Course Diagram

3D Transformations

Dead Reckoning

Basic Probability

Bayesian Inference

Extended Kalman/Information Filter

Projective camera geometry

Multi View Geometry

Feature Matching

Bundle Adjustment

VAN, SLAM

Graphical models

Incremental Smoothing and Mapping (iSAM)

Advanced topics (subject to progress in class)

Multi-Robot SLAM & VAN

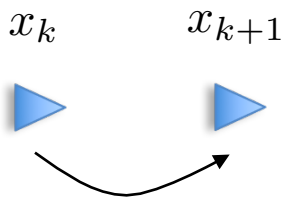
Belief Space Planning

Objectives of this Lecture

- Get familiar with inertial and dead reckoning navigation
- Provide high-level overview of common approaches

Dead Reckoning - Concept

- Use sensors to estimate motion relative to previous time
- Calculate new pose (state) from previous pose and estimated motion



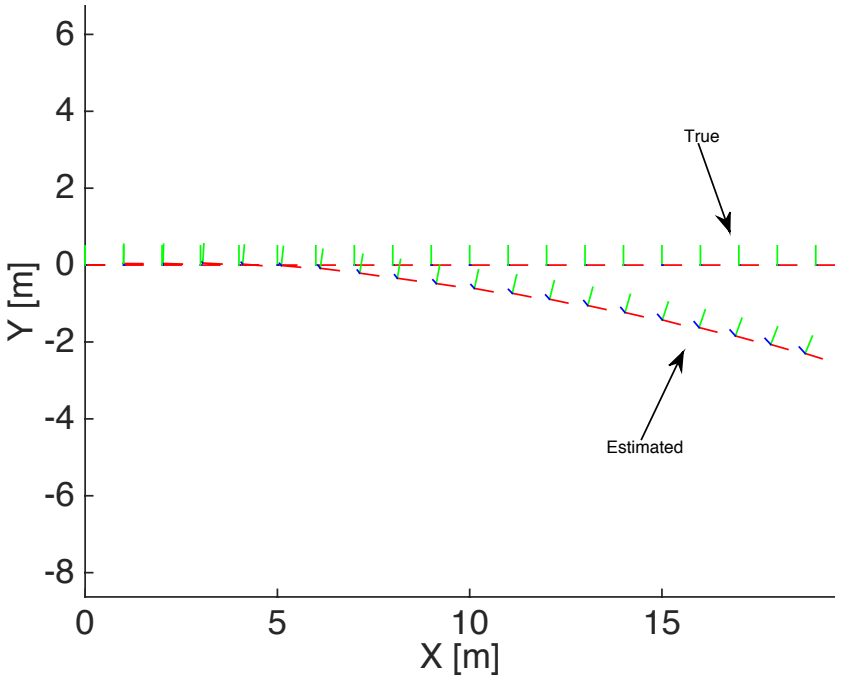
Poses: $x_{k+1} = x_k \oplus T_{k,k+1}$

More generally, navigation state: $x_{k+1} = f\left(x_k, u_k\right)$

Question - what happens over time?

Dead Reckoning - Concept

- All approaches
  - Calculate **noisy** displacement relative to previous state
  - Drift over time (i.e. develop error)
  - Performance depends on sensor quality



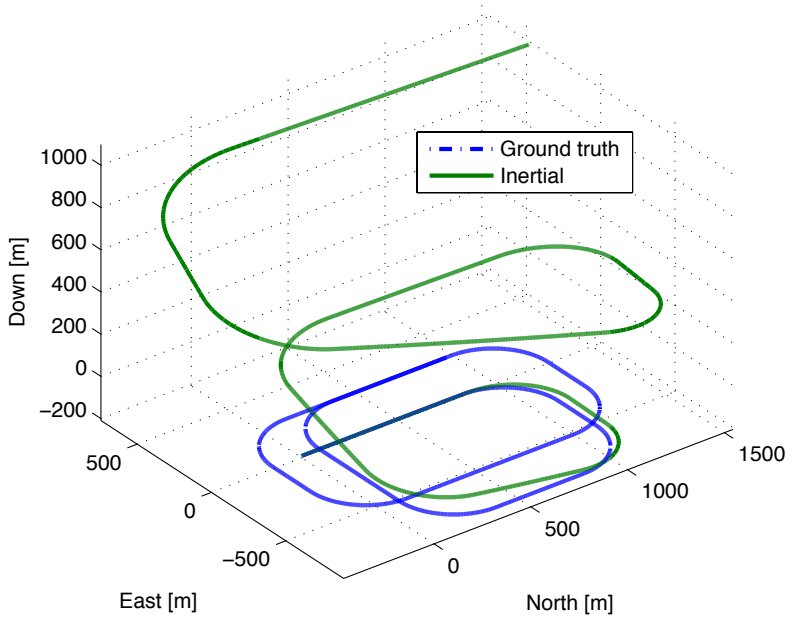
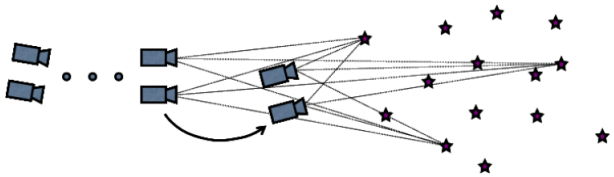


Image from Indelman12fusion

Dead Reckoning - Concept

Common odometry sensors/methods

- Inertial sensors (IMU)
- Wheel odometry
- Visual odometry
- Laser odometry (e.g. via ICP)
- …



Inertial Measurement Unit (IMU)

IMU Xsens MTi-G

- Coordinate frames:
  - {I}: IMU frame
  - {G}: Global frame
- IMU measurements: noisy, measured in IMU frame
- Accelerometers:

$$a_m(t) = R_G^I\left(a^G(t) - g^G\right) + n_a(t)$$

  - Measure specific force
  - Describes acceleration without gravity
- Gyroscopes:

$$\omega_m(t) = \omega(t) + n_g(t)$$

  - Measure angular velocity

Time Evolution of Navigation State

- Time evolution of the navigation state: $\dot{x} = h_c\left(x, a_m, \omega_m\right)$

Position: $\dot{p}_I^G\left(t\right) = v_I^G\left(t\right)$

Velocity: $\dot{v}_I^G\left(t\right) = a^G\left(t\right)$

Orientation: $\dot{R}_I^G\left(t\right) = R_I^G\left(t\right)\Omega\left(\omega\right)$ $\quad \Omega\left(\omega\right) = \left[\omega^I\right]_\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$

- Discrete domain $\quad x_{k+1} = h\left(x_k, z_k^{IMU}\right)$
  - Calculated by numerical integration
  - Euler method, Runge-Kutta etc.

Inertial Navigation System (INS)

- Diagram - inertial navigation (strapdown)
  - Accounts for Earth rotation
  - Full details in navigation systems course (086759)

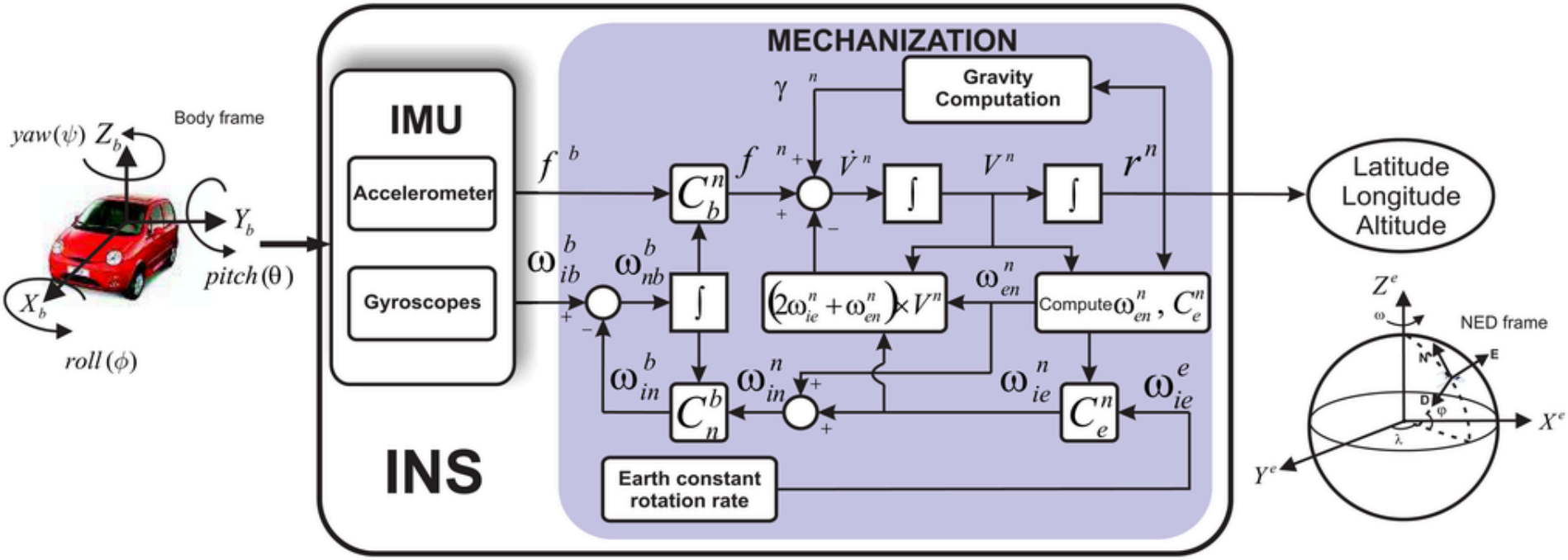


Image from: Schultz, C.E. INS and GPS Integration. M.Sc. Thesis, Technical University of Denmark, July 2006

Inertial Navigation System (INS)

- Inertial navigation drifts over time
  - Cannot be used for long time without aiding from other sensors
  - Unless IMU sensors are very good (expensive!)

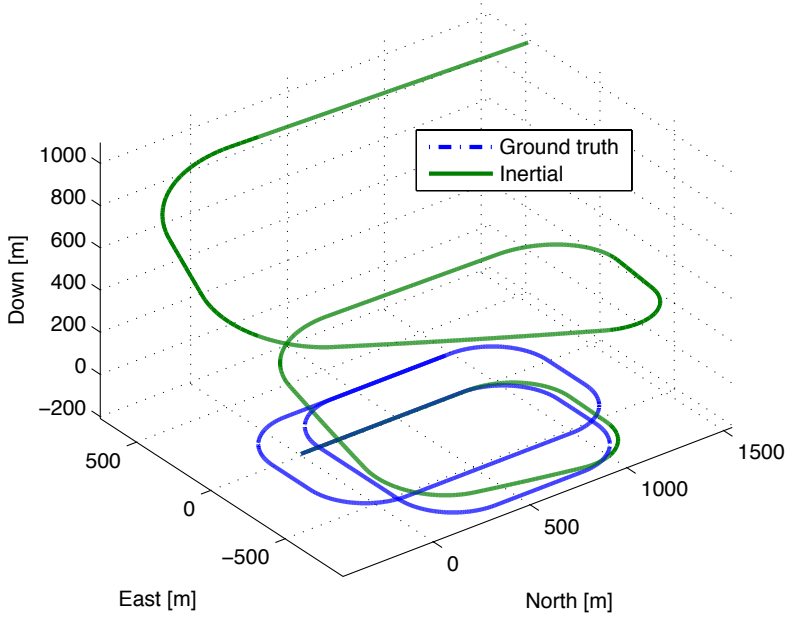
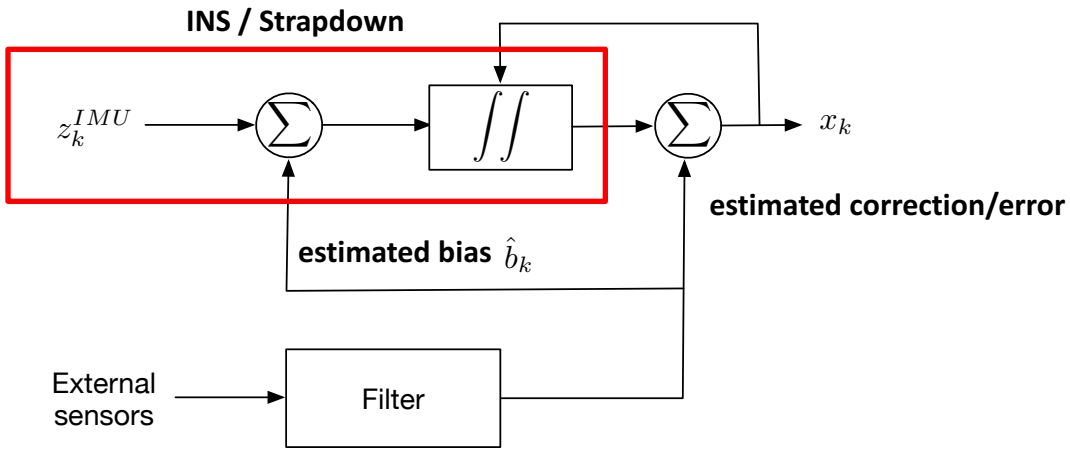


Image from Indelman12fusion

Navigation-Aiding Concept

- Use additional sensors to
  - Correct inertial navigation solution
  - Estimate IMU error model (online calibration)

- Most common: GPS



Navigation-Aiding Concept

- IMU measurements (from previous slides):

$$\omega_m(t) = \omega(t) + n_g(t)$$

$$a_m(t) = R_G^I\left(a^G(t) - g^G\right) + n_a(t)$$

- Noise is not Gaussian
- Need to model IMU error, such that residual error can be assumed zero-mean Gaussian
- Most common: bias

$$\omega_m(t) = \omega(t) + b_g(t) + n_g(t)$$

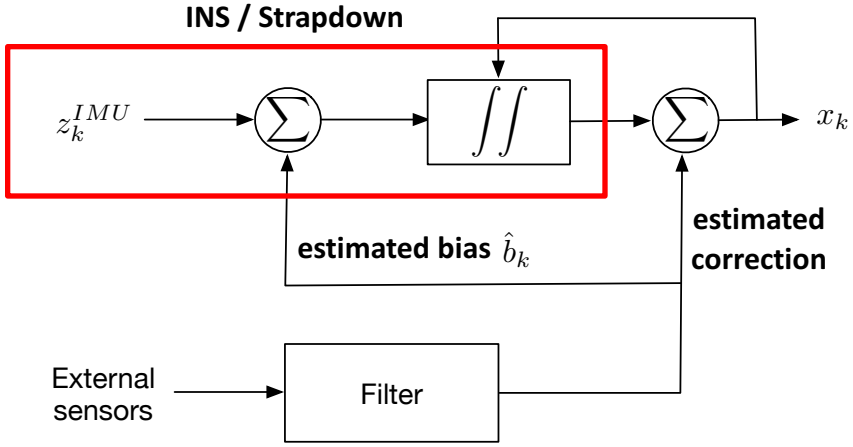
$$a_m(t) = R_G^I\left(a^G(t) - g^G\right) + b_a(t) + n_a(t)$$

Navigation-Aiding Concept

- Navigation state with basic IMU calibration parameters:

$$x \doteq \begin{bmatrix} p^T & v^T & \Psi^T & b_a^T & b_g^T \end{bmatrix}^T \in \mathbb{R}^{15}$$

accelerometer & gyroscope bias



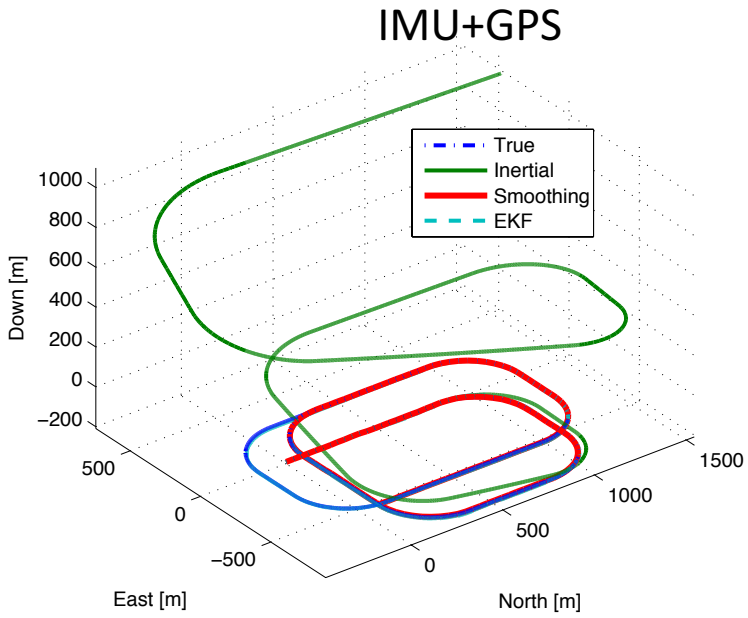


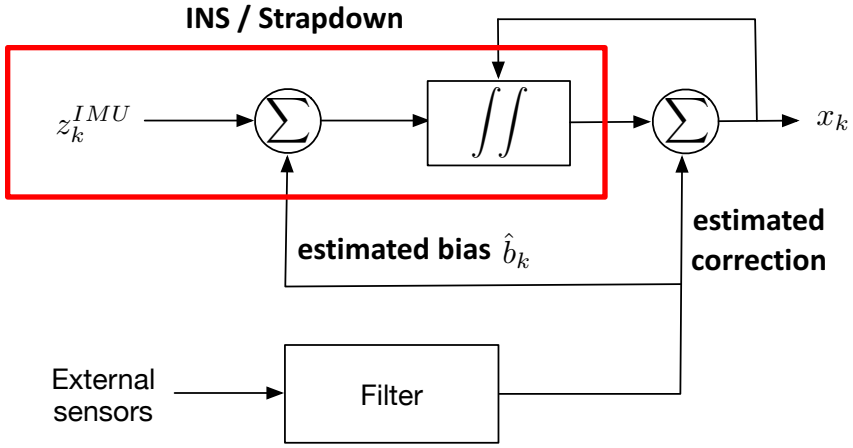
Image from Indelman12fusion

Navigation-Aiding Concept

- Navigation state with basic IMU calibration parameters:

$$x \doteq \begin{bmatrix} p^T & v^T & \Psi^T & b_a^T & b_g^T \end{bmatrix}^T \in \mathbb{R}^{15}$$

accelerometer & gyroscope bias



IMU+GPS

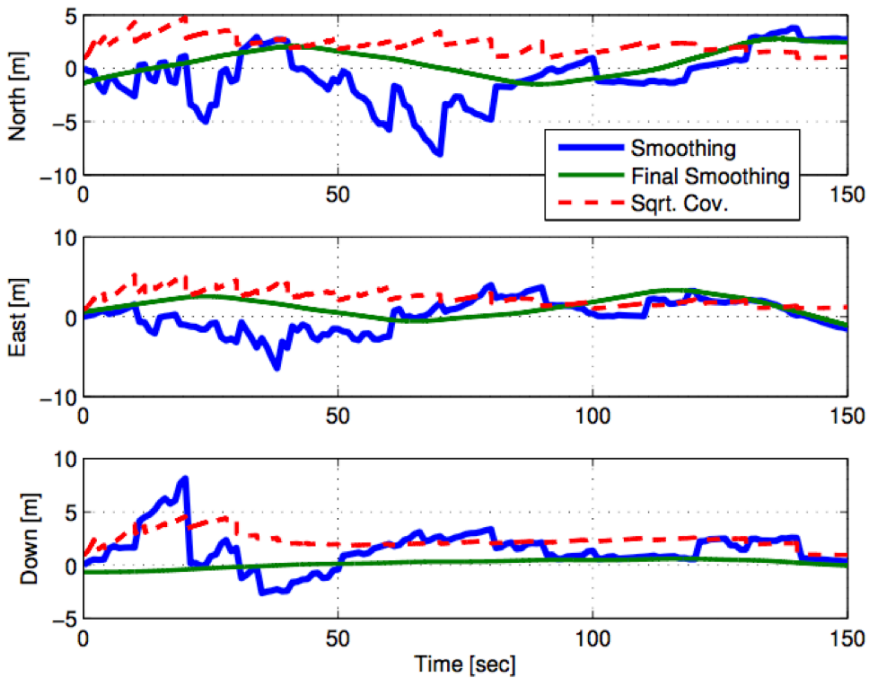
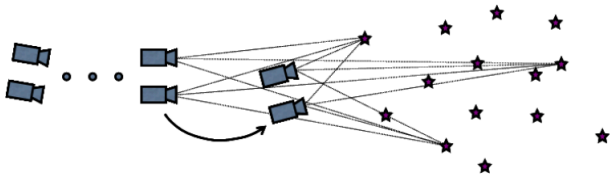


Image from Indelman12fusion

Dead Reckoning - Concept

Common odometry sensors/methods

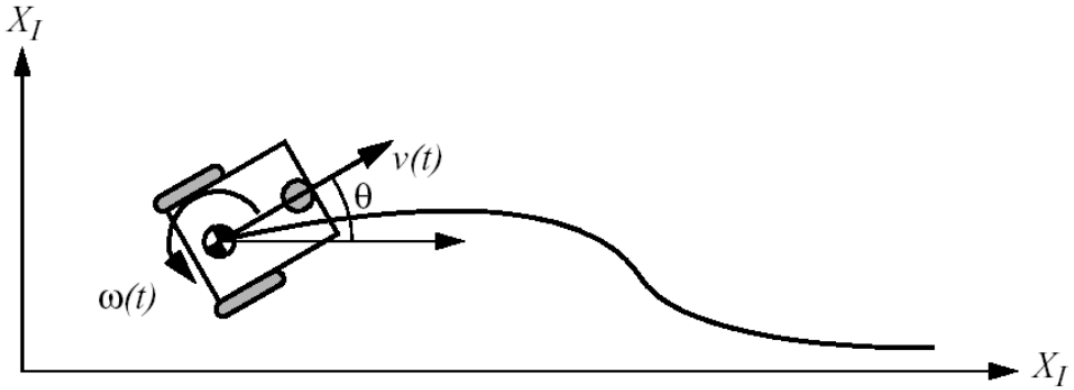
- Inertial sensors (IMU)
- Wheel odometry
- Visual odometry
- Laser odometry (e.g. via ICP)
- …

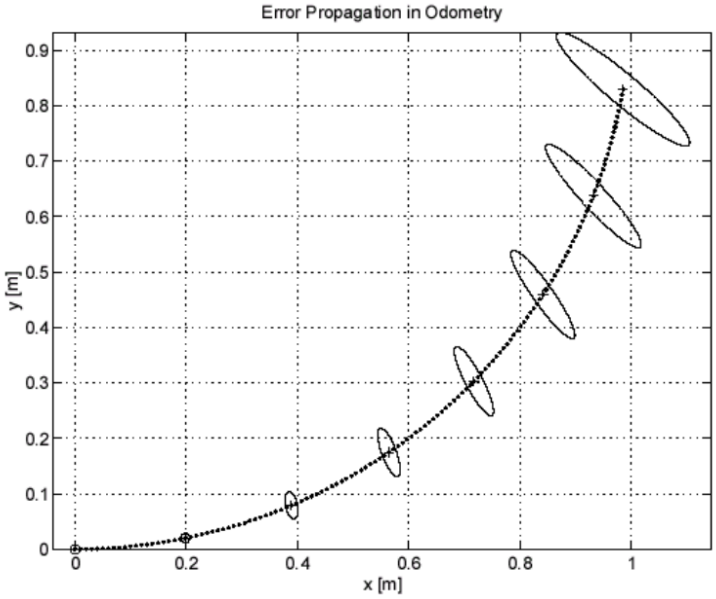


Wheel Odometry

- Use wheel sensors to update position
- Straightforward to implement
- As earlier, errors accumulate, unbounded
- Numerous error sources (e.g. wheel slippage)

Pioneer 3DX





VISUAL ODOMETRY

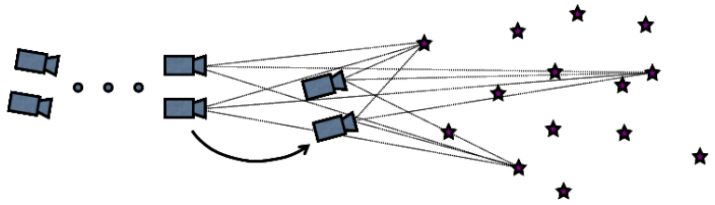
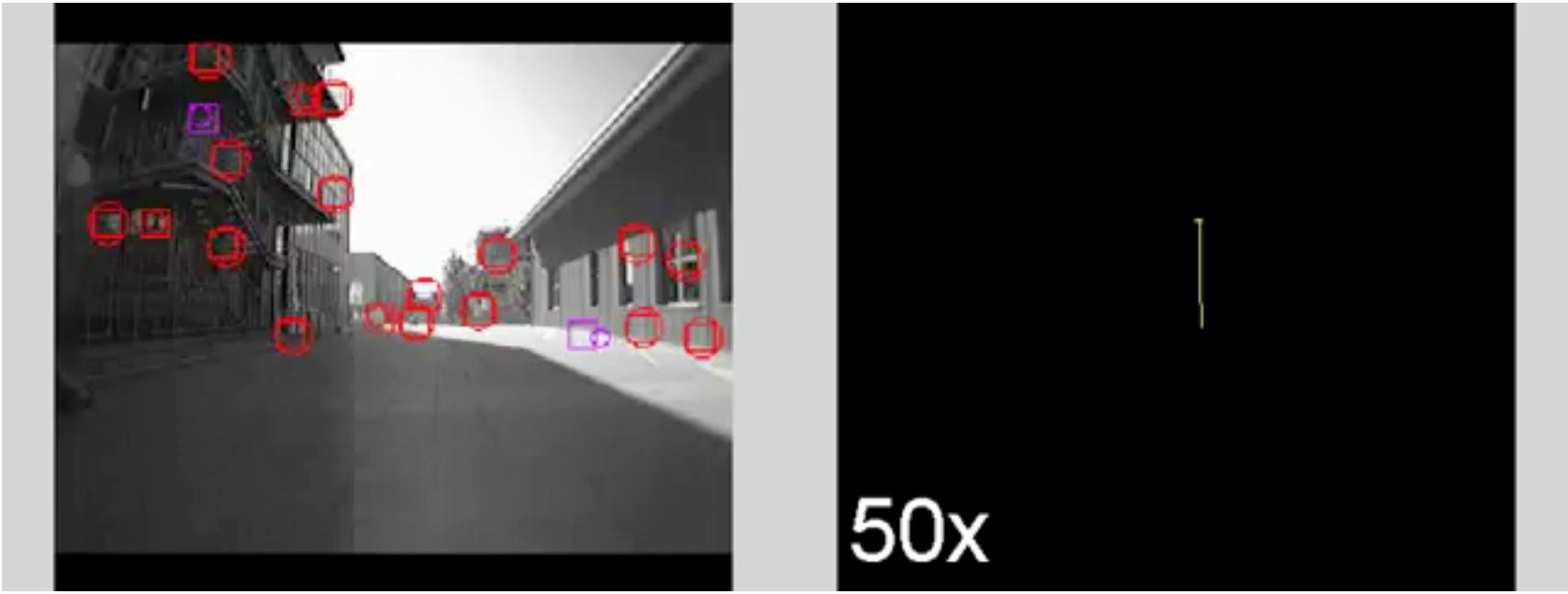


Image from: “FrameSLAM: from Bundle Adjustment to Realtime Visual Mappping”, TRO 2008

Visual Odometry (VO)

- VO: Estimate platform motion (e.g. relative pose) from captured imagery
  - More precisely, from changes that motion induces on images captured by onboard camera(s)



Video by Javier Civera

Working Principle (High Level)

- Stereo VO
  - Estimate 3D points from current stereo frame
  - Estimate pose of the next stereo frame using estimated 3D points

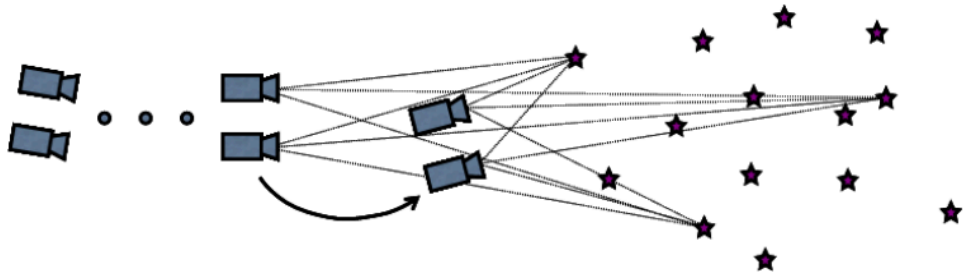


Image from: “FrameSLAM: from Bundle Adjustment to Realtime Visual Mappping”, TRO 2008

Working Principle (High Level)

- Monocular VO (single camera)
  - Extract and match image features
  - Estimate camera motion using multiple view geometry (essential matrix, homography)
  - Translation is estimated up to scale

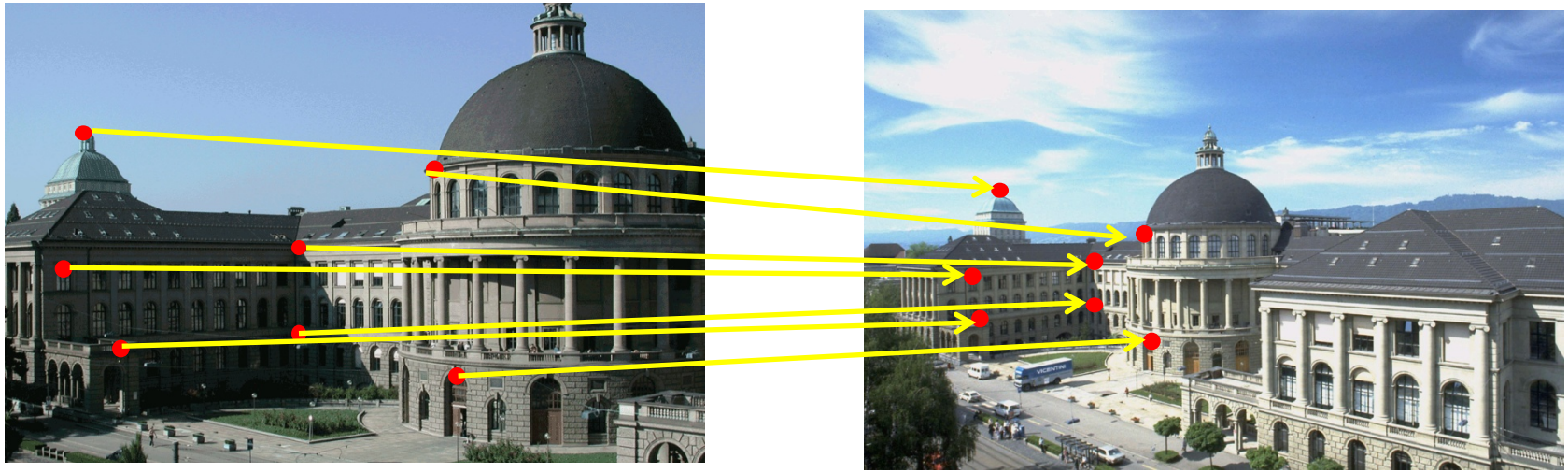
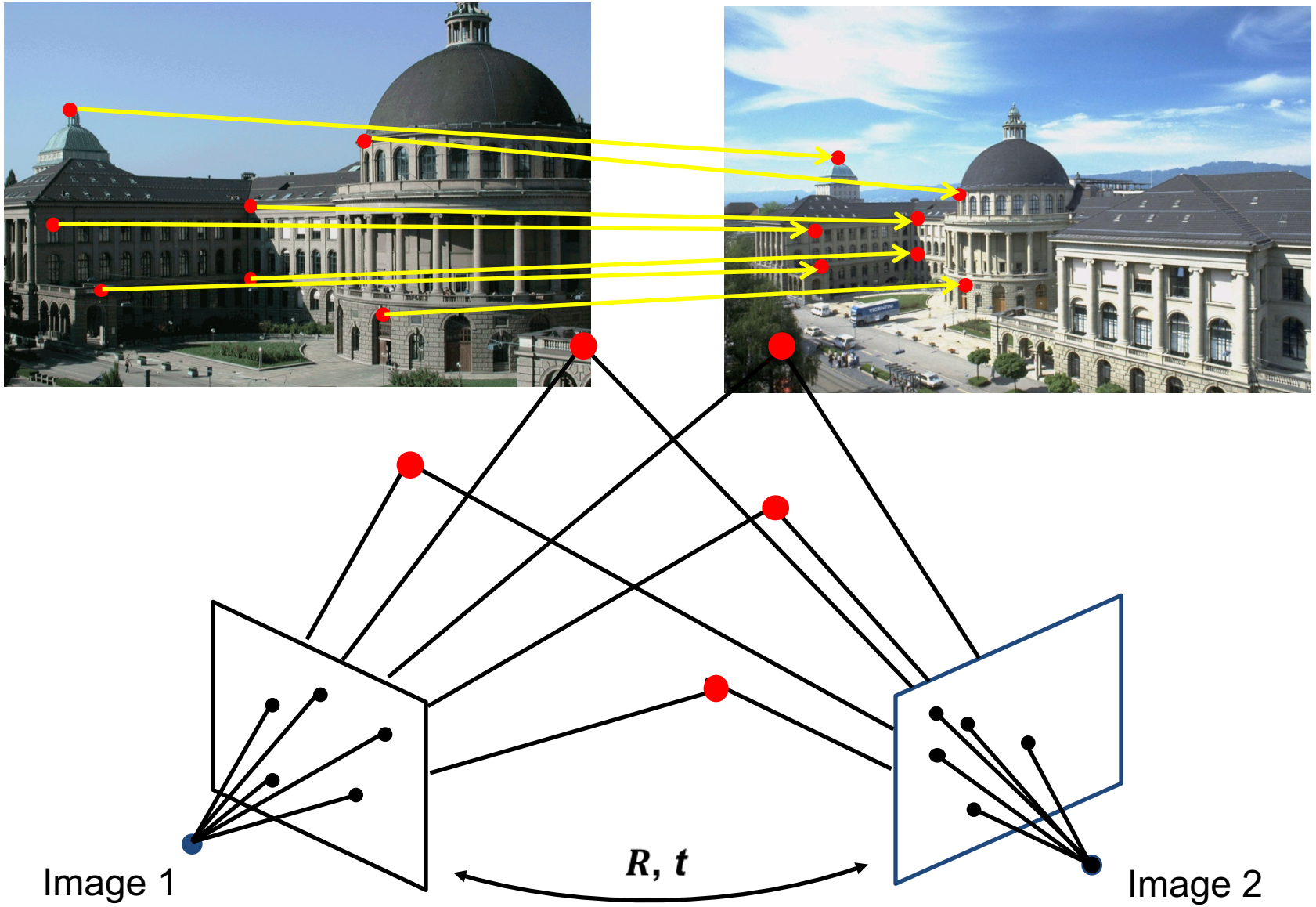


Illustration by Davide Scaramuzza

Working Principle (High Level)



Slide by Davide Scaramuzza

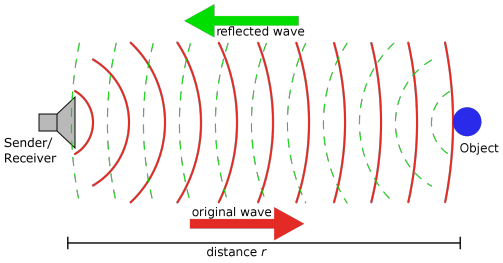
Visual Odometry (VO)

- Drift still accumulates
- More accurate than wheel odometry (not affected by wheel slip)
- Complementary to additional methods (IMU, wheel odometry, GPS)
- Important in GPS-denied environments
- Can be combined with SLAM (more details in a few lectures)

- Extensively used in numerous applications in the last ~20 years
  - e.g., NASA's Spirit and Opportunity Mars-exploration rovers

Additional Sensors

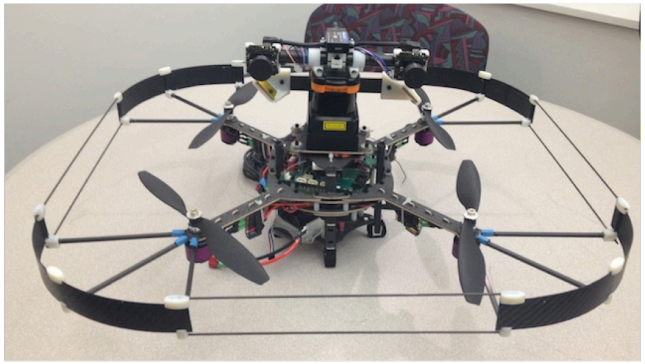
- Sonar (e.g. underwater)

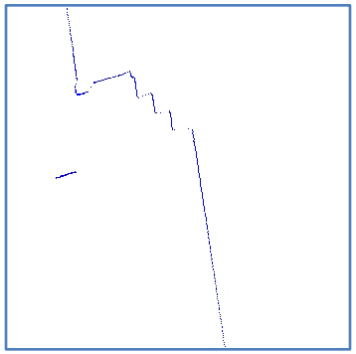


- Laser range finder

SICK

Hokuyo



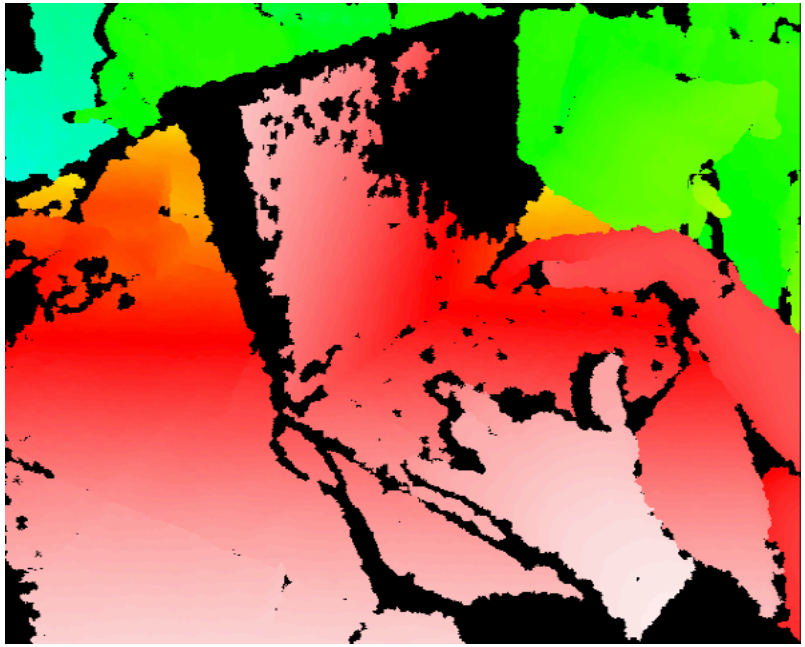


Additional Sensors

- Structured Light
  - Project a known pattern in IR
  - Calculate depth from simple geometry
  - Known as RGB-D sensors

Kinect

Projected IR pattern



Depth map

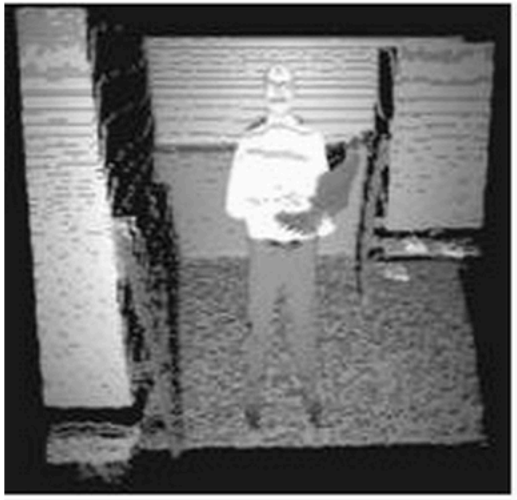
Source: wikipedia

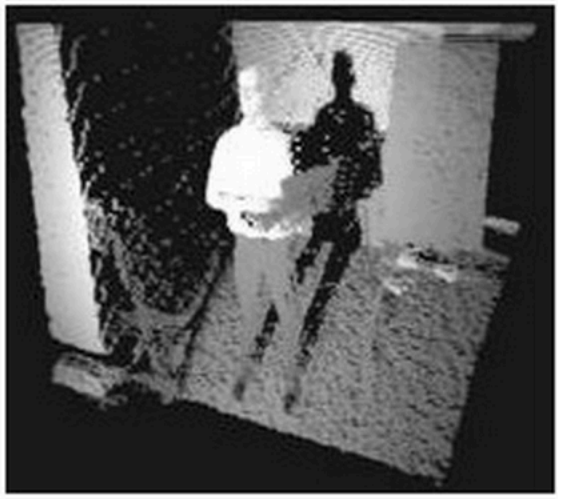
Additional Sensors

- Time of Flight (TOF) cameras
  - Distance is calculated by measuring time-of-flight of a light signal between camera and subject, for each point on image
  - e.g. Kinect 2.0, …

Kinect 2.0

- Provide point clouds – how to match?

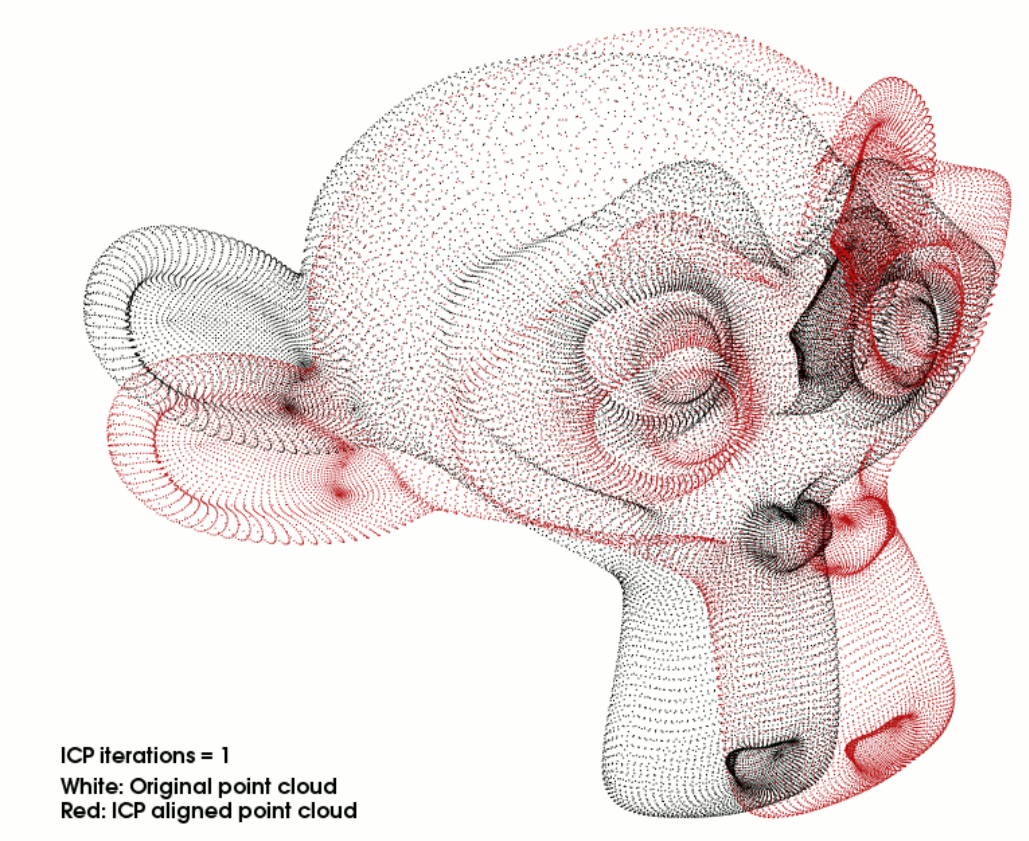




Images from http://www.geometh.ethz.ch/research/range

Iterative Closest Point (ICP)

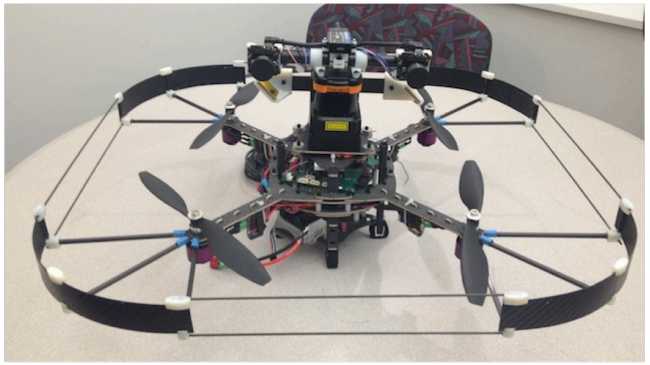
- Algorithm to minimize distance between two point clouds

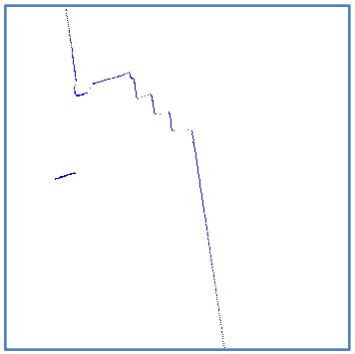


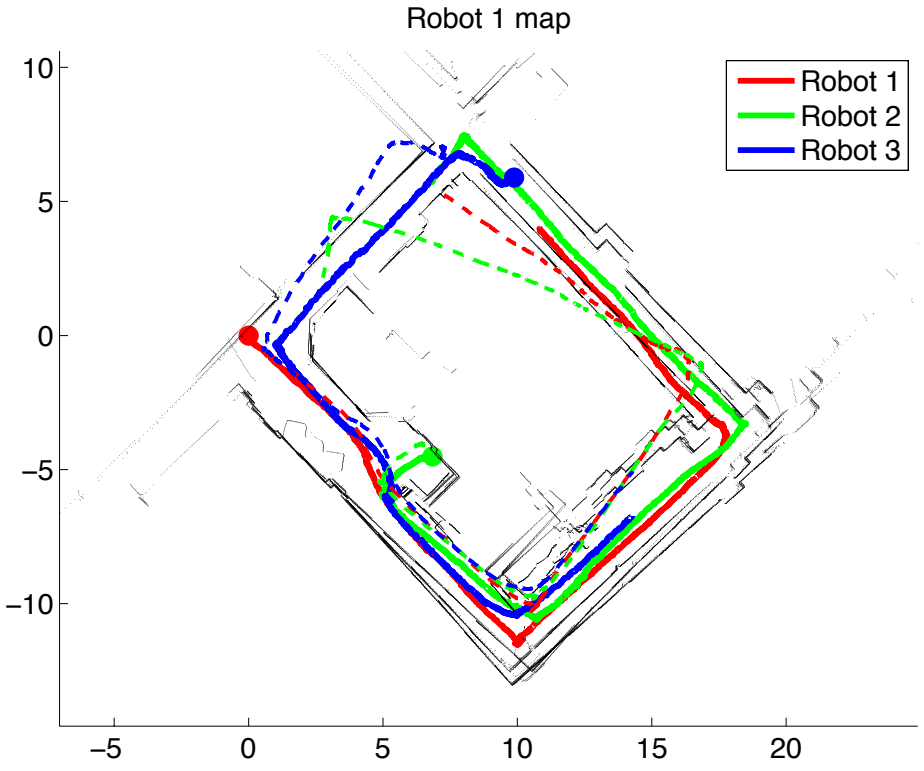
Animation by PCL, from http://pointclouds.org/documentation/tutorials/interactive_icp.php

Iterative Closest Point (ICP)

- Still frame-to-frame motion estimation
- Drift develops over time







Images from Indelman16csm

Dead Reckoning - Summary

- We described in high-level: inertial navigation, wheel odometry, and monocular and stereo visual odometry
- All methods perform frame-to-frame motion estimation
  - Vary in frequency, error sources etc.
- Drift accumulates over time
- How to improve estimation accuracy?

Next Lecture(s) …

3D Transformations | Dead Reckoning | Basic Probability

Bayesian Inference | Extended Kalman/Information Filter

Projective camera geometry | Multi View Geometry | Feature Matching

Bundle Adjustment | VAN, SLAM | Graphical models

Incremental Smoothing and Mapping (iSAM)

Advanced topics (subject to progress in class)

Multi-Robot SLAM & VAN | Belief Space Planning