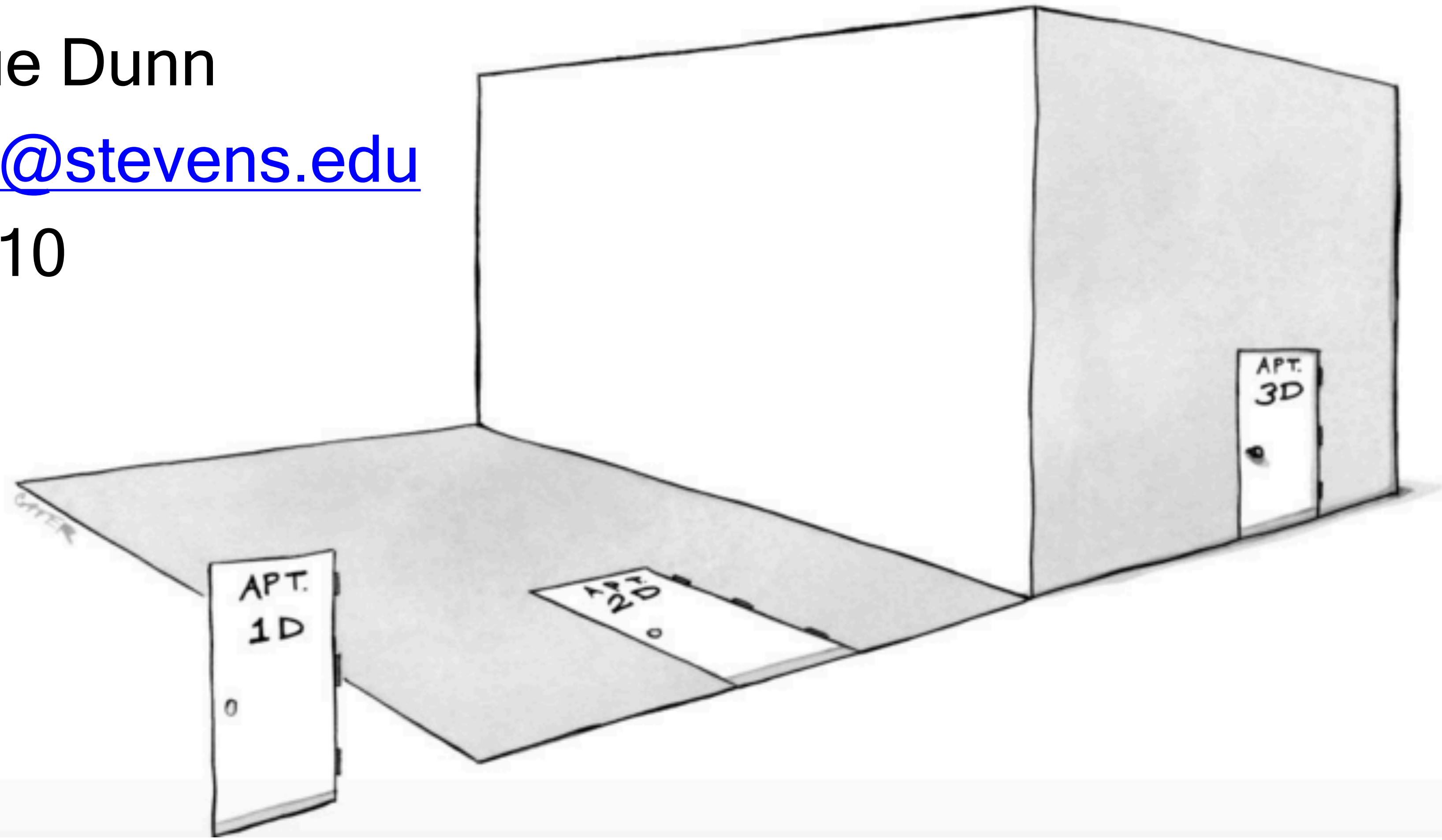
CS 532: 3D Computer Vision
Lecture 5

Enrique Dunn
edunn@stevens.edu
Lieb 310



Mid-Term Exam

- October 27th

Lecture Outline

- Feature tracking

Based on slides by Derek Hoiem, also partially based on sources by C. Tomasi, T. Kanade and T. Svoboda

- Intro to Covariance Matrices
- Simultaneous Localization and Mapping
  - Based on slides by William Green (then at Drexel)
  - See also “An Introduction to the Kalman Filter” by Greg Welch and Gary Bishop
    http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

Feature Matching

- Given a feature in I, how to find the best match in J?

- So far we have searched for best match by testing all possible translations by integer number of pixels
  - Restricted to be purely horizontal in stereo case

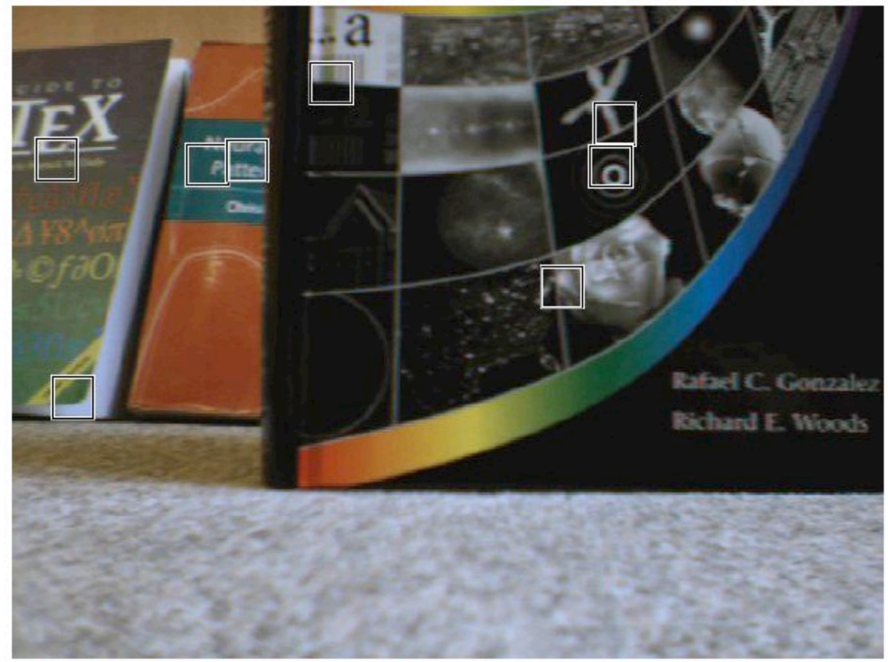
Kanade-Lucas-Tomasi Tracking

- Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th International Conference on Artificial Intelligence, pages 674–679, August 1981.
- Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.
- Jianbo Shi and Carlo Tomasi. Good features to track. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 593–600, 1994.
- Code: http://www.ces.clemson.edu/~stb/klt/

Camera Motion

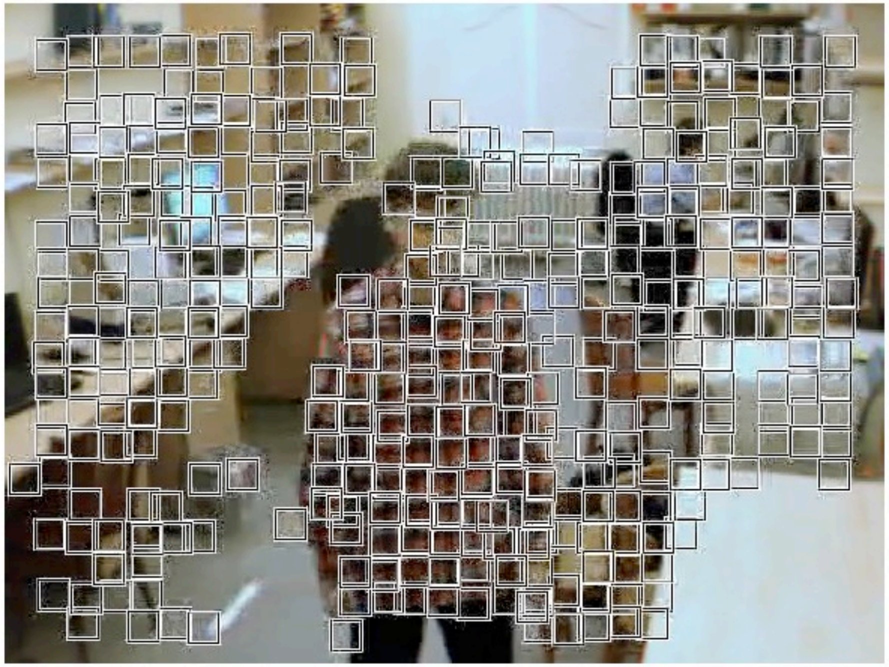
I

J

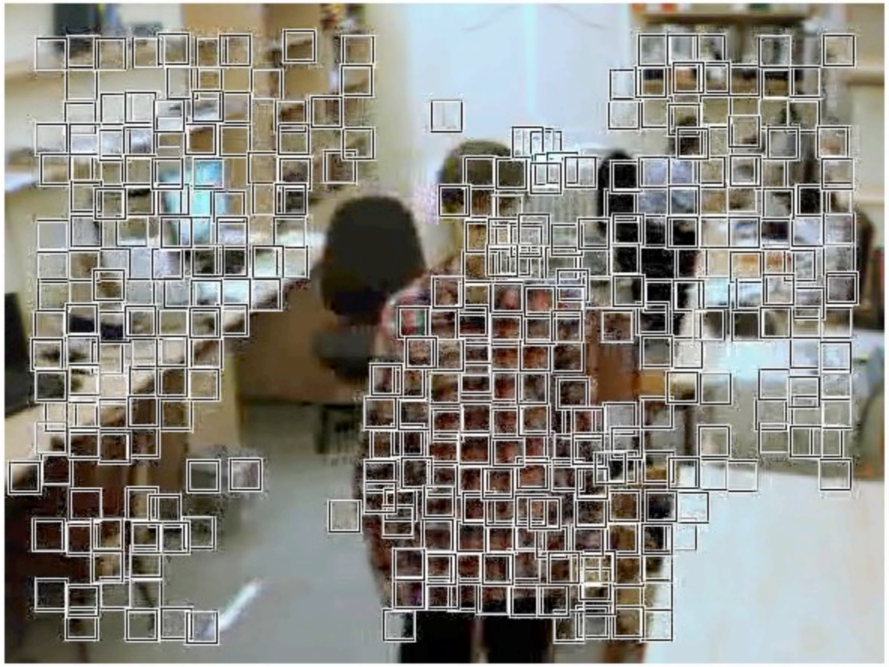


Object Motion

I



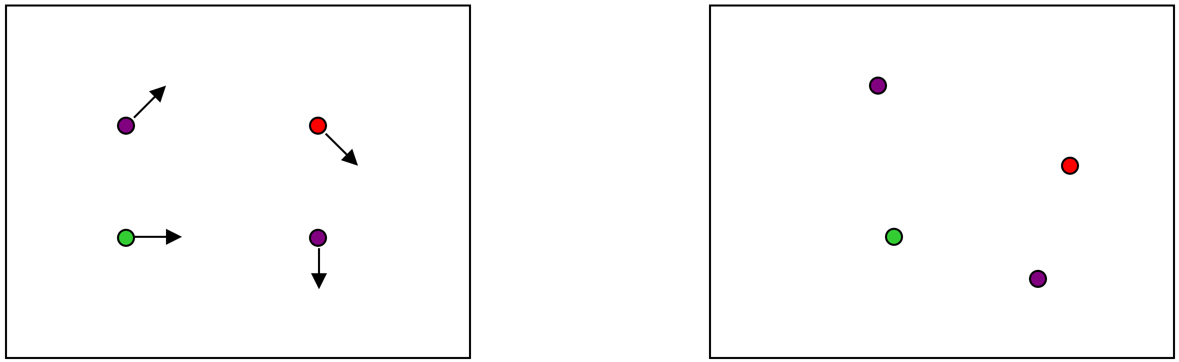
J



Feature Tracking

- Challenges
  - Figure out which features can be tracked
  - Efficiently track across frames
  - Some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)
  - Drift: small errors can accumulate as appearance model is updated
  - Points may appear or disappear: need to be able to add/delete tracked points

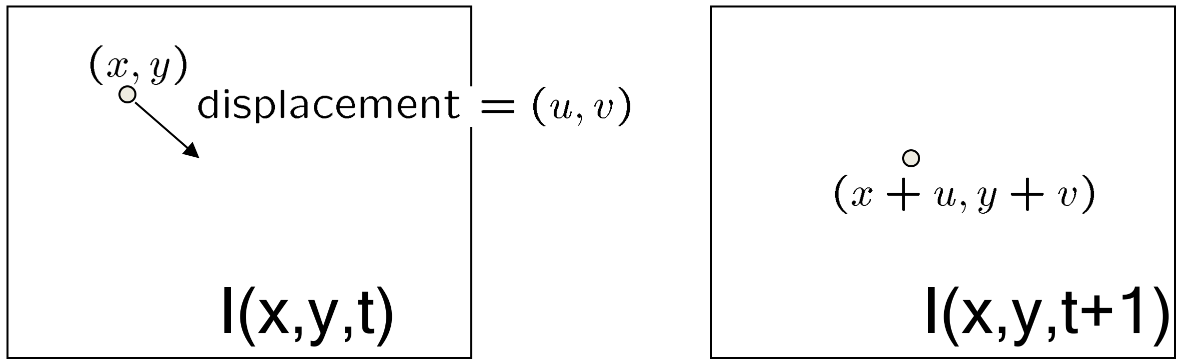
Feature Tracking



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

- Given two subsequent frames, estimate the point translation
- Key assumptions of Lucas-Kanade Tracker
  - **Brightness constancy:** projection of the same point looks the same in every frame
  - **Small motion:** points do not move very far
  - **Spatial coherence:** points move like their neighbors

The Brightness Constancy Constraint



- Brightness Constancy Equation:

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

Take Taylor expansion of I(x+u, y+v, t+1) at (x,y,t) to linearize the right side:

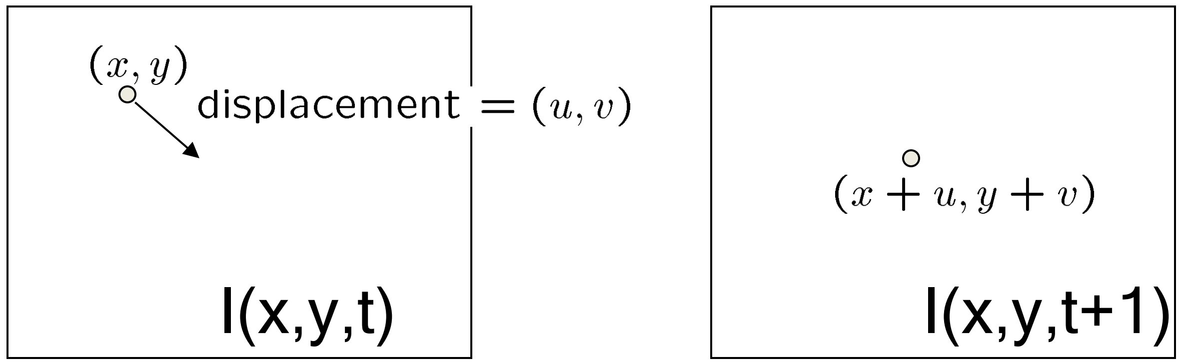
Image derivative along x   Difference over frames

$$I(x+u, y+v, t+1) \approx I(x, y, t) + \boxed{I_x} \cdot u + I_y \cdot v + \boxed{I_t}$$

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

Hence, $I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \rightarrow \nabla \mathrm{I} \cdot [\mathrm{u} \ \ \mathrm{v}]^{\mathrm{T}} + \mathrm{I_t} = 0$

The Brightness Constancy Constraint



- Brightness Constancy Equation:

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

Take Taylor expansion of I(x+u, y+v, t+1) at (x,y,t) to linearize the right side:

Image derivative along x　　Difference over frames

$$I(x+u, y+v, t+1) \approx I(x, y, t) + \boxed{I_x} \cdot u + I_y \cdot v + \boxed{I_t}$$

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

Hence, $I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \rightarrow \nabla \mathrm{I} \cdot [\mathrm{u} \;\; \mathrm{v}]^{\mathrm{T}} + \mathrm{I_t} = 0$

Taylor Expansion

$$f(x)$$

$$f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+\cdots.$$

$$\sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(x-a)^n$$

The Brightness Constancy Constraint

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

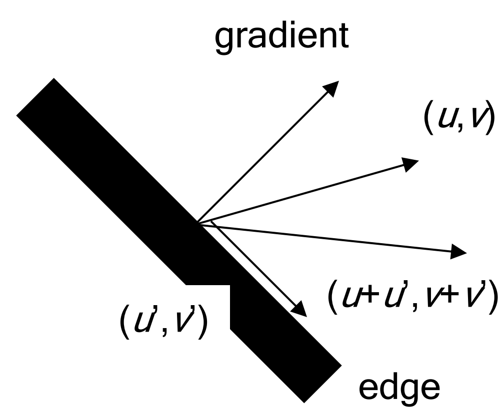
$$\nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$$

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

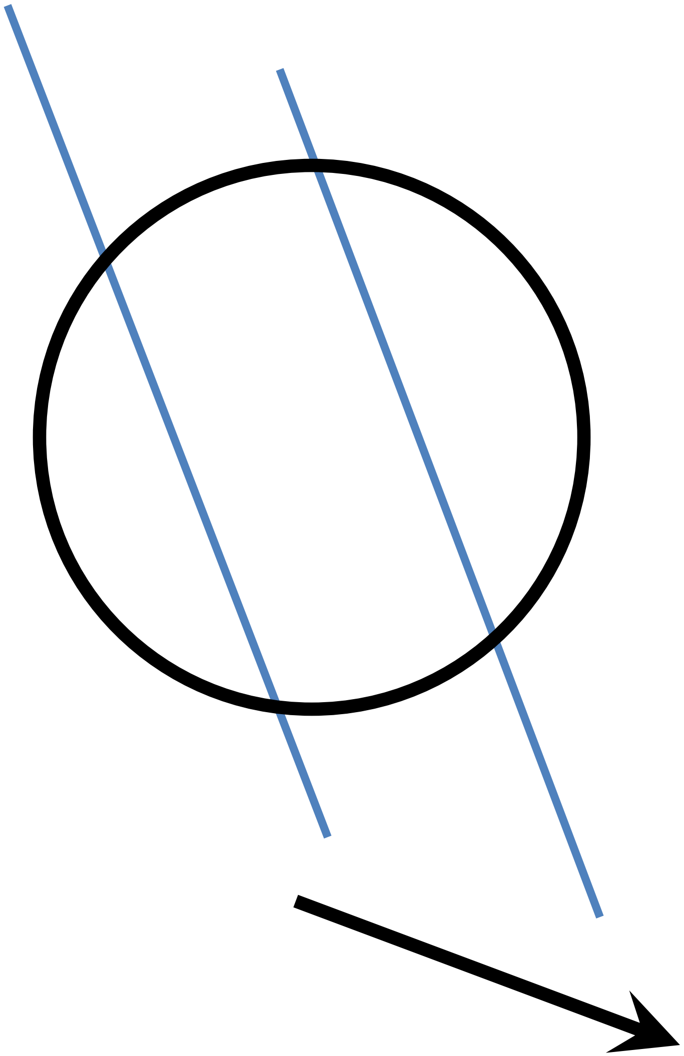
The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

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

$$\nabla I \cdot \begin{bmatrix} u' & v' \end{bmatrix}^T = 0$$

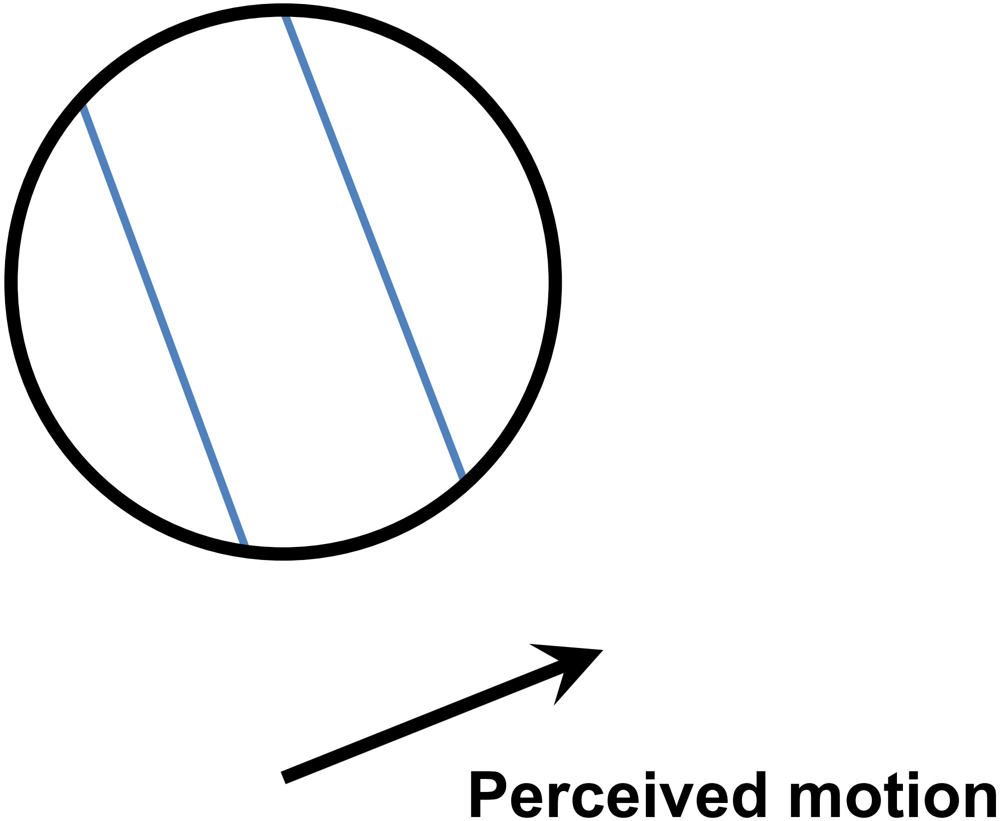


The Aperture Problem



Actual motion

The Aperture Problem



The Barber Pole Illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

The Barber Pole Illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

Solving the Ambiguity…

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

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

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

Solving the Ambiguity…

- Least squares problem:

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix} \qquad \underset{25\times2}{A} \; \underset{2\times1}{d} = \underset{25\times1}{b}$$

Matching Patches across Images

- Overconstrained linear system

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix} \qquad \underset{25\text{x}2}{A} \ \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Least squares solution for *d* given by $(A^T A)\ d = A^T b$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

The summations are over all pixels in the K x K window

Conditions for Solvability

Optimal (u, v) satisfies Lucas-Kanade equation

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

When is this solvable? I.e., what are good points to track?

- **A^TA** should be invertible
- **A^TA** should not be too small due to noise
  – eigenvalues λ_1 and λ_2 of **A^TA** should not be too small
- **A^TA** should be well-conditioned
  – λ_1 / λ_2 should not be too large (λ_1 = largest eigenvalue)

Does this remind you of anything?

Criteria for Harris corner detector

M = A^TA is the *second moment matrix* ! (Harris corner detector…)

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

- Eigenvectors and eigenvalues of A^TA relate to edge direction and magnitude
    - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
    - The other eigenvector is orthogonal to it

Low-texture region

$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude

– small λ_1, small λ_2

Edge

$$\sum \nabla I (\nabla I)^T$$

- gradients very large or very small
- large λ_1, small λ_2

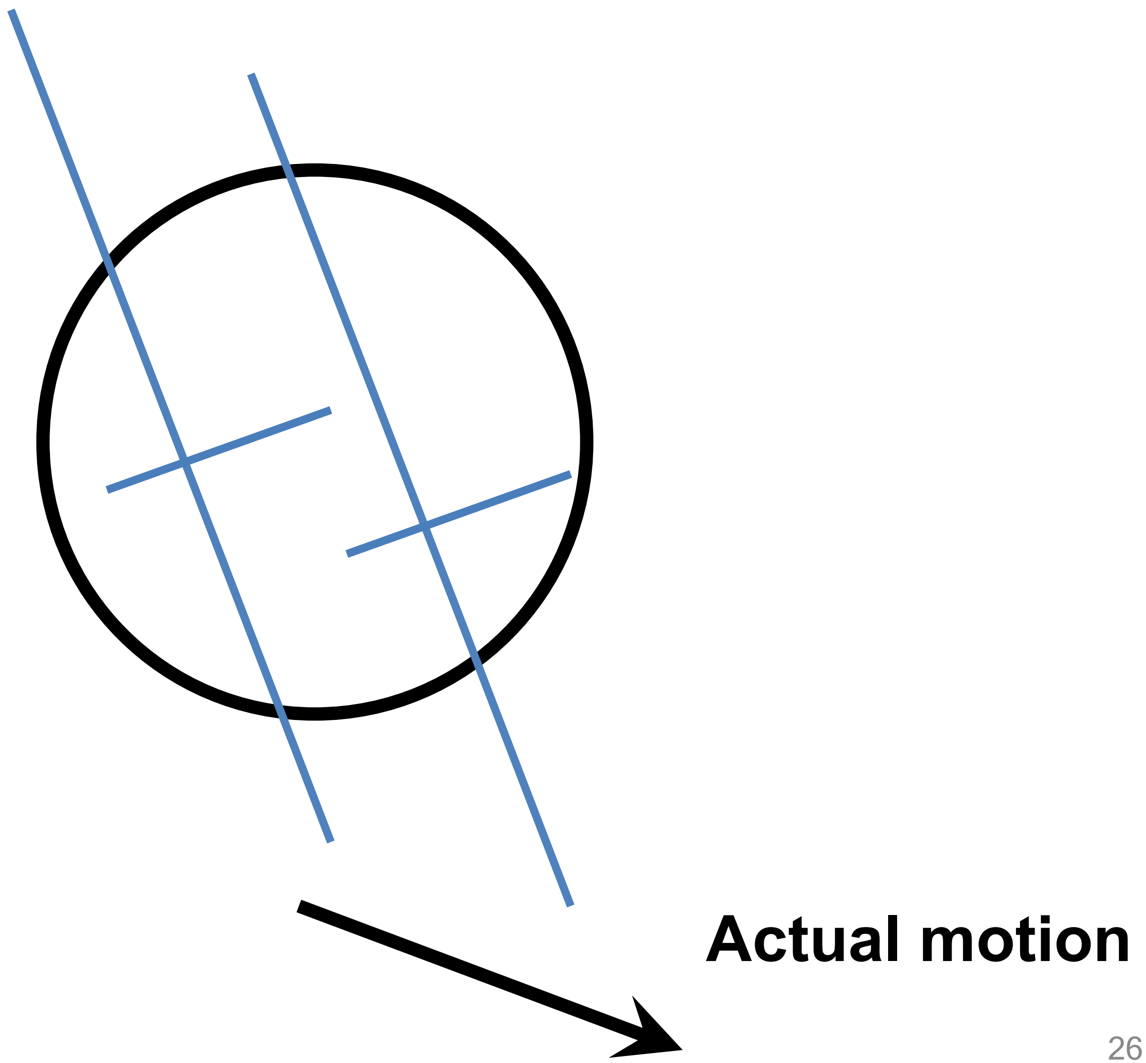
High-texture Region

$$\sum \nabla I (\nabla I)^T$$

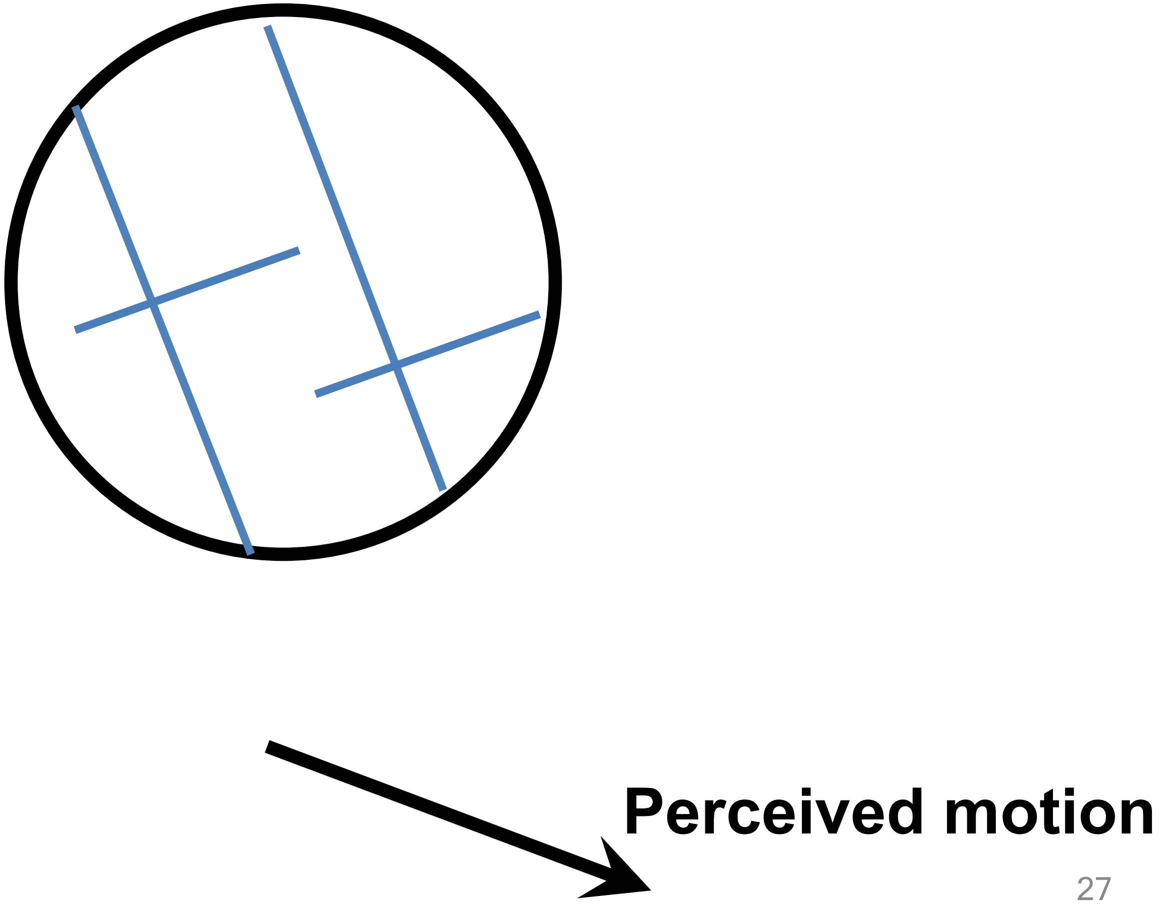
– gradients are different, large magnitudes

– large λ_1, large λ_2

The Aperture Problem Resolved



The Aperture Problem Resolved



Revisiting the small motion assumption

- Is this motion small enough?
  - Probably not–it's much larger than one pixel (2nd order terms dominate)
  - How might we solve this problem?

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Dealing with Larger Motion: Iterative Refinement

Original (x,y) position

$I_t = I(x', y', t+1) - I(x, y, t)$

1. Initialize (x',y') = (x,y)
2. Compute (u,v) by

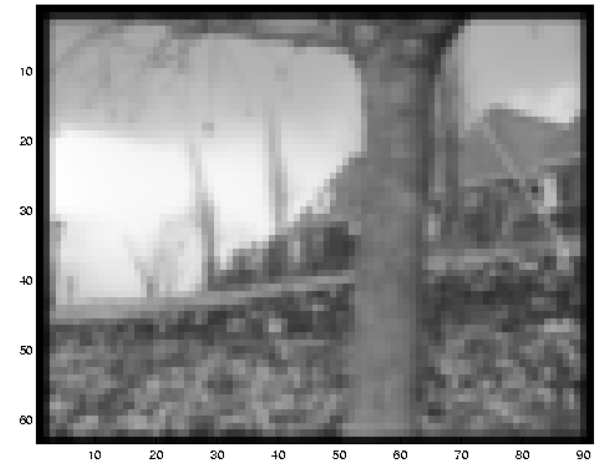
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

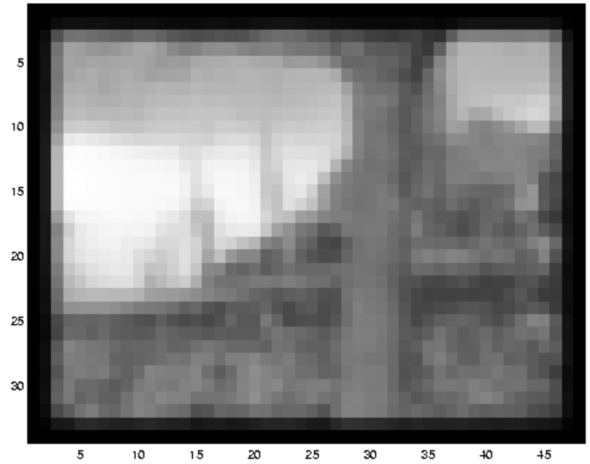
2nd moment matrix for feature patch in first image

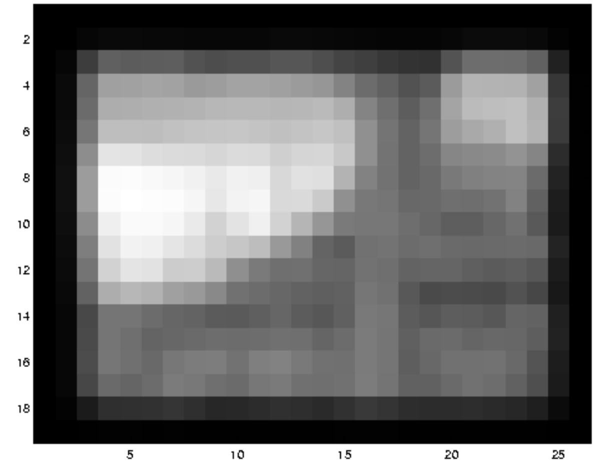
displacement

3. Shift window by (u, v): `x'=x'+u; y'=y'+v;`
4. Recalculate I_t
5. Repeat steps 2-4 until change is small
   - Use interpolation for subpixel values

Reduce the resolution!



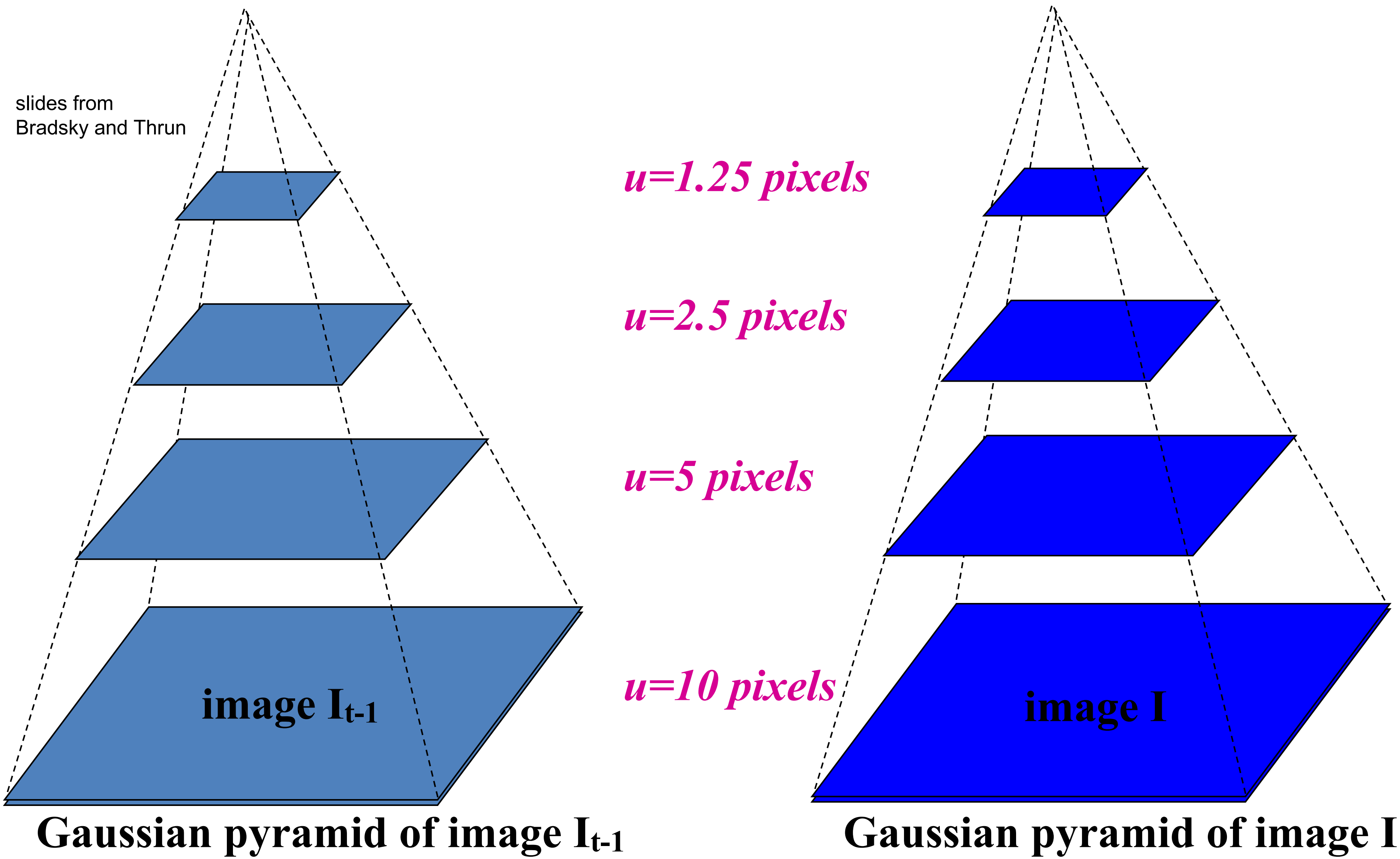




* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

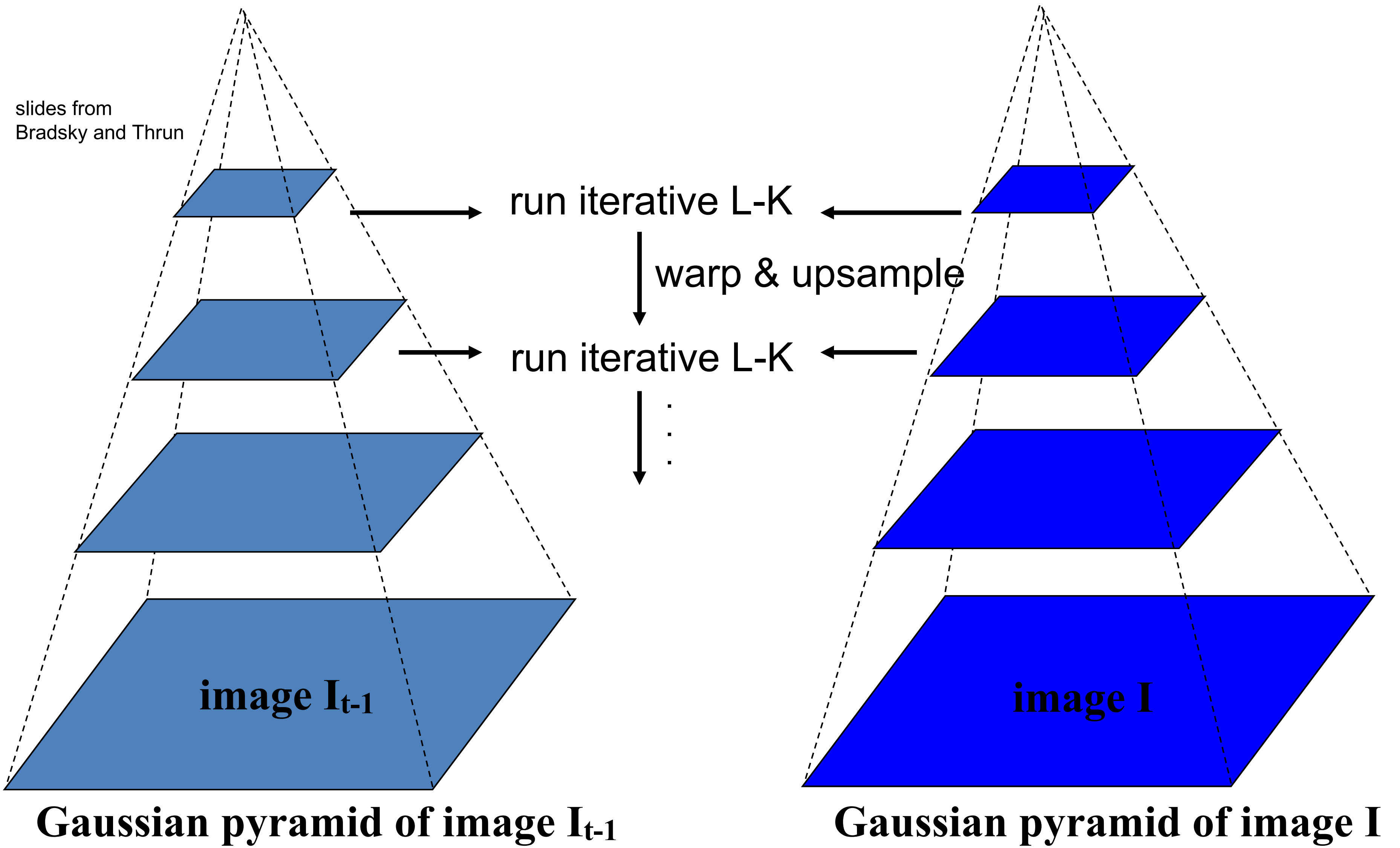
Coarse-to-fine optical flow estimation

slides from
Bradsky and Thrun

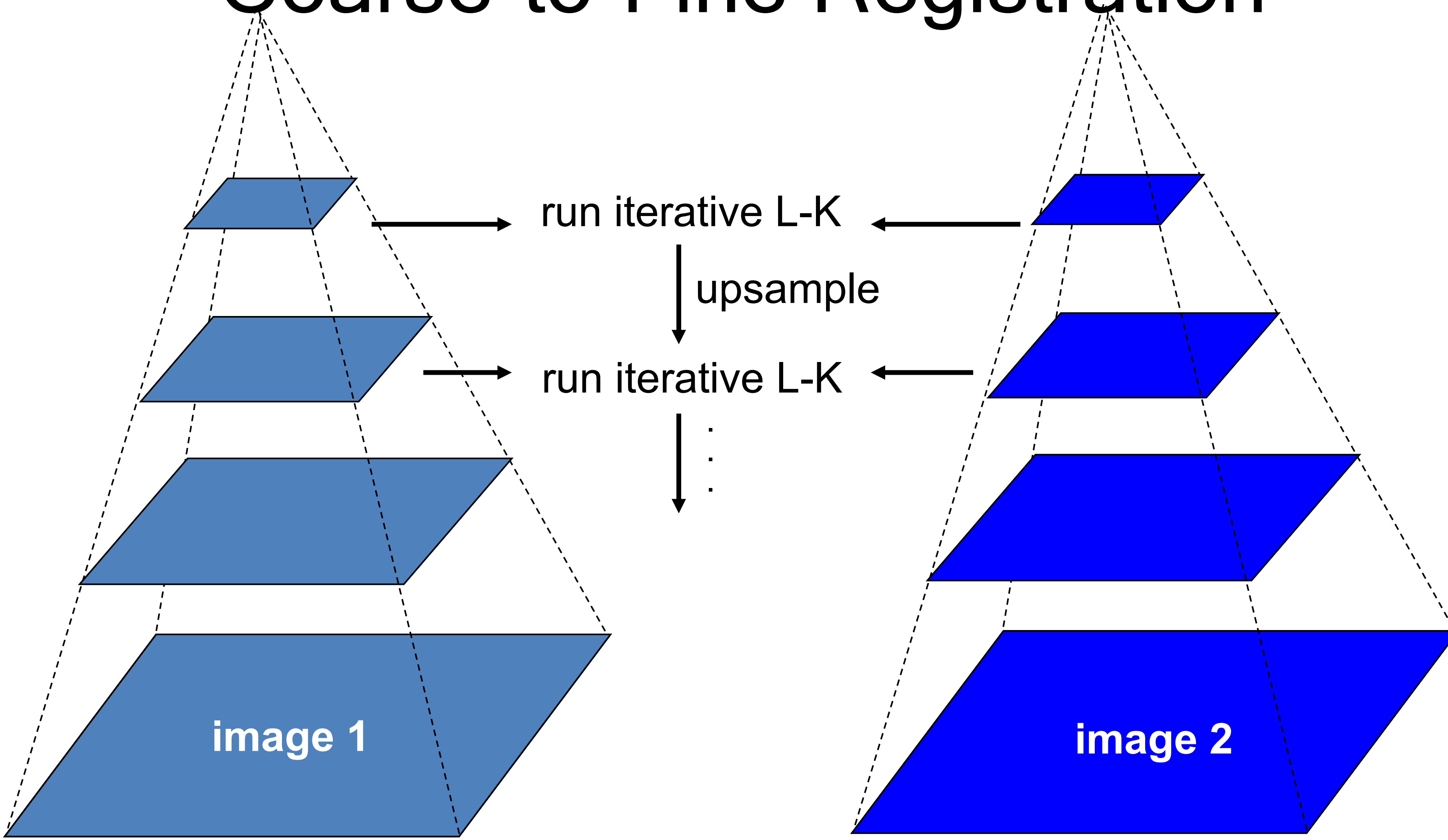


Coarse-to-fine optical flow estimation

slides from
Bradsky and Thrun



Dealing with Larger Motion: Coarse-to-Fine Registration



Gaussian pyramid of image 1 (t) **Gaussian pyramid of image 2 (t+1)**

Shi-Tomasi Feature Tracker

- Find good features using eigenvalues of second-moment matrix (e.g., Harris detector or threshold on the smallest eigenvalue)
  - Key idea: “good” features to track are the ones whose motion can be estimated reliably
- Track from frame to frame with Lucas-Kanade
  - This amounts to assuming a translation model for frame-to-frame feature movement
- Check consistency of tracks by *affine* registration to the first observed instance of the feature
  - Affine model is more accurate for larger displacements
  - Comparing to the first frame helps to minimize drift

Tracking Example

Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

Summary of KLT tracking

- Find a good point to track (Harris corners)
- Use intensity second moment matrix and difference across frames to find displacement
- Iterate and use coarse-to-fine search to deal with larger movements
- When creating long tracks, check appearance of registered patch against appearance of initial patch to find points that have drifted

Covariance

Covariance

- Covariance is a numerical measure that shows how much two random variables change together

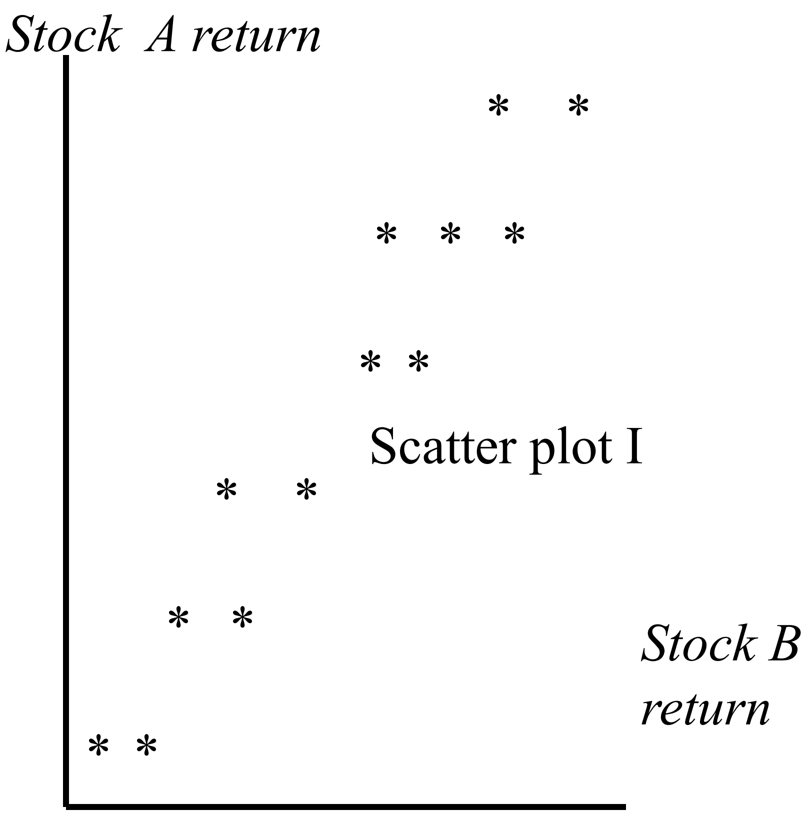
$$\mathrm{cov}(X,Y) = \mathrm{E}\left[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])\right],$$

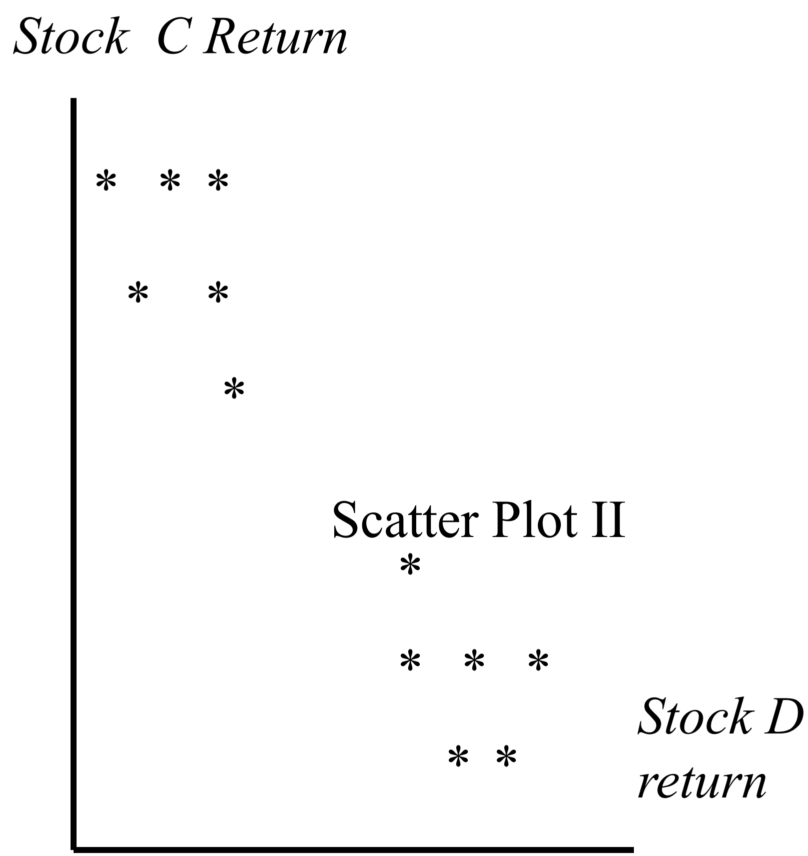
$$\mathrm{cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - E(X))(y_i - E(Y)).$$

- Positive covariance: if one increases, the other is likely to increase
- Negative covariance: …
- More precisely: the covariance is a measure of the *linear* dependence between the two variables

Covariance Example

Relationships between the returns of different stocks





Correlation Coefficient

- One may be tempted to conclude that if the covariance is larger, the relationship between two variables is stronger (in the sense that they have stronger linear relationship)
- The correlation coefficient is defined as:

$$\rho_{jk} = \frac{E\left[(Y_{ij} - \mu_j)(Y_{ik} - \mu_k)\right]}{\sigma_j \sigma_k}$$

Correlation Coefficient

$$\rho_{jk} = \frac{E\left[(Y_{ij} - \mu_j)(Y_{ik} - \mu_k)\right]}{\sigma_j \sigma_k}$$

- The correlation coefficient, unlike covariance, is a measure of dependence free of scales of measurement of Y_{ij} and Y_{ik}
- By definition, correlation must take values between −1 and 1
- A correlation of 1 or −1 is obtained when there is a perfect linear relationship between the two variables

Covariance Matrix

- For the vector of repeated measures, $Y_i = (Y_{i1}, Y_{i2}, \ldots, Y_{in})$, we define the covariance matrix, $\mathrm{Cov}(Y_i)$:

$$
\mathrm{Cov}\begin{pmatrix} Y_{i1} \\ Y_{i2} \\ \vdots \\ Y_{in} \end{pmatrix} = \begin{pmatrix} \mathrm{Var}(Y_{i1}) & \mathrm{Cov}(Y_{i1}, Y_{i2}) & \cdots & \mathrm{Cov}(Y_{i1}, Y_{in}) \\ \mathrm{Cov}(Y_{i2}, Y_{i1}) & \mathrm{Var}(Y_{i2}) & \cdots & \mathrm{Cov}(Y_{i2}, Y_{in}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}(Y_{in}, Y_{i1}) & \mathrm{Cov}(Y_{in}, Y_{i2}) & \cdots & \mathrm{Var}(Y_{in}) \end{pmatrix}
$$

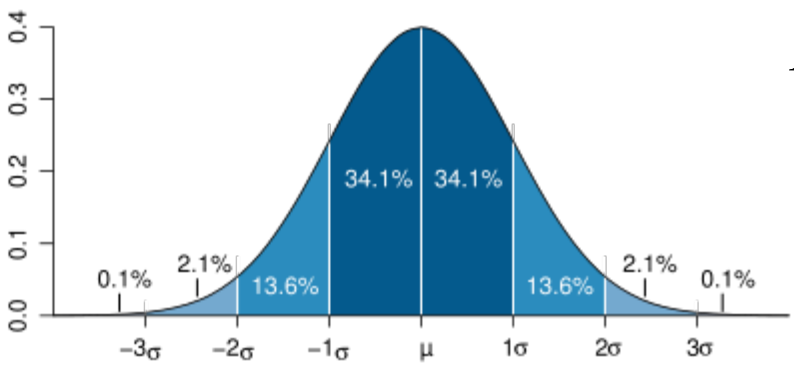
$$
= \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{pmatrix},
$$

where $\mathrm{Cov}(Y_{ij}, Y_{ik}) = \sigma_{jk} = \sigma_{kj} = \mathrm{Cov}(Y_{ik}, Y_{ij})$.

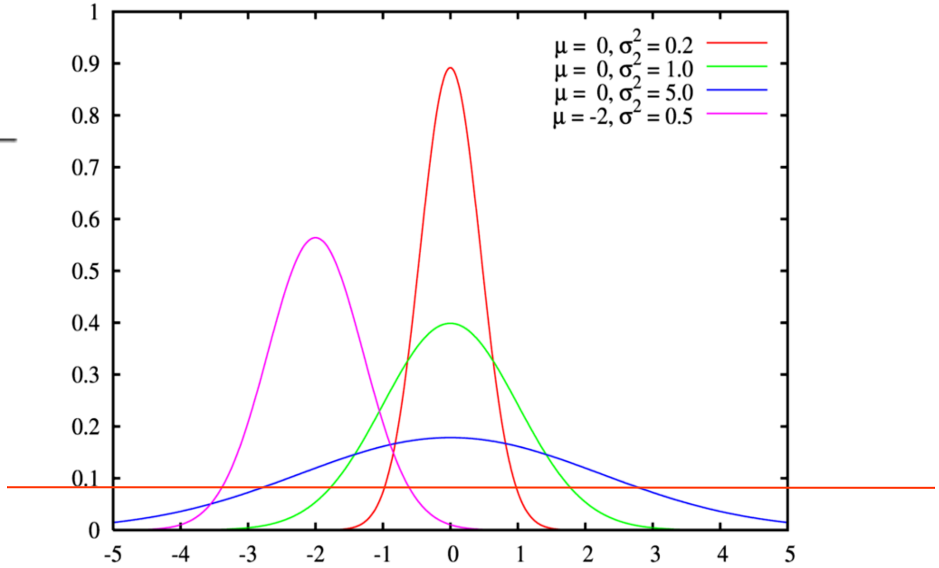
- It is a symmetric, square matrix

Variance and Confidence Intervals

- Single Gaussian (normal) random variable



$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = N(\mu, \sigma^2)$$



Multivariate Normal Density

– The multivariate normal density in d dimensions is:

$$P(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^t \Sigma^{-1}(x-\mu)\right]$$

where:

$x = (x_1, x_2, \ldots, x_d)^t$

$\mu = (\mu_1, \mu_2, \ldots, \mu_d)^t$ mean vector

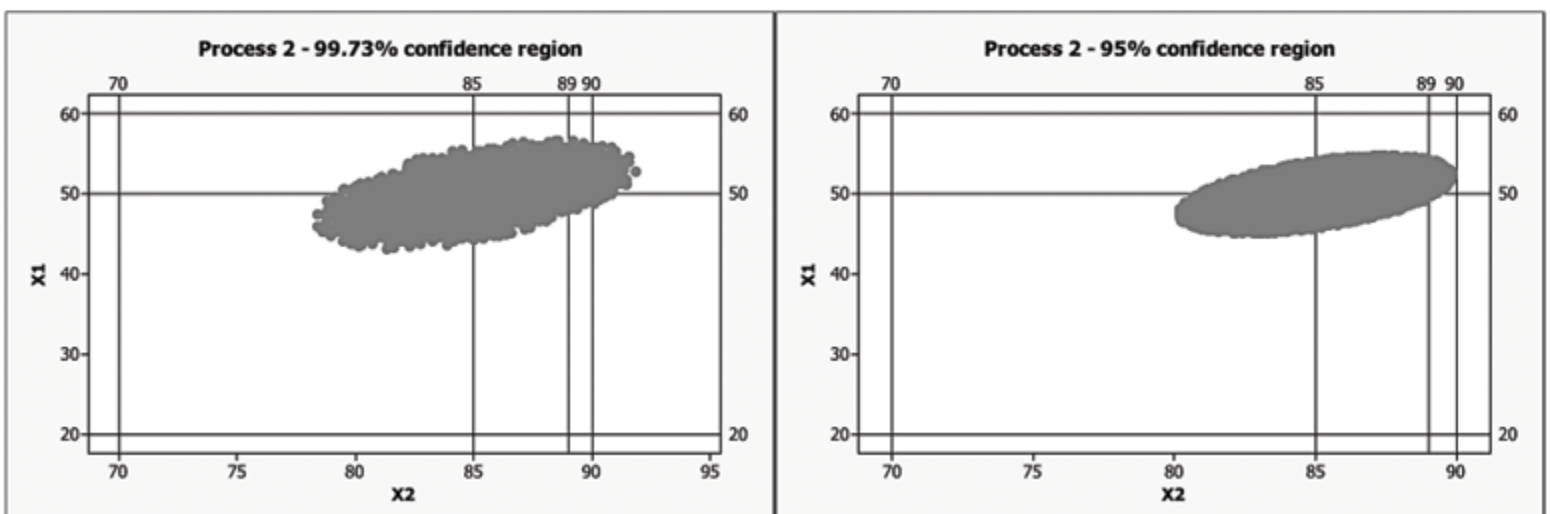
Σ = d×d covariance matrix

$|\Sigma|$ and Σ^{-1} are the determinant and inverse respectively

P(x) is larger for smaller exponents!

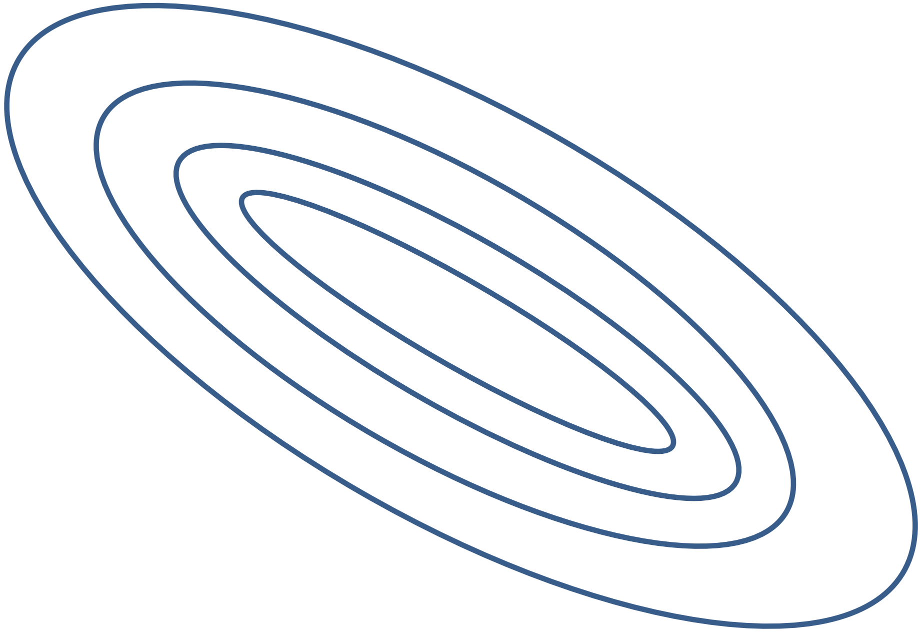
Confidence Intervals: Multi-Variate Case

- Same concept: how large is the area that contains X% of samples drawn from the distribution
- Confidence intervals are ellipsoids for normal distribution



Confidence Intervals: Multi-Variate Case

- Increasing X%, increases the size of the ellipsoids, but not their orientation and aspect ratio

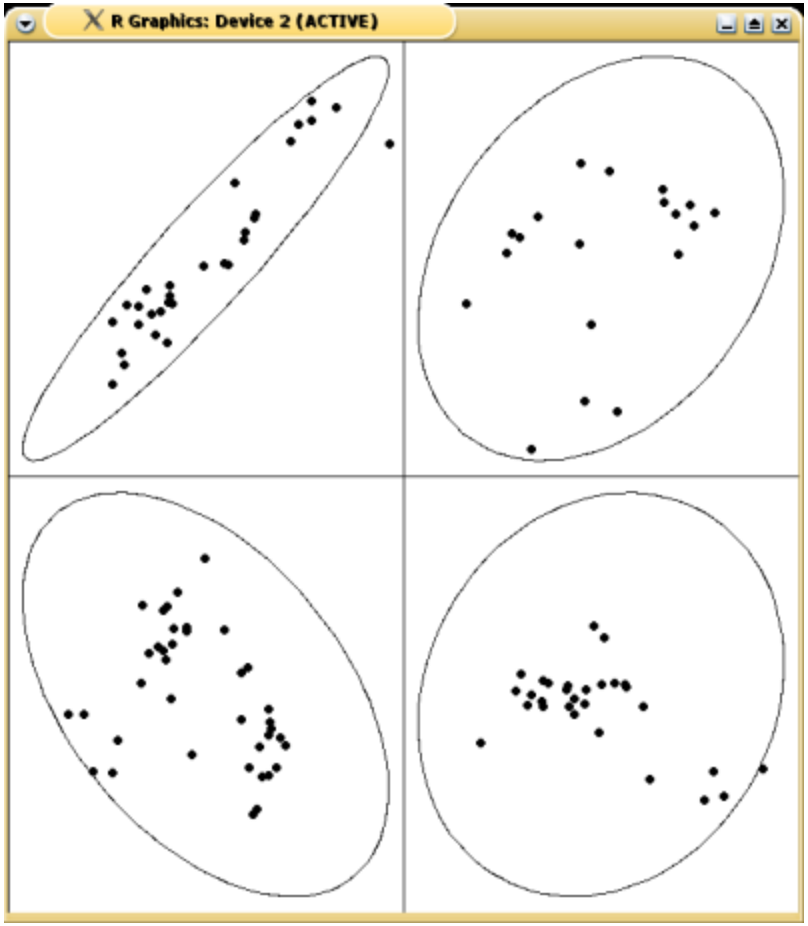


The Multi-Variate Normal Density

- Σ is positive semi definite (**$x^t \Sigma x >= 0$**)
  - If **$x^t \Sigma x = 0$ for non-zero x then det(Σ)=0.** This case is not interesting, **p(x) is not defined**
    - Two or more parameters are linearly dependent
- So we will assume Σ is positive definite (**$x^t \Sigma x > 0$**)
- If Σ is positive definite then so is Σ^{-1}

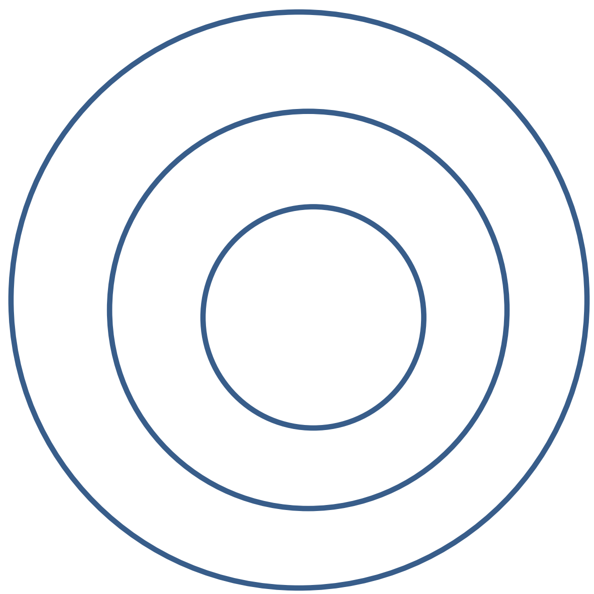
Confidence Intervals: Multi-Variate Case

- Covariance matrix determines the shape



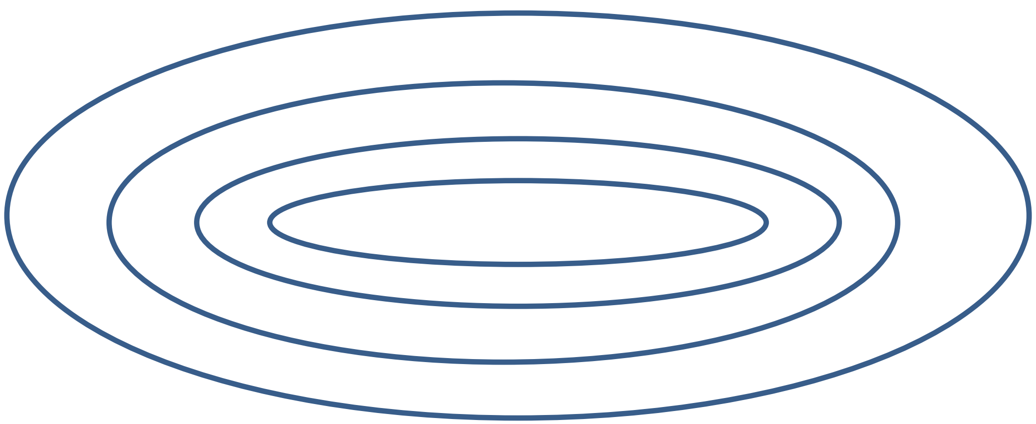
Confidence Intervals: Multi-Variate Case

- Case I: $\Sigma = \sigma^2 I$
  - All variables are uncorrelated and have equal variance
- Confidence intervals are circles



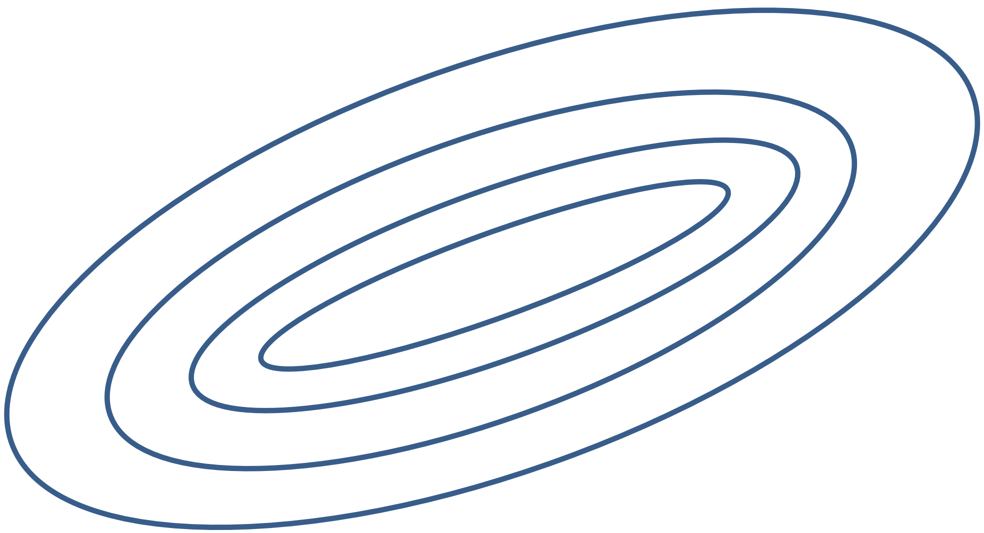
Confidence Intervals: Multi-Variate Case

- Case II: Σ diagonal, with unequal elements
  - All variables are uncorrelated but have different variances
- Confidence intervals are axis-aligned ellipsoids



Confidence Intervals: Multi-Variate Case

- Case III: Σ arbitrary
  - Variables may be correlated and have different variances
- Confidence intervals are arbitrary ellipsoids



Intro to SLAM

Visual SLAM

Introduction

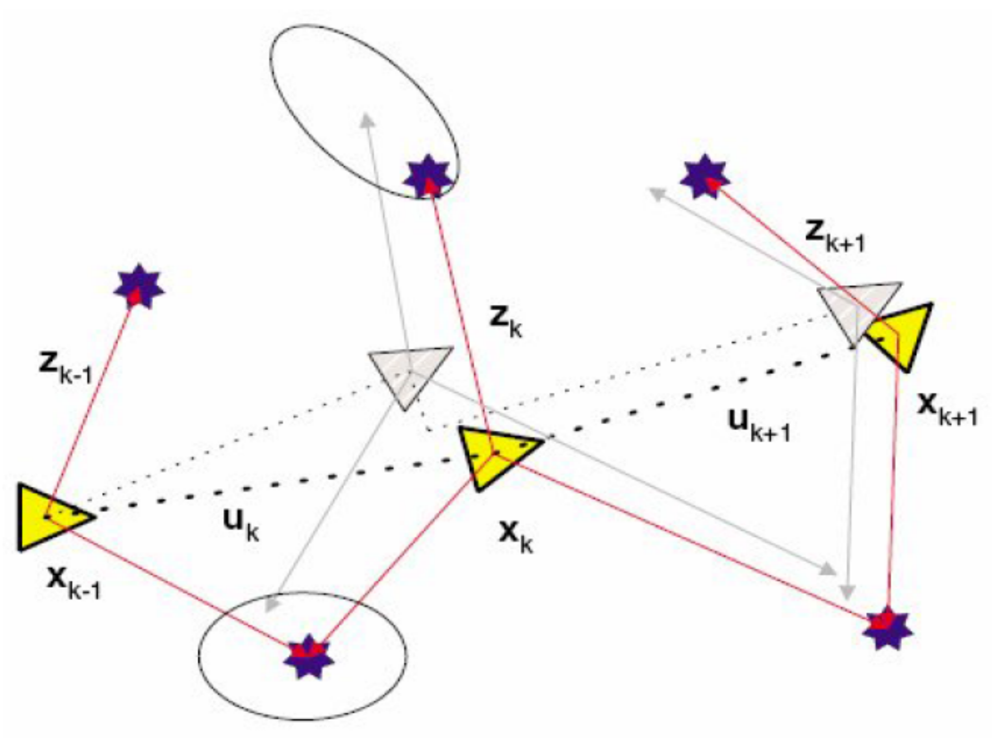
SLAM Objective

- Place a robot in an unknown location in an unknown environment and have the robot incrementally build a map of this environment while simultaneously using this map to compute vehicle location
- A solution to SLAM was seen as the “Holy Grail”
  - Would enable robots to operate in an environment without a priori knowledge of obstacle locations
- A little more than 10 years ago it was shown that a solution is possible!

The Localization Problem

- A map m of landmark locations is known a priori
- Take measurements of landmark location z_k (i.e. distance and bearing)
- Determine vehicle location x_k based on z_k
  - Need filter if sensor is noisy

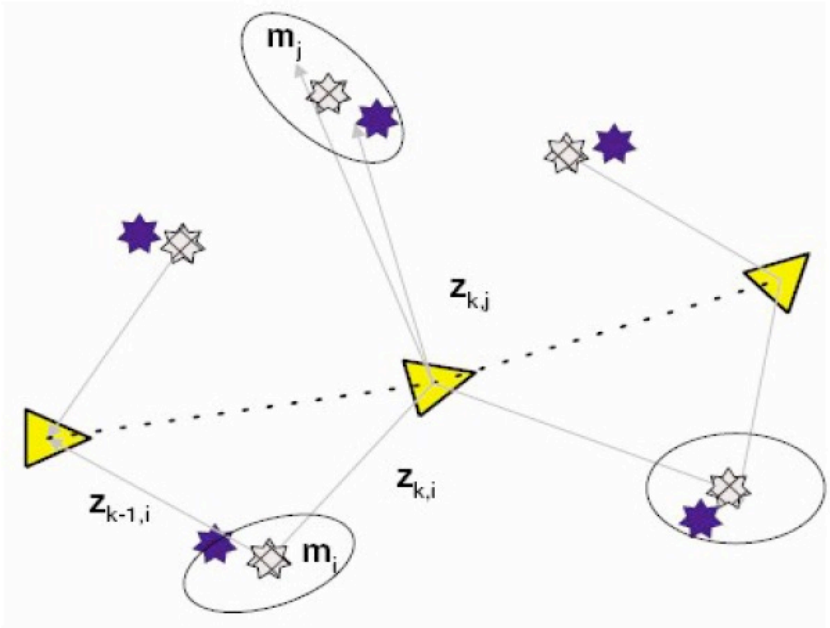
- x_k: location of vehicle at time k
- u_k: a control vector applied at k-1 to drive the vehicle from x_{k-1} to x_k
- z_k: observation of a landmark taken at time k
- X^k: history of states $\{x_1, x_2, x_3, \ldots, x_k\}$
- U^k: history of control inputs $\{u_1, u_2, u_3, \ldots, u_k\}$
- m: set of all landmarks



The Mapping Problem

- The vehicle locations X^k are provided
- Take measurements of landmark location z_k (i.e. distance and bearing)
- Build map m based on z_k
  - Need filter if sensor is noisy

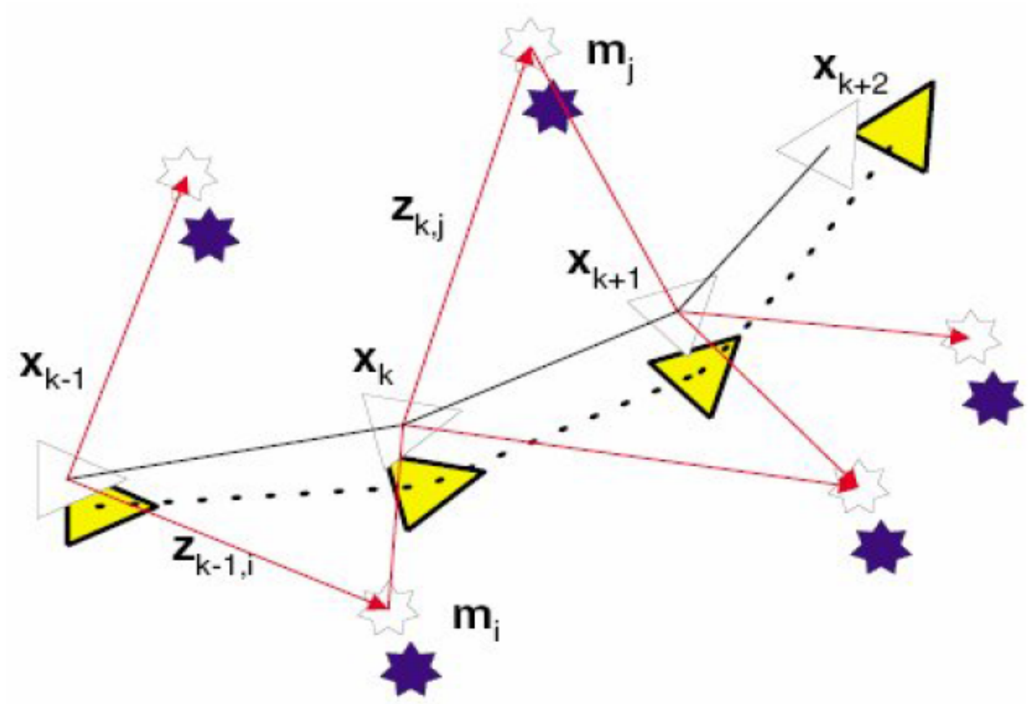
- X^k: history of states $\{x_1, x_2, x_3, \ldots, x_k\}$
- z_k: observation of a landmark taken at time k
- m_i: true location of i^{th} landmark
- m: set of all landmarks



Simultaneous Localization and Mapping

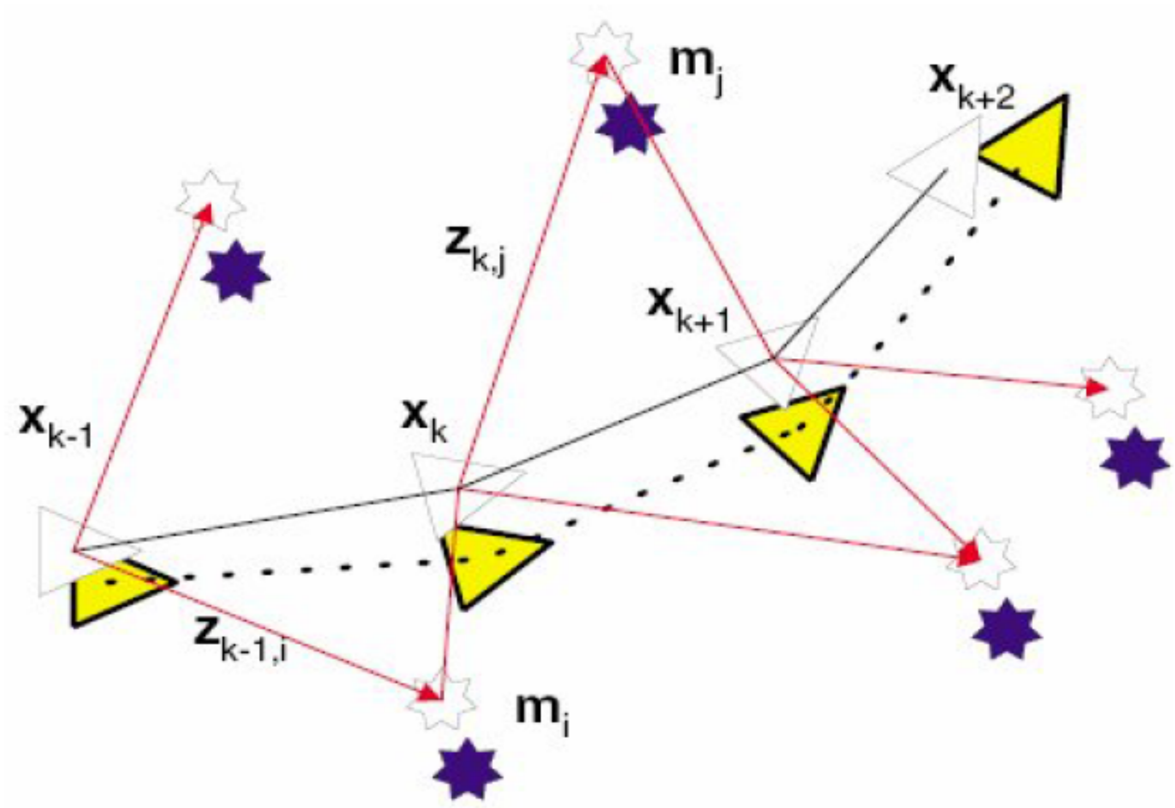
- From knowledge of observations Z^k
- Determine vehicle location X^k
- Build map m of landmark locations

- x_k: location of vehicle at time k
- u_k: a control vector applied at k-1 to drive the vehicle from x_{k-1} to x_k
- m_i: true location of i^{th} landmark
- z_k: observation of a landmark taken at time k
- X^k: history of states $\{x_1, x_2, x_3, \dots, x_k\}$
- U^k: history of control inputs $\{u_1, u_2, u_3, \dots, u_k\}$
- m: set of all landmarks
- Z^k: history of all observations $\{z_1, z_2, \dots, z_k\}$



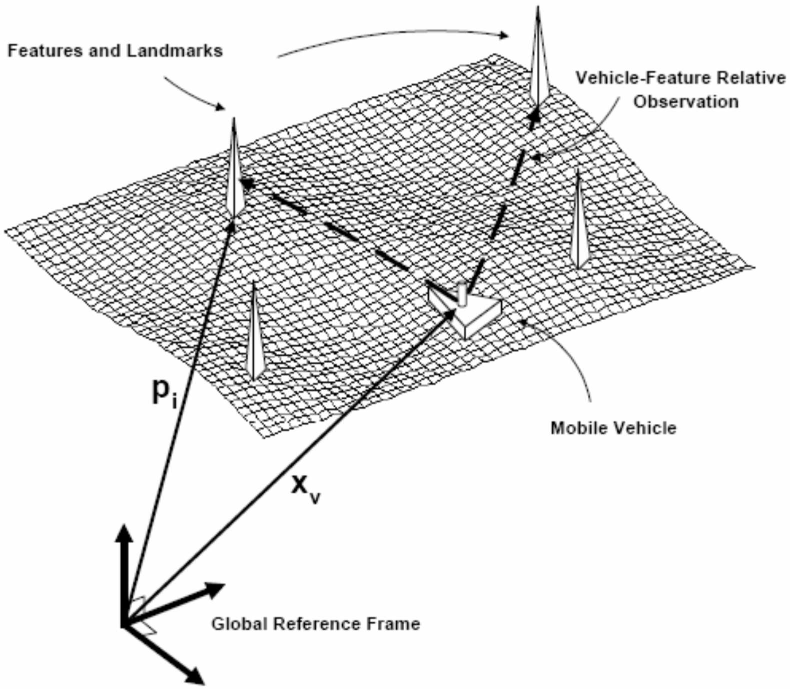
Simultaneous Localization and Mapping

- Localization and mapping are coupled problems
- A solution can only be obtained if the localization and mapping processes are considered together



SLAM Fundamentals

- A vehicle with a known kinematic model moving through an environment containing a population of landmarks (process model)
- The vehicle is equipped with a sensor that can take measurements of the relative location between any individual landmark and the vehicle itself (observation model)



Process Model

- For better understanding, a linear model of the vehicle is assumed
- If the state of the vehicle is given as $x_v(k)$ then the vehicle model is

$$x_v(k+1) = F_v(k)x_v(k) + u_v(k+1) + w_v(k+1)$$

- where
  - $F_v(k)$ is the state transition matrix
  - $u_v(k)$ is a vector of control inputs
  - $w_v(k)$ is a vector of uncorrelated process noise errors with zero mean and covariance $Q_v(k)$
- The state transition equation for the i^{th} landmark is

$$p_i(k+1) = p_i(k) = p_i$$

- SLAM considers all landmarks stationary

Process Model

- The augmented state vector containing both the state of the vehicle and the state of all landmark locations is

$$x(k) = \begin{bmatrix} x_v^T(k) & p_1^T & \dots & p_N^T \end{bmatrix}^T$$

Observation Model

- Assuming the observation to be linear, the observation model for the i^{th} landmark is given as

$$z(k) = H_i x(k) + v_i(k)$$

- where
  - $v_i(k)$ is a vector of uncorrelated observation errors with zero mean and variance $R_i(k)$
  - H_i is the observation matrix that relates the sensor output $z_i(k)$ to the state vector $x(k)$ when observing the i^{th} landmark and is written as

$$H_i = [-H_v, 0 \ldots 0, H_{pi}, 0 \ldots 0]$$

- Re-expressing the observation model

$$z(k) = H_{pi} p - H_v x_v(k) + v_i(k)$$

Estimation Process

- Objective
  - The state of the discrete-time process x_k needs to be estimated based on the measurement z_k
  - This is the exact definition of the Kalman filter
- Kalman Filter
  - Recursively computes estimates of state x(k) which is evolving according to the process and observation models
  - The filter proceeds in three stages
    - Prediction
    - Observation
    - Update

Estimation Process

Prediction

- After initializing the filter (i.e. setting values for $\hat{x}$(k) and P(k)), a prediction is generated for
  - The a priori state estimate

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + u(k)$$

  - The a priori observation relative to the i^th landmark

$$\hat{z}_i(k+1|k) = H_i(k)\hat{x}(k+1|k)$$

  - The a priori state covariance (e.g. a measure of how uncertain the states computed by the process model are)

$$P(k+1|k) = F(k)P(k|k)F^T(k) + Q(k)$$

Estimation Process

Observation

- Following the prediction, an observation $z_i(k+1)$ of the i^{th} landmark is made using the observation model
- An innovation and innovation covariance matrix are calculated
  - Innovation is the discrepancy between the actual measurement z_k and the predicted measurement $\hat{z}(k)$

$$v_i(k+1) = z_i(k+1) - \hat{z}_i(k+1 \mid k)$$

$$S_i(k+1) = H_i(k) P(k+1 \mid k) H_i^T(k) + R_i(k+1)$$

Estimation Process

Update

- The state estimate and corresponding state estimate covariance are then updated according to

$$\hat{x}(k+1 \mid k+1) = \hat{x}(k+1 \mid k) + W_i(k+1)\, v_i(k+1)$$

$$P(k+1 \mid k+1) = P(k+1 \mid k) - W_i(k+1) S(k+1) W_i^T(k+1)$$

- where the gain matrix $W_i(k+1)$ is given by

$$W_i(k+1) = P(k+1 \mid k) H_i^T(k) S_i^{-1}(k+1)$$

Kalman Filter

- Developed by Rudolph E. Kalman in 1960
- A set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process
- It supports estimations of
  - Past states
  - Present states
  - Future states
- and can do so when the nature of the modeled system is unknown!

Kalman Filter Properties

- Given all measurements up to current time, the Kalman filter algorithm is the optimal Minimum Mean Squared Error (MMSE) estimator of the state
- Provided that:
  - initial state is Gaussian with known mean and covariance;
  - process and observations models are linear;
  - and noise terms are uncorrelated, white, Gaussian, zero mean and with known covariances.

Discrete Kalman Filter

Process Model

- Assumes true state at time k evolves from state (k-1) according to

$$x(k) = F\,x(k-1) + G\,u(k-1) + w(k)$$

- where
  - F is the state transition model (A matrix)
  - G is the control input matrix (B matrix)
  - w(k) is the process noise which is assumed to be white and have a normal probability distribution

$$p(w) \sim N(0,Q)$$

Discrete Kalman Filter

Observation Model

- At time k, a measurement z(k) of the true state x(k) is made according to

$$z(k) = H\, x(k) + v(k)$$

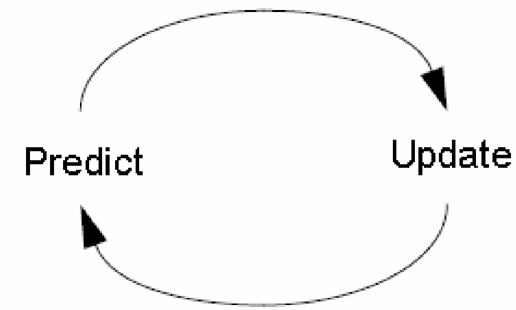
- where
  - H is the observation matrix and relates the measurement z(k) to the state vector x(k)
  - v(k) is the observation noise which is assumed to be white and have a normal probability distribution

$$p(w) \sim N(0,R)$$

Discrete Kalman Filter

Algorithm

- Recursive
  - Only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state
- The state of the filter is represented by two variables
  - x(k): estimate of the state at time k
  - P(k|k): error covariance matrix (a measure of the estimated accuracy of the state estimate)
- The filter has two distinct stages
  - Predict (and observe)
  - Update



Discrete Kalman Filter (Notation 1)

Prediction

- Predicted state $$\boxed{\hat{x}(k \mid k-1)} = F(k)\hat{x}(k-1 \mid k-1) + B(k)u(k-1)$$
- Predicted covariance $$\boxed{P(k \mid k-1)} = F(k)P(k-1 \mid k-1)F(k)^T + Q(k)$$

Observation

- Innovation $$\tilde{y}(k) = z(k) - H(k)\hat{x}(k \mid k-1)$$
- Innovation covariance $$S(k) = H(k)P(k \mid k-1)H(k)^T + R(k)$$

Not the same variable!!

Update

- Optimal Kalman gain $$K(k) = P(k \mid k-1)H(k)^T S(k)^{-1}$$
- Updated state $$\boxed{\hat{x}(k \mid k)} = \hat{x}(k \mid k-1) + K(k)\tilde{y}(k)$$
- Updated covariance $$\boxed{P(k \mid k)} = (I - K(k)H(k))P(k \mid k-1)$$

Not the same variable!!

Discrete Kalman Filter (Notation 2)

Prediction

- Predicted state $\hat{x}(k)^{-} = F(k)\hat{x}(k-1) + Bu(k-1)$
- Predicted estimate covariance $P(k)^{-} = FP(k-1)F^{T} + Q$

Observation

- Innovation $\tilde{y}(k) = z(k) - H\hat{x}(k)^{-}$
- Innovation covariance $S(k) = HP(k)^{-}H^{T} + R$

Update

- Optimal Kalman gain $K(k) = P(k)^{-}HS(k)^{-1}$
- Updated state estimate $\hat{x}(k) = \hat{x}(k)^{-} + K(k)\tilde{y}(k)$
- Updated estimate covariance $P(k) = (I - K(k)H)P(k)^{-}$

Kalman Filter (Alternate Notation)

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}, \qquad z_k = Hx_k + v_k.$$

Time Update ("Predict")

(1) Project the state ahead

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

(2) Project the error covariance ahead

$$P_k^- = AP_{k-1}A^T + Q$$

Measurement Update ("Correct")

(1) Compute the Kalman gain

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

(2) Update estimate with measurement z_k

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

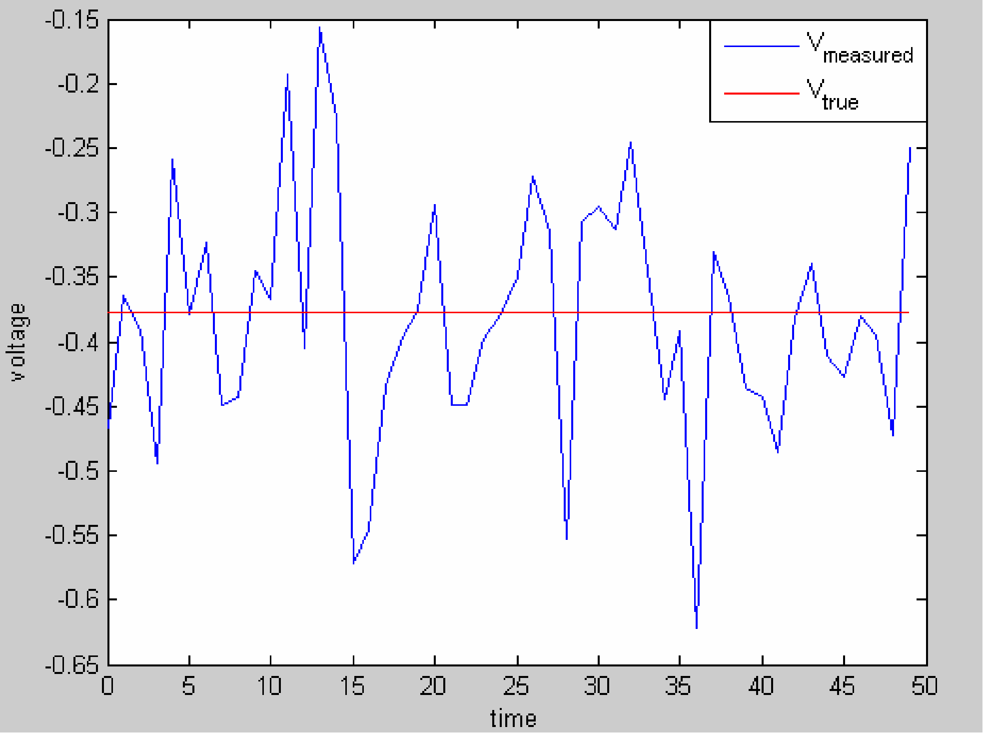
(3) Update the error covariance

$$P_k = (I - K_k H)P_k^-$$

Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Kalman Filter Example

- Estimate a scalar random constant (e.g. voltage)
- - Measurements are corrupted by 0.1 volt RMS white noise



Kalman Filter Example

Process Model

- Governed by the linear difference equation

$$x(k) = Fx(k-1) + Gu(k-1) + w(k)$$

$$x(k) = x(k-1) + w(k) \longrightarrow$$

State doesn’t change (F=1)
No control input (u=0)

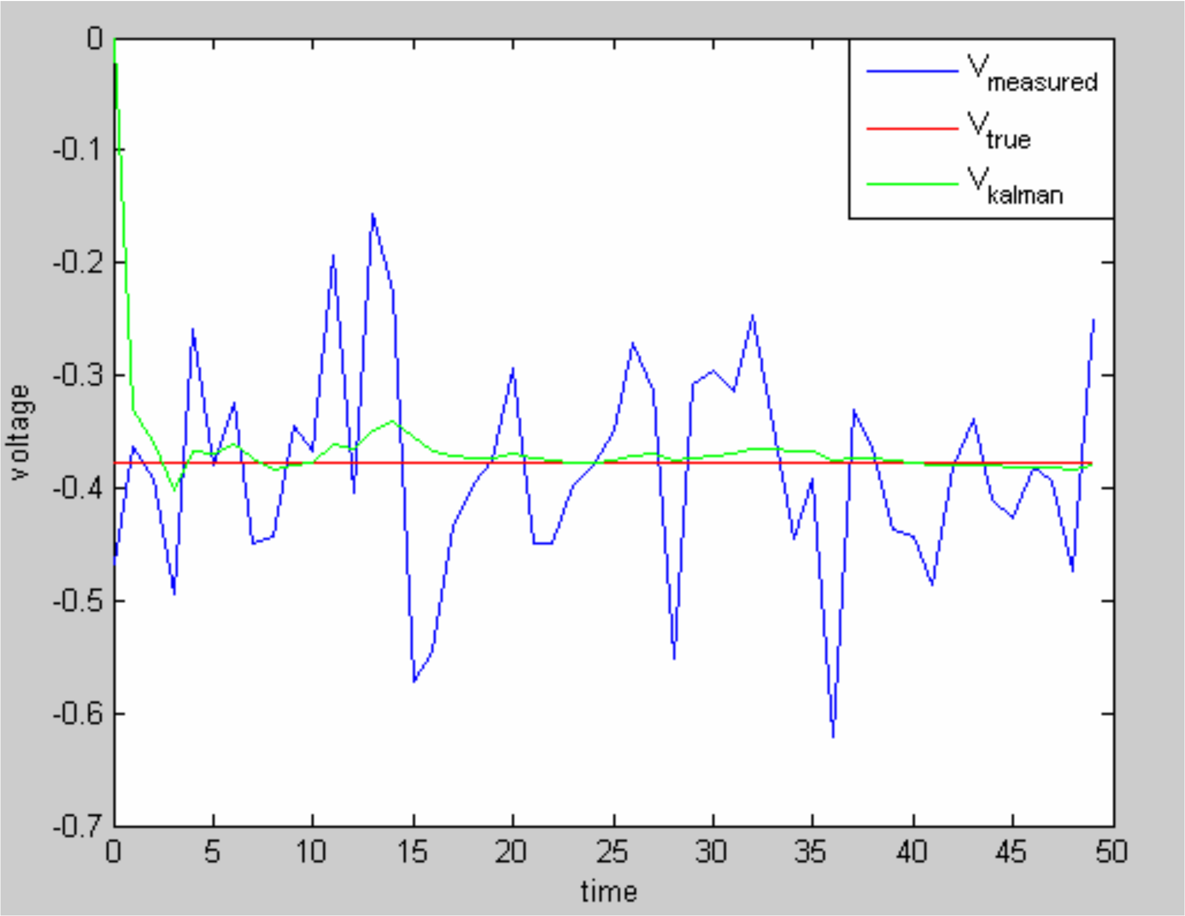
- with a measurement

$$z(k) = Hx(k) + v(k)$$

$$z(k) = x(k) + v(k) \longrightarrow$$

Measurement is of state directly (H=1)

Kalman Filter Example



Extended Kalman Filter

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}), \qquad z_k = h(x_k, v_k),$$

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + W w_{k-1},$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + V v_k.$$

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0), \qquad W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0),$$

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\tilde{x}_k, 0), \qquad V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\tilde{x}_k, 0).$$

Extended Kalman Filter

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}), \qquad z_k = h(x_k, v_k),$$

Time Update ("Predict")

(1) Project the state ahead

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0)$$

(2) Project the error covariance ahead

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$$

Measurement Update ("Correct")

(1) Compute the Kalman gain

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1}$$

(2) Update estimate with measurement z_k

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0))$$

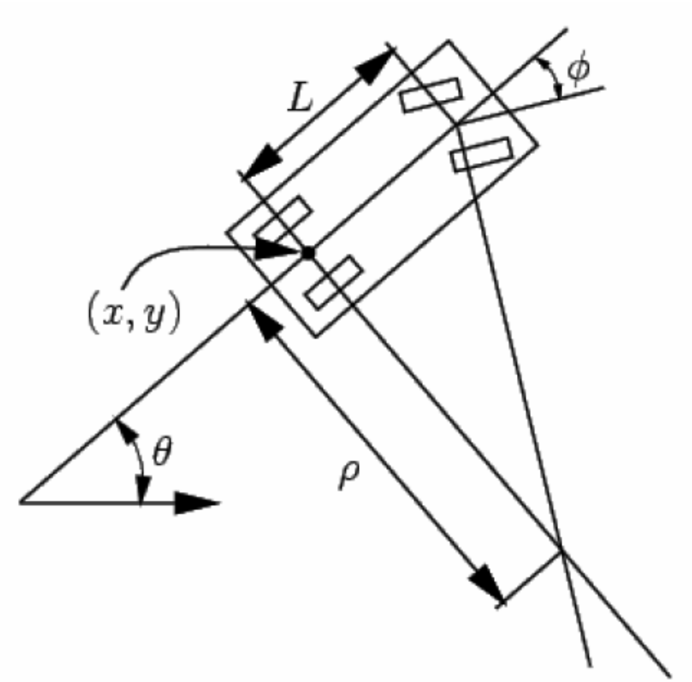
(3) Update the error covariance

$$P_k = (I - K_k H_k) P_k^-$$

Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Motion Tracking Example

Simple Robot Model



Kinematic Equations

$$\dot{x} = V\cos\theta$$
$$\dot{y} = V\sin\theta$$
$$\dot{\theta} = \frac{V\tan\phi}{L}$$

Non-linear!

Simple Robot Model

Kinematic Equations

$$\dot{x} = V\cos\theta$$

$$\dot{y} = V\sin\theta$$

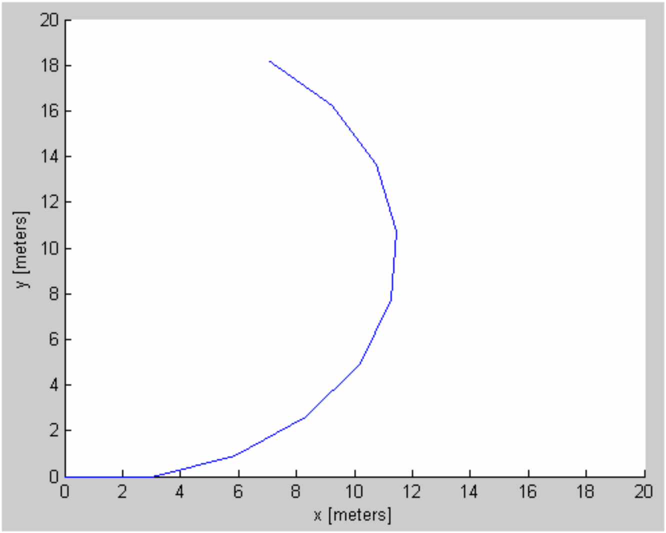
$$\dot{\theta} = \frac{V\tan\phi}{L}$$

f(x,u,w)

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + \Delta t V(k)\cos\theta(k) \\ y(k) + \Delta t V(k)\sin\theta(k) \\ \theta(k) + \dfrac{\Delta t V(k)\tan\phi(k)}{L} \end{bmatrix}$$

Assumptions

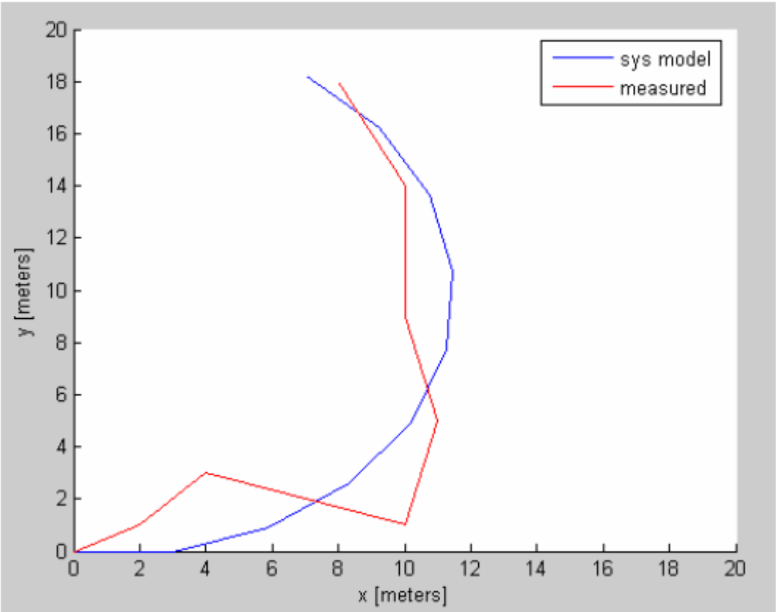
- System inputs
  - Velocity (assumed constant, vel=3)
  - Steering angle (ϕ)
- Δt is fixed and equal to 1
- L=1
- 10 iterations (N=10)

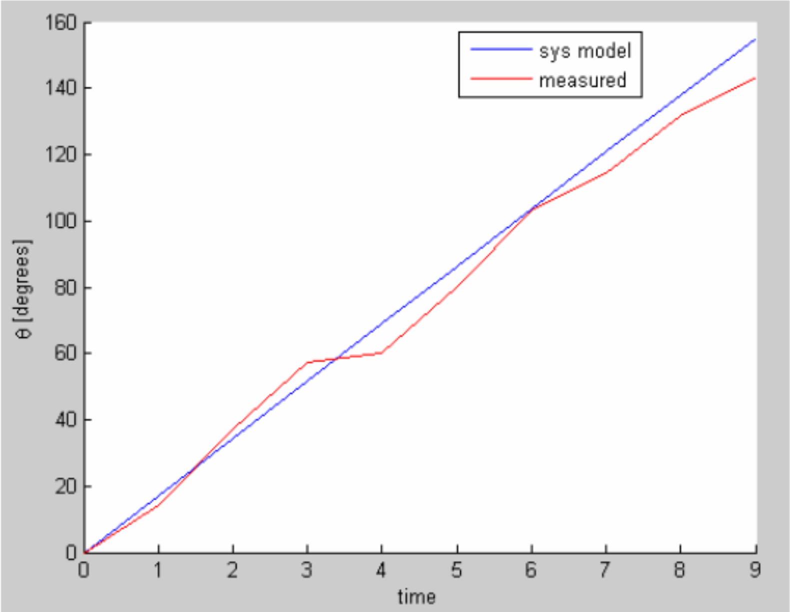


Observation Model

Measurements are taken from an overhead camera, and thus x, y, and θ can be measured directly

$$z(k) = h\big(x(k), v(k)\big) \quad \Rightarrow \quad z(k) = \begin{bmatrix} x(k) + v_x \\ y(k) + v_y \\ \theta(k) + v_\theta \end{bmatrix}$$





EKF

Prediction

$$\hat{x}(k)^- = f\left(\hat{x}(k-1), u(k-1), 0\right)$$ from robot model

$$P(k)^- = \underline{F(k)}P(k-1)\underline{F(k)}^T + \underline{W(k)}Q(k-1)\underline{W(k)}^T$$

$$x(k+1) = f\left(x(k), u(k), w(k)\right) = \begin{bmatrix} x(k) + \Delta t V(k)\cos\theta(k) \\ y(k) + \Delta t V(k)\sin\theta(k) \\ \theta(k) + \dfrac{\Delta t V(k)\tan\phi(k)}{L} \end{bmatrix}$$

Need to calculate Jacobians!

$$F(k) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} & \dfrac{\partial f_1}{\partial \theta} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} & \dfrac{\partial f_2}{\partial \theta} \\ \dfrac{\partial f_3}{\partial x} & \dfrac{\partial f_3}{\partial y} & \dfrac{\partial f_3}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -V\sin\theta \\ 0 & 1 & V\cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$W(k) = \begin{bmatrix} \dfrac{\partial f_1}{\partial w_x} & \dfrac{\partial f_1}{\partial w_y} & \dfrac{\partial f_1}{\partial w_\theta} \\ \dfrac{\partial f_2}{\partial w_x} & \dfrac{\partial f_2}{\partial w_y} & \dfrac{\partial f_2}{\partial w_\theta} \\ \dfrac{\partial f_3}{\partial w_x} & \dfrac{\partial f_3}{\partial w_y} & \dfrac{\partial f_3}{\partial w_\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

EKF

Kalman Gain

$$K(k) = P(k)^{-} \underline{J_h(k)^T} \left(\underline{J_h(k)} P(k)^{-} \underline{J_h(k)^T} + \underline{V(k)} R(k) \underline{V(k)^T} \right)^{-1}$$

$$z(k) = h\big(x(k), v(k)\big) = \begin{bmatrix} x(k) + v_x \\ y(k) + v_y \\ \theta(k) + v_\theta \end{bmatrix}$$

Need to calculate Jacobians!

$$J_h(k) = \begin{bmatrix} \frac{\partial h_1}{\partial x} & \frac{\partial h_1}{\partial y} & \frac{\partial h_1}{\partial \theta} \\ \frac{\partial h_2}{\partial x} & \frac{\partial h_2}{\partial y} & \frac{\partial h_2}{\partial \theta} \\ \frac{\partial h_3}{\partial x} & \frac{\partial h_3}{\partial y} & \frac{\partial h_3}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

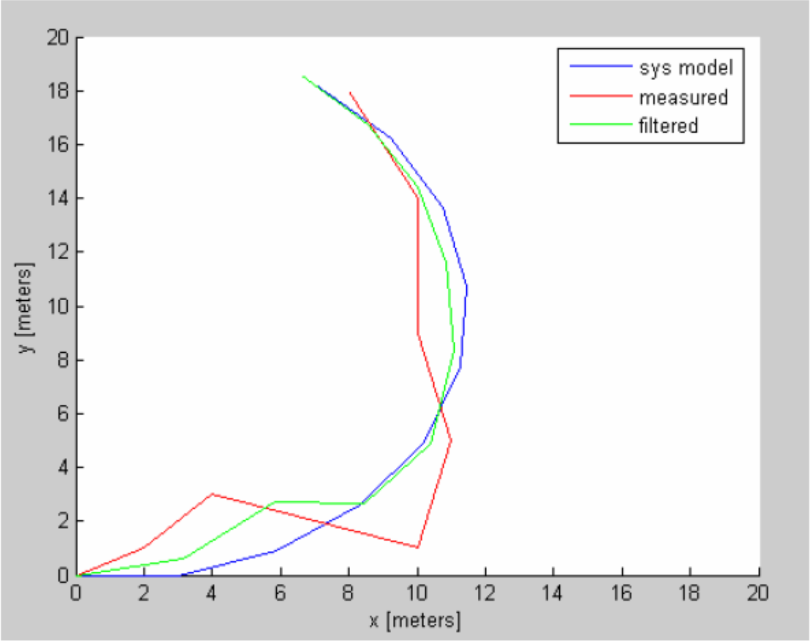
$$V(k) = \begin{bmatrix} \frac{\partial h_1}{\partial v_x} & \frac{\partial h_1}{\partial v_y} & \frac{\partial h_1}{\partial v_\theta} \\ \frac{\partial h_2}{\partial v_x} & \frac{\partial h_2}{\partial v_y} & \frac{\partial h_2}{\partial v_\theta} \\ \frac{\partial h_3}{\partial v_x} & \frac{\partial h_3}{\partial v_y} & \frac{\partial h_3}{\partial v_\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

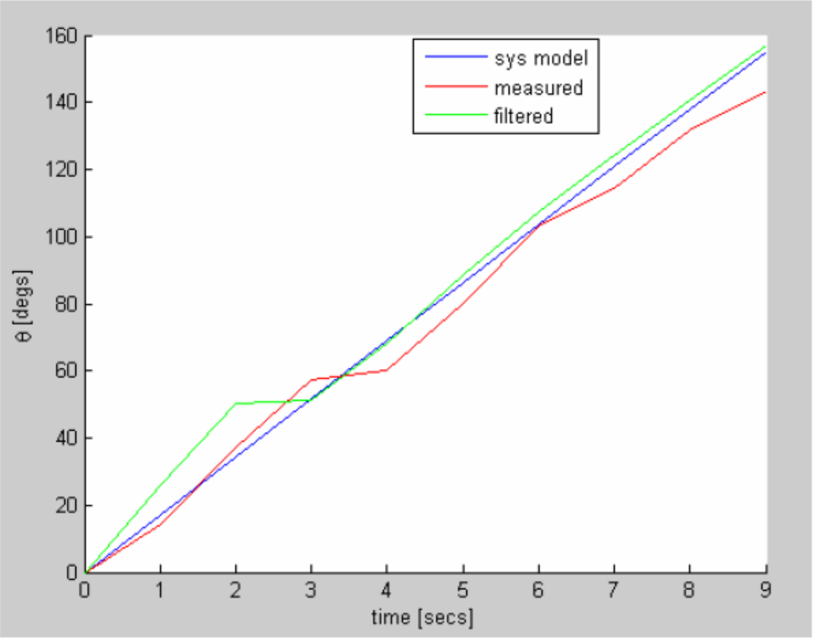
EKF

Measurement Update

$$\hat{x}(k) = \hat{x}(k)^{-} + K(k)\big(z(k) - H\big)$$

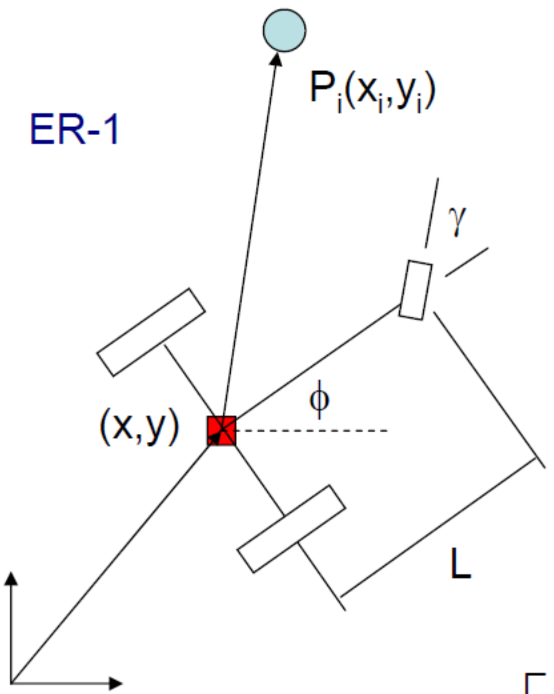
$$P(k) = \big(I - K(k)J_h(k)\big)P(k)^{-}$$





SLAM Example - Single Landmark

Robot Process Model



■ Radar Location

Kinematic Equations

$$\left.\begin{aligned}\dot{x} &= V\cos\varphi \\ \dot{y} &= V\sin\varphi \\ \dot{\varphi} &= \frac{V\tan\gamma}{L}\end{aligned}\right\}$$

Non-linear!

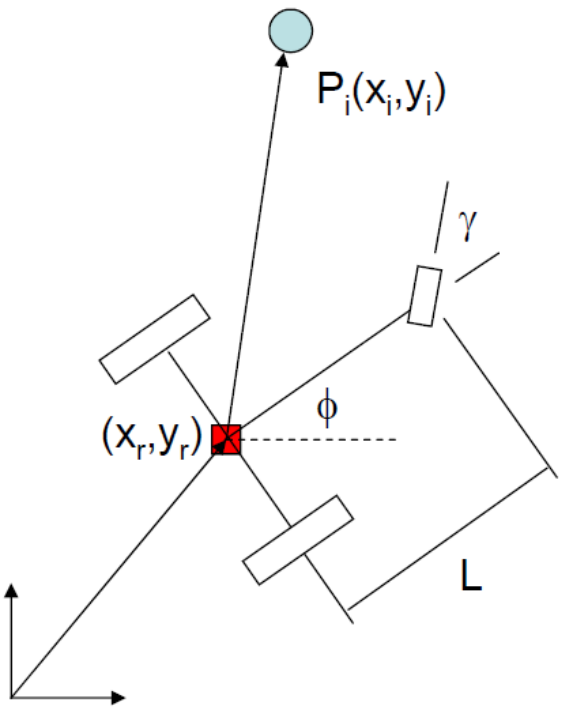
f(x,u,w)

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \varphi(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + \Delta t V(k)\cos\varphi(k) \\ y(k) + \Delta t V(k)\sin\varphi(k) \\ \varphi(k) + \dfrac{\Delta t V(k)\tan\gamma(k)}{L} \end{bmatrix} + w(k)$$

Objective

- Based on system inputs, V and γ (with sensor feedback, i.e. optical encoders) at time k, estimate the vehicle position at time (k+1)

Landmark Process Model



Radar Location

Recall that in the SLAM algorithm, landmarks are assumed to be stationary. Therefore,

$$p_i(k+1) = p_i(k)$$

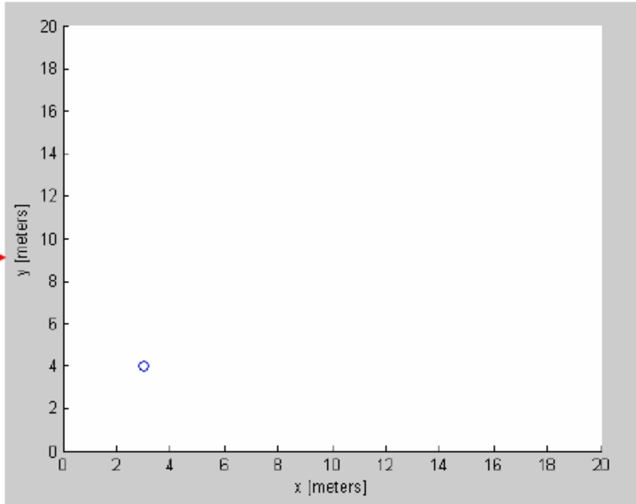
$$\begin{bmatrix} x_i(k+1) \\ y_i(k+1) \end{bmatrix} = \begin{bmatrix} x_i(k) \\ y_i(k) \end{bmatrix}$$

$$\begin{bmatrix} x_1(k+1) \\ y_1(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) \\ y_1(k) \end{bmatrix}$$

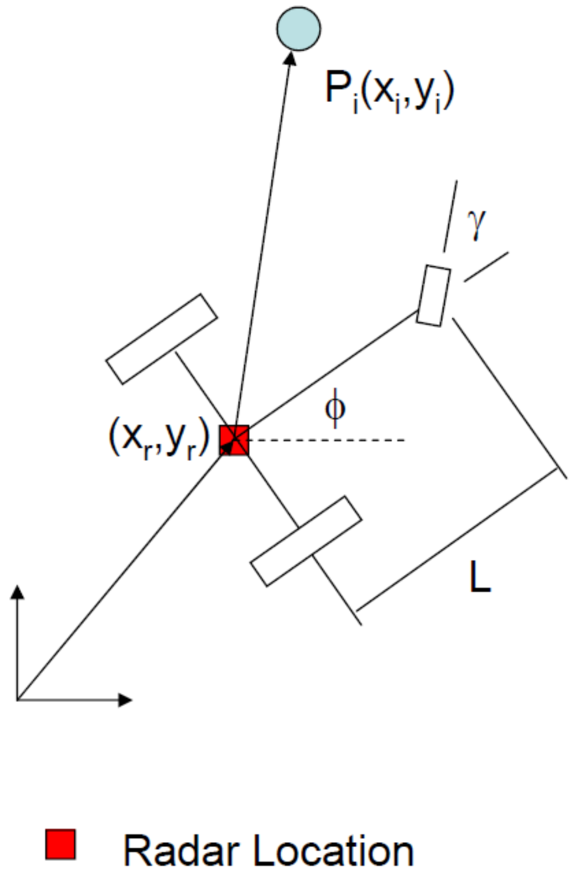
Overall System Process Model

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \varphi(k+1) \\ x_1(k+1) \\ y_1(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + \Delta t V(k)\cos\varphi(k) \\ y(k) + \Delta t V(k)\sin\varphi(k) \\ \varphi(k) + \dfrac{\Delta t V(k)\tan\gamma(k)}{L} \\ x_1(k) \\ y_1(k) \end{bmatrix} + \begin{bmatrix} w_x(k) \\ w_y(k) \\ w_\varphi(k) \\ 0 \\ 0 \end{bmatrix}$$

Landmark 1: (3,4)



Observation Model



$$z(k) = h\big(x(k), v(k)\big)$$

The radar used in the experiment returns the range $r_i(k)$ and bearing $\theta_i(k)$ to a landmark i. Thus, the observation model is

$$r_i(k) = \sqrt{\left(x_i - x_r(k)\right)^2 + \left(y_i - y_r(k)\right)^2} + v_r(k)$$

$$\theta_i(k) = \arctan\left(\frac{y_i - y_r(k)}{x_i - x_r(k)}\right) - \varphi(k) + v_\theta(k)$$

The Estimation Process

Prediction

$$\hat{x}(k)^- = f\left(\hat{x}(k-1), u(k-1), 0\right)$$

$$P(k)^- = F(k)P(k-1)F(k)^T + W(k)Q(k-1)W(k)^T$$

$$x(k+1) = \begin{bmatrix} x(k) + \Delta t V(k)\cos\varphi(k) \\ y(k) + \Delta t V(k)\sin\varphi(k) \\ \varphi(k) + \dfrac{\Delta t V(k)\tan\gamma(k)}{L} \\ x_1(k) \\ y_1(k) \end{bmatrix} + \begin{bmatrix} w_x(k) \\ w_y(k) \\ w_\varphi(k) \\ 0 \\ 0 \end{bmatrix}$$

$$F(k) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} & \dfrac{\partial f_1}{\partial \varphi} & \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial y_1} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} & \dfrac{\partial f_2}{\partial \varphi} & \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial y_1} \\ \dfrac{\partial f_3}{\partial x} & \dfrac{\partial f_3}{\partial y} & \dfrac{\partial f_3}{\partial \varphi} & \dfrac{\partial f_3}{\partial x_1} & \dfrac{\partial f_3}{\partial y_1} \\ \dfrac{\partial f_4}{\partial x} & \dfrac{\partial f_4}{\partial y} & \dfrac{\partial f_4}{\partial \varphi} & \dfrac{\partial f_4}{\partial x_1} & \dfrac{\partial f_4}{\partial y_1} \\ \dfrac{\partial f_5}{\partial x} & \dfrac{\partial f_5}{\partial y} & \dfrac{\partial f_5}{\partial \varphi} & \dfrac{\partial f_5}{\partial x_1} & \dfrac{\partial f_5}{\partial y_1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta t V(k)\sin\varphi(k) & 0 & 0 \\ 0 & 1 & \Delta t V(k)\cos\varphi(k) & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The Estimation Process

Prediction

$$x(k+1)=\begin{bmatrix} x(k)+\Delta t V(k)\cos\varphi(k) \\ y(k)+\Delta t V(k)\sin\varphi(k) \\ \varphi(k)+\dfrac{\Delta t V(k)\tan\gamma(k)}{L} \\ x_1(k) \\ y_1(k) \end{bmatrix}+\begin{bmatrix} w_x(k) \\ w_y(k) \\ w_\varphi(k) \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{x}(k)^- = f\left(\hat{x}(k-1), u(k-1), 0\right)$$

$$P(k)^- = F(k)P(k-1)F(k)^T + W(k)Q(k-1)W(k)^T$$

$$W(k)=\begin{bmatrix}
\frac{\partial f_1}{\partial w_x} & \frac{\partial f_1}{\partial w_y} & \frac{\partial f_1}{\partial w_\varphi} & \frac{\partial f_1}{\partial w_{x_1}} & \frac{\partial f_1}{\partial w_{y_1}} \\
\frac{\partial f_2}{\partial w_x} & \frac{\partial f_2}{\partial w_y} & \frac{\partial f_2}{\partial w_\varphi} & \frac{\partial f_1}{\partial w_{x_1}} & \frac{\partial f_1}{\partial w_{y_1}} \\
\frac{\partial f_3}{\partial w_x} & \frac{\partial f_3}{\partial w_y} & \frac{\partial f_3}{\partial w_\varphi} & \frac{\partial f_1}{\partial w_{x_1}} & \frac{\partial f_1}{\partial w_{y_1}} \\
\frac{\partial f_4}{\partial w_x} & \frac{\partial f_4}{\partial w_y} & \frac{\partial f_4}{\partial w_\varphi} & \frac{\partial f_4}{\partial w_{x_1}} & \frac{\partial f_4}{\partial w_{y_1}} \\
\frac{\partial f_5}{\partial w_x} & \frac{\partial f_5}{\partial w_y} & \frac{\partial f_5}{\partial w_\varphi} & \frac{\partial f_5}{\partial w_{x_1}} & \frac{\partial f_5}{\partial w_{y_1}}
\end{bmatrix}=\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

The Estimation Process

Kalman Gain

$$K(k) = P(k)^- J_h(k)^T \left(J_h(k) P(k)^- J_h(k)^T + V(k) R(k) V(k)^T \right)^{-1}$$

$$z(k) = \begin{bmatrix} r_i(k) \\ \theta_i(k) \end{bmatrix} = \begin{bmatrix} \sqrt{\left(x_i - \hat{x}(k)^-\right)^2 + \left(y_i - \hat{y}(k)^-\right)^2} \\ \tan^{-1}\left(\dfrac{y_i - \hat{y}(k)^-}{x_i - \hat{x}(k)^-} \right) - \hat{\varphi}(k)^- \end{bmatrix} + v(k)$$

$$J_h(k) = \begin{bmatrix} \dfrac{\partial h_1}{\partial x} & \dfrac{\partial h_1}{\partial y} & \dfrac{\partial h_1}{\partial \varphi} & \dfrac{\partial h_1}{\partial x_1} & \dfrac{\partial h_1}{\partial y_1} \\ \dfrac{\partial h_2}{\partial x} & \dfrac{\partial h_2}{\partial y} & \dfrac{\partial h_2}{\partial \varphi} & \dfrac{\partial h_2}{\partial x_1} & \dfrac{\partial h_2}{\partial y_1} \end{bmatrix} = \begin{bmatrix} \dfrac{x - x_i}{r} & \dfrac{y - y_i}{r} & 0 & \dfrac{x_i - x}{r} & \dfrac{y_i - y}{r} \\ \dfrac{y_i - y}{r^2} & \dfrac{x - x_i}{r^2} & -1 & \dfrac{y - y_i}{r^2} & \dfrac{x_i - x}{r^2} \end{bmatrix}$$

where $r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$

The Estimation Process

Kalman Gain

$$K(k) = P(k)^{-} J_h(k)^T \left(J_h(k) P(k)^{-} J_h(k)^T + V(k) R(k) V(k)^T \right)^{-1}$$

$$z(k) = \begin{bmatrix} r_i(k) \\ \theta_i(k) \end{bmatrix} = \begin{bmatrix} \sqrt{\left(x_i - \hat{x}(k)^{-}\right)^2 + \left(y_i - \hat{y}(k)^{-}\right)^2} \\ \tan^{-1}\left(\dfrac{y_i - \hat{y}(k)^{-}}{x_i - \hat{x}(k)^{-}} \right) - \hat{\varphi}(k)^{-} \end{bmatrix} + v(k)$$

$$V(k) = \begin{bmatrix} \dfrac{\partial h_1}{\partial v_r} & \dfrac{\partial h_1}{\partial v_\theta} \\ \dfrac{\partial h_2}{\partial v_r} & \dfrac{\partial h_2}{\partial v_\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The Estimation Process

Measurement Update

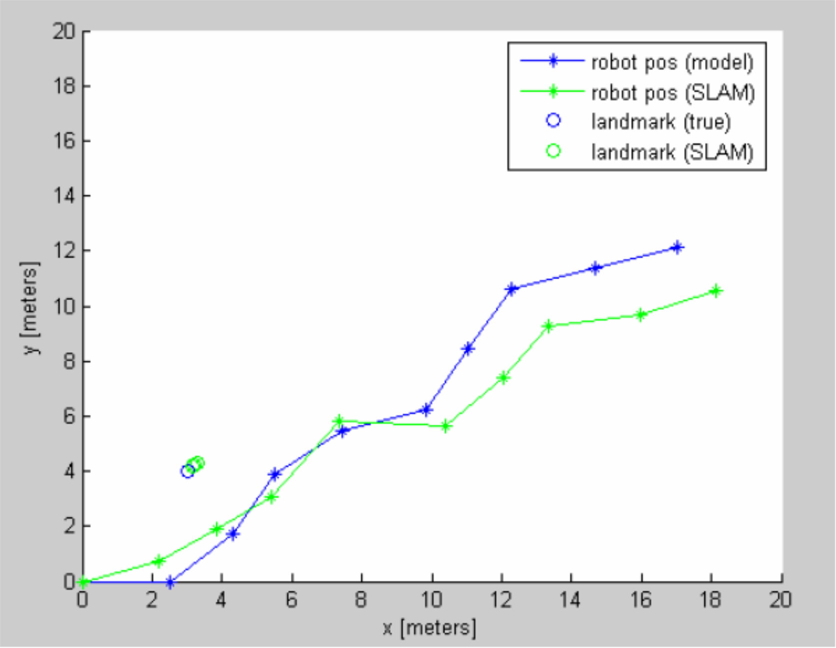
$$\hat{x}(k) = \hat{x}(k)^{-} + K(k)\big(\underline{z(k) - H(k)}\big)$$

$$P(k) = \big(I - K(k)J_h(k)\big)P(k)^{-}$$

Innovation

z(k) is 10 fabricated measurements of range and bearing to landmark 1.

There is only one landmark and it is incorporated into the model from the start.

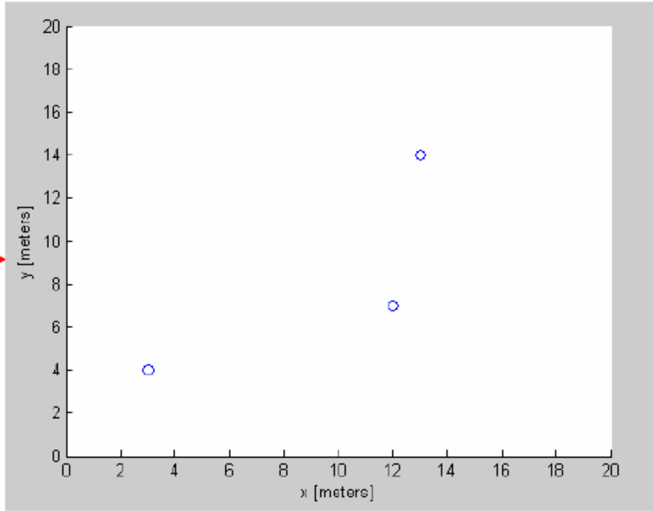


SLAM Example 2 - Multiple Landmarks

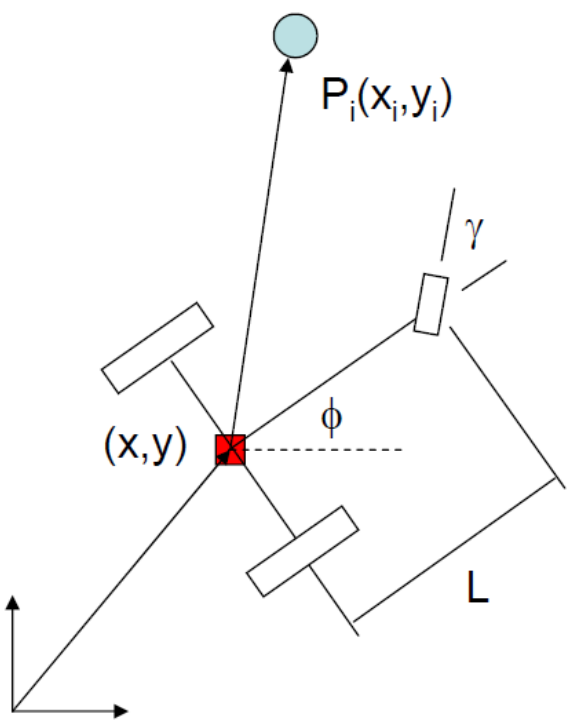
Overall System Process Model

$$
\begin{bmatrix} x(k+1) \\ y(k+1) \\ \varphi(k+1) \\ p_1(k+1) \\ \vdots \\ p_N(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + \Delta t V(k) \cos\varphi(k) \\ y(k) + \Delta t V(k) \sin\varphi(k) \\ \varphi(k) + \dfrac{\Delta t V(k) \tan\gamma(k)}{L} \\ p_1(k) \\ \vdots \\ p_N(k) \end{bmatrix} + \begin{bmatrix} w_x(k) \\ w_y(k) \\ w_\varphi(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix}
$$

Landmark 1: (3,4)
Landmark 2: (12,7)
Landmark 3: (13,14)



Observation Model



Radar Location

$$z(k) = h\big(x(k), v(k)\big)$$

The radar used in the experiment returns the range $r_i(k)$ and bearing $\theta_i(k)$ to a landmark i. Thus, the observation model is

$$r_i(k) = \sqrt{\big(x_i - x_r(k)\big)^2 + \big(y_i - y_r(k)\big)^2} + v_r(k)$$

$$\theta_i(k) = \arctan\left(\frac{y_i - y_r(k)}{x_i - x_r(k)}\right) - \varphi(k) + v_\theta(k)$$

The Estimation Process

Prediction

$$x(k+1)=\begin{bmatrix} x(k)+\Delta t V(k)\cos\varphi(k) \\ y(k)+\Delta t V(k)\sin\varphi(k) \\ \varphi(k)+\dfrac{\Delta t V(k)\tan\gamma(k)}{L} \end{bmatrix}+w(k)$$

$$\hat{x}(k)^{-} = f\left(\hat{x}(k-1), u(k-1), 0\right)$$

$$P(k)^{-} = F(k)P(k-1)F(k)^{T} + W(k)Q(k-1)W(k)^{T}$$

Initially, before landmarks are added

$$F(k)=\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \varphi} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \varphi} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \varphi} \end{bmatrix}=\begin{bmatrix} 1 & 0 & -\Delta t V(k)\sin\varphi(k) \\ 0 & 1 & \Delta t V(k)\cos\varphi(k) \\ 0 & 0 & 1 \end{bmatrix} \qquad W(k)=\begin{bmatrix} \frac{\partial f_1}{\partial w_x} & \frac{\partial f_1}{\partial w_y} & \frac{\partial f_1}{\partial w_\varphi} \\ \frac{\partial f_2}{\partial w_x} & \frac{\partial f_2}{\partial w_y} & \frac{\partial f_2}{\partial w_\varphi} \\ \frac{\partial f_3}{\partial w_x} & \frac{\partial f_3}{\partial w_y} & \frac{\partial f_3}{\partial w_\varphi} \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The Estimation Process

Kalman Gain

$$K(k) = P(k)^{-} J_h(k)^T \left(J_h(k) P(k)^{-} J_h(k)^T + V(k) R(k) V(k)^T \right)^{-1}$$

$$z(k) = \begin{bmatrix} r_i(k) \\ \theta_i(k) \end{bmatrix} = \begin{bmatrix} \sqrt{\left(x_i - \hat{x}(k)^{-}\right)^2 + \left(y_i - \hat{y}(k)^{-}\right)^2} \\ \tan^{-1}\left(\dfrac{y_i - \hat{y}(k)^{-}}{x_i - \hat{x}(k)^{-}} \right) - \hat{\varphi}(k)^{-} \end{bmatrix} + v(k)$$

Initially, before landmarks are added

$$J_h(k) = \begin{bmatrix} \dfrac{\partial h_1}{\partial x} & \dfrac{\partial h_1}{\partial y} & \dfrac{\partial h_1}{\partial \varphi} \\ \dfrac{\partial h_2}{\partial x} & \dfrac{\partial h_2}{\partial y} & \dfrac{\partial h_2}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} \dfrac{x - x_i}{r} & \dfrac{y - y_i}{r} & 0 \\ \dfrac{y_i - y}{r^2} & \dfrac{x - x_i}{r^2} & -1 \end{bmatrix} \qquad V(k) = \begin{bmatrix} \dfrac{\partial h_1}{\partial v_r} & \dfrac{\partial h_1}{\partial v_\theta} \\ \dfrac{\partial h_2}{\partial v_r} & \dfrac{\partial h_2}{\partial v_\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

where $r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$

The Estimation Process

Measurement Update

$$\hat{x}(k) = \hat{x}(k)^{-} + K(k)\big(z(k) - H(k)\big)$$

$$P(k) = \big(I - K(k)J_h(k)\big)P(k)^{-}$$

Now, if a landmark is observed at t(k+1), the state model is updated

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \varphi(k+1) \\ x_1(k+1) \\ y_1(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + \Delta t V(k)\cos\varphi(k) \\ y(k) + \Delta t V(k)\sin\varphi(k) \\ \varphi(k) + \dfrac{\Delta t V(k)\tan\gamma(k)}{L} \\ x_1(k) \\ y_1(k) \end{bmatrix} + \begin{bmatrix} w_x(k) \\ w_y(k) \\ w_\varphi(k) \\ 0 \\ 0 \end{bmatrix}$$

$$x_1(k+1) = x(k) + r\cos\theta \qquad y_1(k+1) = y(k) + r\sin\theta$$

The Estimation Process

Prediction (2)

$$\hat{x}(k)^- = f\left(\hat{x}(k-1), u(k-1), 0\right)$$

$$P(k)^- = F(k)P(k-1)F(k)^T + W(k)Q(k-1)W(k)^T$$

$$x(k+1) = \begin{bmatrix} x(k) + \Delta t V(k)\cos\varphi(k) \\ y(k) + \Delta t V(k)\sin\varphi(k) \\ \varphi(k) + \dfrac{\Delta t V(k)\tan\gamma(k)}{L} \\ x_1(k) \\ y_1(k) \end{bmatrix} + \begin{bmatrix} w_x(k) \\ w_y(k) \\ w_\varphi(k) \\ 0 \\ 0 \end{bmatrix}$$

$$F(k) = \begin{bmatrix} \dfrac{\partial f}{\partial(x, y, \varphi)} & 0 \\ 0 & I^{2N \, x \, 2N} \end{bmatrix}$$

where N is the number of landmarks

The Estimation Process

Kalman Gain (2)

$$K(k) = P(k)^{-} J_h(k)^T \left(J_h(k) P(k)^{-} J_h(k)^T + V(k) R(k) V(k)^T \right)^{-1}$$

If observing the 1st landmark

$$J_h(k) = \begin{bmatrix} \dfrac{\partial h}{\partial(x, y, \varphi)} & \dfrac{\partial h}{\partial(x_i, y_i)} & 0 & \ldots & 0 \end{bmatrix}$$

If observing the 2nd landmark

$$J_h(k) = \begin{bmatrix} \dfrac{\partial h}{\partial(x, y, \varphi)} & 0 & \dfrac{\partial h}{\partial(x_i, y_i)} & 0 & \ldots & 0 \end{bmatrix}$$

Must repeat for each landmark!!

The Estimation Process

Measurement Update (2)

$$\hat{x}(k) = \hat{x}(k)^{-} + K(k)\big(z(k) - H(k)\big)$$

$$P(k) = \big(I - K(k)J_{h}(k)\big)P(k)^{-}$$