Module IN 2018

Introduction to Augmented Reality

Prof. Gudrun Klinker
with contributions from M. Huber, D. Pustka, F. Echtler

Sensor Fusion and Registration

SS 2018

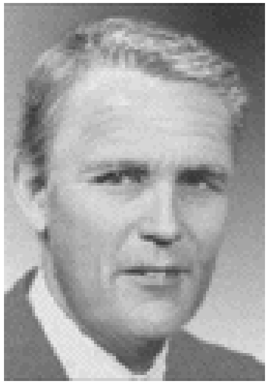
Agenda

1. The Kalman Filter
2. Sensor Fusion
3. Calibration and Registration

1. Kalman Filter

Literature

- G. Welch and G. Bishop, “An Introduction to the Kalman Filter”, SIGGRPAPH 2001 Course 8, http://www.cs.unc.edu/~welch/kalman
- A. Gelb (editor), “Applied Optimal Estimation”
- R.E. Kalman, “A new Approach to Linear Filtering and Prediction Problems”, Transactions ASME, 1960



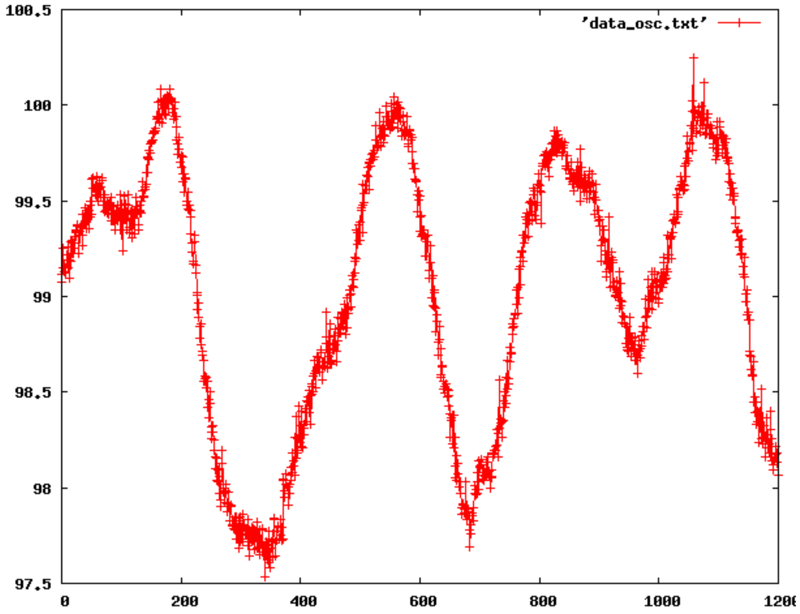
Rudolf Kálmán
https://de.wikipedia.org

1. The Kalman Filter

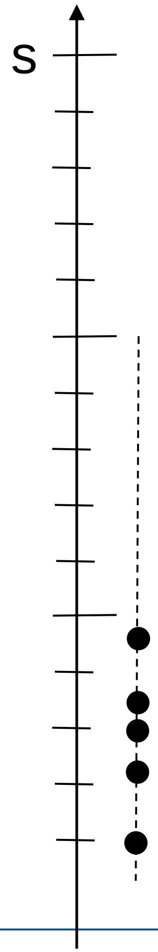
1.1 Motivation

Sensor measurements always subject to noise

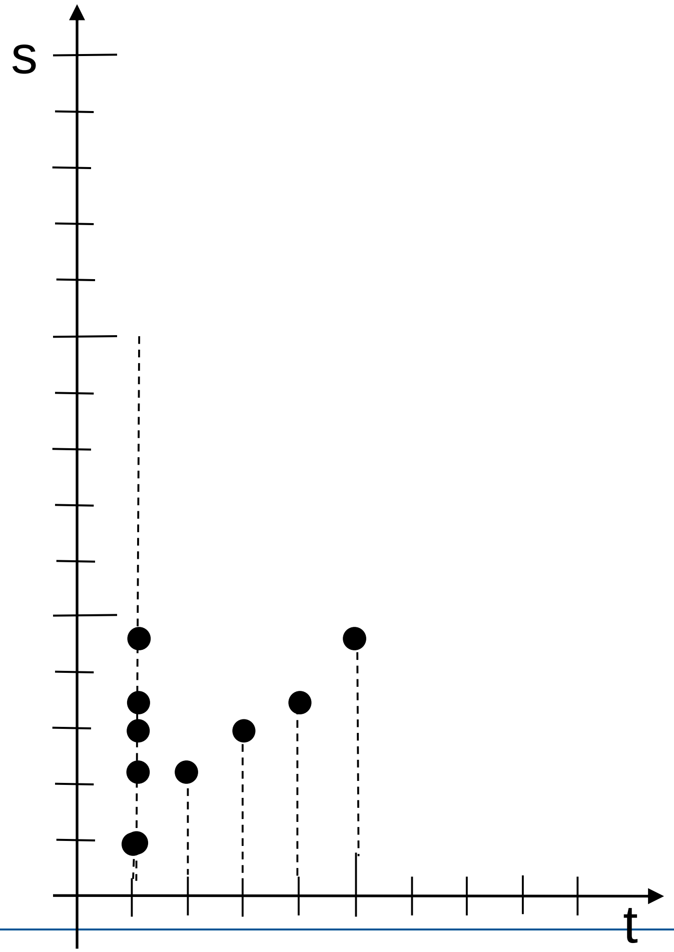
➡ Filtering



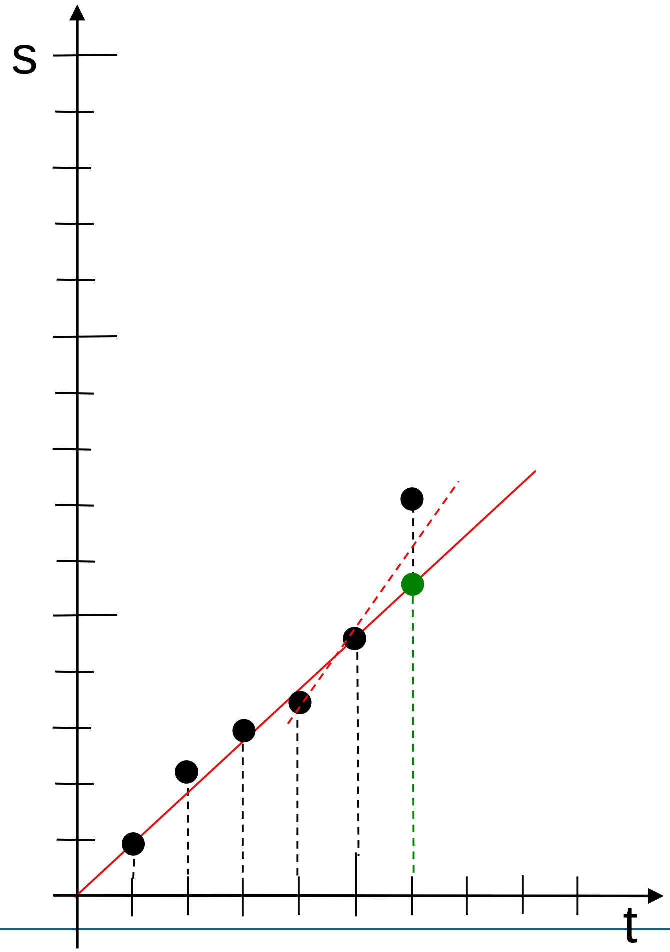
1.1 Motivation



1.1 Motivation



1.1 Motivation



- Sensor measurements
  - complex motion or noisy data?
- Motion model
  - $s(t) = \int v(t)\, dt$
  - e.g., constant speed: $s(t) = v \bullet t$
- Motion prediction
  - $s(t + \Delta t) = s(t) + v \bullet \Delta t$
- New measurement
  - update model "to some extent"

1.1 Motivation

Example

- Multiple sensors
  - Different ideas about time
  - Disagree on measured value

➡ Sensor-Fusion

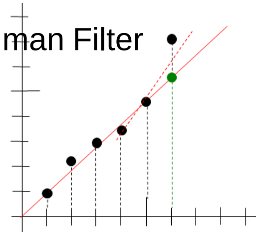
1. The Kalman Filter

1.2 Dynamic Process Model

- Model system as dynamic process
  - Estimation parameters
  - Noise parameters
- Model is application specific

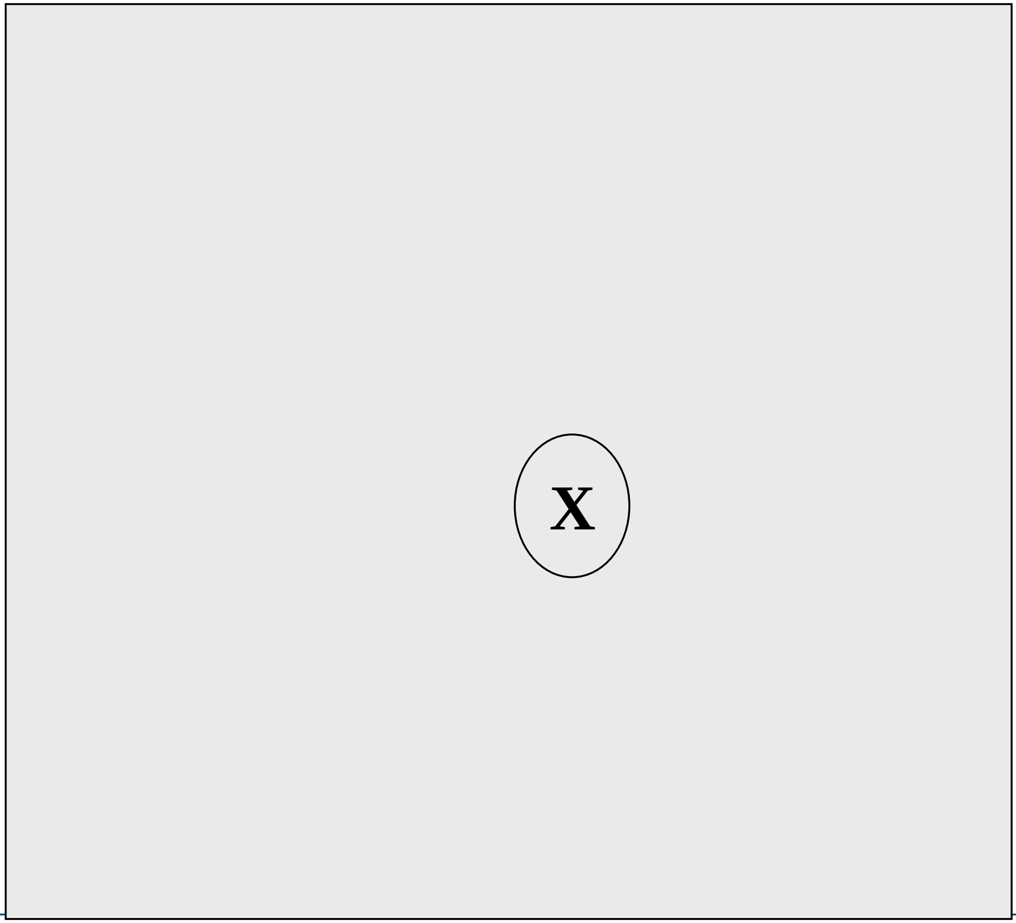
- Kalman Filter is a set of techniques
- Optimal estimator for linear, time-discrete, dynamic systems

1. Kalman Filter

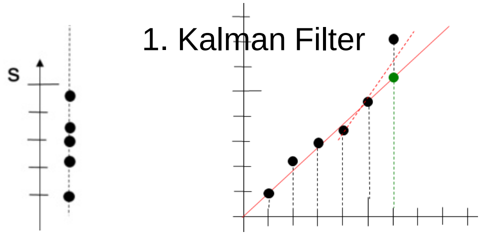


1.2 Dynamic Process Model

- State variable of process: $\mathbf{X}$
  (e.g: a motion model)
- Goal: Predict / approximate state
- Concrete characteristics dependent on application

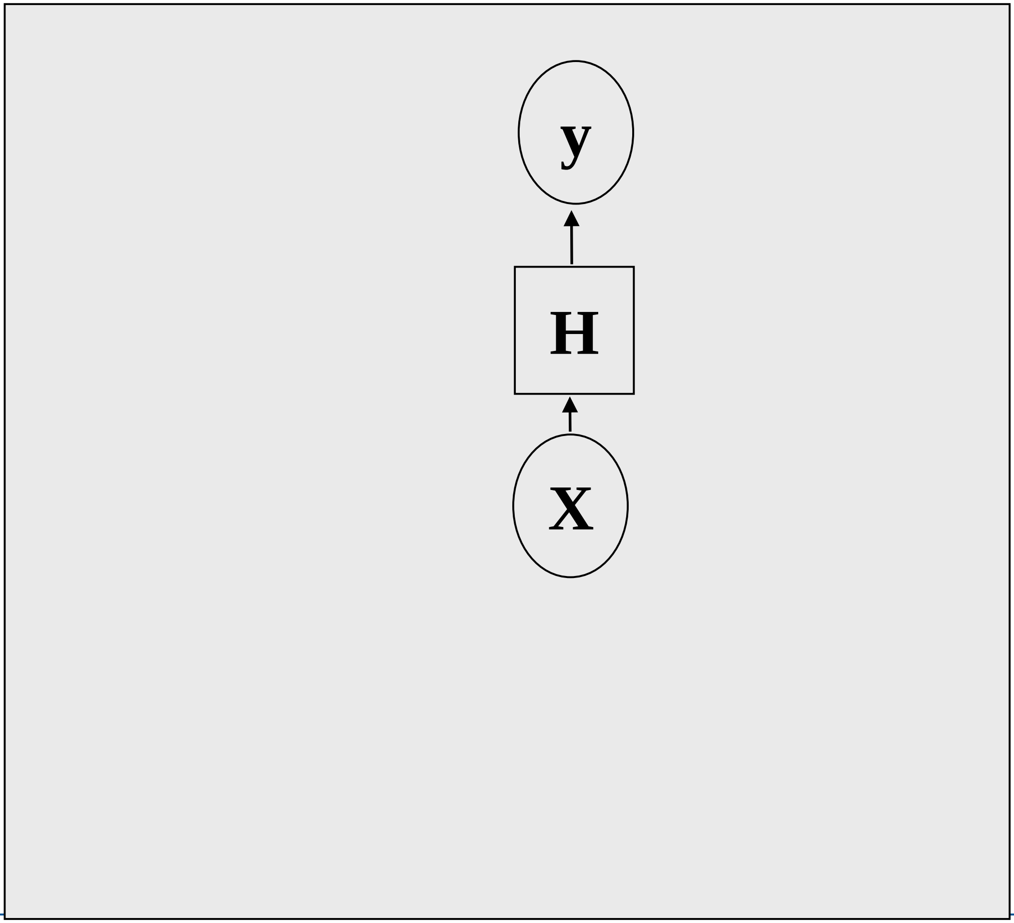


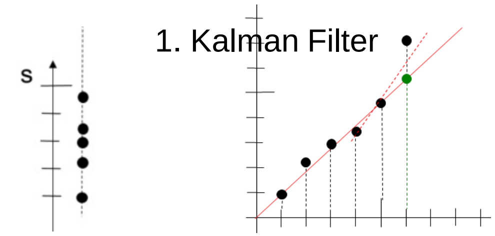
1.2 Dynamic Process Model



- What do we have to work with?
- Observations **y**

- Model: Linear projection **H** of state **X** to observable **y**

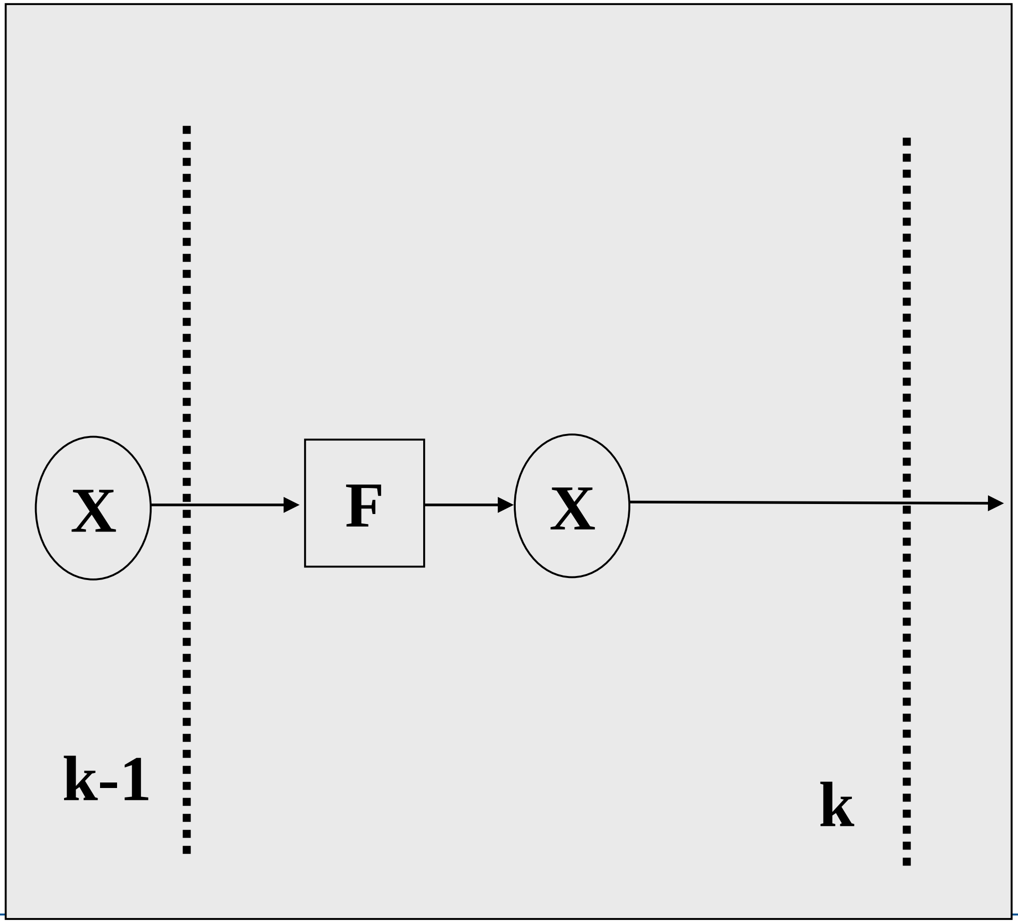




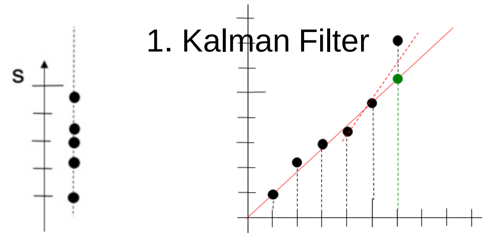
1.2 Dynamic Process Model

- State evolves on its own
  - Dynamic characteristics
  - User interaction

- Transition matrix **F**: Updates state **X** from time step k-1 to k

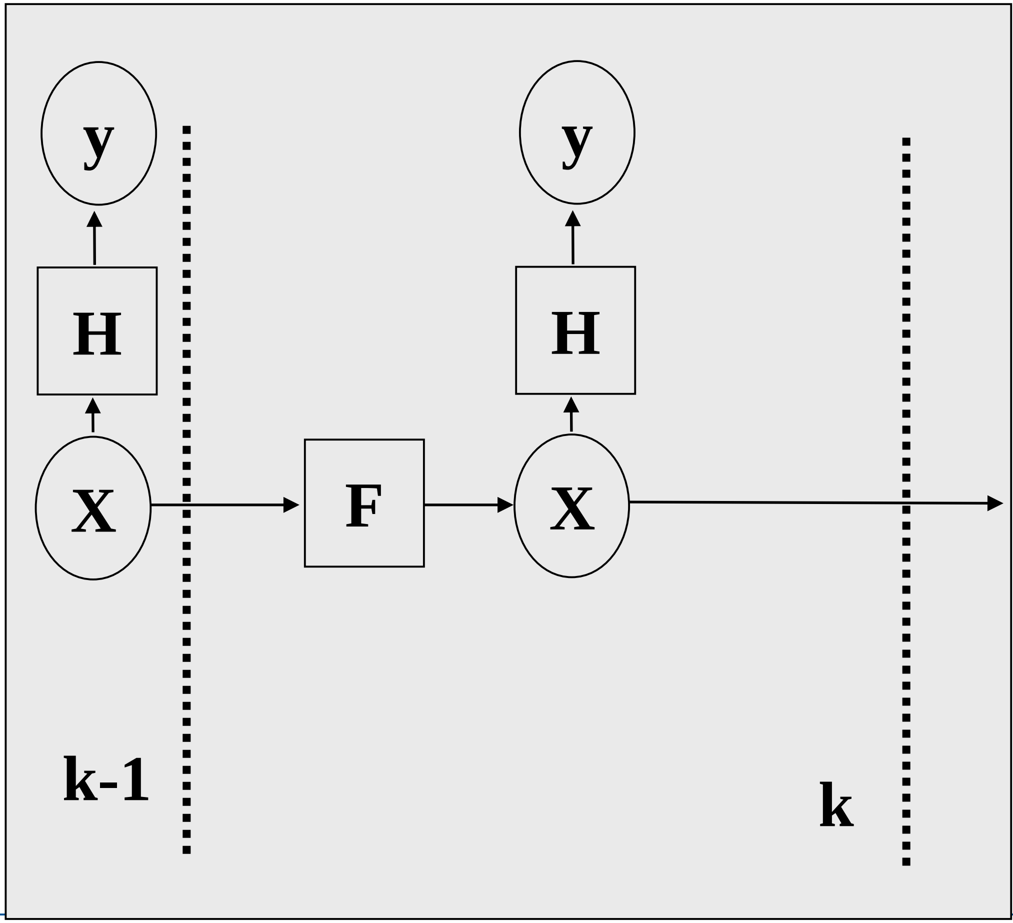


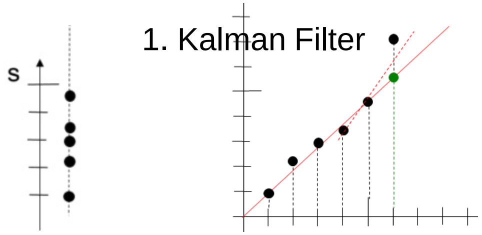
1.2 Dynamic Process Model



Summary thus far: Combination of

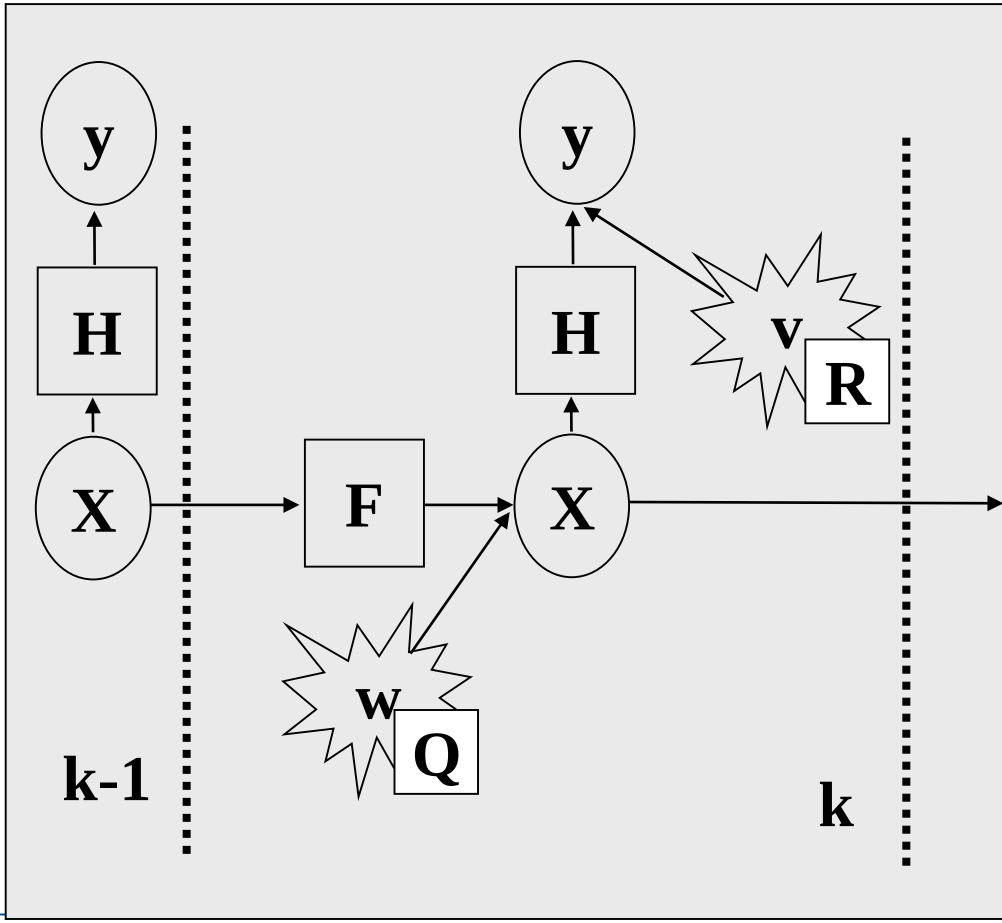
- State X_{k-1} at time step k-1
- Transition matrix F to transform X_{k-1} into X_k at time step k
- Projection matrix H to project internal system state X into observable "symptoms" y (at every time step …, k-1, k, …)





1.2 Dynamic Process Model

- Include two sources of noise
- Sensors are noisy
  - => Measurement noise $\mathbf{v}_k$ (described by covariance matrix R)
- Process is noisy
  - Indeterministic behavior
  - Unmodeled dynamic properties
  - Unmodeled external $\mathbf{w}_k$ influences
  - => Process Noise (described by covariance matrix Q)



1. The Kalman Filter

1.1 Motivation

1.2 Dynamic Process Model

→ 1.3 Mathematical Formulation

1.4 Outlook

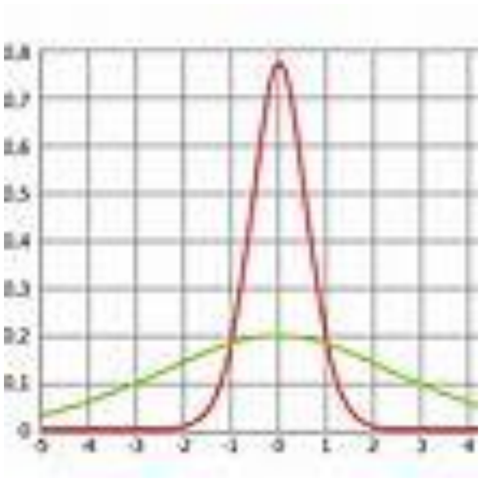
1.3 Mathematical Formulation

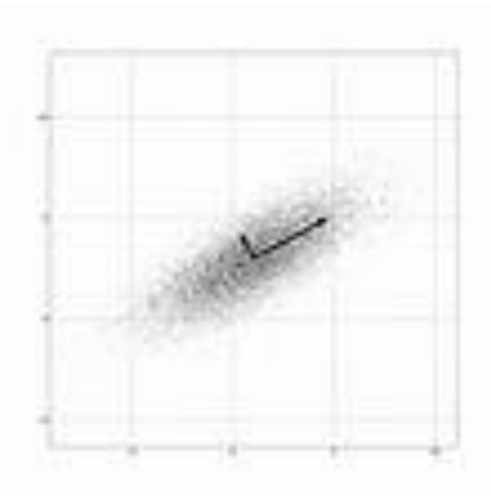
- Basic Approach: Recursive Filter
  - Predict next state from last state (Predict Step)
  - Update state estimate from measurement (Update Step)
- Optimality: Minimizes estimation error

1.3 Mathematical Formulation

- Reminder: Covariance matrix

$$Cov(X,Y) = E[(X - E[X])(Y - E[Y])] \qquad Var(X) = Cov(X,X)$$





[wikipedia.de]

1.3 Mathematical Formulation

- Reminder: Covariance matrix

$$Cov(X,Y) = E[(X - E[X])(Y - E[Y])] \qquad Var(X) = Cov(X,X)$$

- Generalization of variance to multivariate statistics

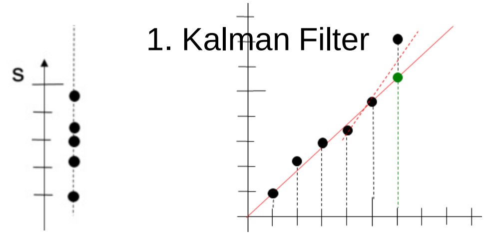
$$\underline{X} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} \qquad \Sigma = \begin{bmatrix} Cov(X_1,X_1) & Cov(X_1,X_2) & \cdots & Cov(X_1,X_n) \\ Cov(X_2,X_1) & Cov(X_2,X_2) & \cdots & Cov(X_2,X_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_n,X_1) & Cov(X_n,X_2) & \cdots & Cov(X_n,X_n) \end{bmatrix}$$

$$\Sigma = E[(\underline{X} - E[\underline{X}])(\underline{X} - E[\underline{X}])^T]$$

1.3 Mathematical Formulation

- Parameterizes multivariate distributions
- Modelled noise
  - Process Noise $\mathbf{w}_k \sim \mathrm{N}[0, \mathbf{Q}_k]$
  - Measurement Noise $\mathbf{v}_k \sim \mathrm{N}[0, \mathbf{R}_k]$
- Independent, white, Gaussian noise
- Q: modeling uncertainty
  - Larger Q => track large changes in data more closely
- R: how much to trust measurements
  - Large R => considers measurements as not very accurate
  - Smaller R => follow measurements more closely

1.3 Mathematical Formulation



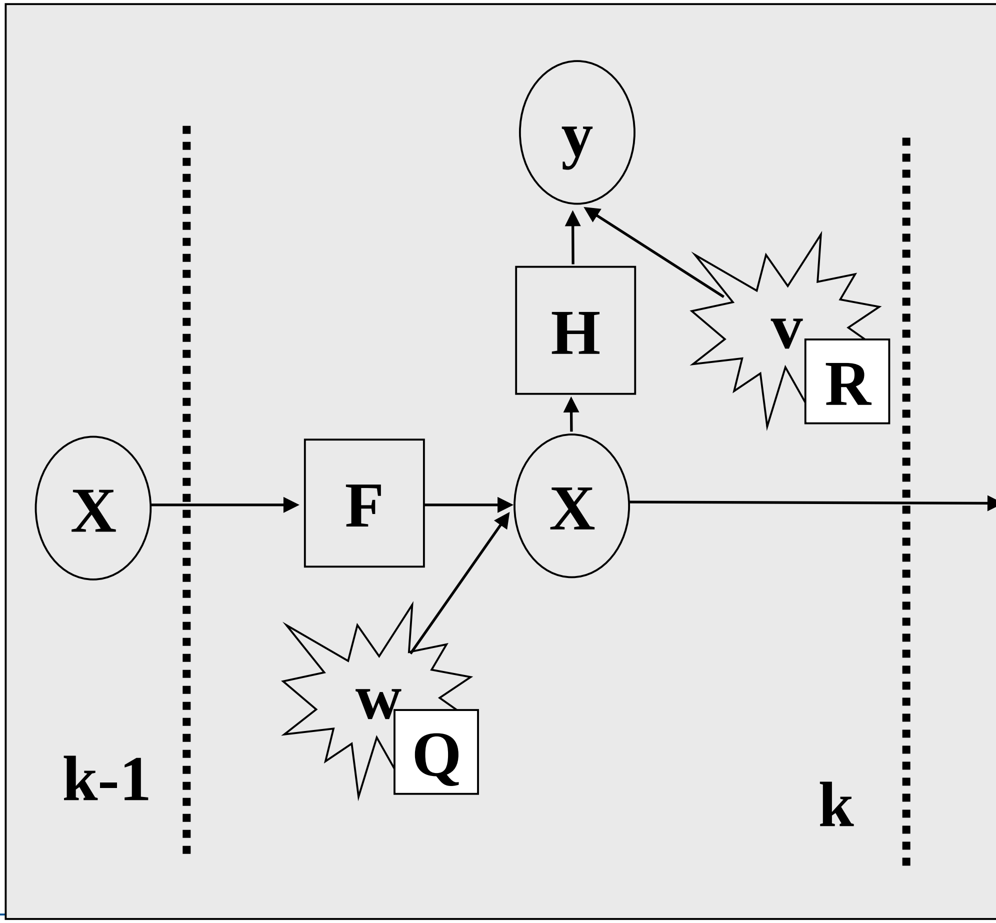
- Formulate model equations
  - Process equation

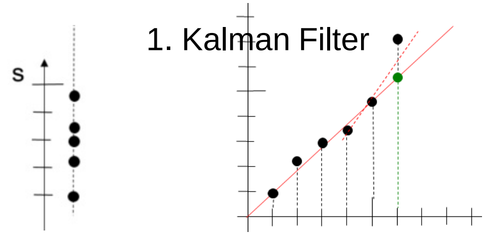
    $$\mathbf{x}_k = \mathbf{F}_{k,k-1}\mathbf{x}_{k-1} + \mathbf{w}_{k-1}$$

  - Measurement equation

    $$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

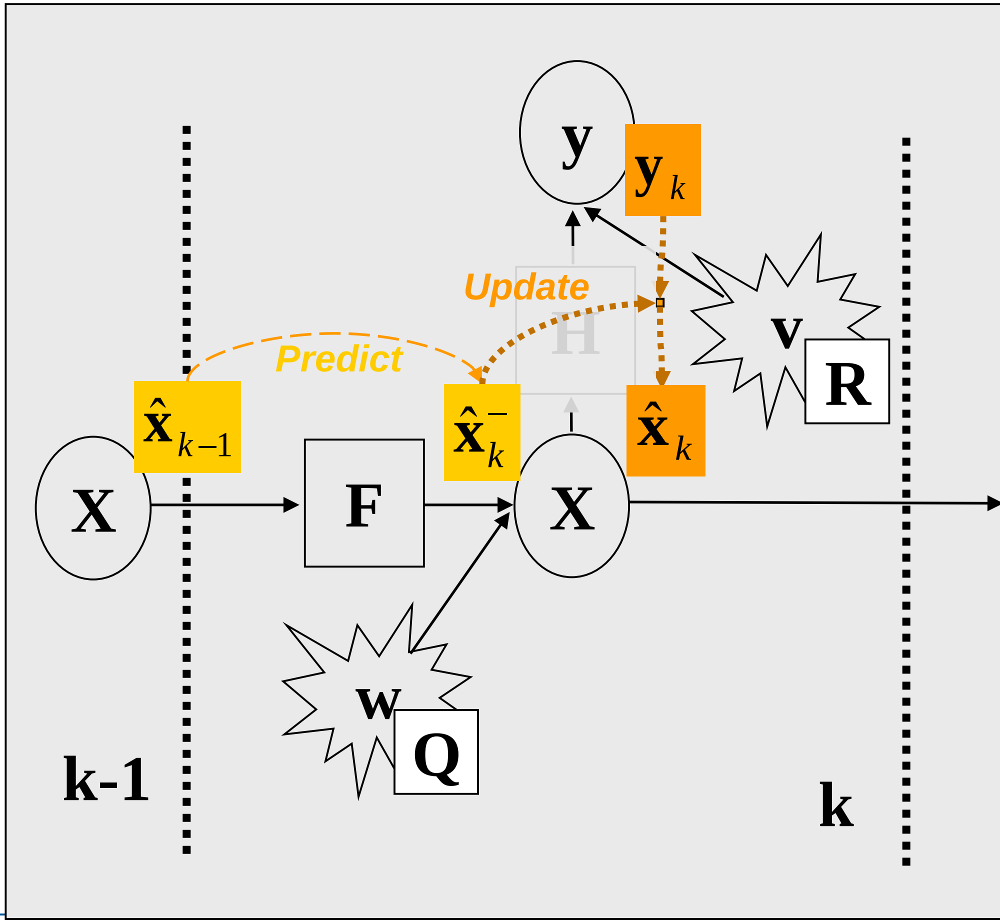
- With
  - State vector $\mathbf{x}_k$
  - Transition matrix $\mathbf{F}_{k,k-1}$
  - Observable $\mathbf{y}_k$
  - Measurement matrix $\mathbf{H}_k$
  - Noise $\mathbf{w}_k, \mathbf{v}_k$



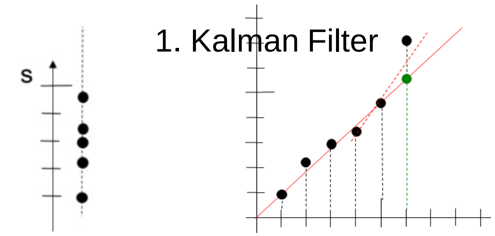


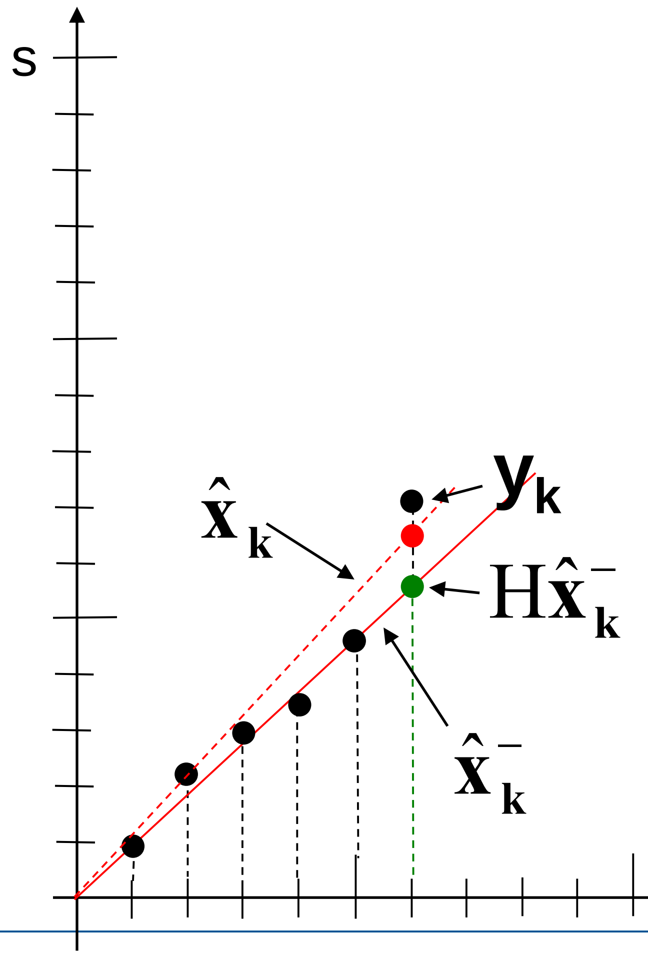
1.3 Mathematical Formulation

- Kalman Filter approximates true state $\mathbf{x}_k$ with estimate $\hat{\mathbf{x}}_k$
- At time step k:
  - A priori estimate using information of step k-1: $\hat{\mathbf{x}}_k^-$ (Prediction)
  - Measurement
  - => Improve a posteriori estimate (Update)
    $$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{G}_k(\mathbf{y}_k - \mathbf{H}\hat{\mathbf{x}}_k^-)$$
- Choose $\mathbf{G}_k$ such that error $\mathrm{E}[|\hat{\mathbf{x}}_k - \mathbf{x}_k|^2]$ is minimized
- $\mathbf{G}_k$ is called the Kalman-Gain



1.3 Mathematical Formulation





The a posteriori estimate $\mathbf{\hat{x}}_\mathbf{k}$ is a linear combination of the difference between measurement $\mathbf{y}_\mathbf{k}$ and measurement prediction $\mathrm{H}\mathbf{\hat{x}}_\mathbf{k}^-$ and the a priori state $\mathbf{\hat{x}}_\mathbf{k}^-$

$$\boxed{\mathbf{\hat{x}}_\mathbf{k} = \mathbf{\hat{x}}_\mathbf{k}^- + \mathrm{G}(\mathbf{y}_\mathbf{k} - \mathrm{H}\mathbf{\hat{x}}_\mathbf{k}^-)}$$

Kalman gain, blending factor

residual, innovation

1. The Kalman Filter

1.4 Outlook

- Non linear model with

$$\mathbf{x}_k = f(k, \mathbf{x}_{k-1}) + \mathbf{w}_{k-1} \qquad \mathbf{y}_k = h(k, \mathbf{x}_k) + \mathbf{v}_k$$

- Extended Kalman Filter (EKF)
  - Use functions where possible, linearize estimation around current estimate
  - Use Jacobians as transition/measurement matrices

$$\mathbf{F}_{k+1,k} = \frac{\partial f(k, \mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}=\hat{\mathbf{x}}_k} \qquad \mathbf{H}_k = \frac{\partial h(k, \mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}=\hat{\mathbf{x}}_k^-}$$

  - Notes:
    - F, H may change every step
    - Fundamental flaw: distributions of random variable no longer normal
- Unscented Kalman Filter
  - Uses deterministic sampling
  - Does not require calculation of Jacobians

Agenda

1. The Kalman Filter
2. Sensor Fusion
3. Calibration and Registration

2. Sensor Fusion

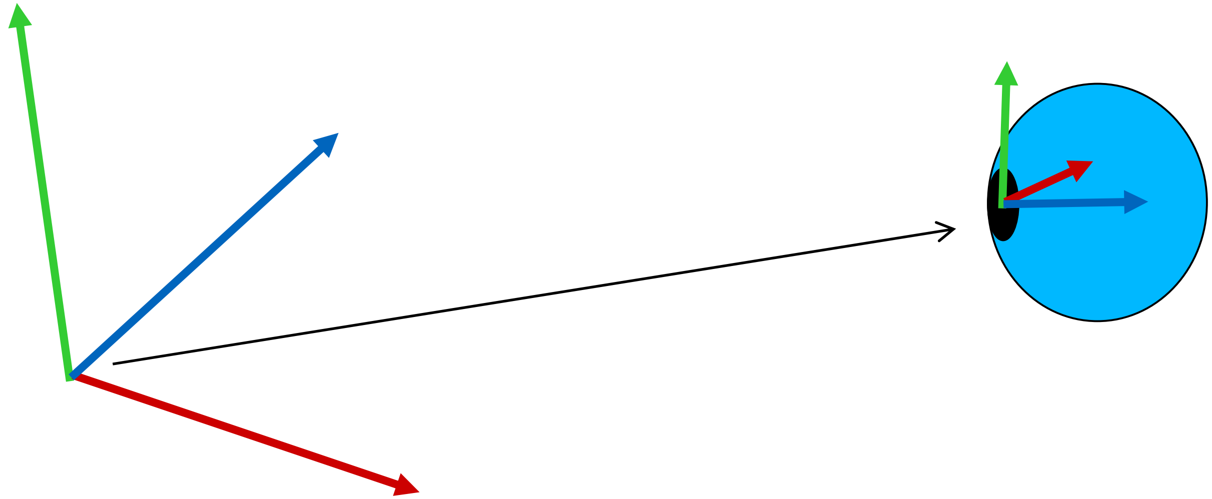
Literature

- Ronald Azuma and Gary Bishop. *Improving static and dynamic registration in an optical see-through HMD*. SIGGRAPH 1994, pages 197-204
- Greg Welch and Gary Bishop. *SCAAT: Incremental Tracking with Incomplete Information*. SIGGRAPH 1997, pages 333-344

2. Sensor Fusion

2.1 6DOF Tracking: Problem Definition

- Objective: 6DOF tracking of an object (e.g. HMD)
- Object pose expressed as transformation in 3-space
- Tracker measurements are noisy!



2. Sensor Fusion

2.2 State Vector

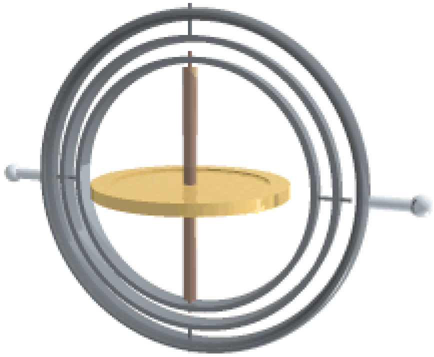
General Issue

- What do we need to know about the system?
  - Which parameters to estimate?
- Position
- Orientation
- Dynamic system
  - Translation velocity (positional changes, 1rst derivative)
  - Rotation velocity
- More derivatives possible
  - Accelerations (velocity changes, 2nd derivative) w.r.t. position and/or orientation

2.2 State Vector

Representation of Rotations

- So far: Pose represented as 4x4-Matrix
- 16 elements, but only 6 DOF → impracticable in state vector
- Separate translation (3 elements) and rotation
- Choices for rotation
  - Euler angles
    - Problems: gimbal lock, non-trivial multiplication
  - 3-element Axis-Angle
    - Vector direction: axis
    - Vector magnitude: angle
    - Problem: non-trivial multiplication
  - Quaternions



2.2 State Vector

Quaternions

- 4-element representation of rotation
- Generalization of complex numbers with imaginary $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$
- Only quaternions with $|\mathbf{q}|=1$ are pure rotations
  - All rotations lie on a 4D hyper-sphere
  - May require frequent normalization
- Inverse rotation: $\mathbf{q}^*$ (conjugate)
- Simple multiplication $\mathbf{q}_1 \cdot \mathbf{q}_2$
  - Consists only of sums and products
- Rotation of vectors: $\mathbf{x}' = \mathbf{q} \cdot \mathbf{x} \cdot \mathbf{q}^*$
  - Extend vector with 0 as real to construct a quaternion

2.2 State Vector

Quaternions and Axis-Angle

- Given
  - Rotation axis $\mathbf{x}$
  - $|\mathbf{x}| = 1$
  - Rotation angle θ
- The corresponding quaternion is

$$\mathbf{q} = (s\, x_1,\, s\, x_2,\, s\, x_3,\, c)$$

- where

$$s = \sin(\theta / 2)$$
$$c = \cos(\theta / 2)$$

2.2 State Vector

Complete State Vector

- Position
  - 3-Vector $\mathbf{p}$
- Translational velocity
  - 3-Vector $\mathbf{v}$
- Orientation
  - Quaternion $\mathbf{r}$
  - Sometimes requires normalization
- Rotation velocity
  - Axis-angle 3-vector $\mathbf{w}$
  - Allows velocities higher than 360° /s

$$\mathbf{x} = \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{r} \\ \mathbf{w} \end{pmatrix}$$

$$\mathbf{x}_k = \mathbf{F}_{k,k-1}\mathbf{x}_{k-1} + \mathbf{w}_{k-1}$$

2. Sensor Fusion

2.3 Motion Model

Example: Simple Linear Model for Translation at Constant Speed

- Simple linear model

  $$\hat{\mathbf{p}}_k^- = \hat{\mathbf{p}}_{k-1} + \Delta t \hat{\mathbf{v}}_{k-1}$$

  $$\hat{\mathbf{v}}_k^- = \hat{\mathbf{v}}_{k-1} \qquad \text{(constant speed)}$$

- Jacobian
  - Linear model → same as state transition **F** matrix for KF
  - Only on $\mathbf{x}_p$=($\mathbf{p}$, $\mathbf{v}$)

$$\begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{x}_k = \mathbf{F}_{k,k-1}\mathbf{x}_{k-1} + \mathbf{w}_{k-1}$$

$$\begin{bmatrix} p_{x_k} \\ p_{y_k} \\ p_{z_k} \\ v_{x_k} \\ v_{y_k} \\ v_{z_k} \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} p_{x_{k-1}} \\ p_{y_{k-1}} \\ p_{z_{k-1}} \\ v_{x_{k-1}} \\ v_{y_{k-1}} \\ v_{z_{k-1}} \end{bmatrix}_{k-1} + \mathbf{w}_{k-1}$$

2.3 Motion Model

Example: Rotation at Constant Rotational Speed

- Time update function

$$\hat{\mathbf{r}}_k^- = \hat{\mathbf{r}}_{k-1} \cdot \mathbf{q}(\Delta t \hat{\mathbf{w}}_{k-1})$$

$$\hat{\mathbf{w}}_k^- = \hat{\mathbf{w}}_{k-1}$$

- $\mathbf{q}(\mathbf{x})$ converts axis-angle to quaternion
  - Non-linear → EKF required
- Non-trivial Jacobian
  - Compute using symbolic math software (e.g. Mathematica, Maple)

$$\begin{bmatrix} \mathbf{J}_{r1} & \mathbf{J}_{r2} \\ 0 & \mathbf{I} \end{bmatrix}$$

2.3 Motion Model

Combined Time Update Equations

$$\hat{\mathbf{x}}_k^- = \begin{pmatrix} \hat{\mathbf{p}}_k^- \\ \hat{\mathbf{v}}_k^- \\ \hat{\mathbf{r}}_k^- \\ \hat{\mathbf{w}}_k^- \end{pmatrix} = \mathbf{f}(\hat{\mathbf{x}}_{k-1}) = \begin{pmatrix} \hat{\mathbf{p}}_{k-1} + \Delta t \hat{\mathbf{v}}_{k-1} \\ \hat{\mathbf{v}}_{k-1} \\ \hat{\mathbf{r}}_k^- \cdot \mathbf{q}(\Delta t \hat{\mathbf{w}}_{k-1}) \\ \hat{\mathbf{w}}_{k-1} \end{pmatrix}$$

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{Q}_k$$

$$\mathbf{A}_k = \left.\frac{\delta \mathbf{f}(\hat{\mathbf{x}})}{\delta \hat{\mathbf{x}}}\right|_{\hat{\mathbf{x}}=\hat{\mathbf{x}}_{k-1}} = \begin{bmatrix} \mathbf{I} & \Delta t\,\mathbf{I} & 0 & 0 \\ 0 & \mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{J}_{r1} & \mathbf{J}_{r2} \\ 0 & 0 & 0 & \mathbf{I} \end{bmatrix}$$

2. Sensor Fusion

2.4 Measurement Update

6DOF-Tracker Integration

- Absolute 6DOF tracker directly measures $\mathbf{p}_\mathrm{k}$ and $\mathbf{r}_\mathrm{k}$

$$\hat{\mathbf{y}}_k^- = \mathbf{h}(\hat{\mathbf{x}}_k^-) = \mathbf{h}\begin{pmatrix} \hat{\mathbf{p}}_k^- \\ \hat{\mathbf{v}}_k^- \\ \hat{\mathbf{r}}_k^- \\ \hat{\mathbf{w}}_k^- \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{p}}_k^- \\ \hat{\mathbf{r}}_k^- \end{pmatrix}$$

$$\mathbf{H} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & 0 & I & 0 \end{bmatrix}$$

- Note: $\mathbf{v}$, $\mathbf{w}$ are not measured explicitly!

2.4 Measurement Update

Gyroscope Integration

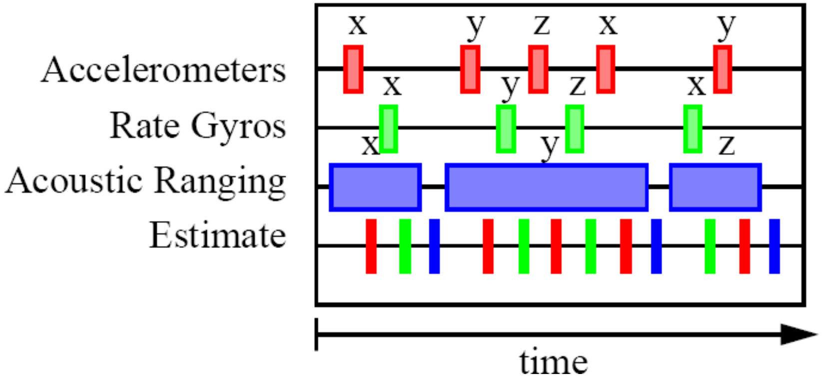
- Gyroscope directly measures $\mathbf{w}_\mathrm{k}$

$$\hat{\mathbf{y}}_k^- = \mathbf{h}(\hat{\mathbf{x}}_k^-) = \mathbf{h}\begin{pmatrix} \hat{\mathbf{p}}_k^- \\ \hat{\mathbf{v}}_k^- \\ \hat{\mathbf{r}}_k^- \\ \hat{\mathbf{w}}_k^- \end{pmatrix} = \left(\hat{\mathbf{w}}_k^-\right)$$

2.4 Measurement Update

The Actual Sensor Fusion Step

- Multiple different sensors
- For each measurement:
  1. Time update (to time of measurement)
  2. Measurement update (can be sensor-specific)
- Measurements should arrive in order!
- Requires timestamps!



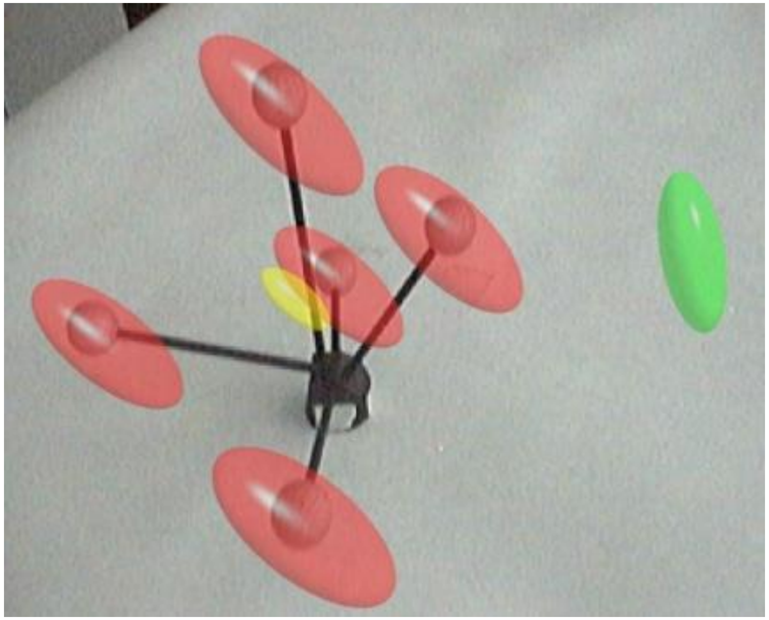
2.4 Measurement Update

Tracker Covariance

- How to set matrix $\mathbf{R}$ (measurement covariance)?
- Describes error distribution of tracker
- Trivial version:

$$\mathbf{R} = \sigma^2 \mathbf{I}$$

- Better: Compute $\mathbf{R}$ for each measurement based on the actual observations

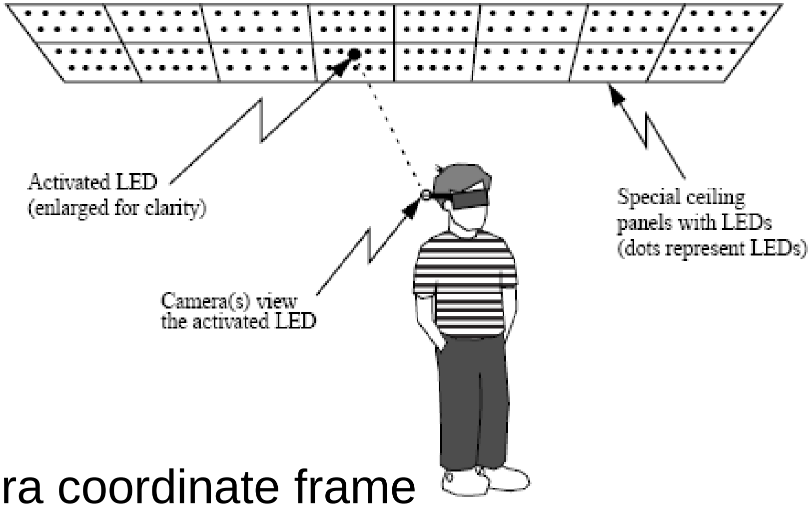


2. Sensor Fusion

2.5 Further Applications

2D Image Measurements

- EKF can directly integrate 2D image measurements



- Measurement equation

$$h(\hat{\mathbf{x}}_k^-) = \frac{\left(\mathbf{K}(\hat{\mathbf{r}}_k^- \mathbf{a} \hat{\mathbf{r}}_k^{-*} + \hat{\mathbf{t}}_k^-)\right)_{x,y}}{\left(\mathbf{K}(\hat{\mathbf{r}}_k^- \mathbf{a} \hat{\mathbf{r}}_k^{-*} + \hat{\mathbf{t}}_k^-)\right)_z}$$

1. Rotates the world point $\mathbf{a}$ into camera coordinate frame
2. Applies intrinsic camera matrix $\mathbf{K}$
3. Computes perspective division (de-homogenization)
   → Non-linear!

2.5 Further Applications

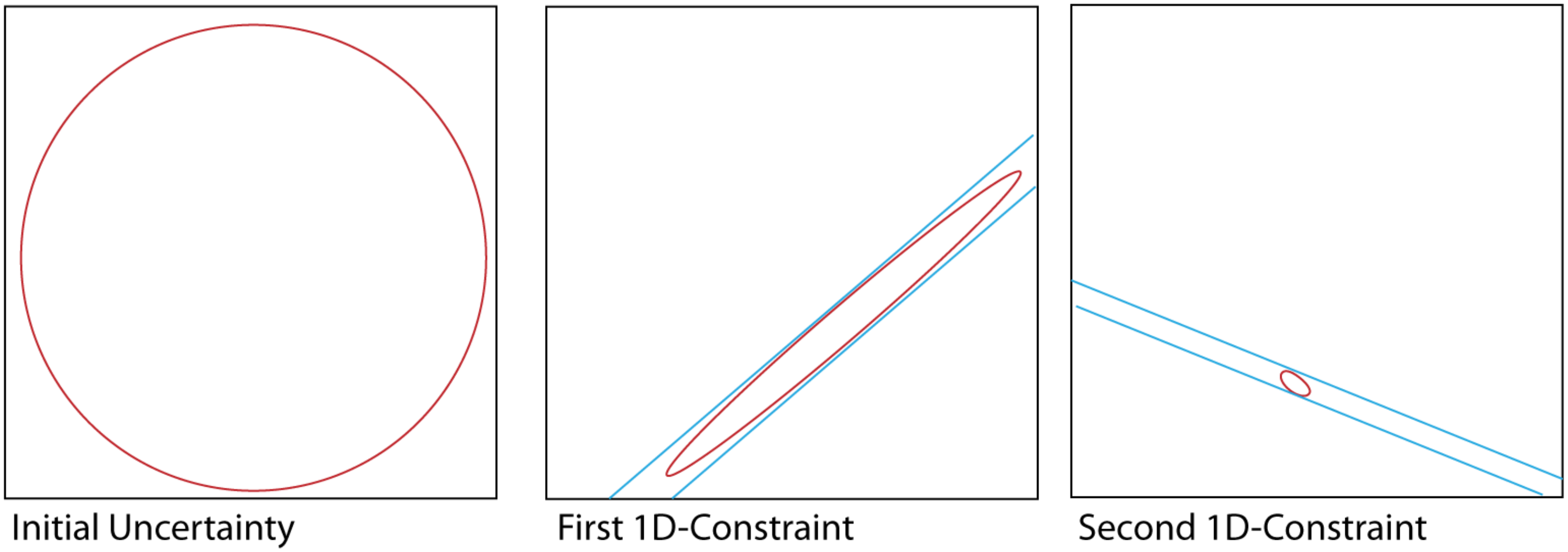
SCAAT

- State vector has 13 DOF
- (E)KF can compute all 13 elements given only one of (a large set of) 2D observations at a time
  - → Single Constraint At A Time (SCAAT)
- However: System must be observable
  - Needs different points

2.5 Further Applications

SCAAT – Why does it work?

- Example: Observation of a 2D point by two 1D cameras



2.5 Further Applications

Auto-Calibration / SLAM

- What if world point $\mathbf{a}$ is not known exactly?
- Idea: Integrate $\mathbf{a}$ into state vector

$$\mathbf{x} = (\mathbf{p}, \mathbf{v}, \mathbf{r}, \mathbf{w}, \mathbf{a}_1, \ldots, \mathbf{a}_n)^T$$

- Measurement equation $\mathbf{h}$ same as before

$$h(\hat{\mathbf{x}}_k^-) = \frac{\left(K(\hat{\mathbf{r}}_k^- \mathbf{a}_i \hat{\mathbf{r}}_k^{-*} + \hat{\mathbf{t}}_k^-)\right)_{x,y}}{\left(K(\hat{\mathbf{r}}_k^- \mathbf{a}_i \hat{\mathbf{r}}_k^{-*} + \hat{\mathbf{t}}_k^-)\right)_z}$$

- Jacobian H includes derivative of h w.r.t. a
- Kalman covariance matrix P stores uncertainty of each individual point and the dependencies between them!
- If new points are added while running
  - Simultaneous Localization and Mapping (SLAM)

2.5 Further Applications

Agenda

1. The Kalman Filter
2. Sensor Fusion
3. Calibration and Registration

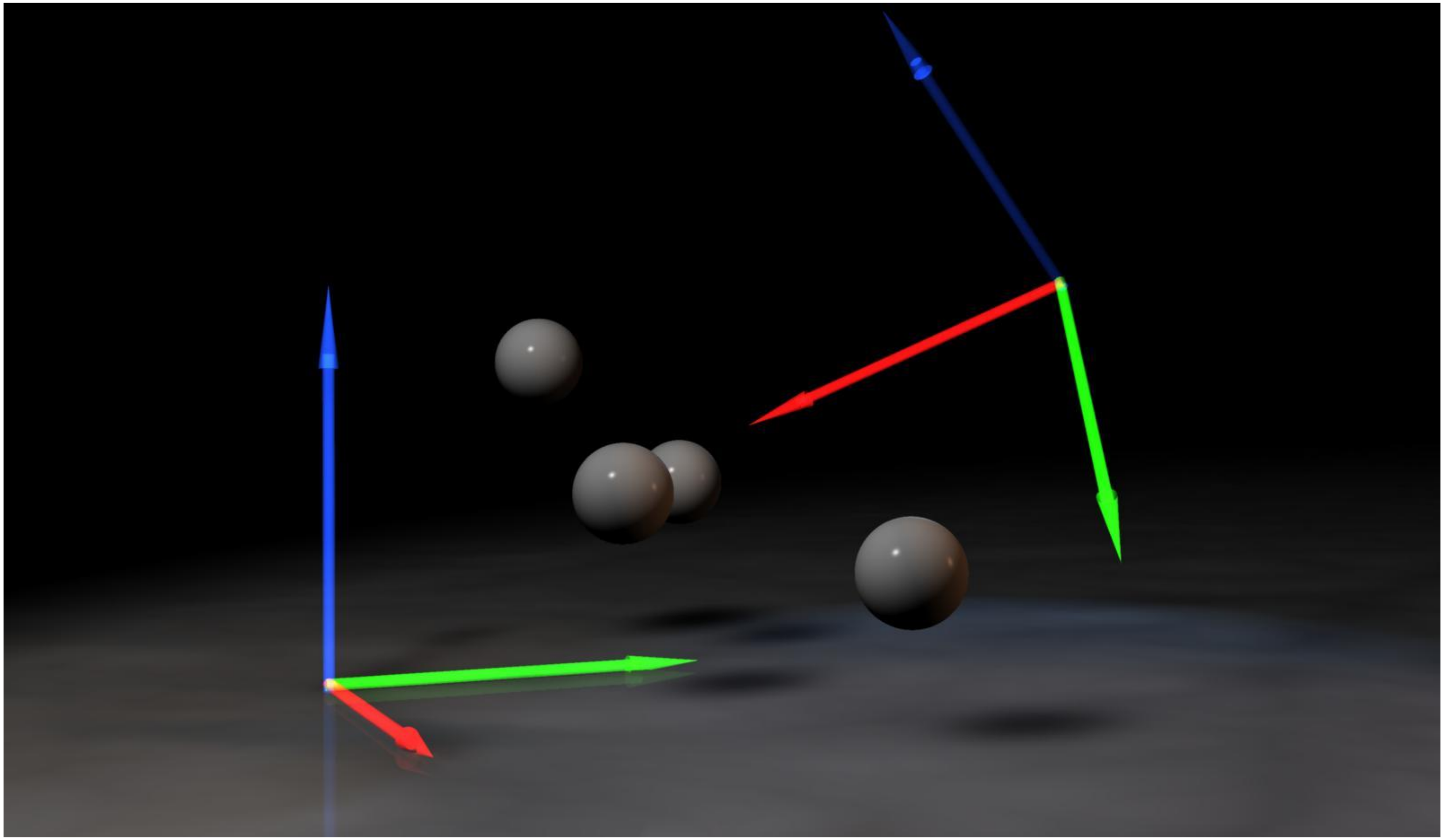
3. Calibration and Registration

→ 3.1 Absolute Orientation

3.2 Hand-Eye Calibration

3.1 Absolute Orientation

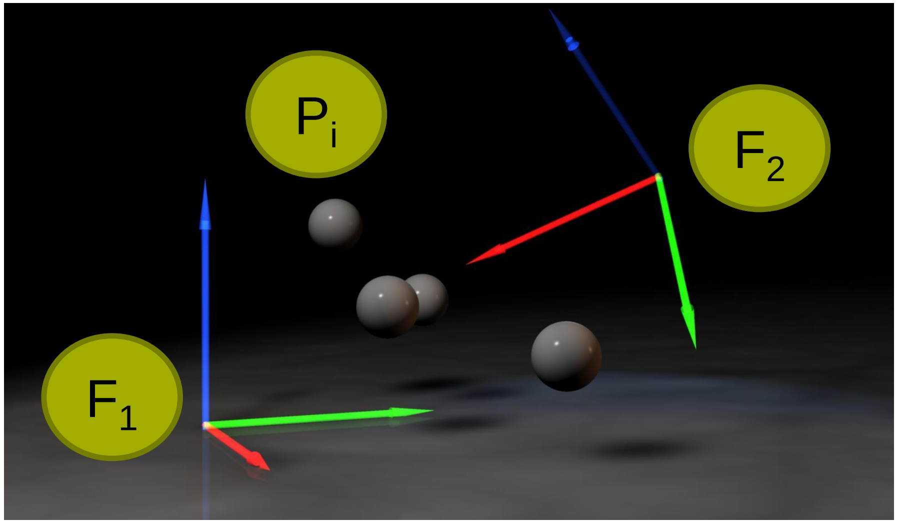
3.1 Absolute Orientation



3.1 Absolute Orientation

Setup and Motivation

- Setup
  - Two different coordinate frames, F_1 and F_2
  - Given pairs of coordinates of points in both frames P_i (i = 1..n)

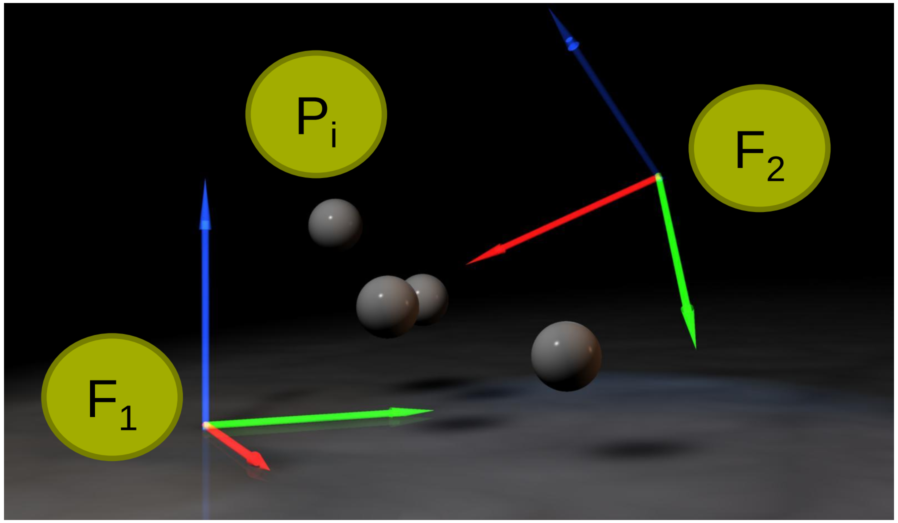


- Task
  - Determine the relationship between the two coordinate frames

3.1 Absolute Orientation

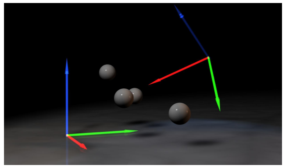
Setup and Motivation

- Possible Transformations
  - Rotation
  - Translation
  - Scaling (not expressible by pose)



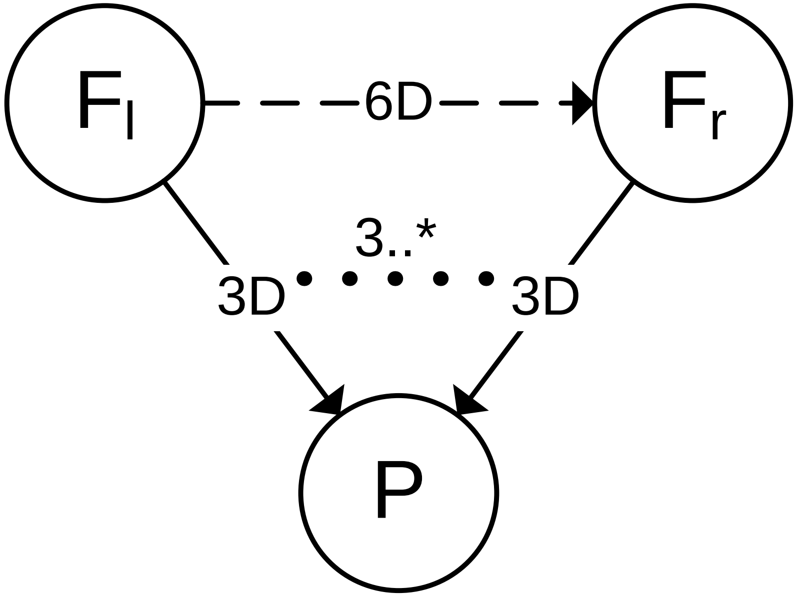
- Horn, B. K. P.
  Closed-form solution of absolute orientation using unit quaternions

3.1 Absolute Orientation



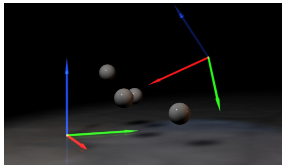
Setup and Motivation

- Spatial Relationship Graph
  - Tracked Relationships
    - $F_l \rightarrow P$
    - $F_r \rightarrow P$
  - Unknown Relationships
    - $F_l \rightarrow F_r$



3.1 Absolute Orientation

3.1 Absolute Orientation



Solution

- For any vector r_l in the left frame find parameters such that

  $$r_r = R(r_l) + t$$

  is the corresponding vector in the right frame
- Let the corresponding point coordinates be

  $$\{r_{l,i}\} \text{ and } \{r_{r,i}\}$$

- Strategy
  - First determine the rotation R
  - Translation follows easily

3.1 Absolute Orientation

Solution

- Calculate the centroids of the points

$$\bar{r}_l = \frac{1}{n}\sum_i r_{l,i} \qquad \bar{r}_r = \frac{1}{n}\sum_i r_{r,i}$$

- Normalize the coordinates

$$r'_{l,i} = r_{l,i} - \bar{r}_l \qquad r'_{r,i} = r_{r,i} - \bar{r}_r$$

- Note that given the rotation R we can calculate the translation

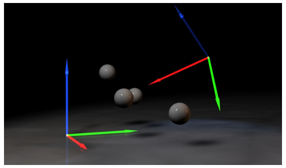
$$t = \bar{r}_r - R(\bar{r}_l)$$

3.1 Absolute Orientation

3.1.1 Setup and Motivation

3.1.2 Solution

→ 3.1.3 Applications



3.1 Absolute Orientation

Applications

- 3D-3D-Pose estimation

3. Calibration and Registration

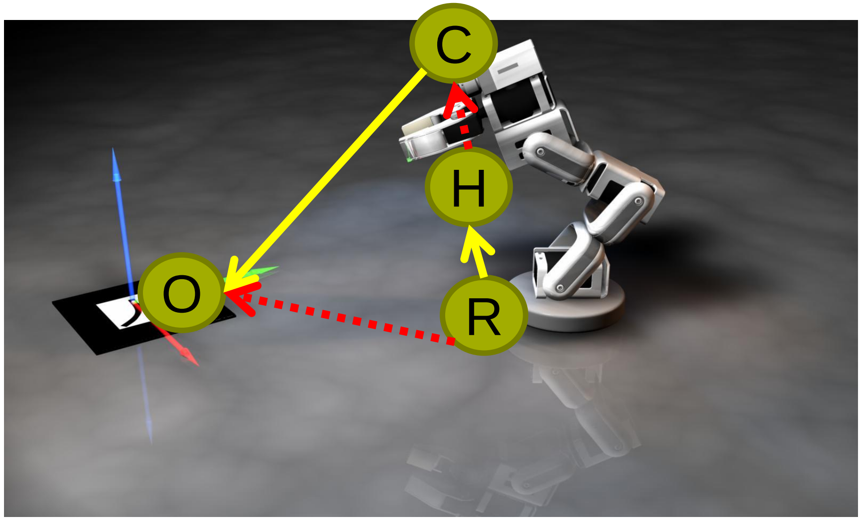
3.2 Hand-Eye Calibration

3.2 Hand-Eye Calibration

Setup and Motivation

Source: faro.com

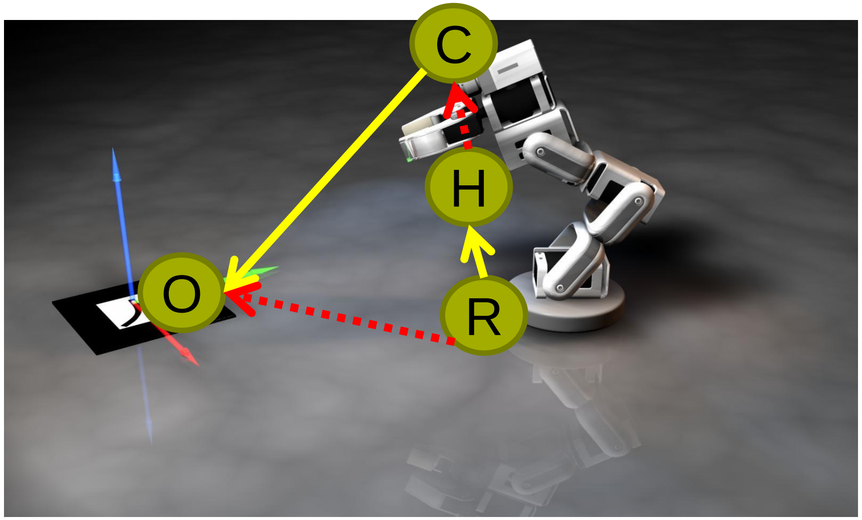
3.2 Hand-Eye Calibration



Setup and Motivation

- Involved Coordinate Frames
  - Robot-Base (R)
  - Hand (H)
  - Camera (C)
  - Object (O)
- Hand pose is known by forward kinematics
- Camera tracks Object
- Unknown:
  - Pose of Camera, relative to robot Hand
  - Pose of Object, relative to Robot base

3.2 Hand-Eye Calibration

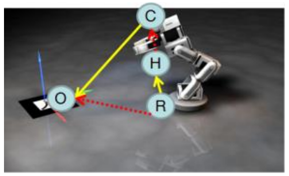


Setup and Motivation

- Task
  - Find the Object pose in the Robot base coordinate frame
- Problem
  - Transformation Camera to Hand is unknown

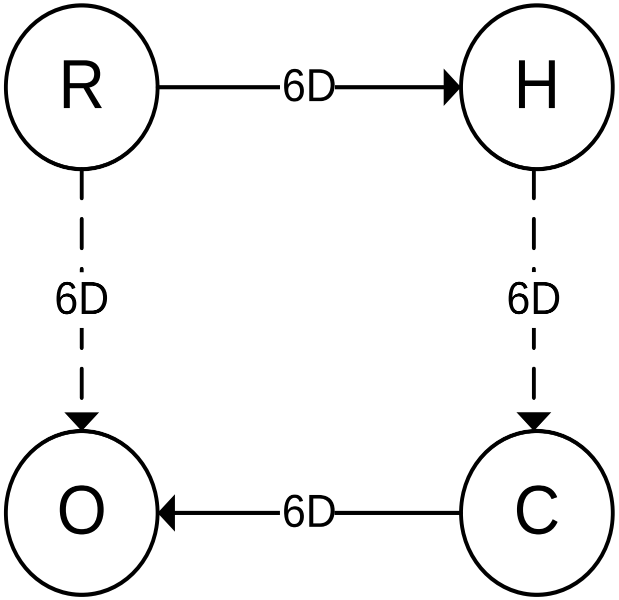
- Calibration (registration) problem

3.2 Hand-Eye Calibration



Setup and Motivation

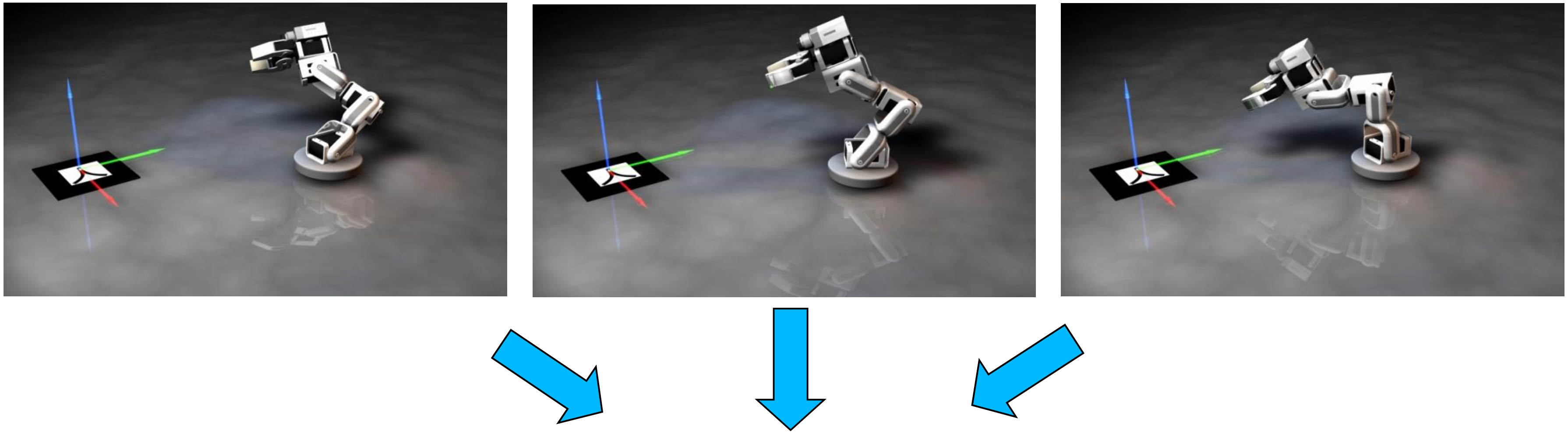
- Spatial Relationship Graph
  - Tracked Relationships
    - R → H
    - C → O
  - Unknown Relationships
    - H → C (Calibration)
    - R → O



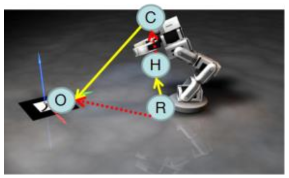
3.2 Hand-Eye Calibration

3.2 Hand-Eye Calibration

Basic Approach



- Move robot arm while keeping the object fixed; use several robot postures
- At least 3 different postures are needed

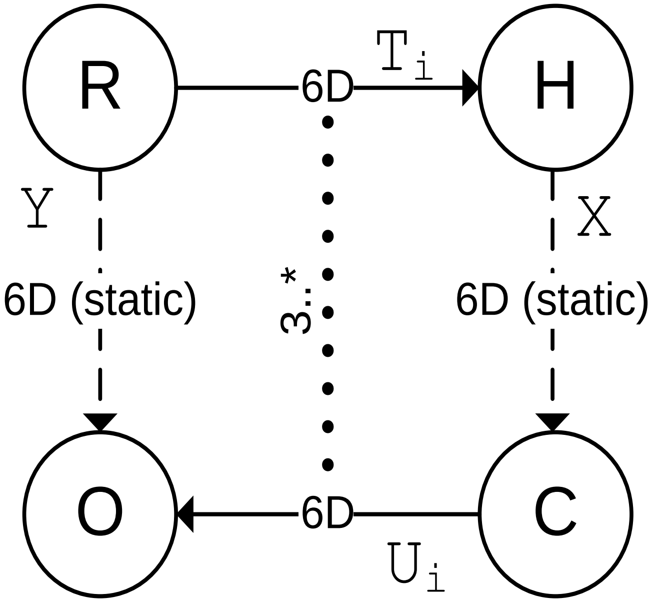


3.2 Hand-Eye Calibration

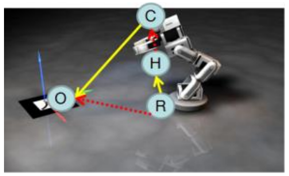
Basic Approach

- Represent poses as 4x4 matrices
- For the i^th robot configuration:

$$Y = T_i X U_i$$



3.2 Hand-Eye Calibration



Basic Approach

- For the i[th] robot configuration:

$$Y = T_i X U_i$$

- Combining two distinct configurations `i` and `j`:

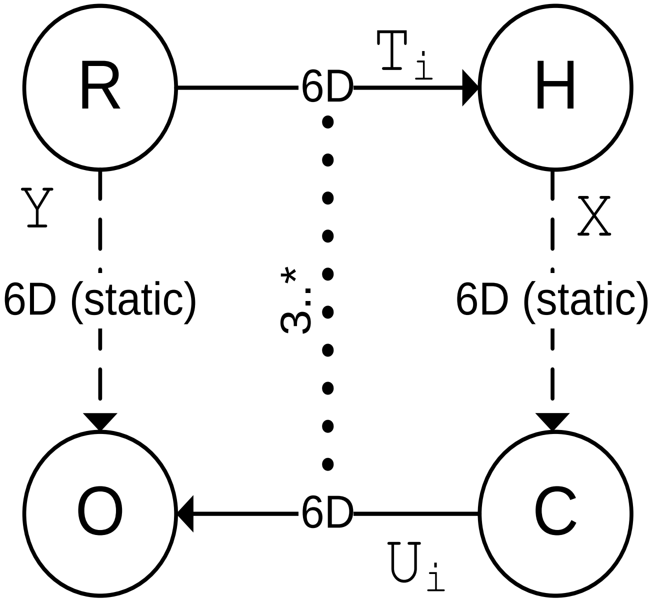
$$T_j X U_j = T_i X U_i$$

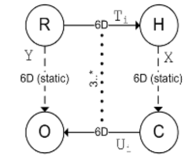
- Simplifying using

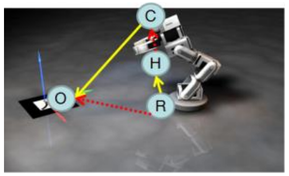
$$A = T_i^{-1} T_j \qquad B = U_i U_j^{-1}$$

- yields:

$$AX = XB$$







3.2 Hand-Eye Calibration

Basic Approach

$$AX = XB$$

- Solution of equation not unique
- One equation per pair i, j

- At least two equations simultaneously needed
  $\rightarrow$ ≥3 robot configurations

- System is over constrained
  - Solve for X minimizing error
  - e.g. Linear least squares

3.2 Hand-Eye Calibration

3.2 Hand-Eye Calibration

Solutions

$$AX = XB$$

- Using

$$A = \left[\begin{array}{ccc|c} & R_a & & T_a \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad B = \left[\begin{array}{ccc|c} & R_b & & T_b \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad X = \left[\begin{array}{ccc|c} & R_x & & T_x \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

- we can split the equation into rotation and translation

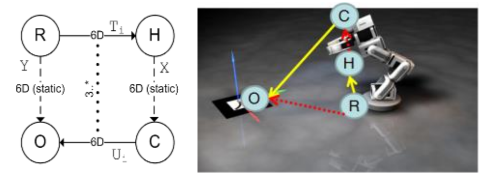
$$\begin{cases} R_a R_x = R_x R_b \\ R_a T_x + T_a = R_x T_b + T_x \end{cases}$$

3.2 Hand-Eye Calibration

Solution Strategies

- Solve for rotation first and determine translation
  - Tsai, R.Y., Lenz, R.K.
    Real Time Versatile Robotics Hand/Eye Calibration using 3D Machine Vision
  - Shiu, Y.C., Ahmad, S.
    Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX=XB
  - Chou, Jack C. K. and Kamel, M.,
    Quaternions Approach to Solve the Kinematic Equation of Rotation of a Sensor-Mounted Robotic Manipulator
- Solve for rotation and translation simultaneously
  - K. Daniilidis,
    Hand-eye calibration using dual quaternions

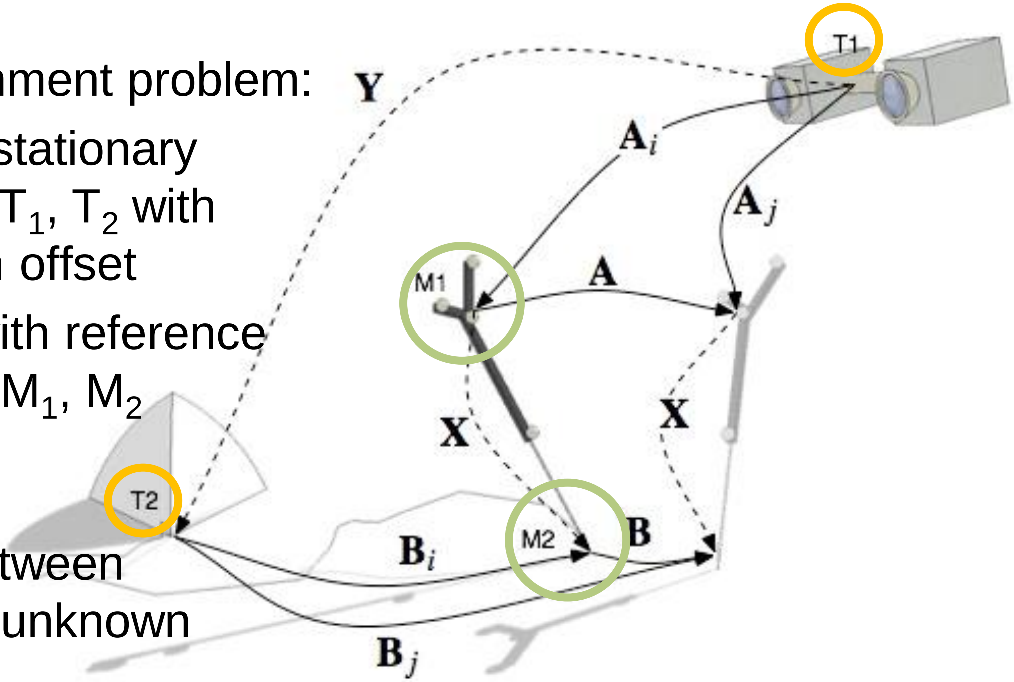
3.2 Hand-Eye Calibration

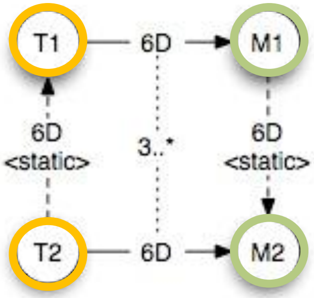


3.2 Hand-Eye Calibration

Applications

- Tracker alignment problem:
  - Several stationary trackers T_1, T_2 with unknown offset
  - Target with reference markers M_1, M_2 for each sensor; offset between markers unknown
- Examples:
  - Magnetic ↔ Optical
  - Inertial ↔ Optical





Thank you!