



Simultaneous Localization and Mapping (SLAM)

Lecture 01

Introduction

SLAM Objective

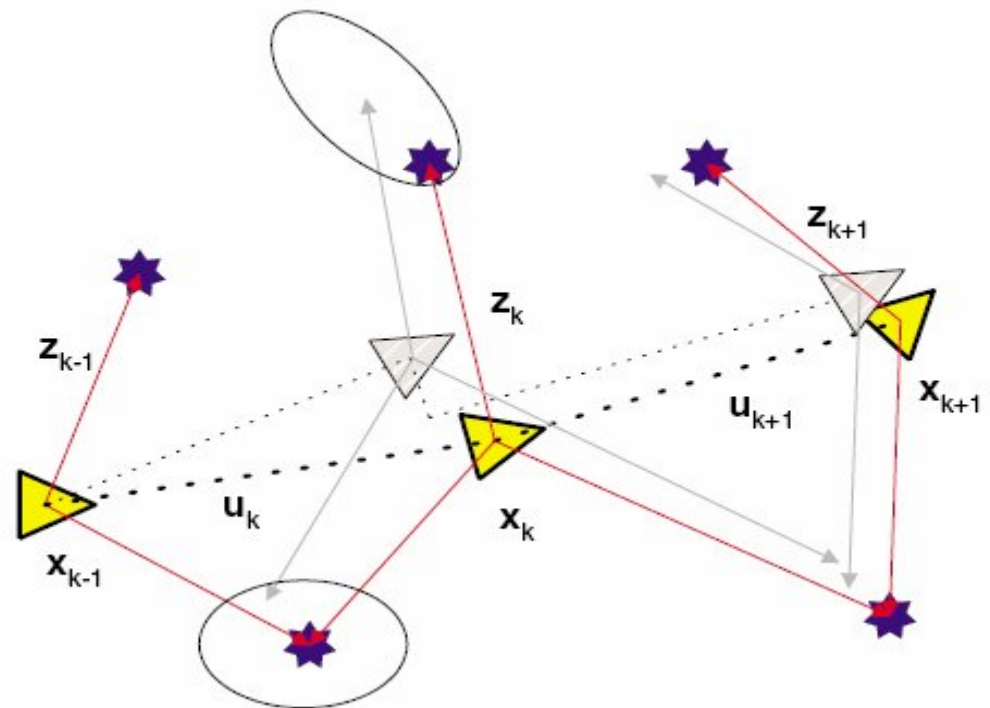
- Place a robot in an unknown location in an unknown environment and have the robot incrementally build a map of this environment while simultaneously using this map to compute vehicle location
- SLAM began with seminal paper by R. Smith, M. Self, and P. Cheeseman in 1990
- A solution to SLAM has been seen as the “Holy Grail”
 - Would enable robots to operate in an environment without **a priori** knowledge of obstacle locations
- Research over the last decade has shown that a solution is possible!!

The Localization Problem

Defined

- 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 $\{x_1, x_2, x_3, \dots, x_k\}$
- U^k : history of control inputs $\{u_1, u_2, u_3, \dots, u_k\}$
- m : set of all landmarks

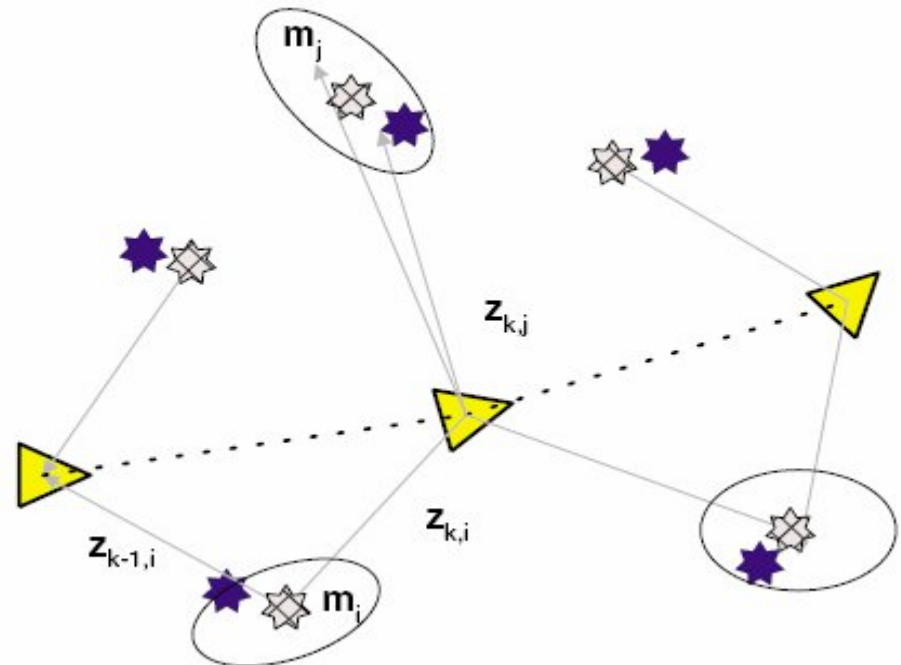


The Mapping Problem

Defined

- The vehicle locations X^k are provided
- Take measurement 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 $\{x_1, x_2, x_3, \dots, x_k\}$
- z_k : observation of a landmark taken at time k
- m_i : true location of the i^{th} landmark
- m : set of all landmarks

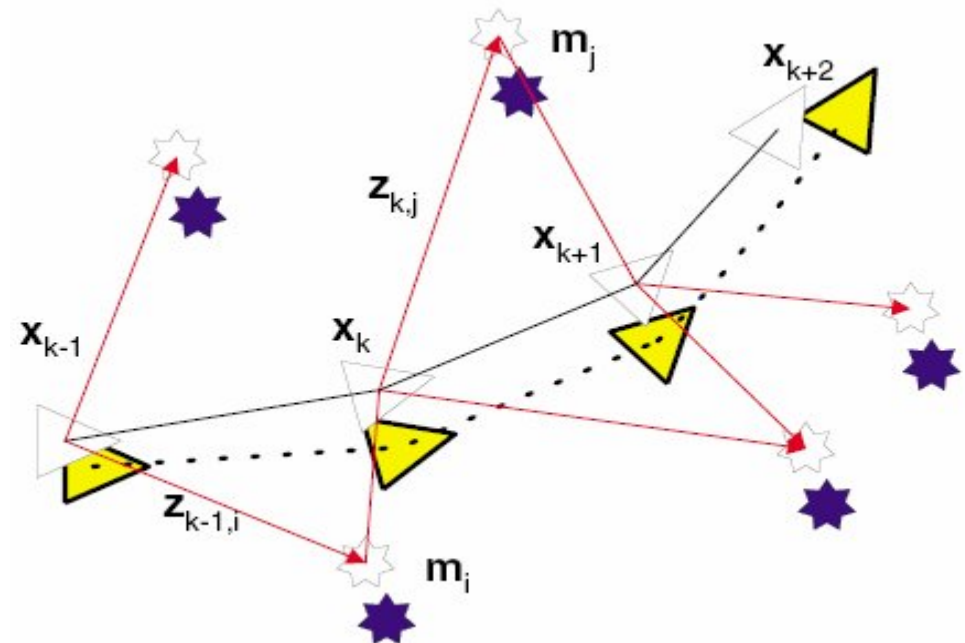


Simultaneous Localization and Mapping

Defined

- From knowledge of observations Z^k
 - Determine vehicle locations 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 $\{z_1, z_2, \dots, z_k\}$

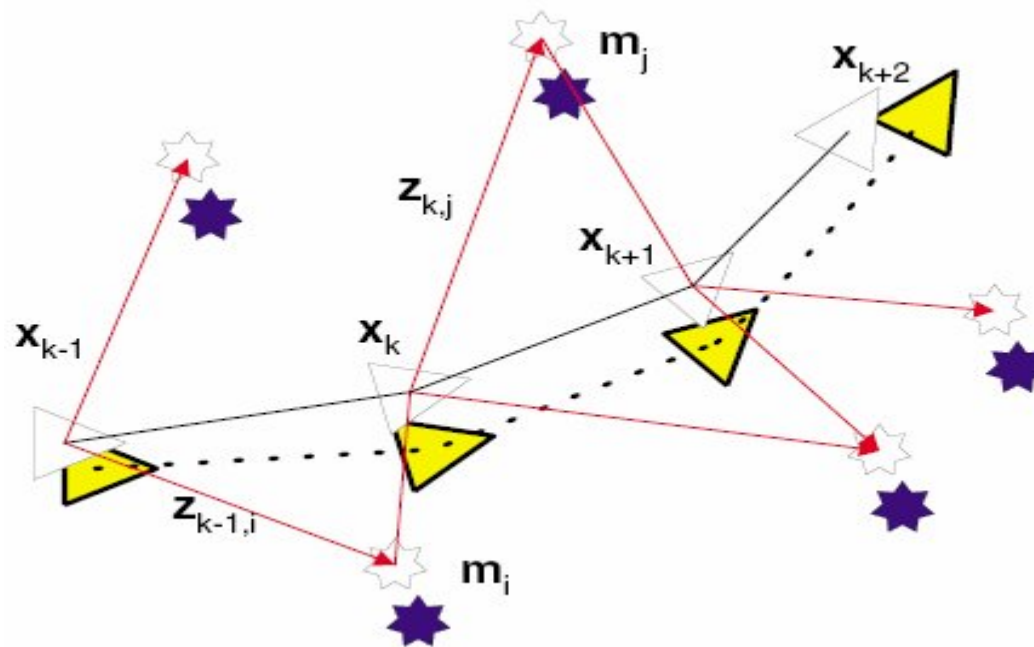


H. Durrant-Whyte, D. Rye, E. Nebot, "Localisation of Automatic Guided Vehicles", ISRR 1995

Simultaneous Localization and Mapping

Characteristics

- Localization and mapping are coupled problems
 - Two quantities are to be inferred from a single measurement
- A solution can only be obtained if the localization and mapping processes are considered together

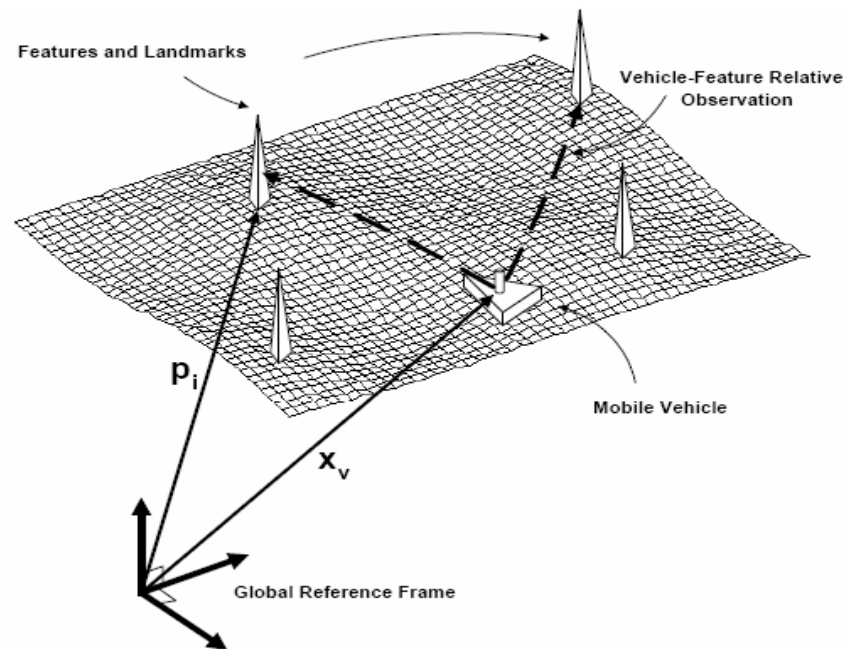


H. Durrant-Whyte, D. Rye, E. Nebot, "Localisation of Automatic Guided Vehicles",
Robotics Research: The 7th International Symposium (ISRR 1995)

SLAM Fundamentals

Setting

- 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**)



SLAM Fundamentals

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!

SLAM Fundamentals

Process Model

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

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

- The **state transition model** for the complete system is now

$$\begin{bmatrix} x_v(k+1) \\ p_1 \\ \vdots \\ p_N \end{bmatrix} = \begin{bmatrix} F_v(k) & 0 & \dots & 0 \\ 0 & I_{p_1} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & I_{p_N} \end{bmatrix} \begin{bmatrix} x_v(k) \\ p_1 \\ \vdots \\ p_N \end{bmatrix} + \begin{bmatrix} u_v(k+1) \\ 0_{p_1} \\ \vdots \\ 0_{p_N} \end{bmatrix} + \begin{bmatrix} w_v(k+1) \\ 0_{p_1} \\ \vdots \\ 0_{p_N} \end{bmatrix}$$

where

- I_{p_i} is the $\dim(p_i) \times \dim(p_i)$ identity matrix
- 0_{p_i} is the $\dim(p_i)$ null vector

SLAM Fundamentals

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

$$H_i = \begin{bmatrix} -H_v, 0 \dots 0, H_{p_i}, 0 \dots 0 \end{bmatrix}$$

- Re-expressing the observation model

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

Estimation Process

Objective

- The state of our discrete-time process x_k needs to be estimated based on our 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

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

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

Estimation Process

Update

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

$$\hat{x}(k+1 | k+1) = \hat{x}(k+1 | k) + W_i(k+1)w_i(k+1)$$

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

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

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

Kalman Filter

A Closer Look...

Kalman Filter

Background

- 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!



Discrete Kalman Filter

Process Model

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

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

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

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

 covariance

Discrete Kalman Filter

Observation Model

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

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

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

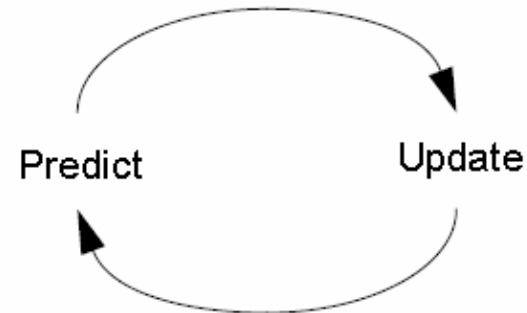
$$p(v) \sim N(0, R)$$

 covariance

Discrete Kalman Filter

Algorithm

- It's 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 $\hat{x}(k | k - 1) = F(k)\hat{x}(k - 1 | k - 1) + B(k)u(k - 1)$
- Predicted covariance $P(k | k - 1) = F(k)P(k - 1 | k - 1)F(k)^T + Q(k)$

Observation

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

Update

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

Not the same variable!!

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

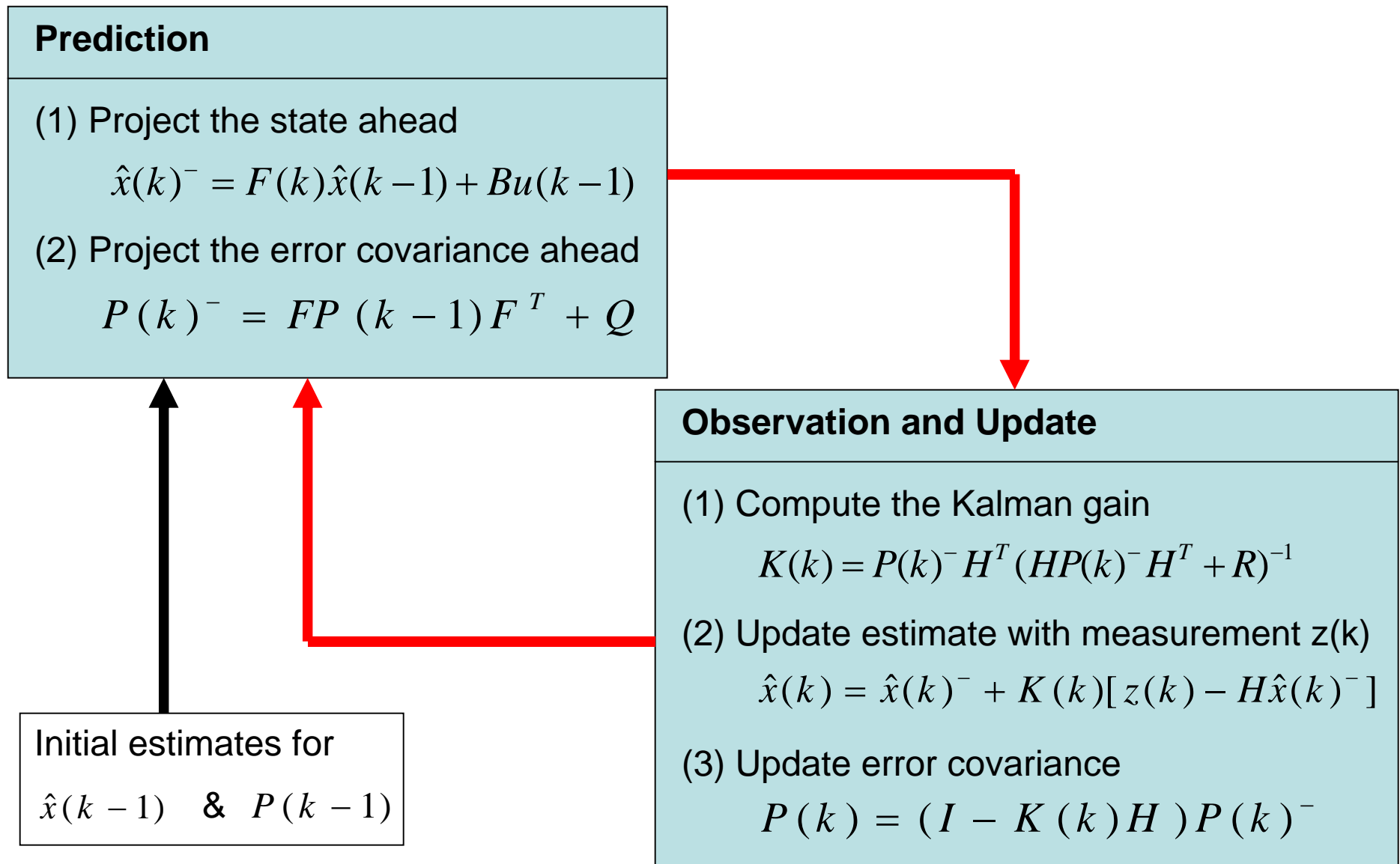
Observation

- 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 $P(k) = (I - K(k)H)P(k)^-$

Discrete Kalman Filter



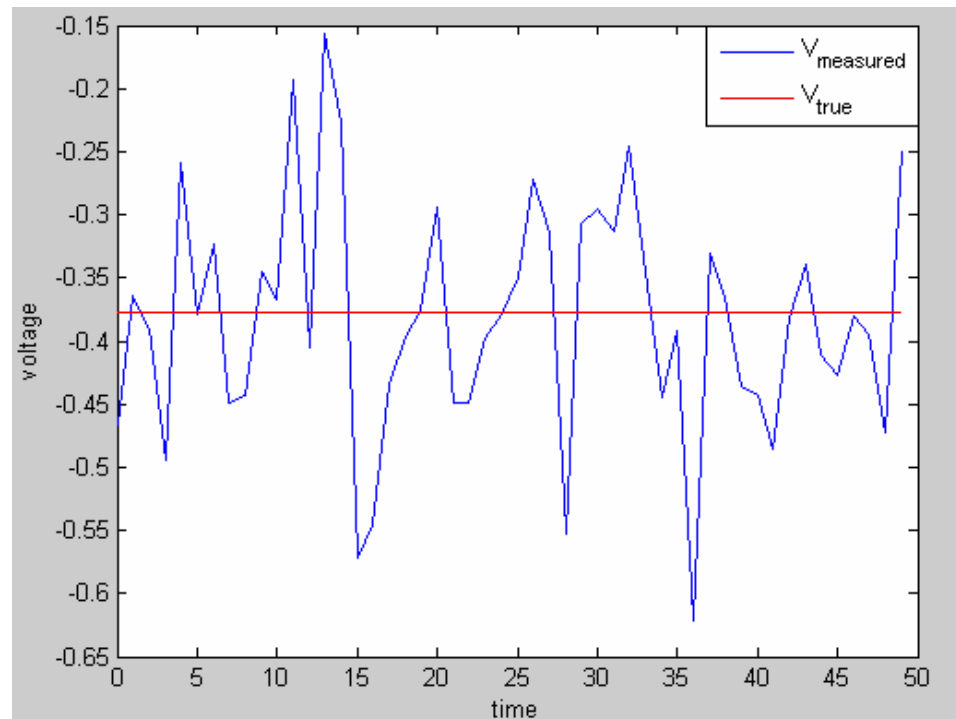
A Kalman Filter in Action

An Example...

Kalman Filter Example

Process Model

- 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

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

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

State doesn't change ($F=0$)
No control input ($u=0$)

with a measurement

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

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

Measurement is of state
directly ($H=1$)

Kalman Filter Example

Output

