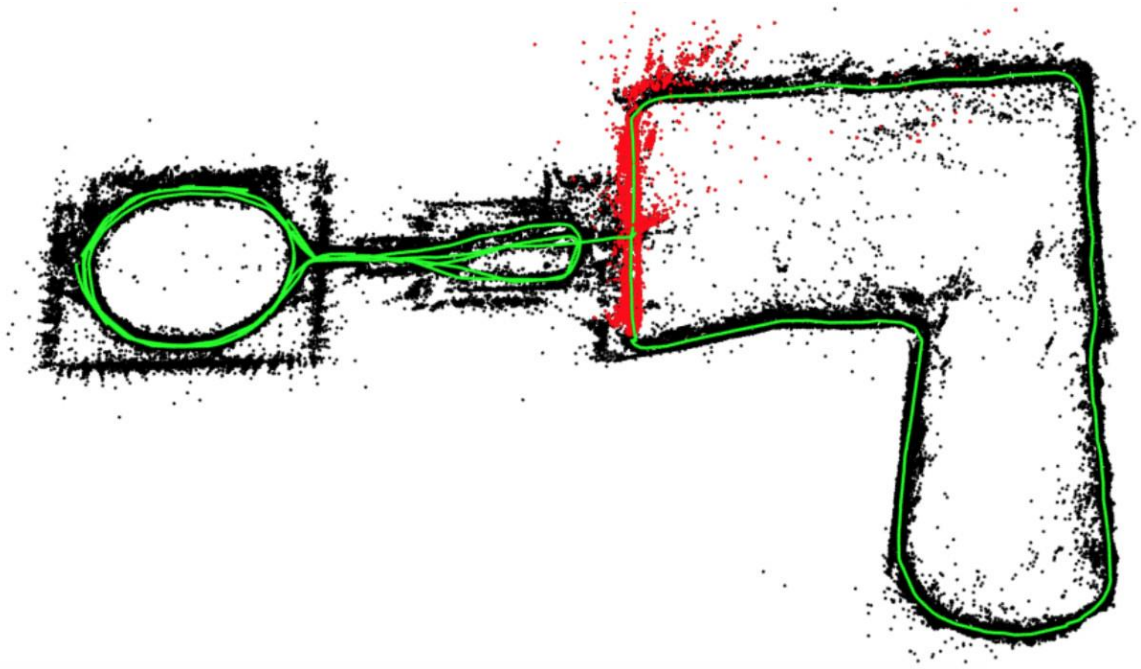
EKF and SLAM



McGill COMP 765

Sept 18th, 2017

Outline

- News and information
- Instructions for paper presentations
- Continue on Kalman filter: EKF and extension to mapping
- Example of a real mapping system: visual structure from motion

Next 3 Lectures

- Sept 21st
  - Normal room
  - Lead by Florian and Travis
  - Required reading (at least quickly) the 2 papers listed on the list
- Sept 26th and 28th
  - MC 103
  - Joint with COMP 417
  - Lead by Jose Ruis, Professor in intelligent control from Mexico

Paper Presentations

- Presenters:
  - Read the paper carefully
  - Prepare roughly 15 minutes of speaking. Can include showing the paper pdf and using their figures, probably better to show some powerpoint summary, say things in your own words
  - Goal is to identify key points of the paper, teach the class something
  - Goal is not to be a complete expert. It's OK to raise confusing points. Remember, all research papers are imperfect!
- Audience:
  - Read the papers ahead, come up with questions on the spot. Fill out feedback forms afterwards.

Kalman Filter: an instance of Bayes' Filter

Assumptions guarantee that if the prior belief before the prediction step is Gaussian

$$
\begin{aligned}
bel(x_t) &= p(x_t|u_{0:t-1}, z_{0:t}) \\
&= \eta\, p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}
\end{aligned}
$$

then the prior belief after the prediction step will be Gaussian

and the posterior belief (after the update step) will be Gaussian.

Linear observations with Gaussian noise

$$
\begin{aligned}
z_t &= Hx_t + n_t \\
&\text{with noise } n_t \sim \mathcal{N}(0, R)
\end{aligned}
$$

Linear dynamics with Gaussian noise

$$
\begin{aligned}
x_t &= Ax_{t-1} + Bu_{t-1} + Gw_{t-1} \\
&\text{with noise } w_{t-1} \sim \mathcal{N}(0, Q)
\end{aligned}
$$

Initial belief is Gaussian

$$bel(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$$

Kalman Filter: an instance of Bayes' Filter

Suppose you replace the linear models with nonlinear models.

Does the posterior $bel(x_t)$ remain Gaussian?

Assumptions guarantee that if the prior belief before the prediction step is Gaussian

$$
\begin{aligned}
bel(x_t) &= p(x_t|u_{0:t-1}, z_{0:t}) \\
&= \eta\, p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}
\end{aligned}
$$

then the prior belief after the prediction step will be Gaussian

and the posterior belief (after the update step) will be Gaussian.

Linear observations with Gaussian noise

$$
\begin{aligned}
z_t &= h(x_t) + n_t \\
&\text{with noise } n_t \sim \mathcal{N}(0, R)
\end{aligned}
$$

Linear dynamics with Gaussian noise

$$
\begin{aligned}
x_t &= f(x_{t-1}, u_{t-1}) + Gw_{t-1} \\
&\text{with noise } w_{t-1} \sim \mathcal{N}(0, Q)
\end{aligned}
$$

Initial belief is Gaussian

$$bel(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$$

Kalman Filter: an instance of Bayes' Filter

Suppose you replace the linear models with nonlinear models.

Does the posterior $bel(x_t)$ remain Gaussian? NO

Assumptions guarantee that if the prior belief before the prediction step is Gaussian

$$
\begin{aligned}
bel(x_t) &= p(x_t|u_{0:t-1}, z_{0:t}) \\
&= \eta\, p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}
\end{aligned}
$$

then the prior belief after the prediction step will be Gaussian

and the posterior belief (after the update step) will be Gaussian.

Linear observations with Gaussian noise

$$z_t = h(x_t) + n_t \quad \text{with noise } n_t \sim \mathcal{N}(0, R)$$

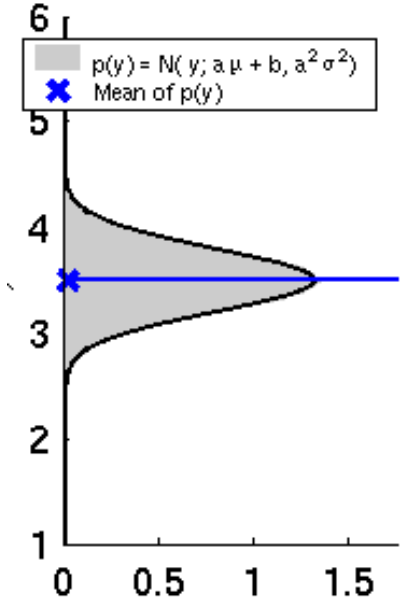
Linear dynamics with Gaussian noise

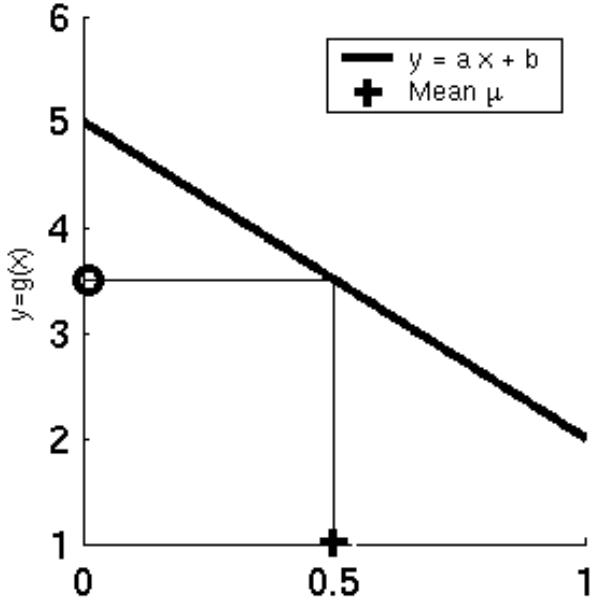
$$x_t = f(x_{t-1}, u_{t-1}) + Gw_{t-1} \quad \text{with noise } w_{t-1} \sim \mathcal{N}(0, Q)$$

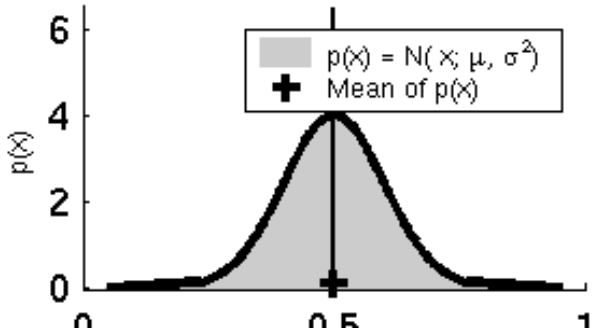
Initial belief is Gaussian

$$bel(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$$

Linearity Assumption Revisited



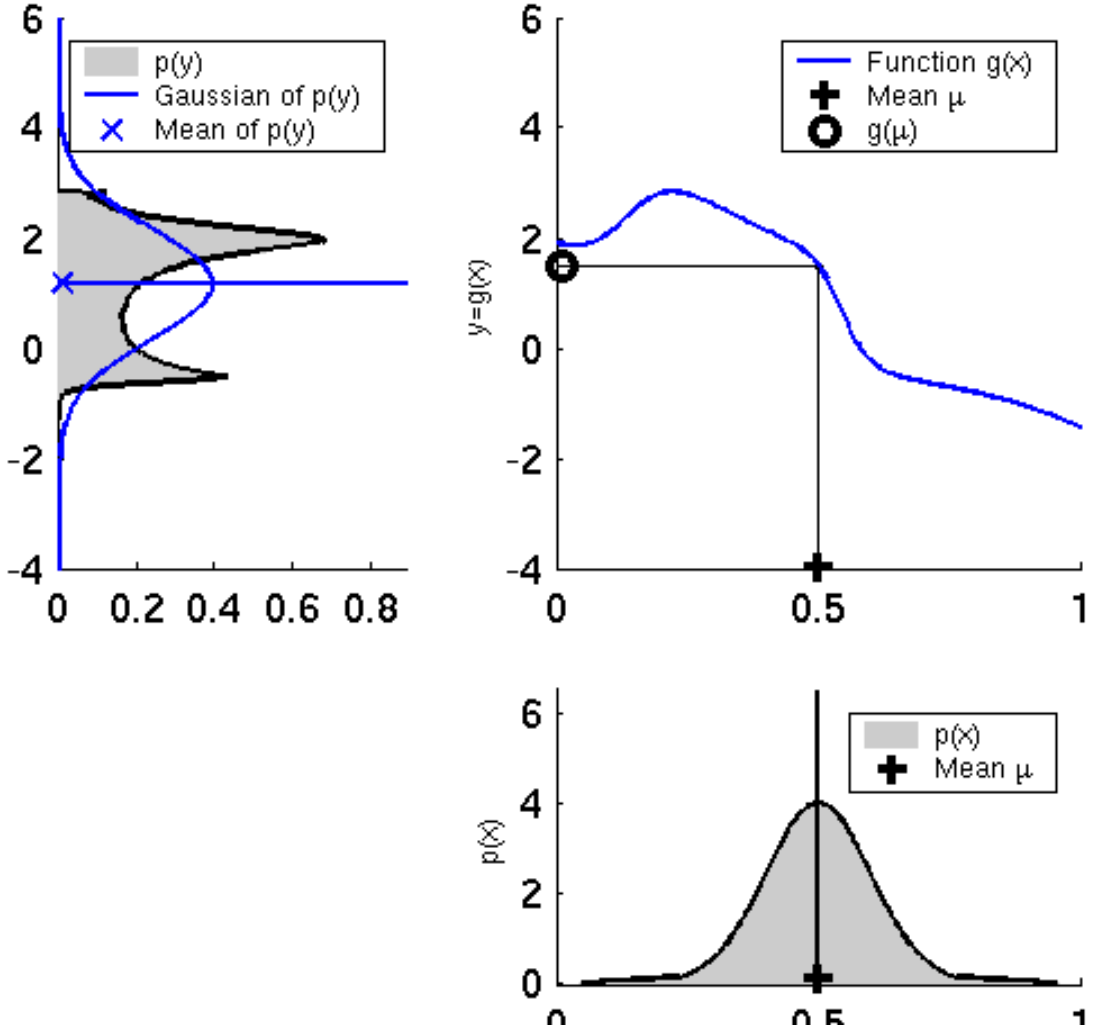




If $y = ax + b$
and $x \sim \mathcal{N}(\mu, \sigma^2)$

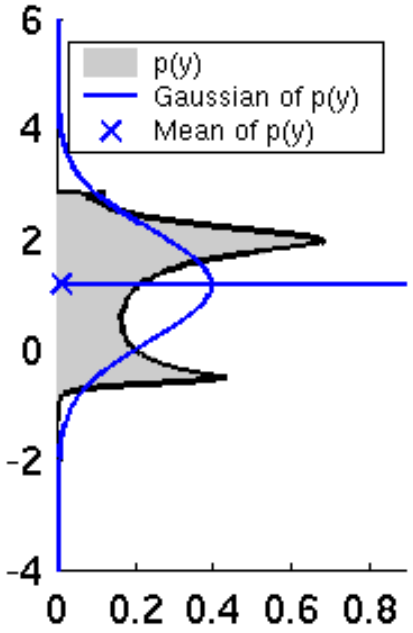
then $y \sim \mathcal{N}(a\mu + b, a^2\sigma^2)$

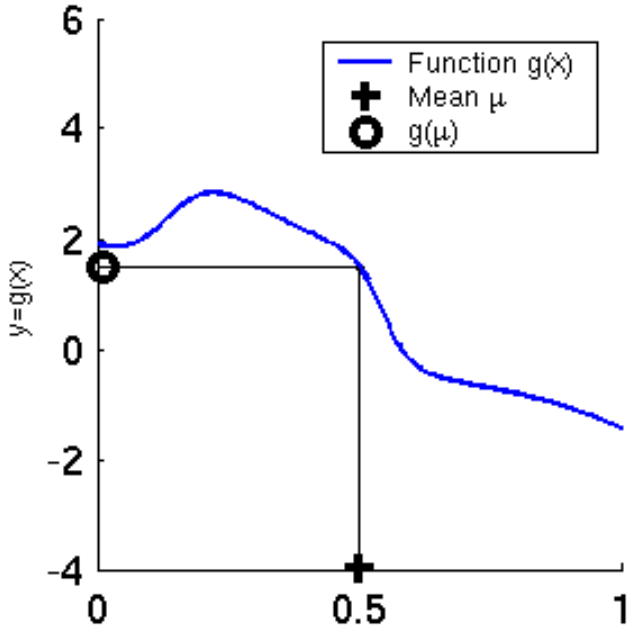
Nonlinear Function

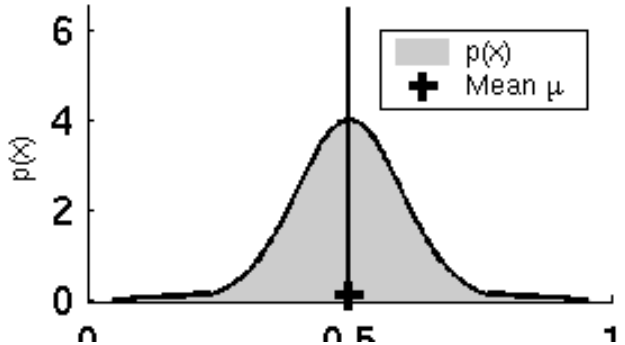


If $y = g(x)$
and $x \sim \mathcal{N}(\mu, \sigma^2)$
then y is not necessarily distributed as a Gaussian.

Nonlinear Function







If $y = g(x)$
and $x \sim \mathcal{N}(\mu, \sigma^2)$
then y is not necessarily distributed as a Gaussian.

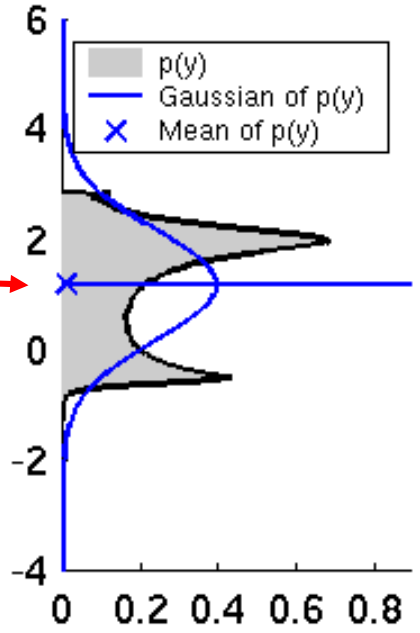
How can we approximate p(y) using a single Gaussian, without having a formula for p(y)?

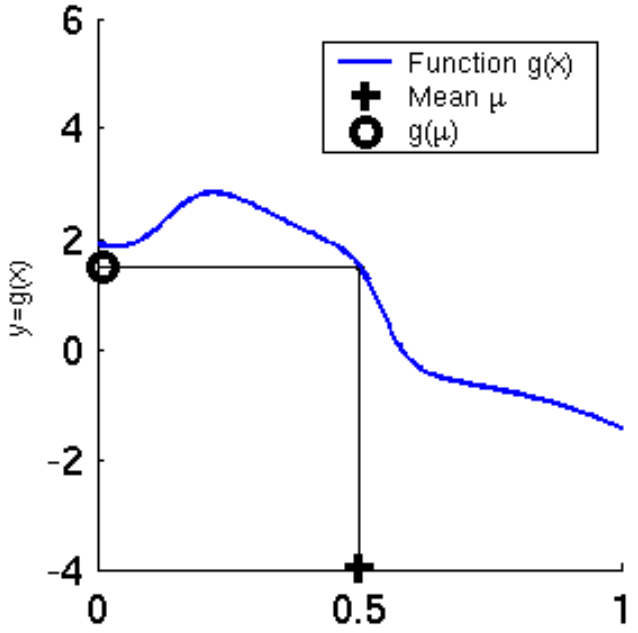
IDEA #1: MONTE CARLO SAMPLING
- Draw many (e.g. 10^6) samples xi ~ p(x)
- Pass them through the nonlinear function yi = g(xi)
- Compute the empirical mean (m) and covariance (S) of the samples yi
- Return Normal(m, S)

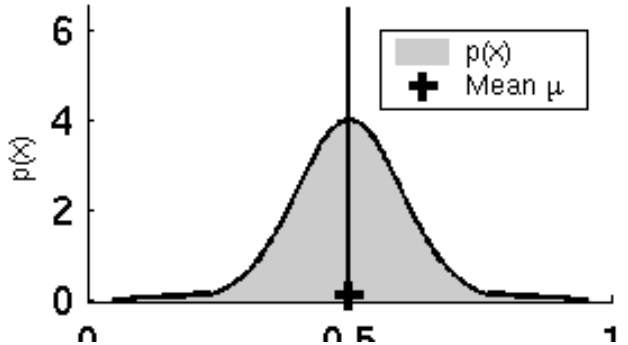
IDEA #2: LINEARIZE THE NONLINEAR FUNCTIONS f, h
- Then we are in the case y = Gx +c, so p(y) is a Gaussian

Nonlinear Function

This is how we computed the Gaussian approximation of p(y).







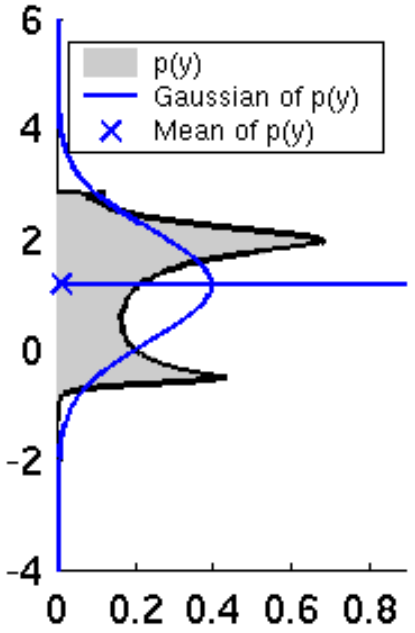
If $y = g(x)$
and $x \sim \mathcal{N}(\mu, \sigma^2)$
then y is not necessarily distributed as a Gaussian.

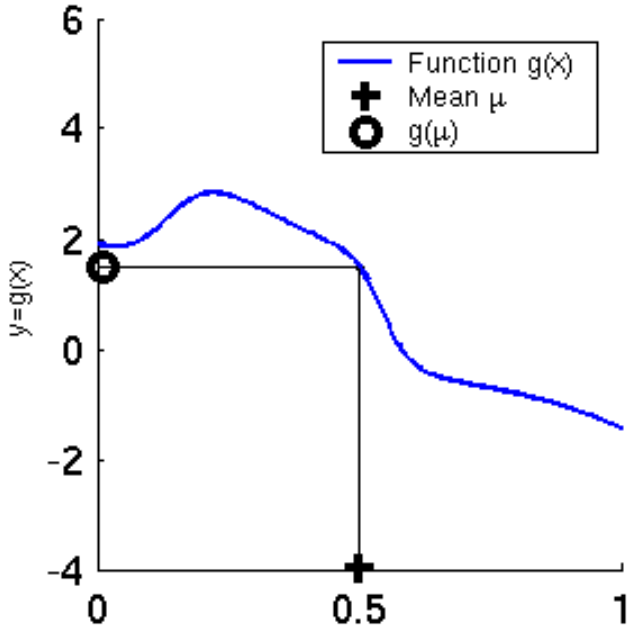
How can we approximate p(y) using a single Gaussian, without having a formula for p(y)?

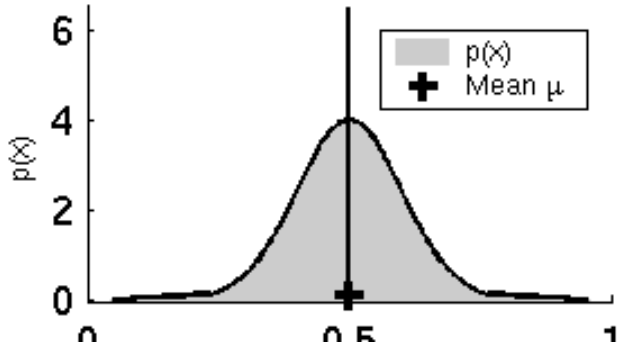
IDEA #1: MONTE CARLO SAMPLING
- Draw many (e.g. 10^6) samples xi ~ p(x)
- Pass them through the nonlinear function yi = g(xi)
- Compute the empirical mean (m) and covariance (S) of the samples yi
- Return Normal(m, S)

IDEA #2: LINEARIZE THE NONLINEAR FUNCTIONS f, h
- Then we are in the case y = Gx +c, so p(y) is a Gaussian

Nonlinear Function







If $y = g(x)$
and $x \sim \mathcal{N}(\mu, \sigma^2)$
then y is not necessarily distributed as a Gaussian.

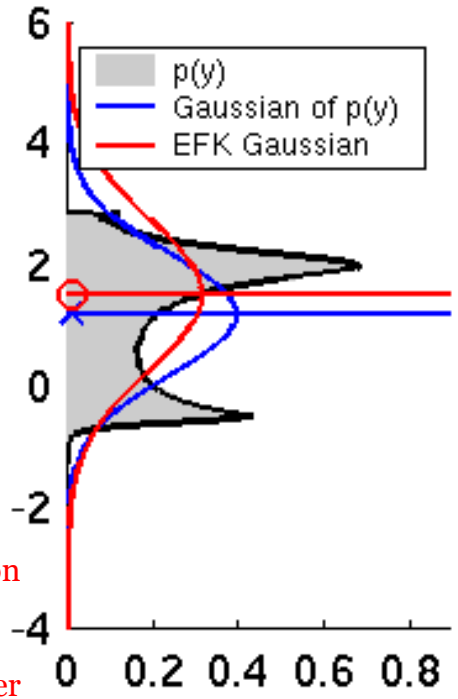
Bad idea to use in Kalman Filter because we want to approximate p(y) efficiently.

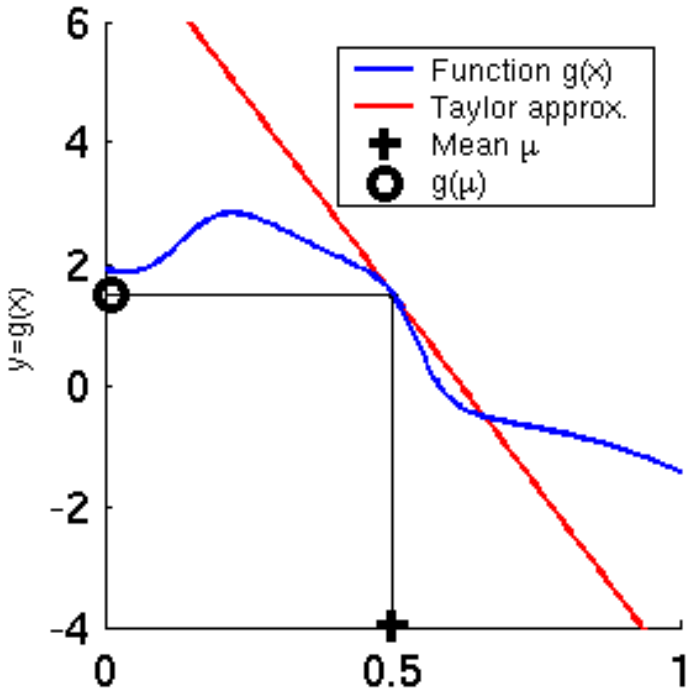
How can we approximate p(y) using a single Gaussian, without having a formula for p(y)?

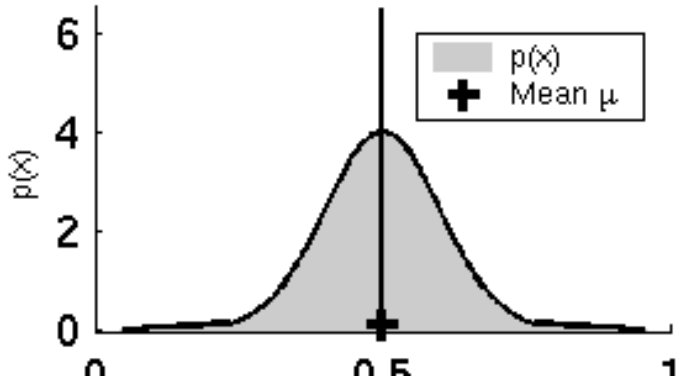
IDEA #1: MONTE CARLO SAMPLING
- Draw many (e.g. 10^6) samples xi ~ p(x)
- Pass them through the nonlinear function yi = g(xi)
- Compute the empirical mean (m) and covariance (S) of the samples yi
- Return Normal(m, S)

IDEA #2: LINEARIZE THE NONLINEAR FUNCTIONS f, h
- Then we are in the case y = Gx +c, so p(y) is a Gaussian

Linearization







Notice how the Linearization approximation differs from the Monte Carlo approximation (which is better, provided sufficiently many samples).

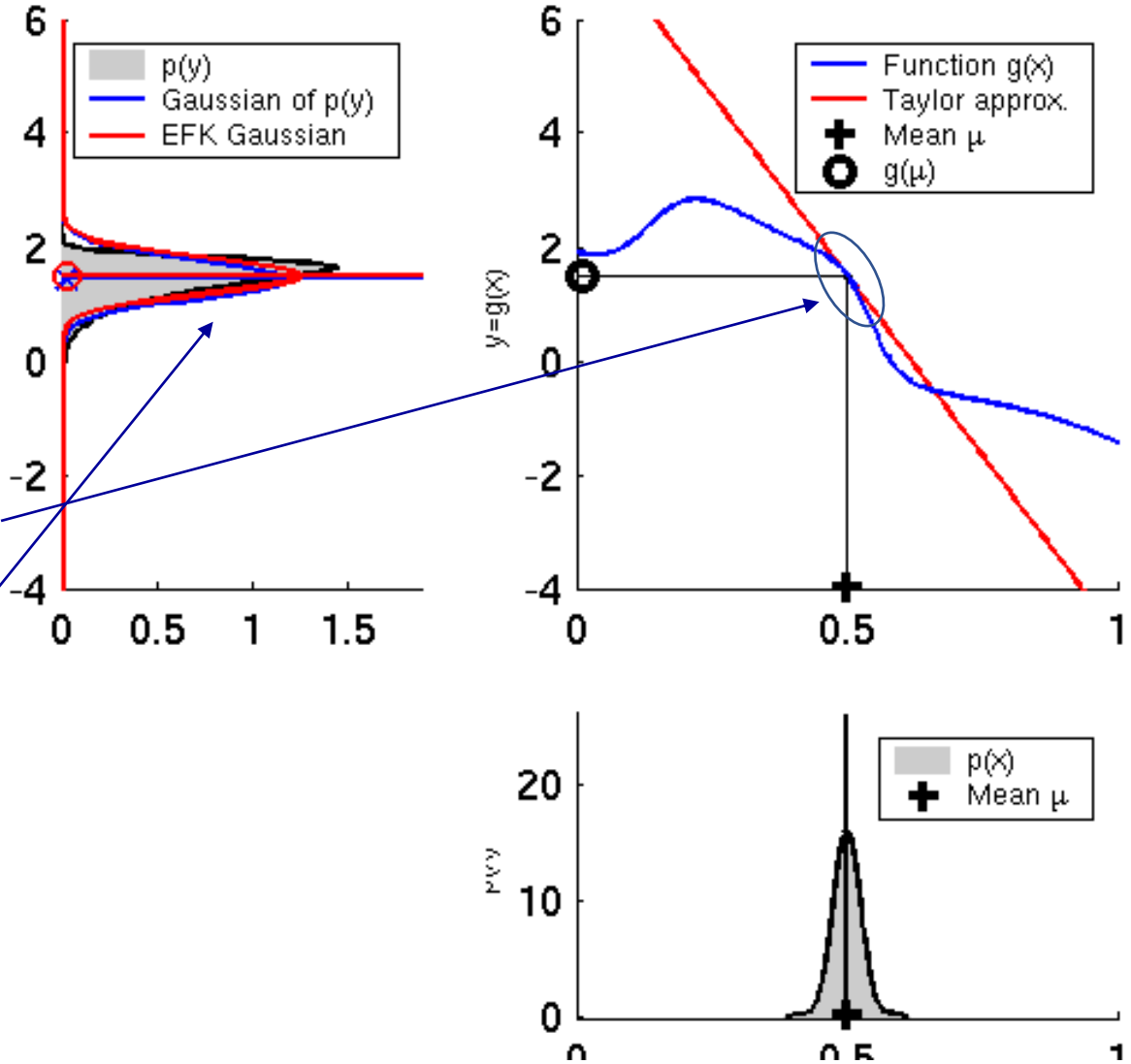
That said, the Linearization approximation can be computed efficiently, and can be integrated into the Kalman Filter estimator → Extended Kalman Filter

Linearization with low approximation error

The quality of the linearization approximation depends on the uncertainty of p(x) but also on the shape of the nonlinearity g(x).

In this example p(x) has small variance so most points will be concentrated around 0.5 and will pass through a very small region of g(x), where g(x) is close to linear.

In this case p(y) is nearly Gaussian, and the Linearization approximation matches the Monte Carlo approximation.

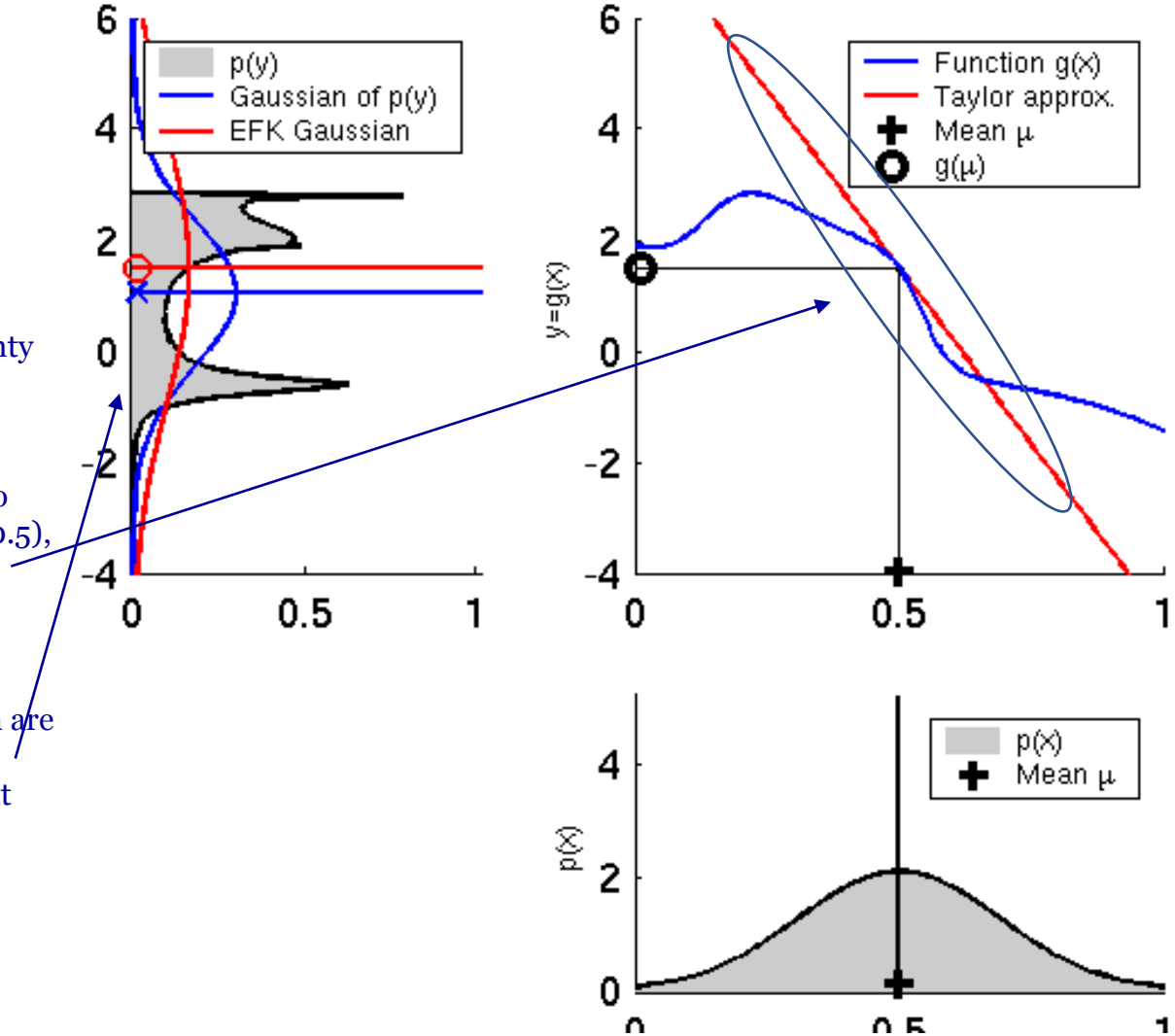


Linearization with high approximation error

The quality of the linearization approximation depends on the uncertainty of p(x) but also on the shape of the nonlinearity g(x).

In this example p(x) has high variance so points g(x) will be spread out around g(0.5), where g(x) is not close to linear.

In this case p(y) is multimodal, and the Linearization approximation matches the Monte Carlo approximation are both suboptimal approximations. Again Monte Carlo is better, provided sufficient samples.



How do we linearize?

- Using the first order Taylor expansion around the mean of the previous update step's state estimate:

$$
\begin{aligned}
x_{t+1} &= f(x_t, u_t) + w_t \\
&\approx f(\mu_{t|t}, u_t) + \frac{\partial f}{\partial x}(\mu_{t|t}, u_t)(x_t - \mu_{t|t}) + w_t \\
&= f(\mu_{t|t}, u_t) + F_t(x_t - \mu_{t|t}) + w_t \\
&= F_t x_t + \boxed{f(\mu_{t|t}, u_t) - F_t \mu_{t|t}} + w_t \\
&= F_t x_t + \bar{u}_t + w_t
\end{aligned}
$$

Constant term with respect to the state

Recall how to compute the Jacobian matrix. For example, if

$$f(x_1, x_2, u) = [x_1 + x_2^2,\ x_2 + 3u,\ x_1^4 - u^2] \in \mathbb{R}^3$$

then the Jacobian of f with respect to (x_1, x_2) at (μ_1, μ_2, u) is

$$
\begin{aligned}
\frac{\partial f}{\partial x_{1:2}}(\mu_1, \mu_2, u_1) &= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} \end{bmatrix} (\mu_1, \mu_2, u_1) \\
&= \begin{bmatrix} 1 & 2\mu_2 \\ 0 & 1 \\ 4\mu_1^3 & 0 \end{bmatrix}
\end{aligned}
$$

How do we linearize?

- Using the first-order Taylor expansion around the mean of the previous prediction step's state estimate:

$$
\begin{aligned}
z_{t+1} &= h(x_{t+1}) + n_{t+1} \\
&\approx h(\mu_{t+1|t}) + \frac{\partial h}{\partial x}(\mu_{t+1|t})(x_t - \mu_{t+1|t}) + n_{t+1} \\
&= h(\mu_{t+1|t}) + H_{t+1}(x_{t+1} - \mu_{t+1|t}) + n_{t+1} \\
&= H_{t+1}x_{t+1} + \boxed{h(\mu_{t+1|t}) - H_{t+1}\mu_{t+1|t}} + n_{t+1} \\
&= H_{t+1}x_{t+1} + \bar{c}_{t+1} + n_{t+1}
\end{aligned}
$$

Constant term with respect to the state

Recall how to compute the Jacobian matrix. For example, if

$$h(x_1, x_2) = [x_1 + x_2^2,\ x_2,\ x_1^4] \in \mathbb{R}^3$$

then the Jacobian of f with respect to (x_1, x_2) at (μ_1, μ_2) is

$$
\begin{aligned}
\frac{\partial h}{\partial x_{1:2}}(\mu_1, \mu_2) &= \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} \\ \frac{\partial h_3}{\partial x_1} & \frac{\partial h_3}{\partial x_2} \end{bmatrix} (\mu_1, \mu_2) \\
&= \begin{bmatrix} 1 & 2\mu_2 \\ 0 & 1 \\ 4\mu_1^3 & 0 \end{bmatrix}
\end{aligned}
$$

Extended Kalman Filter: an instance of Bayes' Filter

Assumptions guarantee that if the prior belief before the prediction step is Gaussian

$$
\begin{aligned}
bel(x_t) &= p(x_t|u_{0:t-1}, z_{0:t}) \\
&= \eta\, p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}
\end{aligned}
$$

then the prior belief after the prediction step will be Gaussian

and the posterior belief (after the update step) will be Gaussian.

Linear observations with Gaussian noise

$$
\begin{aligned}
z_t &= H_t x_t + \bar{c}_t + n_t \\
&\text{with noise } n_t \sim \mathcal{N}(0, R)
\end{aligned}
$$

Linear dynamics with Gaussian noise

$$
\begin{aligned}
x_t &= F_{t-1} x_{t-1} + \bar{u}_{t-1} + G w_{t-1} \\
&\text{with noise } w_{t-1} \sim \mathcal{N}(0, Q)
\end{aligned}
$$

Initial belief is Gaussian

$$bel(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$$

Dynamics

$$x_{t+1} = f(x_t, u_t) + Gw_t$$
$$w_t \sim \mathcal{N}(0, Q)$$

Measurements

$$z_t = h(x_t) + n_t$$
$$n_t \sim \mathcal{N}(0, R)$$

EKF in N dimensions

Init

$$bel(x_0) \sim \mathcal{N}(\mu_{0|0}, \Sigma_{0|0})$$

Prediction Step

$$\mu_{t+1|t} = f(\mu_{t|t}, u_t)$$
$$\Sigma_{t+1|t} = F_t \Sigma_{t|t} F_t^T + GQG^T$$

Update Step

Received measurement $\bar{z}_{t+1}$ but expected to receive $\mu_{z_{t+1}} = h(\mu_{t+1|t})$

Prediction residual is a Gaussian random variable $\delta z \sim \mathcal{N}(\bar{z}_{t+1} - \mu_{z_{t+1}}, S_{t+1})$
where the covariance of the residual is $S_{t+1} = H_{t+1} \Sigma_{t+1|t} H_{t+1}^T + R$

Kalman Gain (optimal correction factor): $K_{t+1} = \Sigma_{t+1|t} H_{t+1}^T S_{t+1}^{-1}$

$$\mu_{t+1|t+1} = \mu_{t+1|t} + K_{t+1}(\bar{z}_{t+1} - \mu_{z_{t+1}})$$
$$\Sigma_{t+1|t+1} = \Sigma_{t+1|t} - K_{t+1} H_{t+1} \Sigma_{t+1|t}$$

EKF Summary

- Efficient: Polynomial in measurement dimensionality k and state dimensionality n:
  $O(k^{2.376} + n^2)$

As in KF, inverting the covariance of the residual is O(k^2.376)

- Not optimal (unlike the Kalman Filter for linear systems)
- Can diverge if nonlinearities are large
- Works surprisingly well even when all assumptions are violated

Example #1: EKF-Localization

> "Using sensory information to locate the robot in its environment is the most fundamental problem to providing a mobile robot with autonomous capabilities." [Cox '91]

- **Given**
  - Map of the environment.
  - Sequence of sensor measurements.
- **Wanted**
  - Estimate of the robot's position.
- **Problem classes**
  - Position tracking
  - Global localization
  - Kidnapped robot problem (recovery)

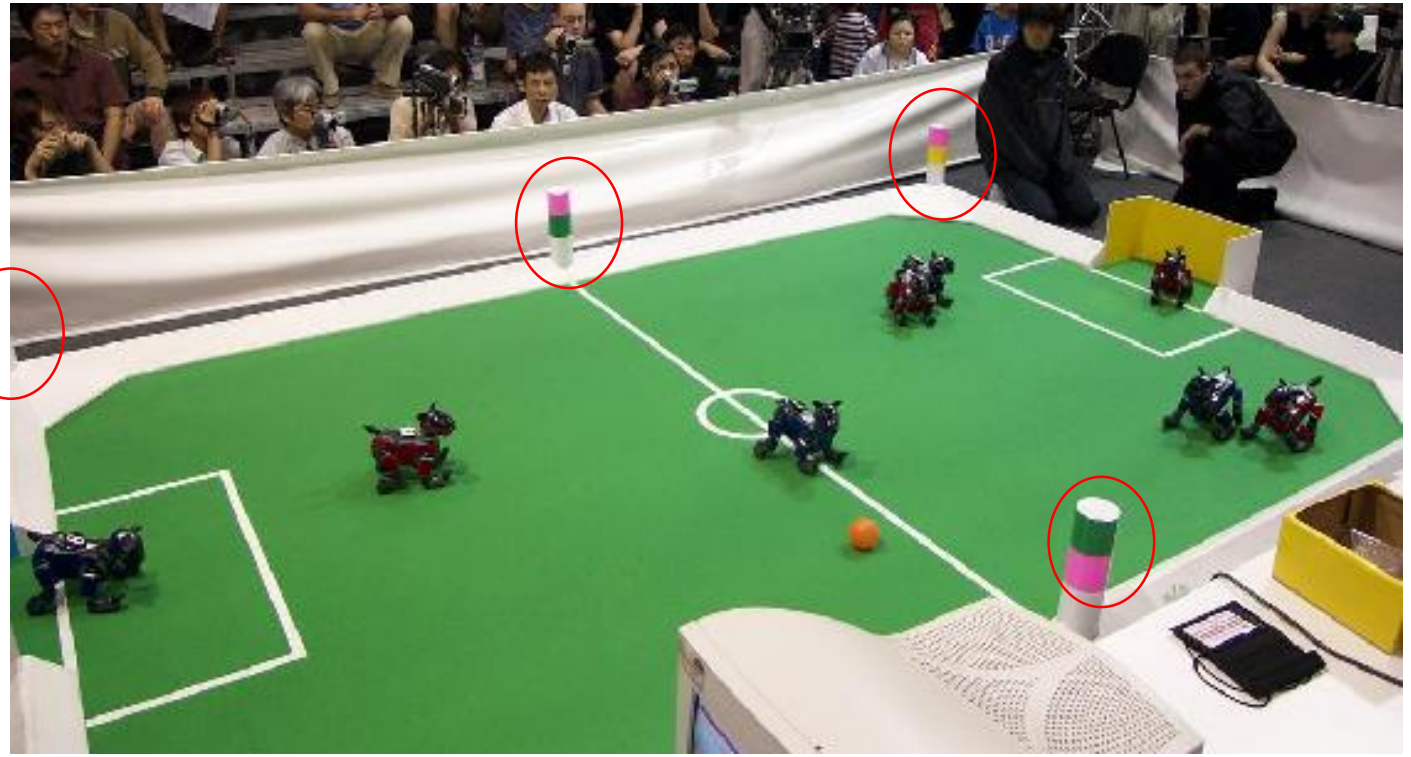
Landmark-based Localization

Landmarks, whose position $(l_x^{(i)}, l_y^{(i)})$ in the world is known.

Each robot measures its range and bearing from each landmark to localize itself.

State of a robot:

$$x_t = \begin{bmatrix} p_x(t) \\ p_y(t) \\ \theta(t) \end{bmatrix}$$



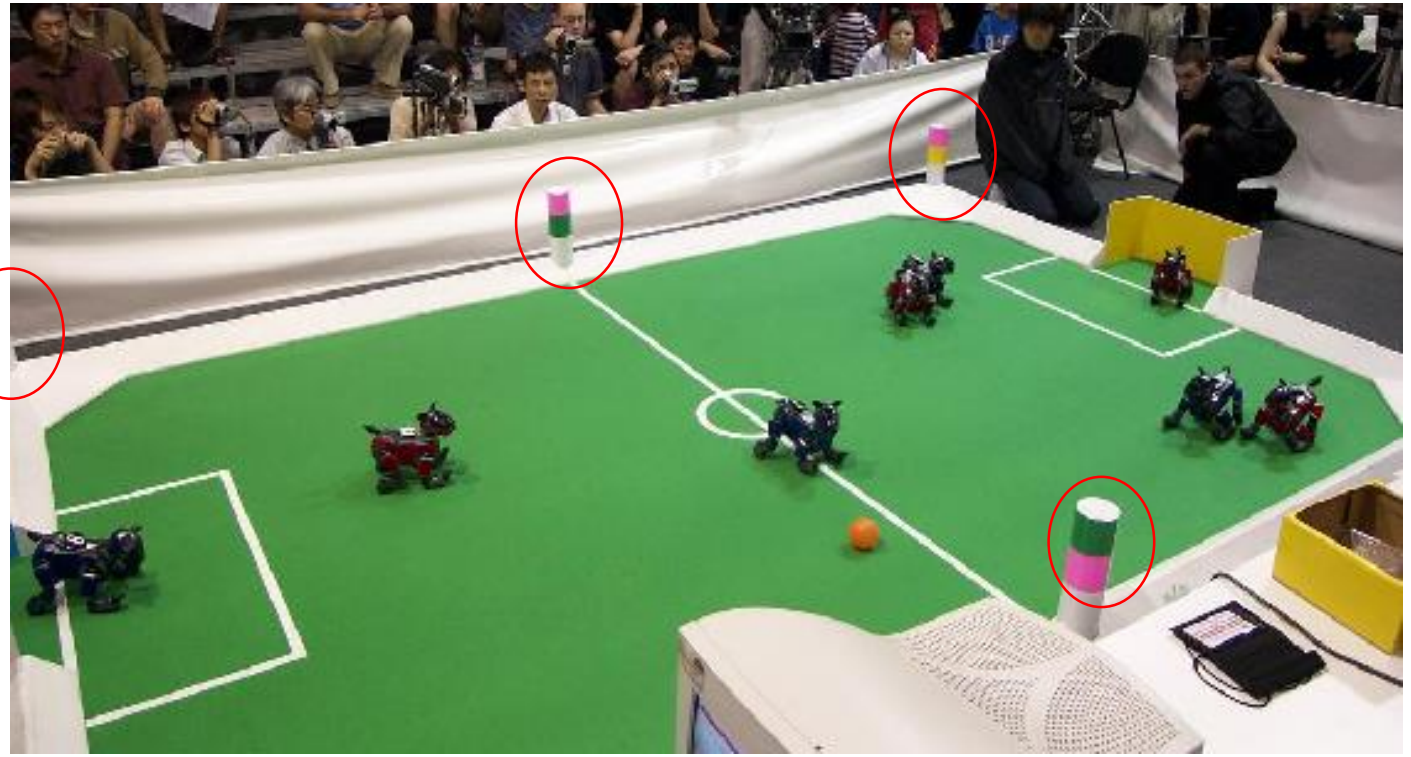
Landmark-based Localization

Measurement at time t,

$$z_t = \begin{bmatrix} \dots \\ z_t^{(i)} \\ \dots \end{bmatrix}$$

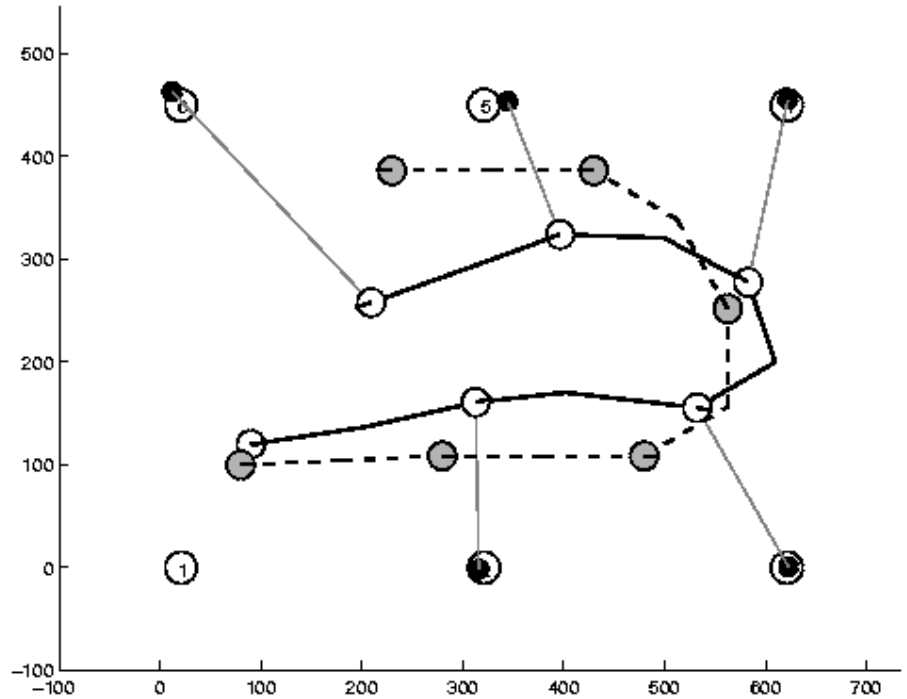
is a variable-sized vector, depending on the landmarks that are visible at time t.

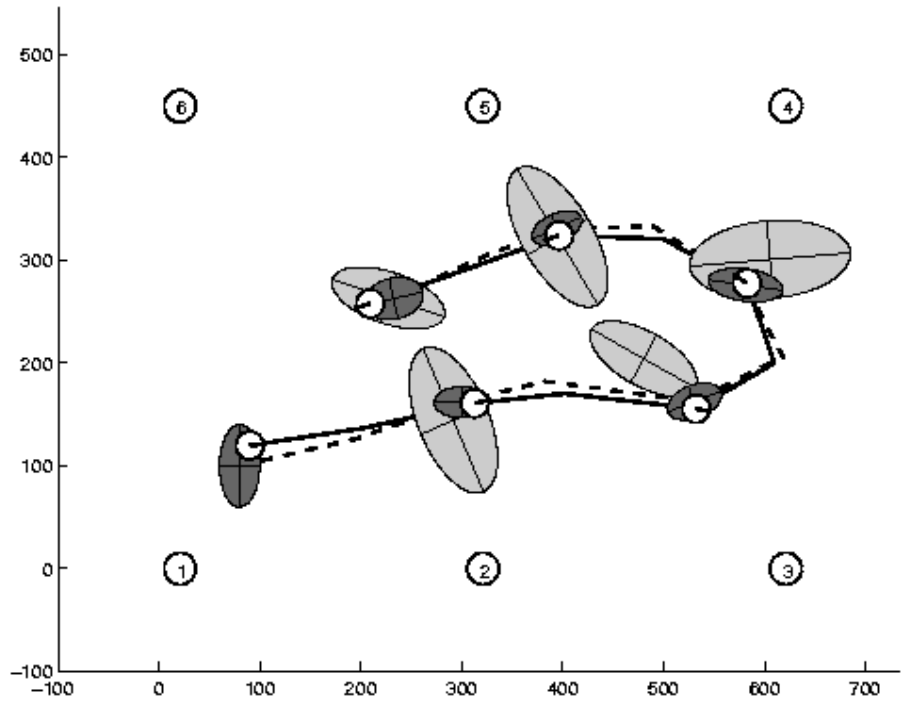
Each measurement is a 2D vector, containing range and bearing from the robot to a landmark.



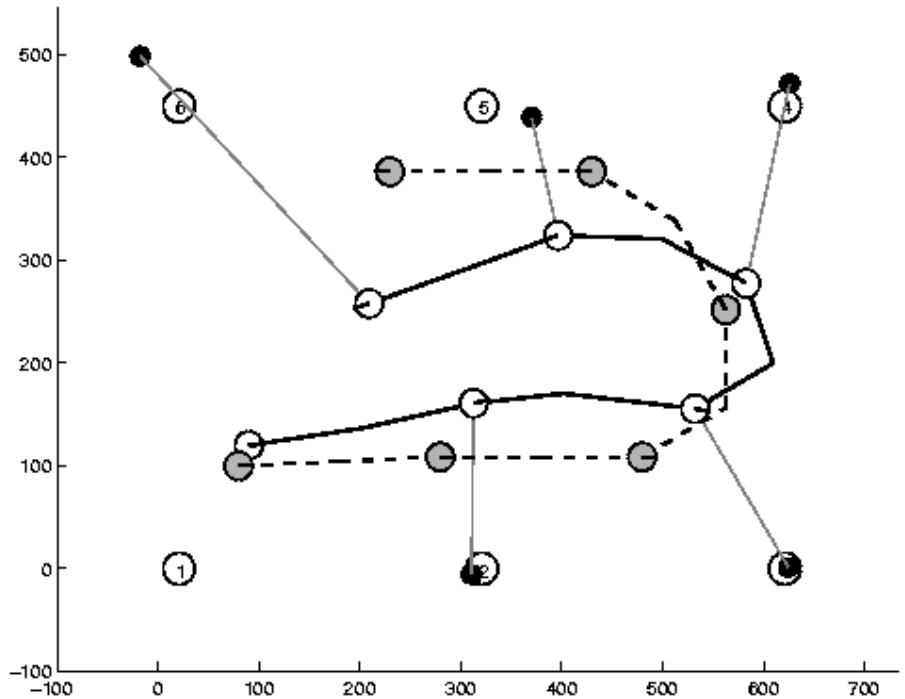
$$z_t^{(i)} = h_i(x_t) = \begin{bmatrix} \sqrt{(p_x(t) - l_x^{(i)})^2 + (p_y(t) - l_y^{(i)})^2} \\ \text{atan2}(p_y(t) - l_y^{(i)}, \quad p_x(t) - l_x^{(i)}) - \theta(t) \end{bmatrix} + n_t$$

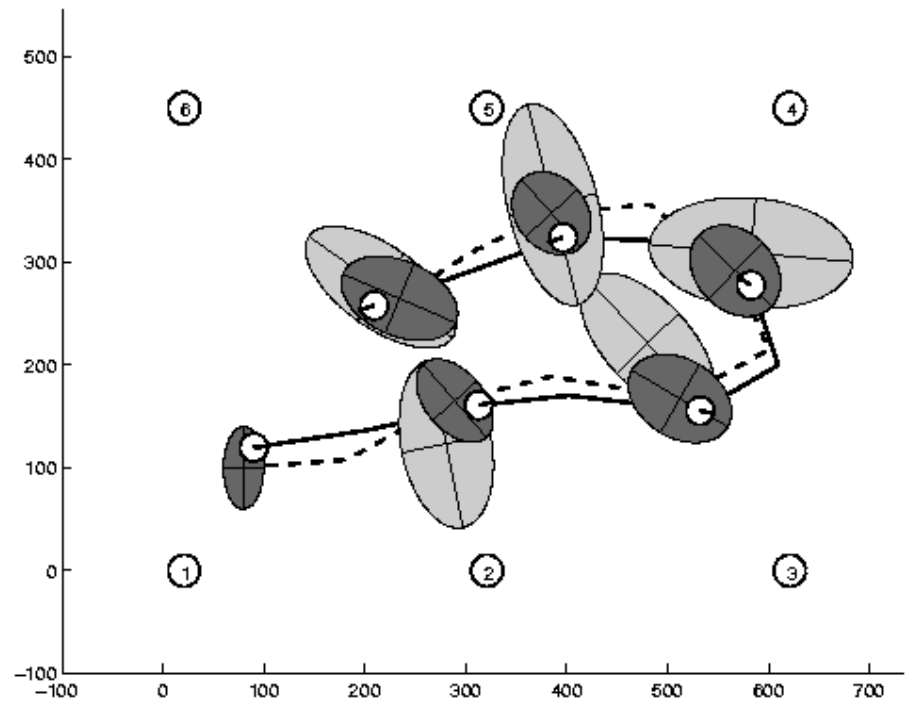
Estimation Sequence (1)



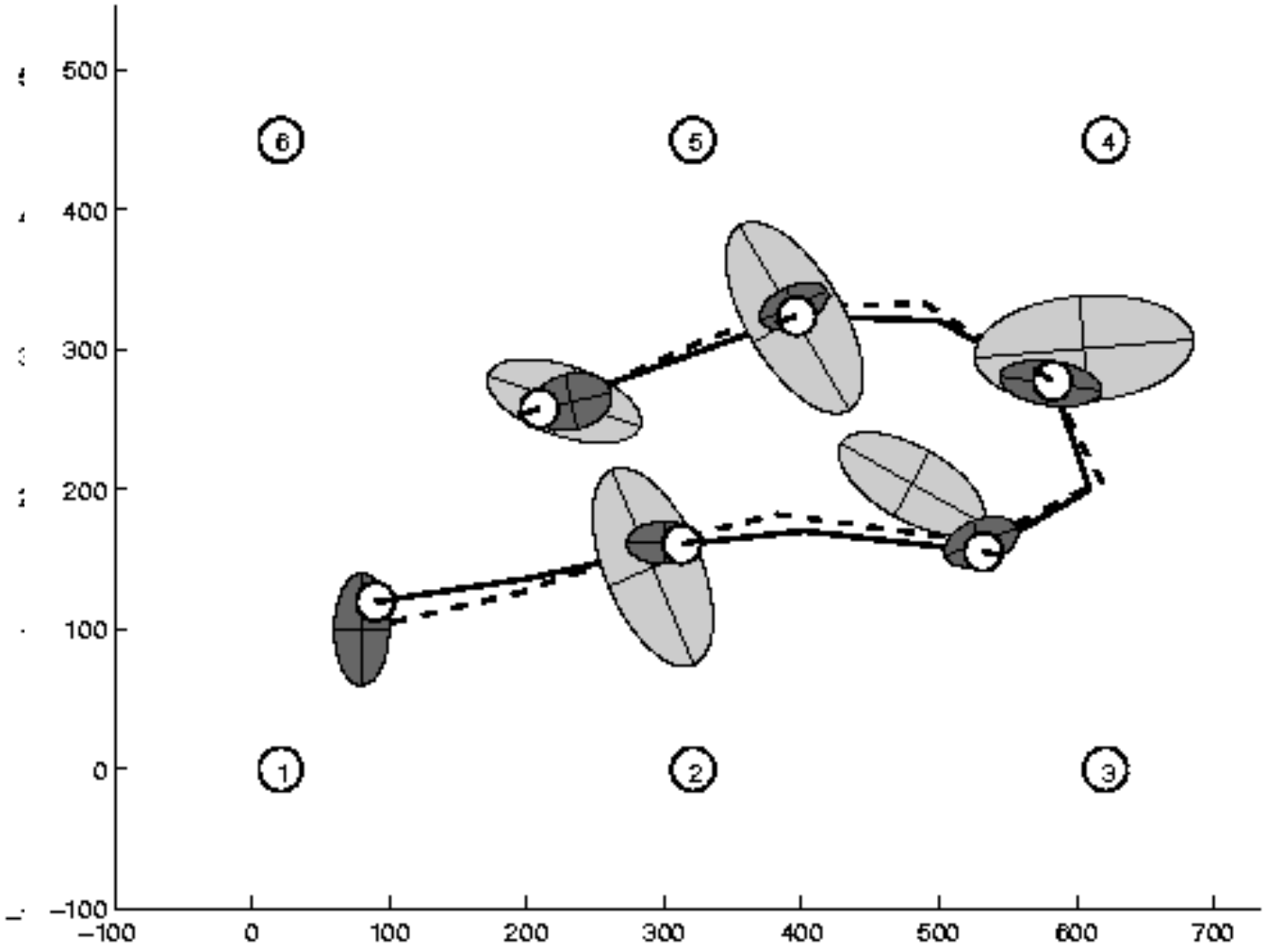


Estimation Sequence (2)





Comparison to true trajectory



Introducing: The SLAM problem

- So far we have assumed measurement model is only a function of the state and observation: this assumes knowledge of the world map (example 3 doors environment)
- Robots often have the need to navigate without a pre-known map. This requires simultaneously building a map and localizing the robot: the SLAM problem (example, jointly determine where the doors and robot are located)

First method: EKF-SLAM

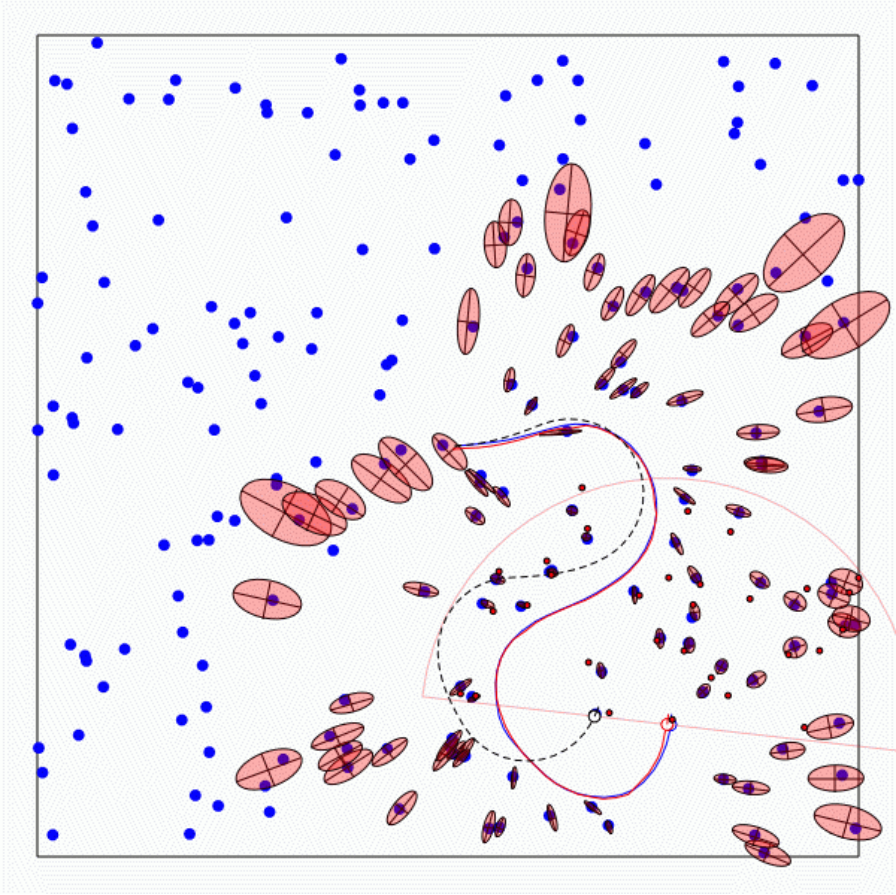
A robot is exploring an unknown, static environment.

Given:

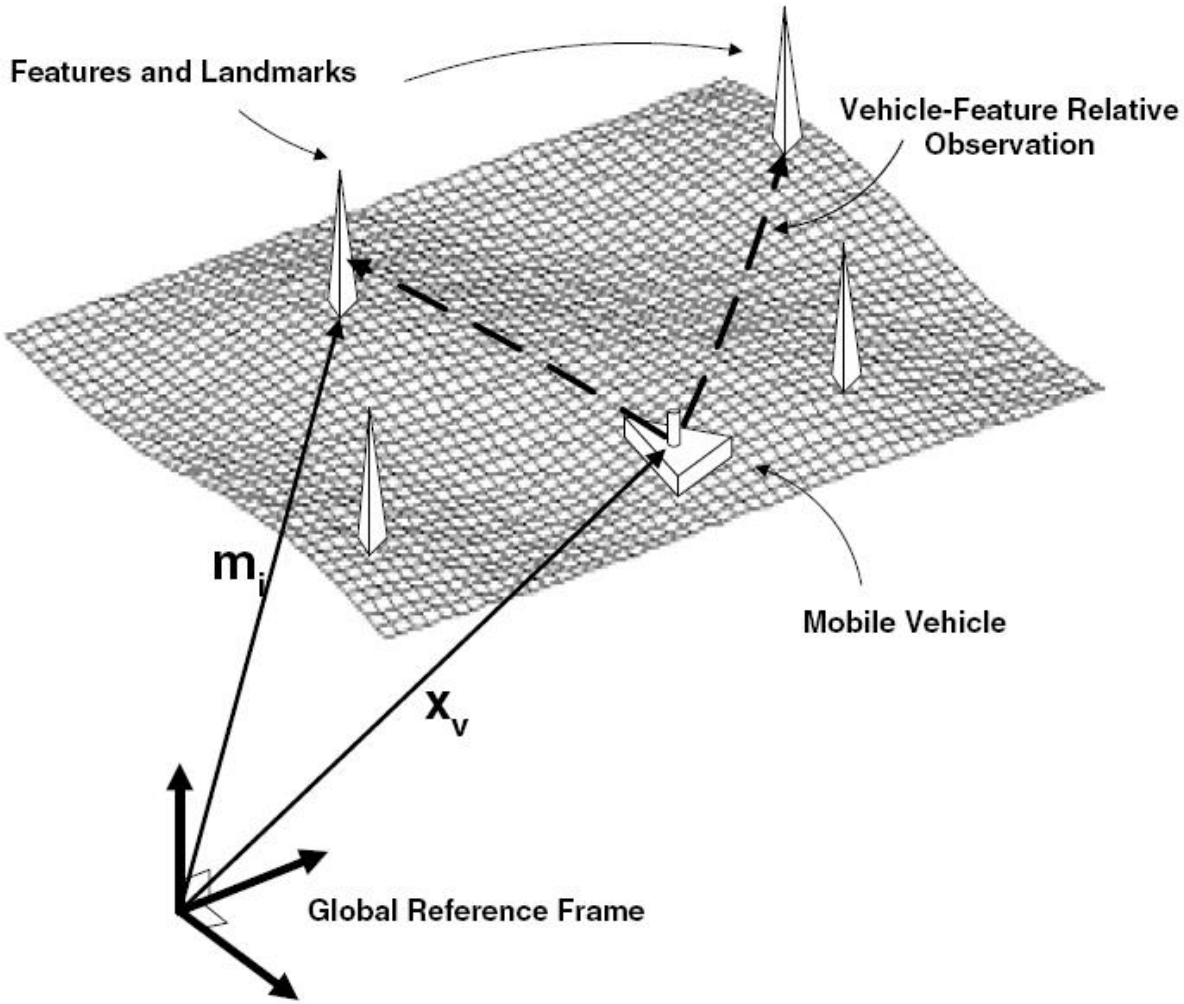
- The robot's controls
- Observations of nearby features

Estimate:

- Map of features
- Path of the robot

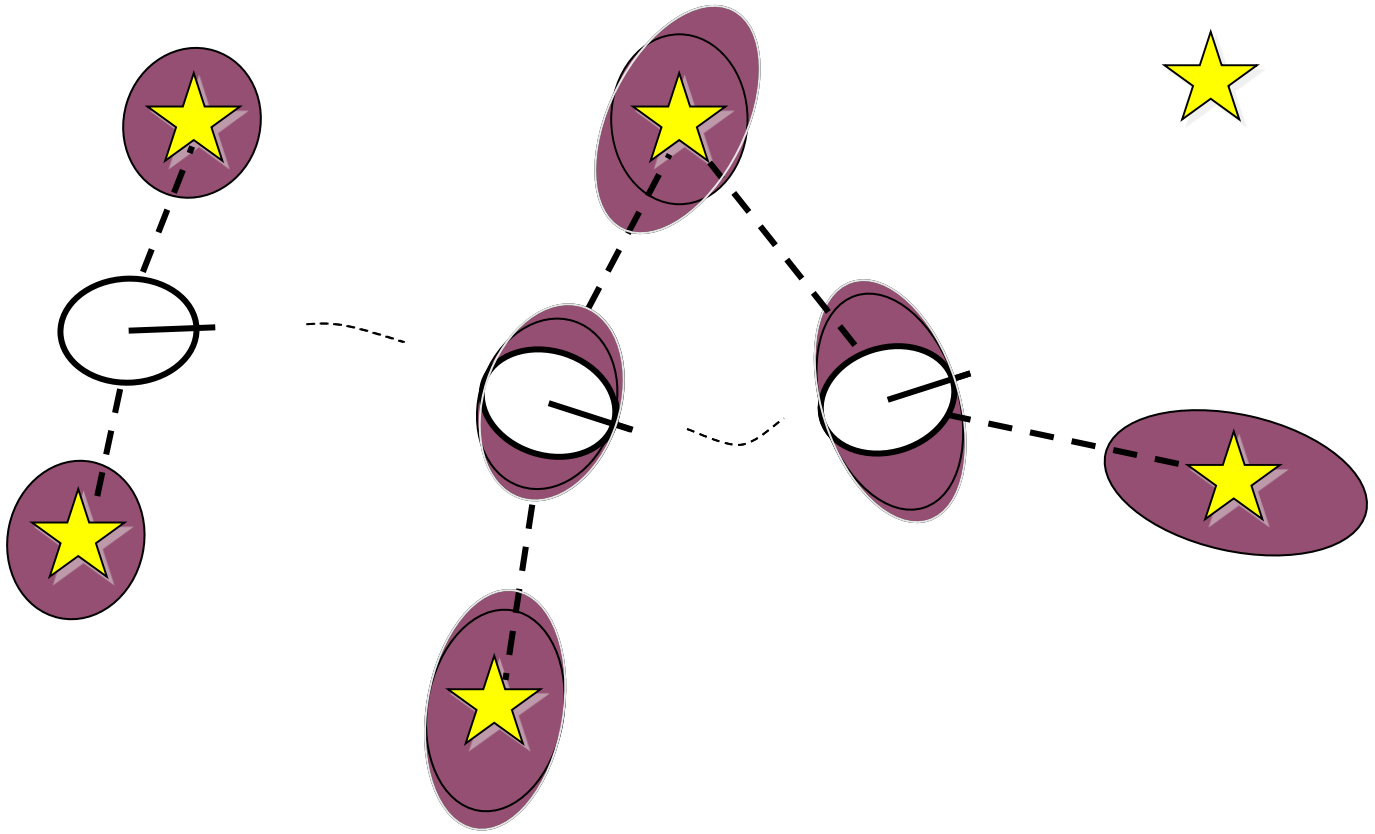


Structure of Landmark-based SLAM



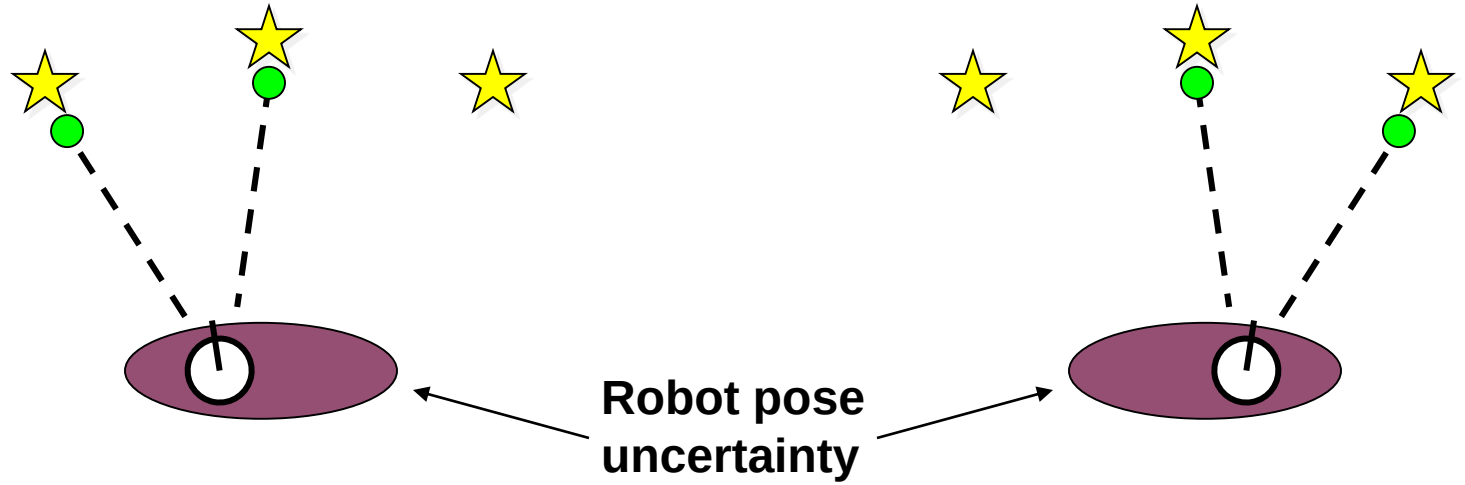
Why is SLAM a hard problem?

SLAM: robot path and map are both **unknown**



Robot path error correlates errors in the map

Why is SLAM a hard problem?



- In the real world, the mapping between observations and landmarks is unknown
- Picking wrong data associations can have catastrophic consequences
- Pose error correlates data associations

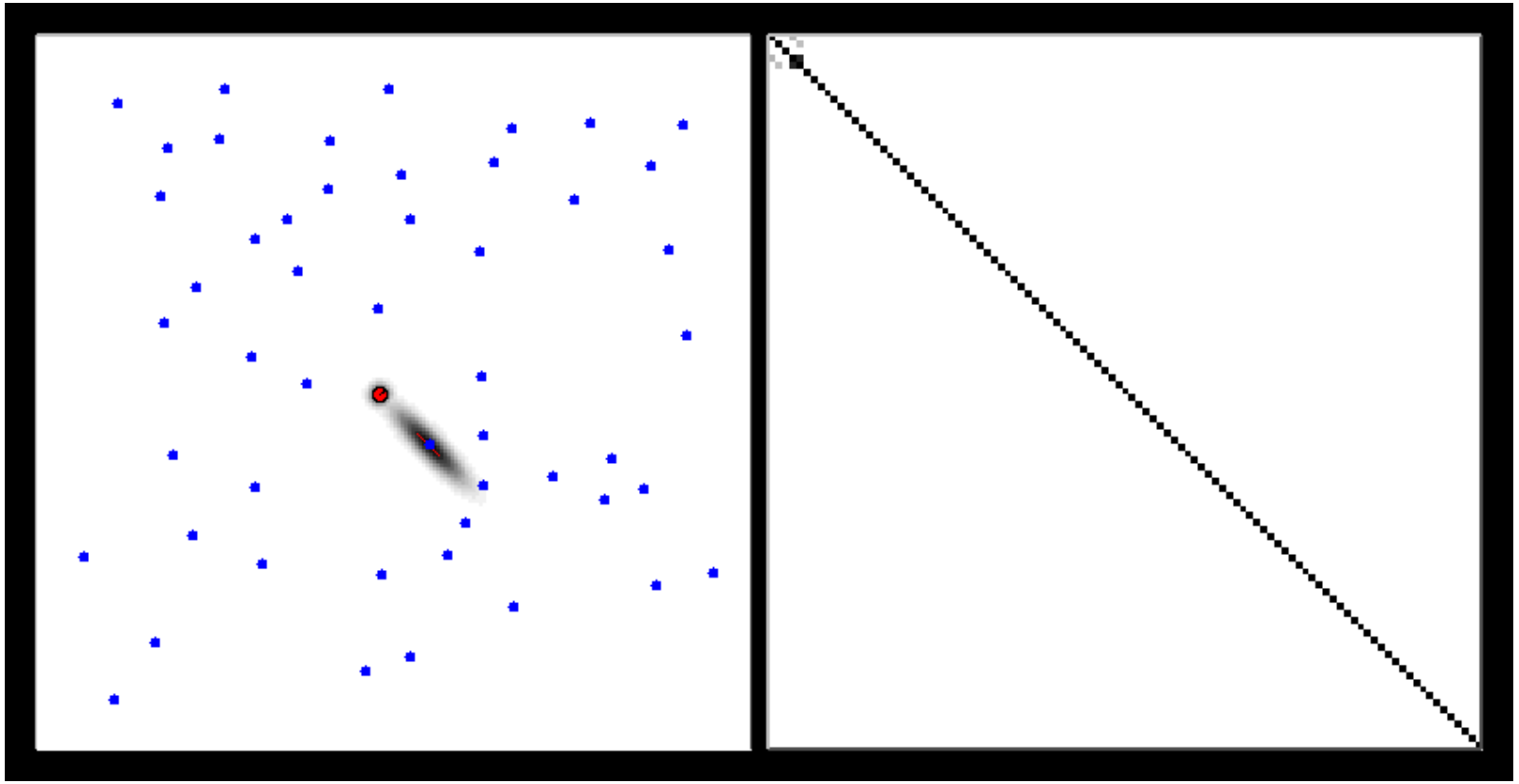
EKF-SLAM

- Map with N landmarks: (3+2N)-dimensional Gaussian

$$Bel(x_t, m_t) = \left\langle \begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix}, \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xl_1} & \sigma_{xl_2} & \cdots & \sigma_{xl_N} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & \sigma_{yl_1} & \sigma_{yl_2} & \cdots & \sigma_{yl_N} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & \sigma_{\theta l_1} & \sigma_{\theta l_2} & \cdots & \sigma_{\theta l_N} \\ \sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} & \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \cdots & \sigma_{l_1 l_N} \\ \sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} & \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \cdots & \sigma_{l_2 l_N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N} & \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \cdots & \sigma_{l_N}^2 \end{pmatrix} \right\rangle$$

- Can handle hundreds of dimensions

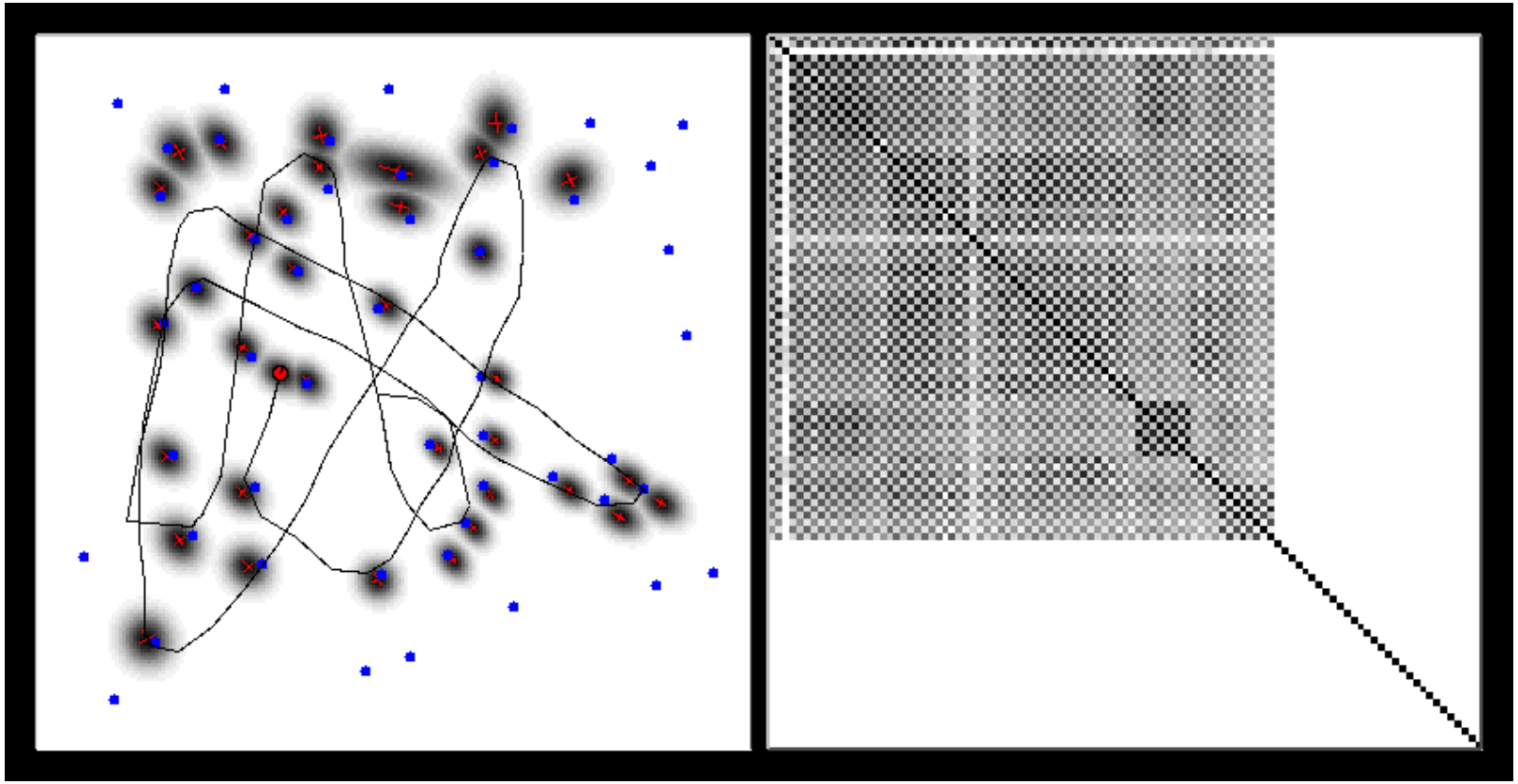
EKF-SLAM



Map

Covariance matrix

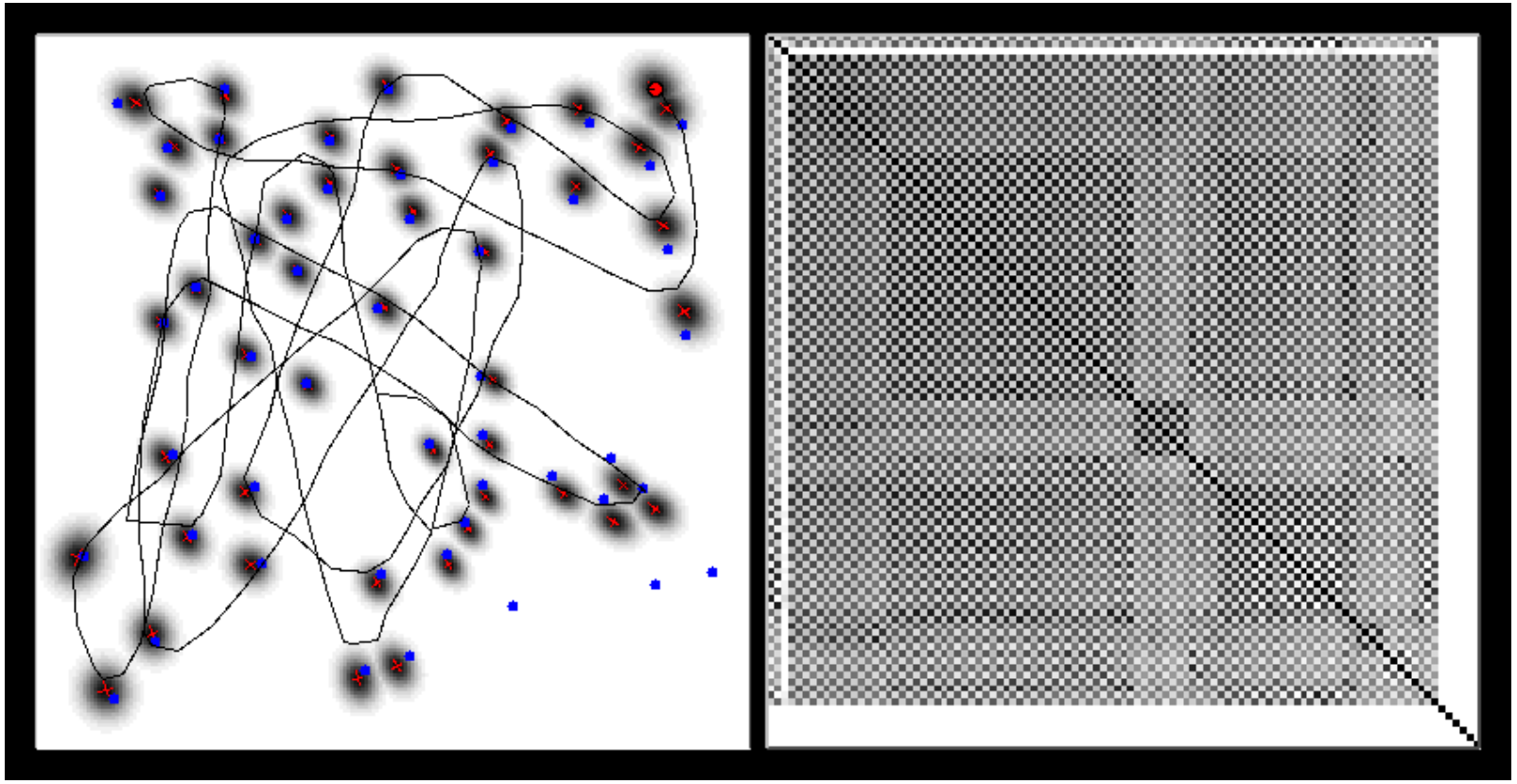
EKF-SLAM



Map

Covariance matrix

EKF-SLAM



Map

Covariance matrix

Properties of EKF-SLAM (Linear Case)

[Dissanayake et al., 2001]

Theorem:

The determinant of any sub-matrix of the map covariance matrix decreases monotonically as successive observations are made.

Theorem:

In the limit the landmark estimates become fully correlated

Example 2: Particle Filter SLAM (the dumb way)

- Augment each particle's state estimate with full map information
- Each particle contains a full copy of the map
- What are the practical implications of this?

Example 3: Rao-Blackwellized Particle Filters (the modern way)

- 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

Structure from Motion

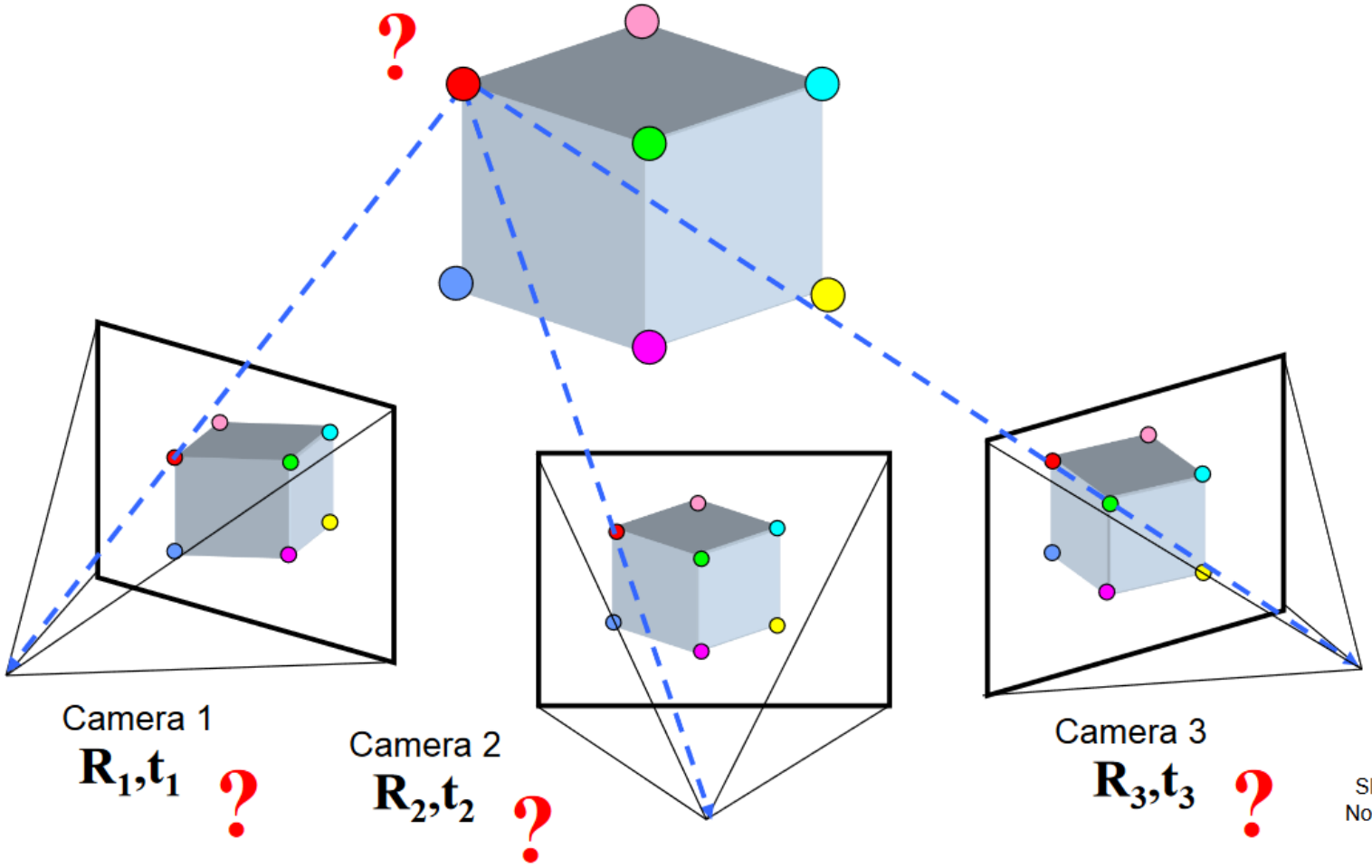
How can we estimate both 3D point positions and the relative camera transformations?

Sometimes also called bundle adjustment

Q: Why is it different than SLAM?

A: SLAM potentially includes

- loop closure
- dynamics constraints
- velocities, accelerations

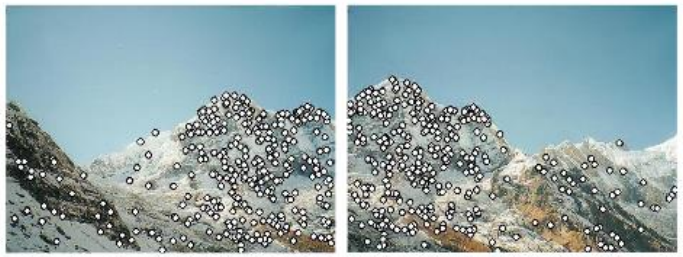


Slide credit: Noah Snavely

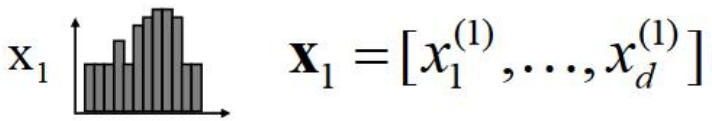
Basic underlying component in many of these problems:
keypoint detection and
matching across images

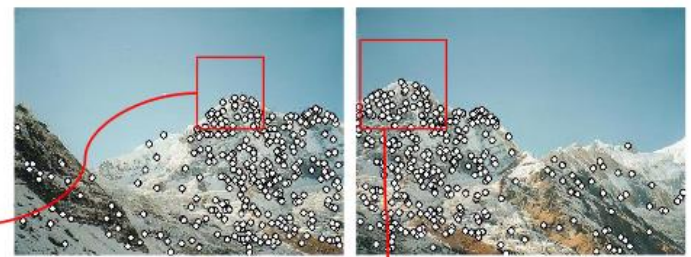
Local features: main components

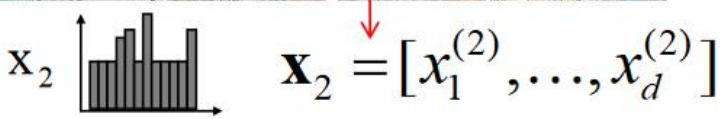
1) Detection:
Find a set of distinctive key points.



2) Description:
Extract feature descriptor around each interest point as vector.

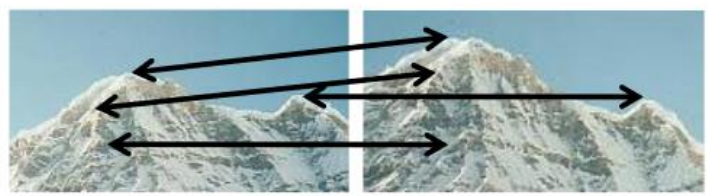






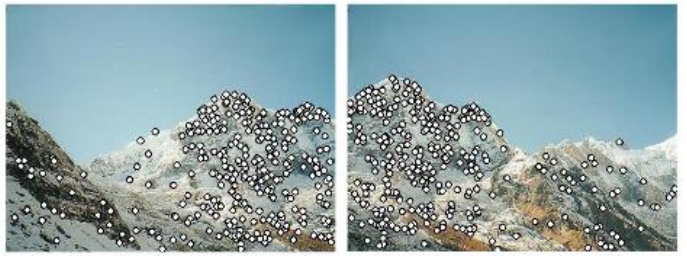
3) Matching:
Compute distance between feature vectors to find correspondence.

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

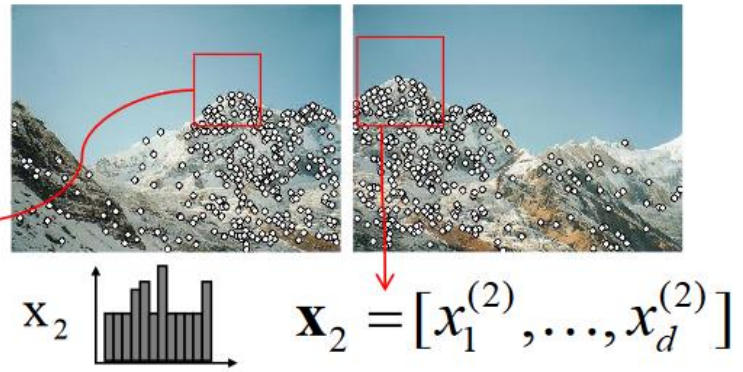


Local features: main components

1) Detection:
Find a set of distinctive key points.

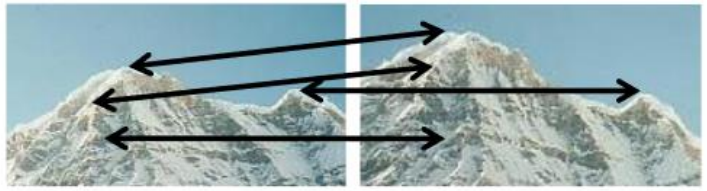


2) Description:
Extract feature descriptor around each interest point as vector.



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)

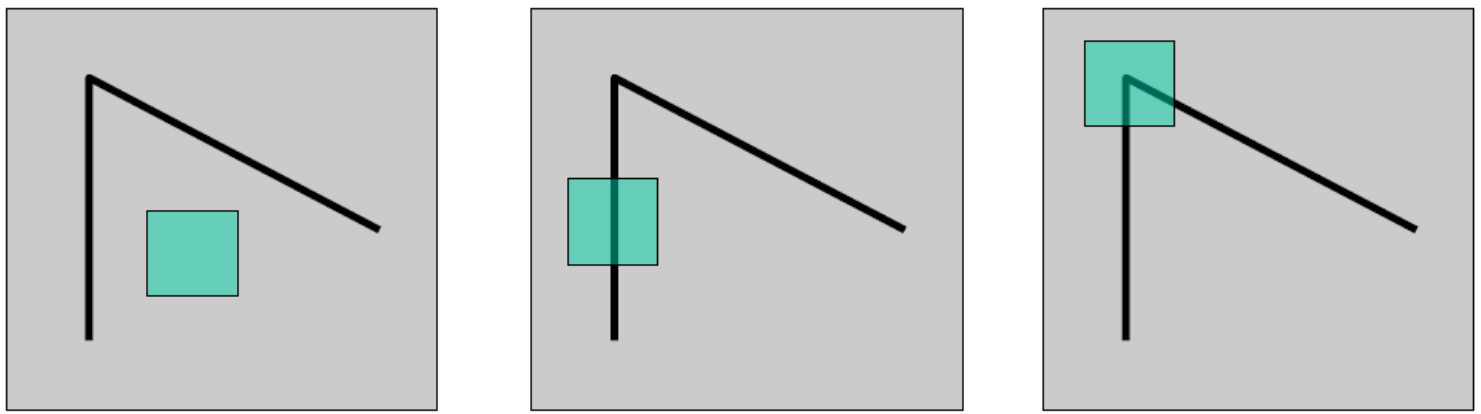
- scale-changes

- 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?

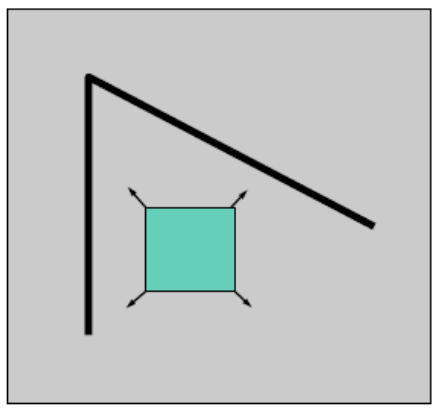


Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

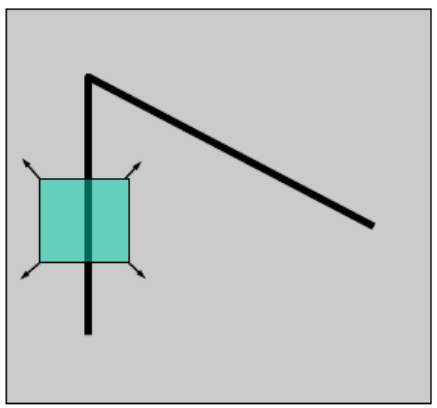
Feature detection

Local measure of feature uniqueness

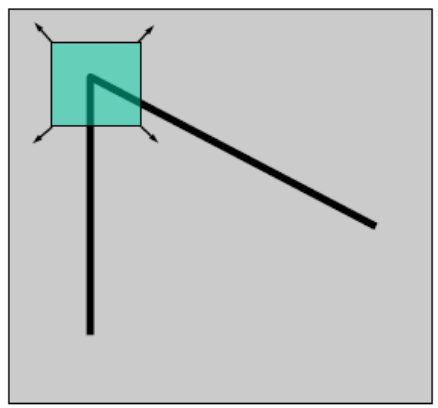
- How does the window change when you shift by a *small amount*?



"flat" region:
no change in all directions



"edge":
no change along the edge direction



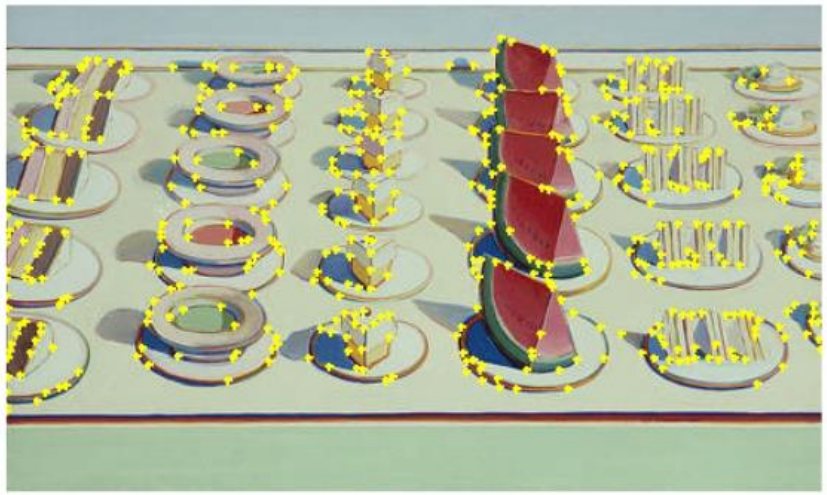
"corner":
significant change in all directions

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

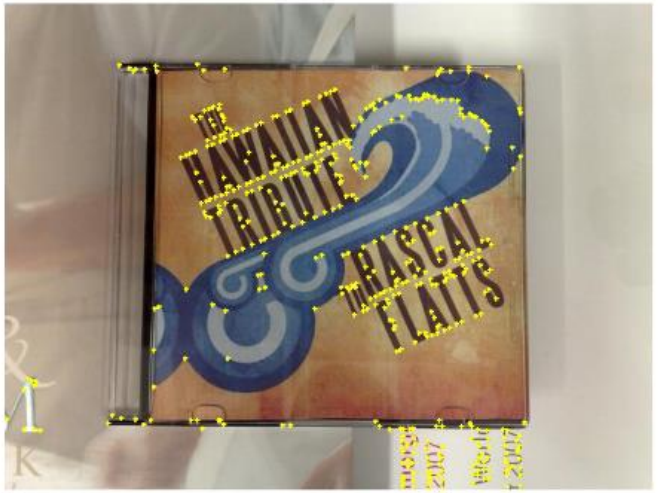
Corner detectors

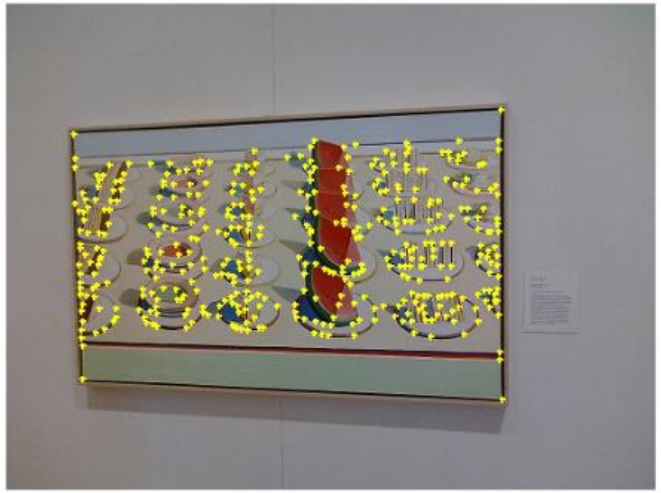
- Harris
- FAST
- Laplacian of Gaussian detector
- SUSAN
- Forstner

Superimposed Harris keypoints

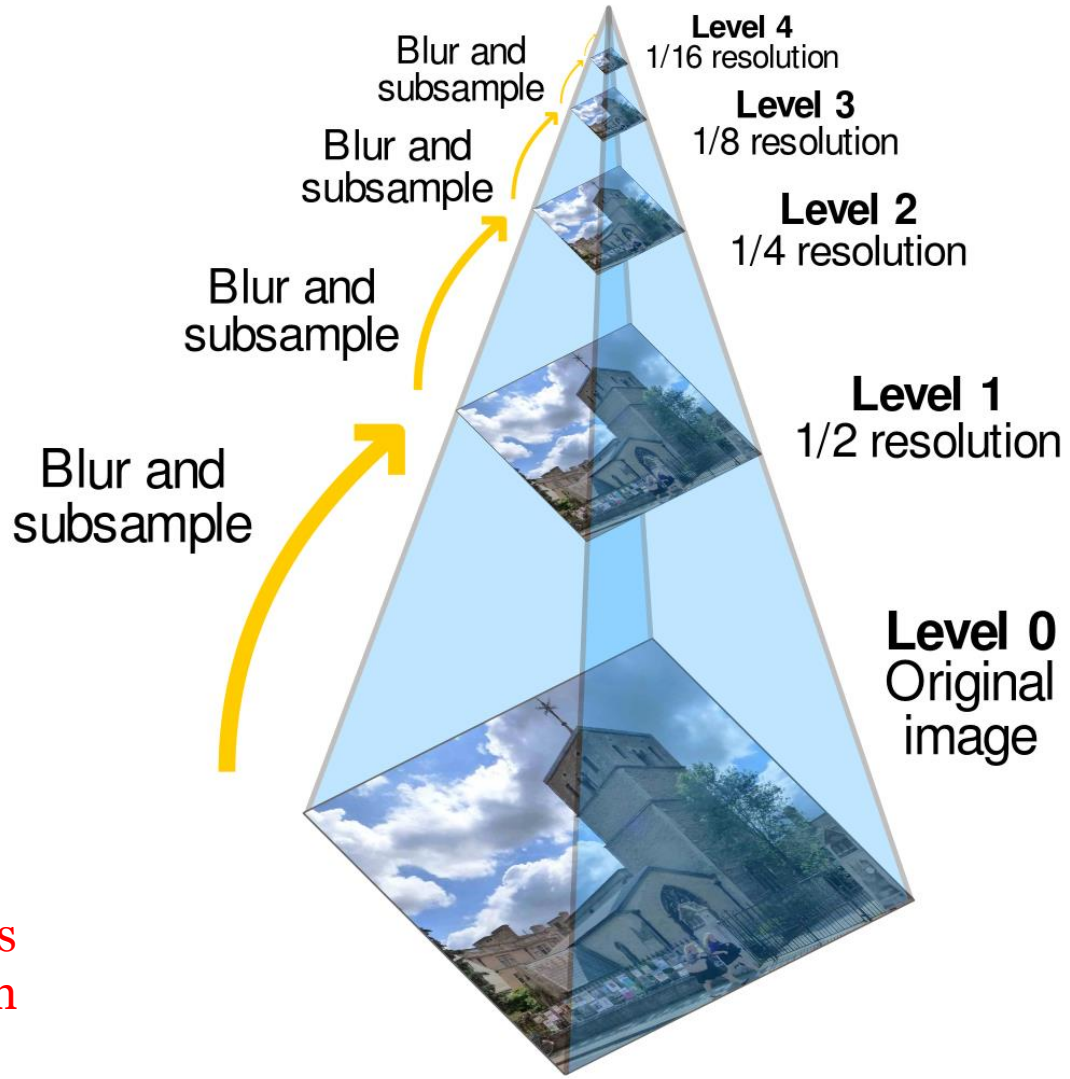


500 strongest keypoints





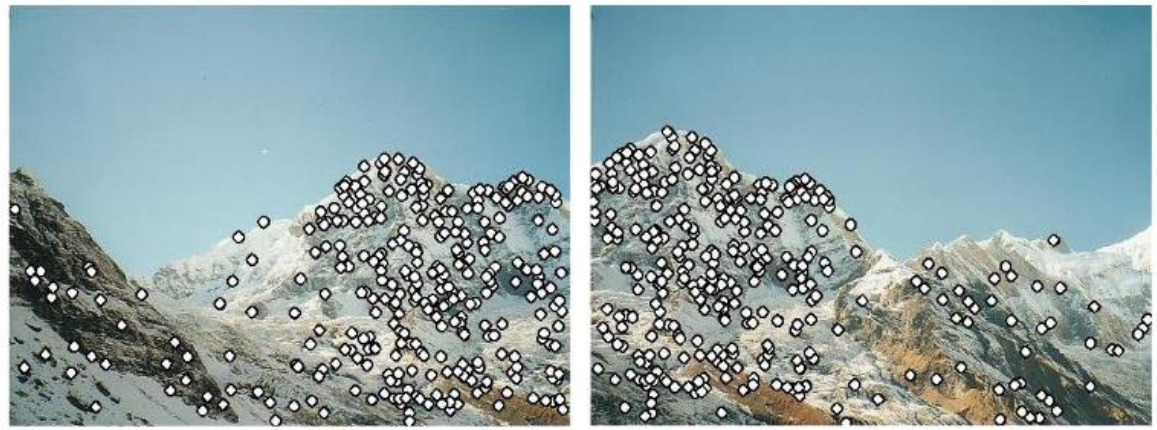
Scale-space representation



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 x 16 window around detection.
- Rotate and scale window according to discovered orientation Θ and scale σ (gain invariance).
- Compute gradients weighted by a Gaussian of variance half the window (for smooth falloff).

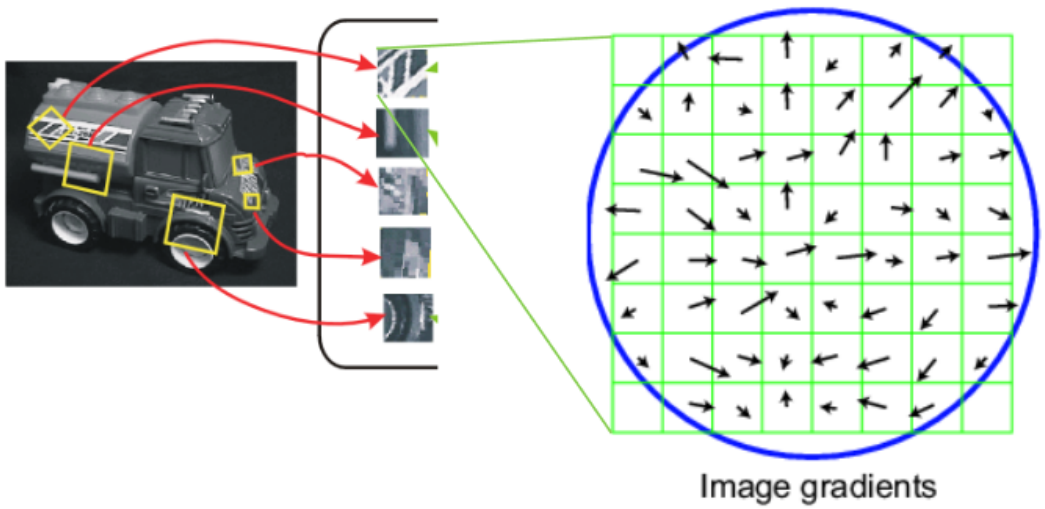
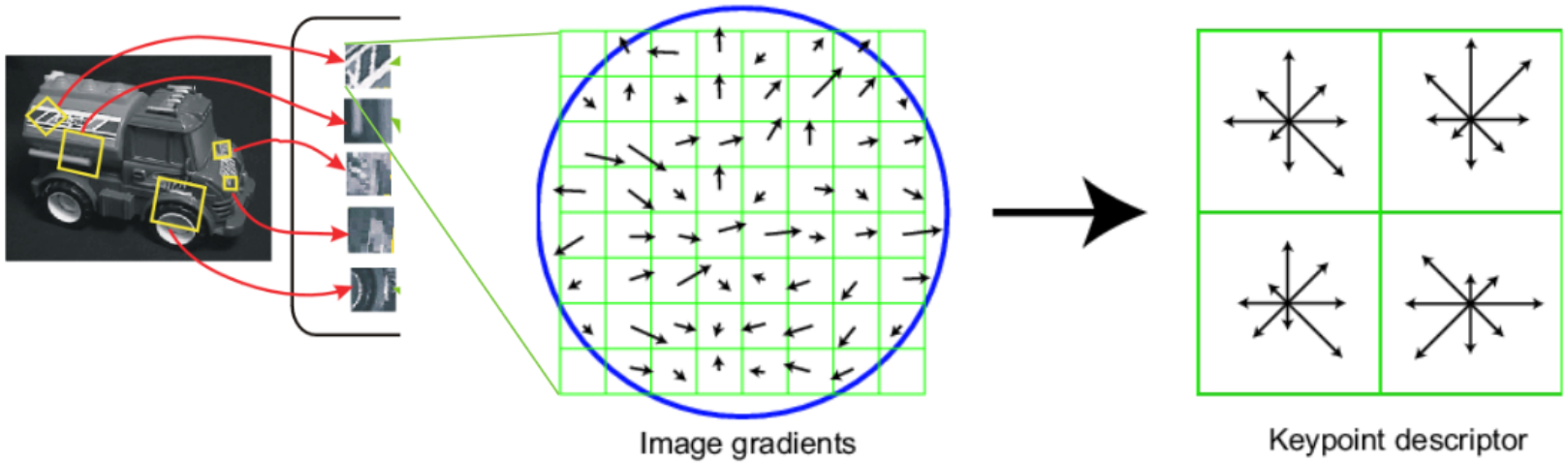


Image gradients

Actually 16x16, only showing 8x8

SIFT vector formation

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

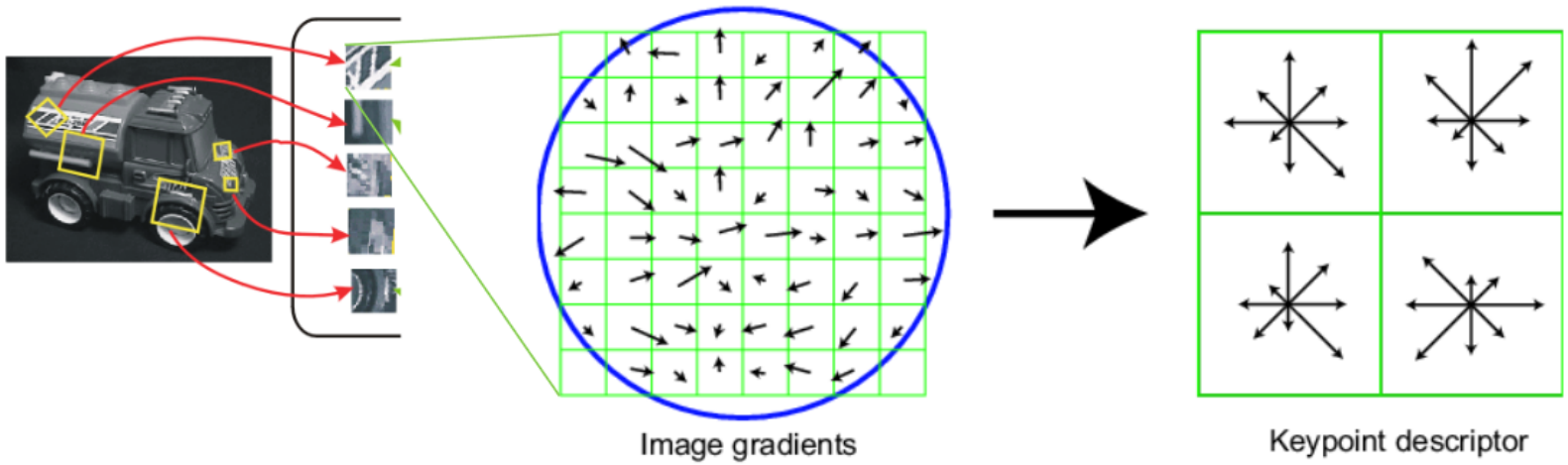


Showing only 2x2 here but is 4x4

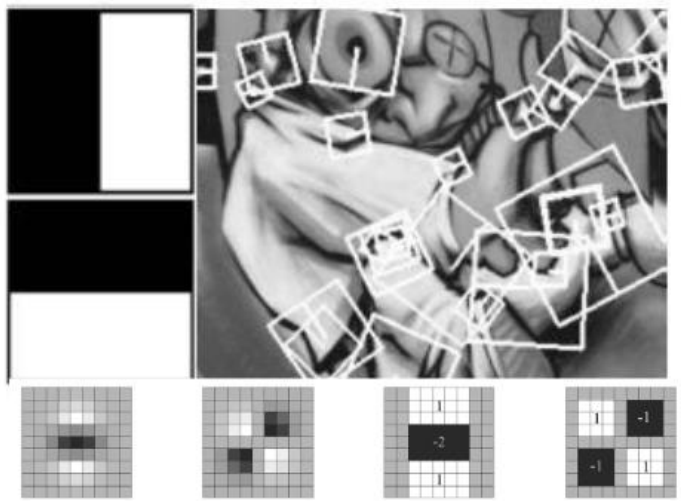
James Hays

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)

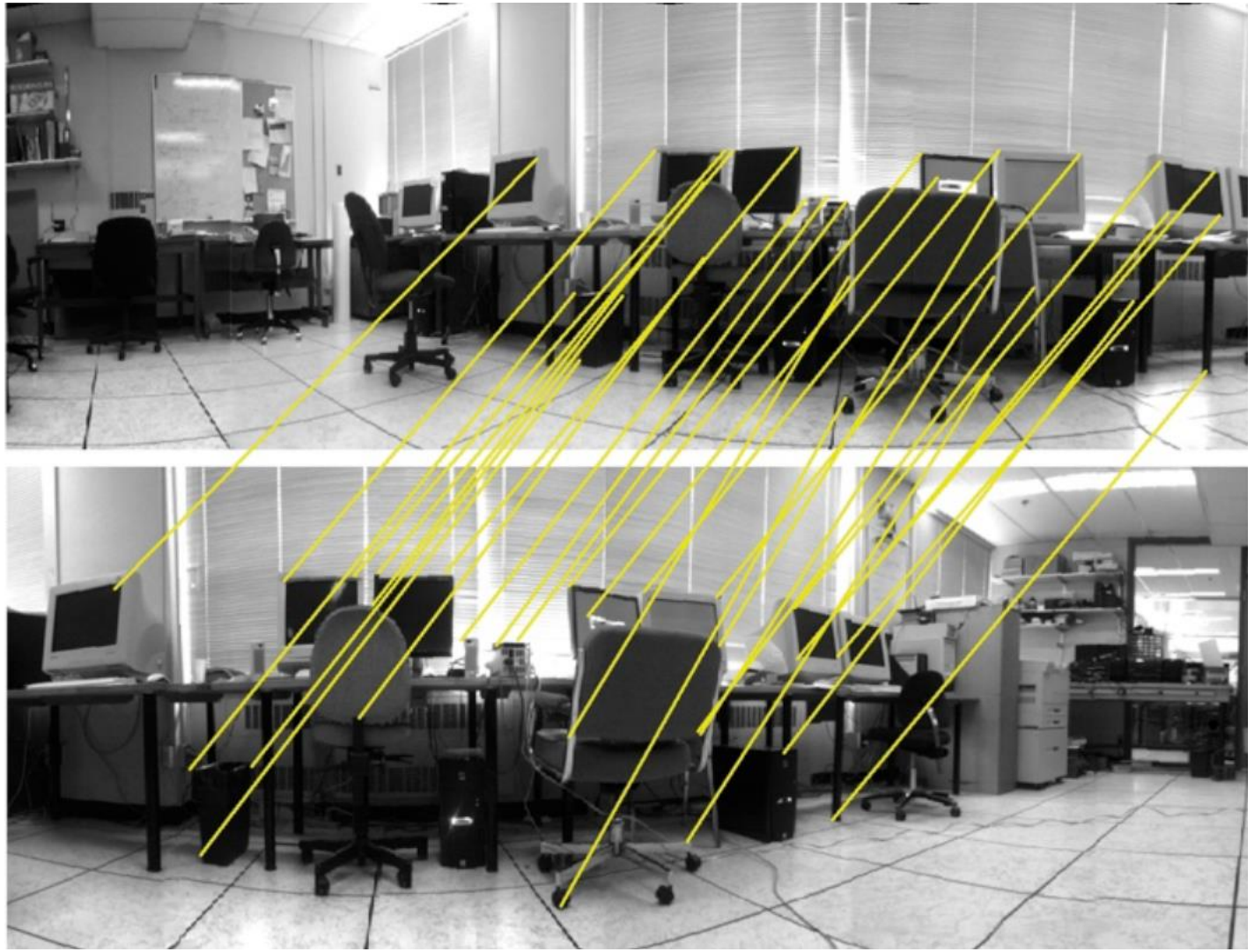
http://www.vision.ee.ethz.ch/~surf

[Bay, ECCV'06], [Cornelis, CVGPU'08]

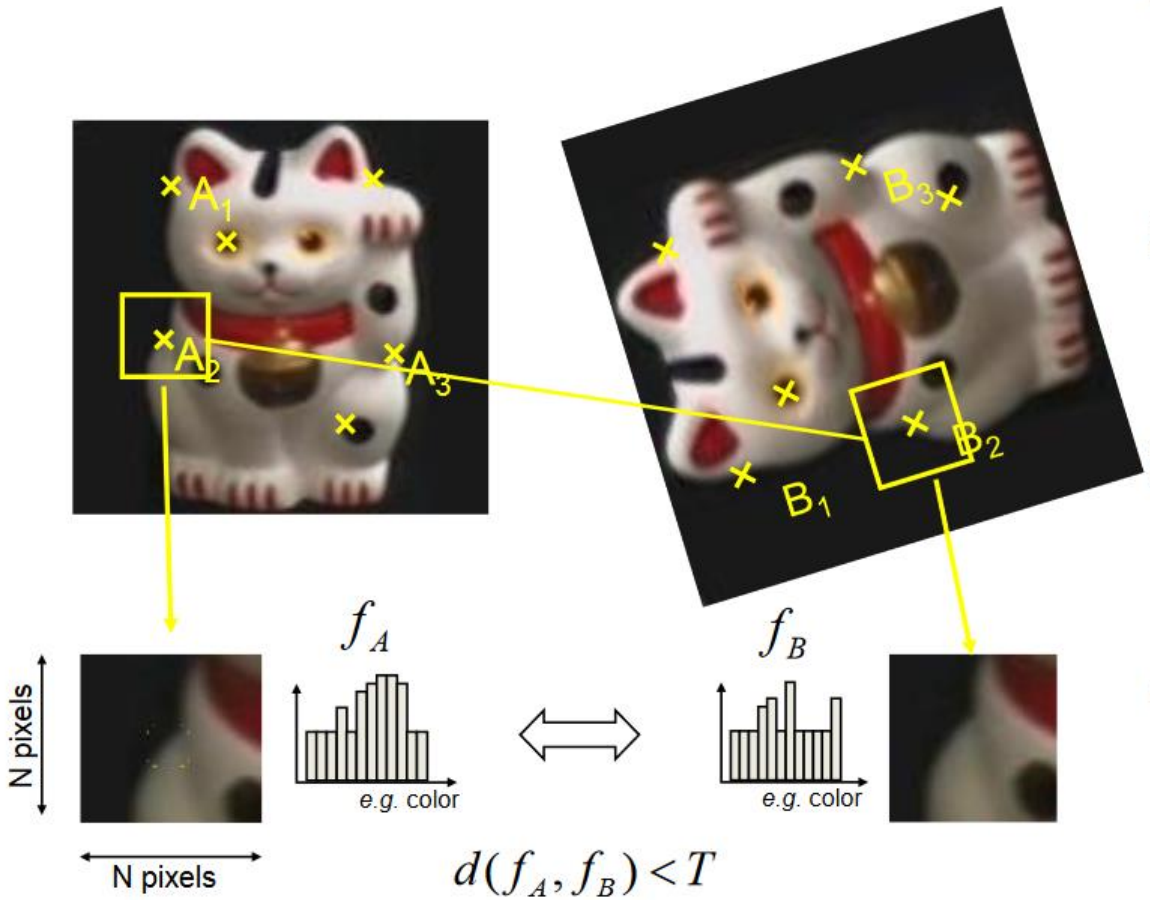
Many other local descriptors

- ORB
- BRIEF
- FREAK
- RootSIFT-PCA

Feature matching



Overview of Keypoint Matching



1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**
4. **Compute a local descriptor from the normalized region**
5. **Match local descriptors**

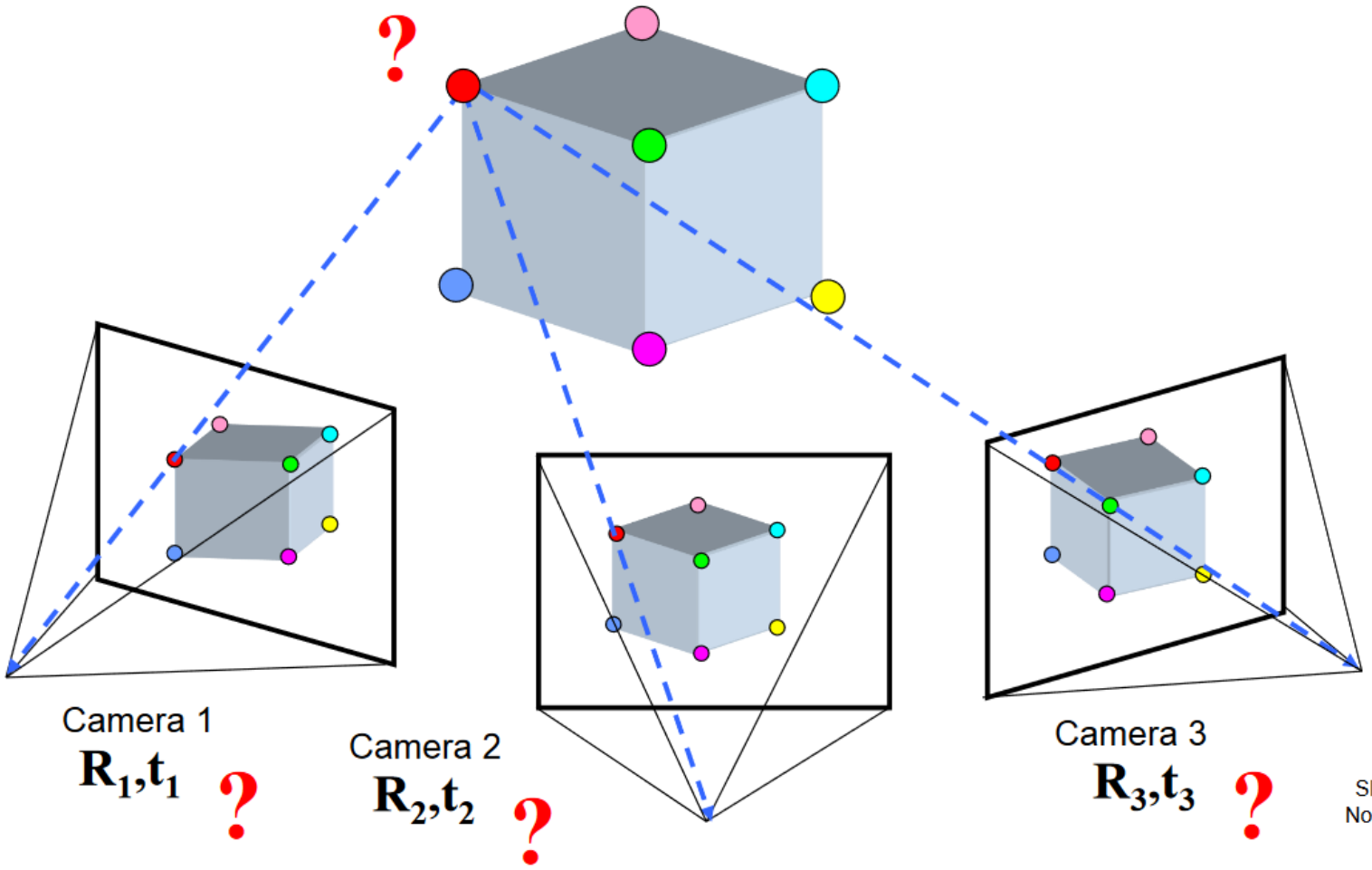
K. Grauman, B. Leibe

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

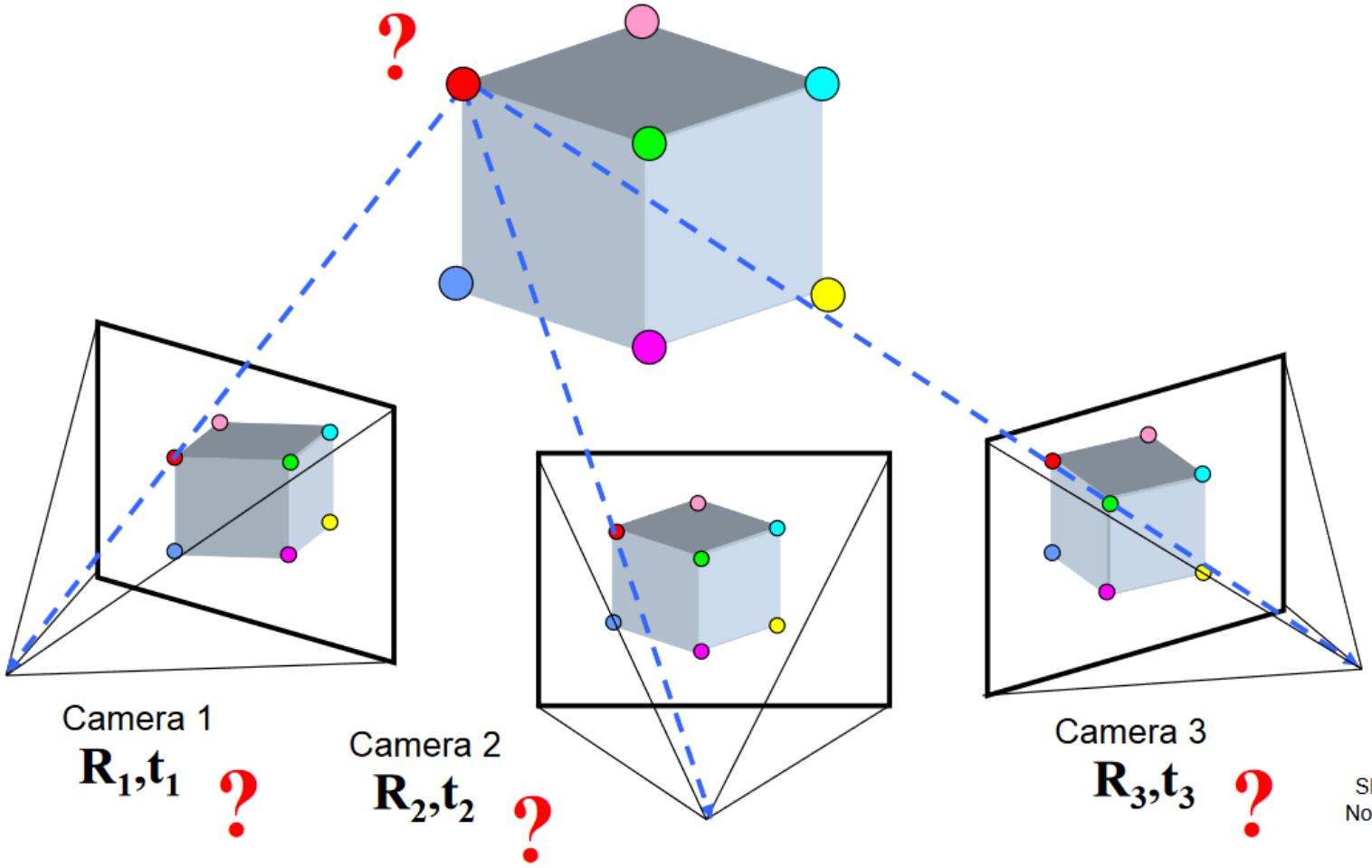
How can we estimate both 3D point positions and the relative camera transformations?

Sometimes also called bundle adjustment

Q: Why is it different than SLAM?

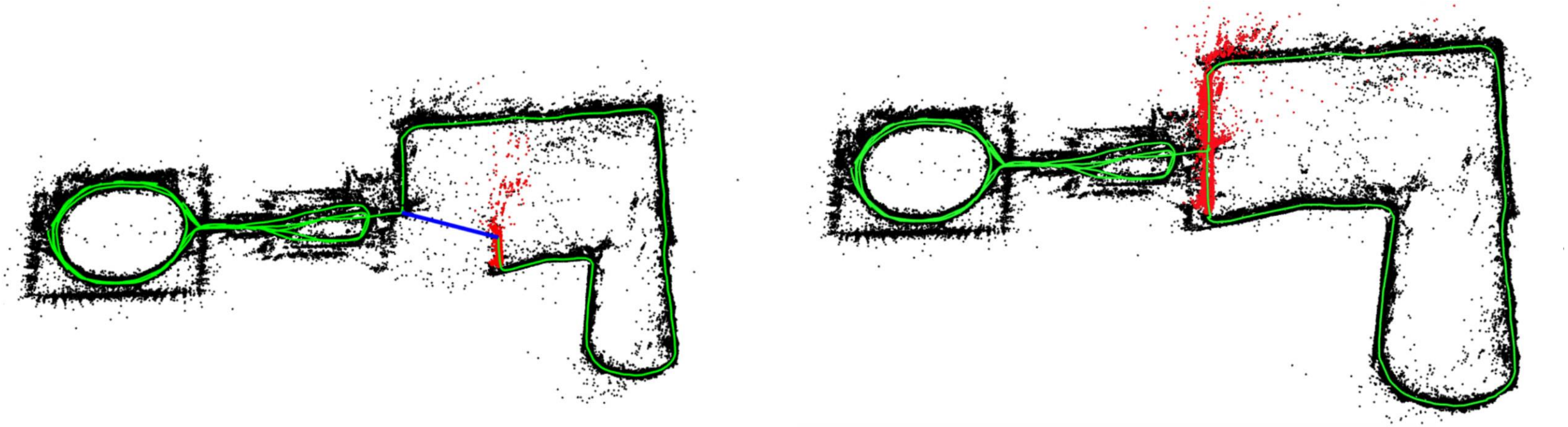
A: SLAM potentially includes

- loop closure
- dynamics constraints
- velocities, accelerations



Slide credit: Noah Snavely

Loop Closure in Visual SLAM



ORB-SLAM, Mur-Artal, Tardos, Montiel, Galvez-Lopez

Bundler (bundle adjustment/structure from motion)

Structure from Motion as Least Squares

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

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

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

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

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