上海科技大学

ShanghaiTech University

INTRODUCTION TO 3D MAPPING AND SLAM

Sören Schwertfeger / 师泽仁

https://robotics.shanghaitech.edu.cn

Outline

- Introduction
- Coordinate System
- Sensors
- Scan Matching via ICP
- Localization, Mapping and SLAM
- Example: Jacobs Plane Mapping

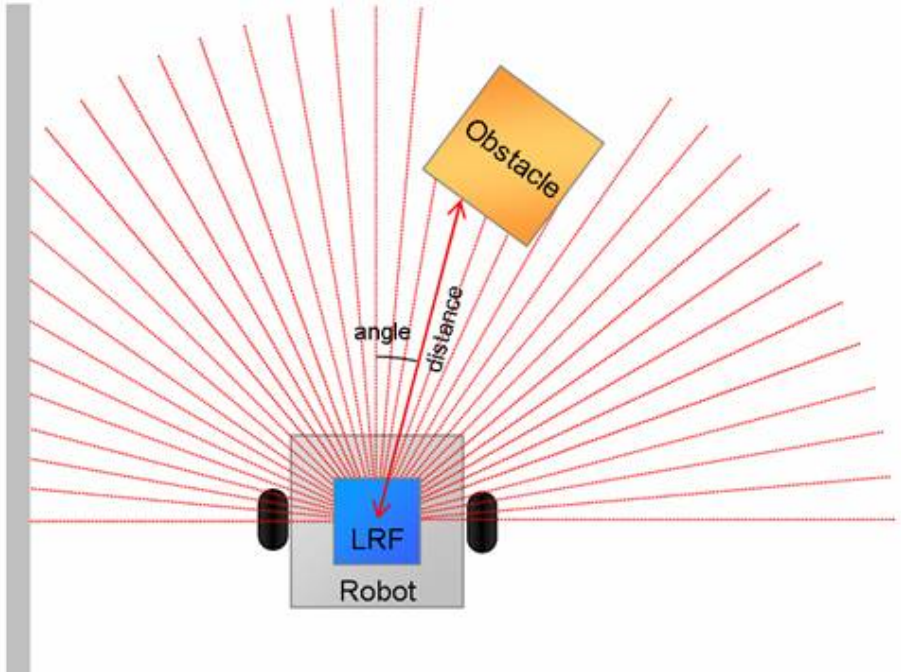
With lots of material from ”Autonomous Mobile Robots” ETH Zürich
R. Siegwart & D. Scaramuzza

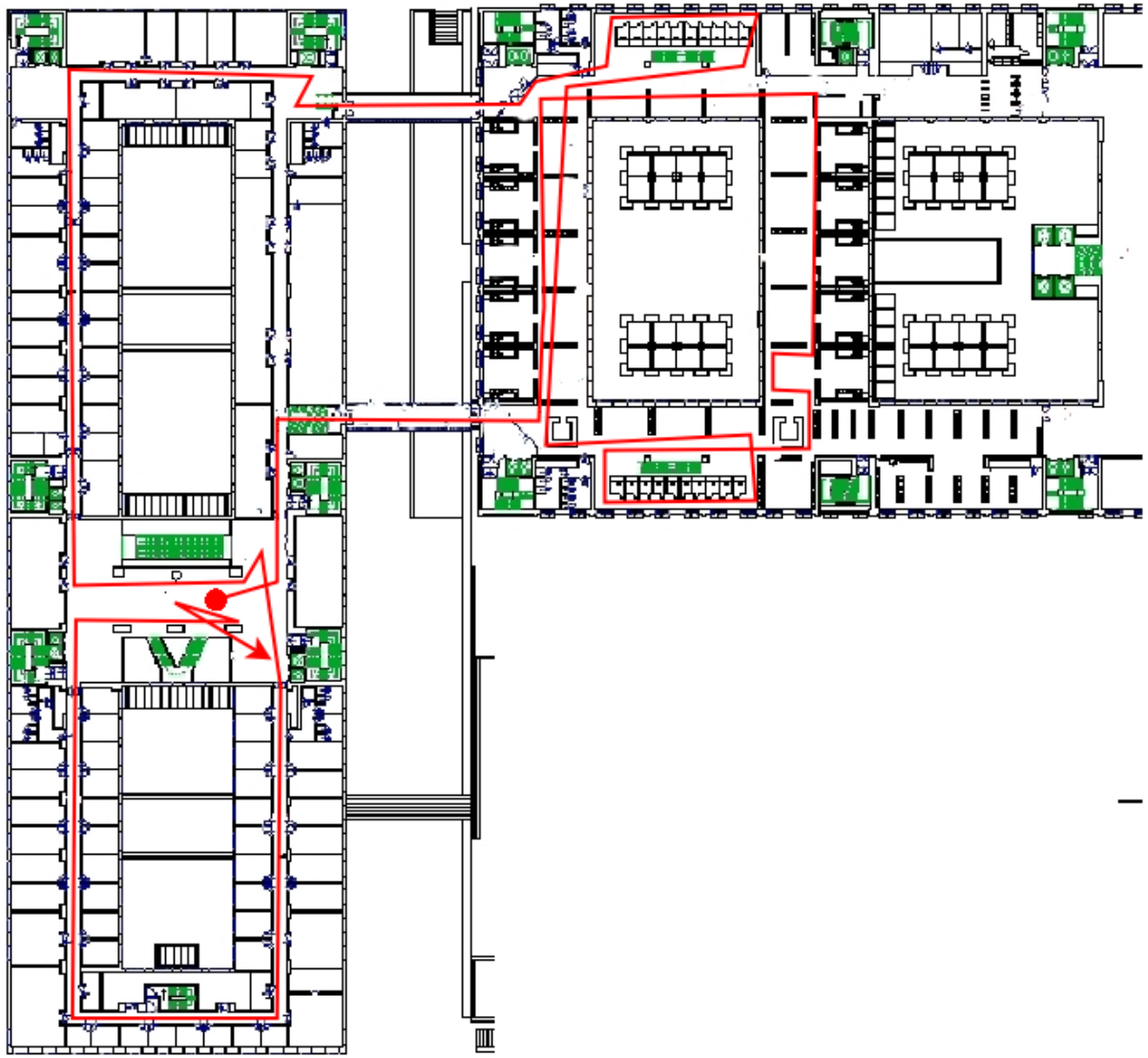
Mapping: Important Robotic Capability

- Applications:
  - Localization: Know where the robot is
  - Navigation: Know about the obstacles surrounding the robot
  - Planning: Plan paths to distant locations
  - Exploration: Explore unknown areas, search for objects
  - Semantic mapping: Know where objects of interest are located
  - Coordination: Coordinate a team of (heterogeneous) robots to cooperatively perform tasks
  - Human Robot Interaction: Communicate about locations

Where am I? Map **Localization**

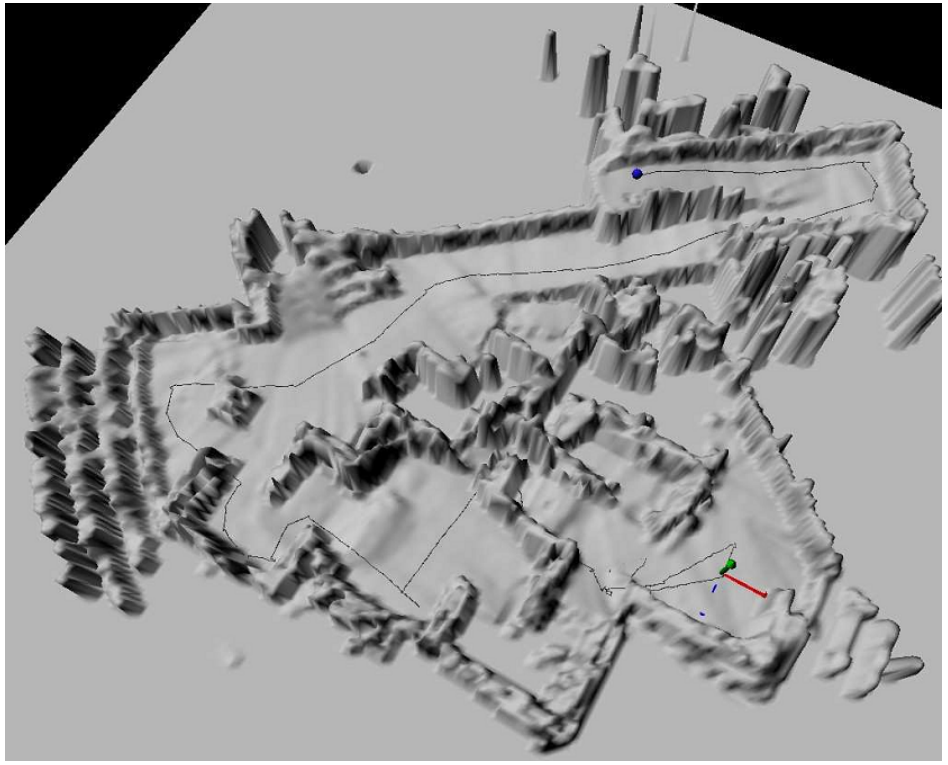
- Use sensor data (LRF) to estimate your position given a map of the environment
- Scan matching (Iterative Closest Point (ICP))





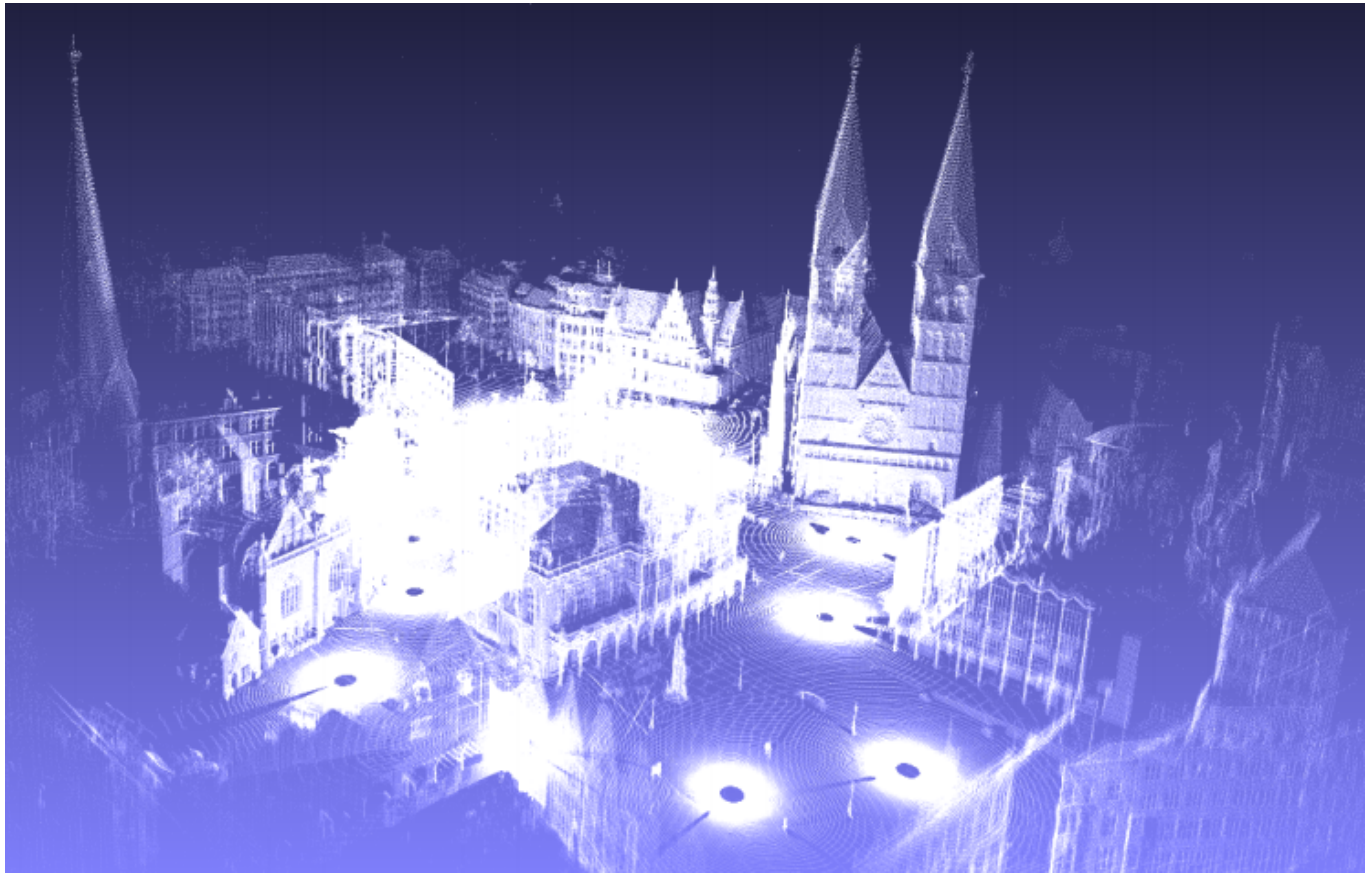
Don’t have a map? Build it : **Mapping**

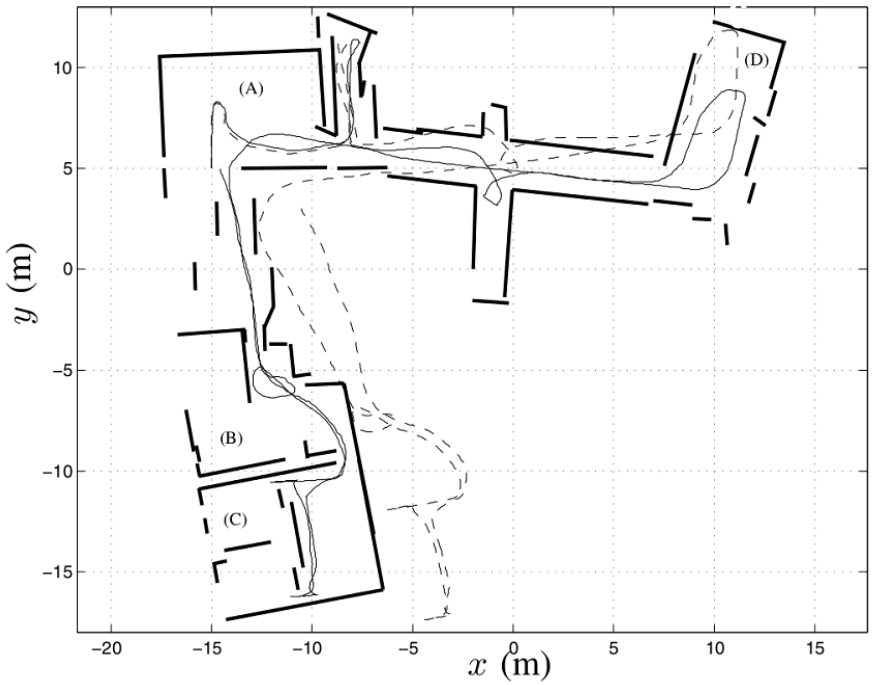
- Take a sensor scan, transform to global frame, put in map
- Errors in localization => Errors in map
- 2D (right), Elevation Map (2.5 D) (below)

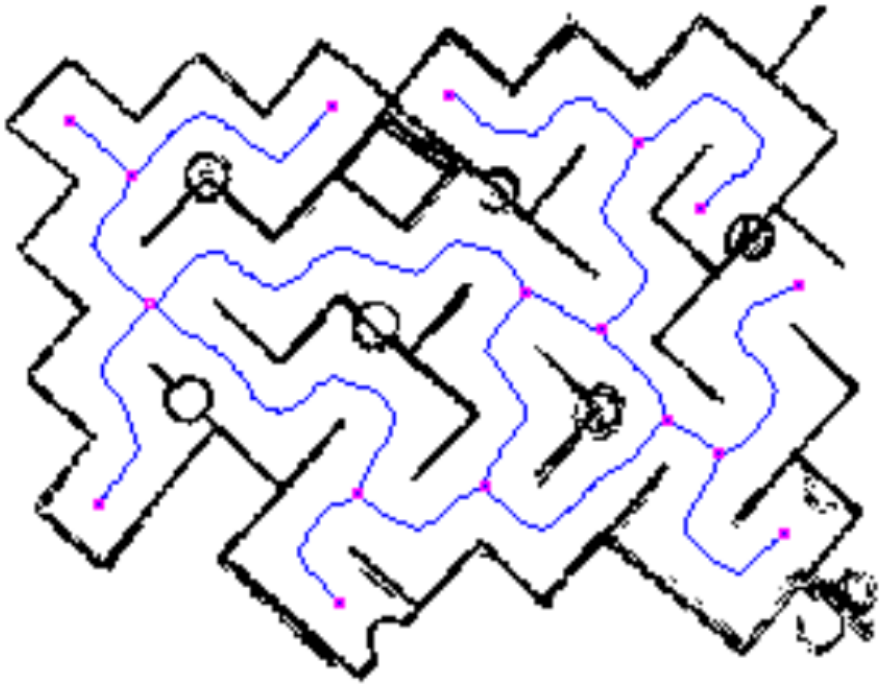


Map representations

- Line Segments, Topology
- 3D Point clouds, …







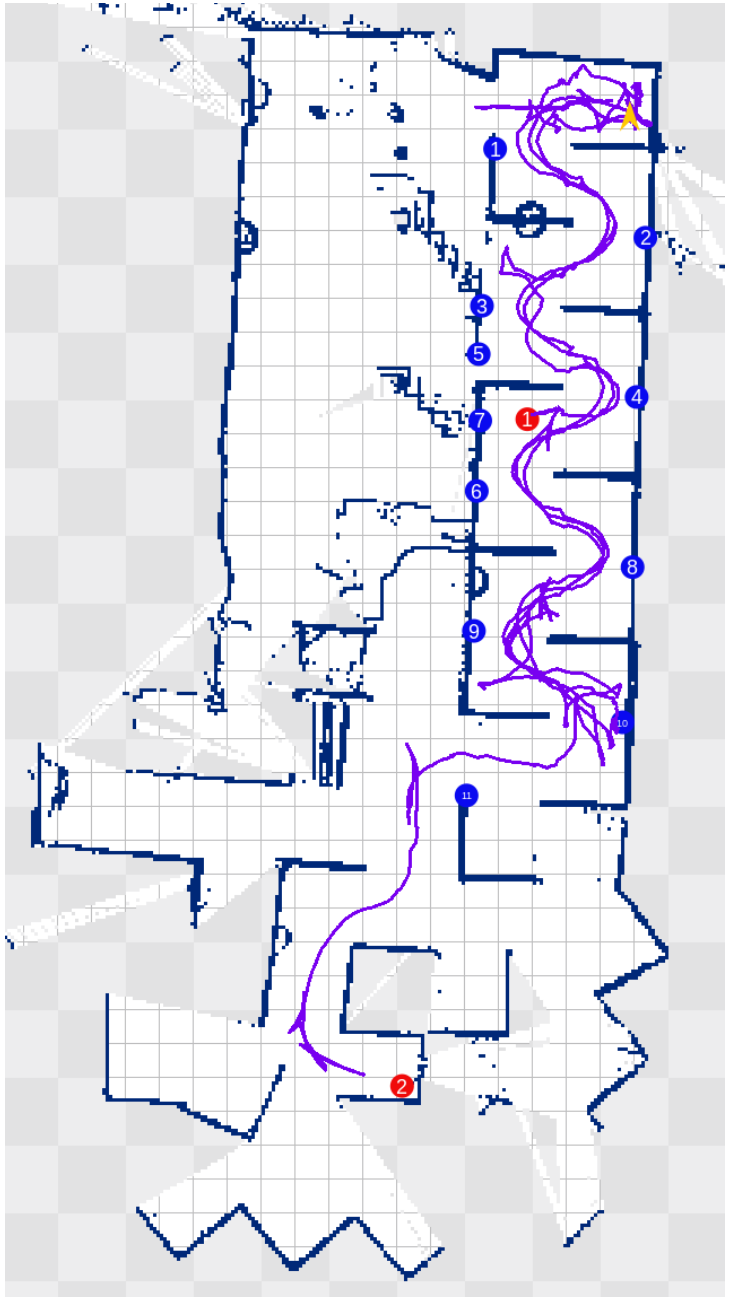
SLAM

- **S**imultaneous **L**ocalization **A**nd **M**apping (SLAM)
- Chicken-and-egg problem

- Different approaches:
  - Extended Kalman filter (EKF)
  - Expectation Maximization (EM)
  - Monte Carlo (Particle Filters)
  - Graph-based algorithms

- Good solutions in 2D and structured environments
  - Hector SLAM (TU-Darmstadt):
    - Open source software for ROS (Robot Operating System)
    - Developed for RoboCup Rescue – universal usage

- In 3D and unstructured terrain: still open



Mapping/ SLAM in other Contexts

- 3D Modelling
- 3D Reconstruction and Modelling
- Structure from Motion
- Bundle Adjustment

- Difference to SLAM often:
  - SLAM is real time
  - SLAM can use other data: Odometry, control input, IMU…

- Autonomous mobile robots move around in the environment. Therefore **ALL** of them:
  - They need to know **where** they **are.**
  - They need to know **where** their **goal** is.
  - They need to know **how** to get there.

- Where am I?
  - Global Positioning System: outdoor, meters of error
  - Guiding system: (painted lines, inductive guides), markers
  - Model of the environment (Map), Localize yourself in this model
    - Build the model online: Mapping
    - Localization: determine position by comparing sensor data with the map
    - Do both at the same time: Simultaneous Localization and Mapping (SLAM)

- Autonomous mobile robots move around in the environment. Therefore **ALL** of them:
  - They need to know **where** they **are**.
  - They need to know **where** their **goal** is.
  - They need to know **how** to get there.

- Where is my goal?
- Two part problem:
  - What is the goal?
    - Expressed using the world model (map)
    - Using object recognition
    - No specific goal (random)
  - Where is that goal?
    - Coordinates in the map
    - Localization step at the end of the object recognition process
    - User input

- Autonomous mobile robots move around in the environment. Therefore **ALL** of them:
  - They need to know **where** they **are**.
  - They need to know **where** their **goal** is.
  - They need to know **how** to get there.

- Different levels:
  - Control:
    - How much power to the motors to move in that direction, reach desired speed
  - Navigation:
    - Avoid obstacles
    - Classify the terrain in front of you
    - Predict the behavior (motion) of other agents (humans, robots, animals, machines)
  - Planning:
    - Long distance path planning
    - What is the way, optimize for certain parameters

Most important capability

(for autonomous mobile robots)

How to get from place A to B?

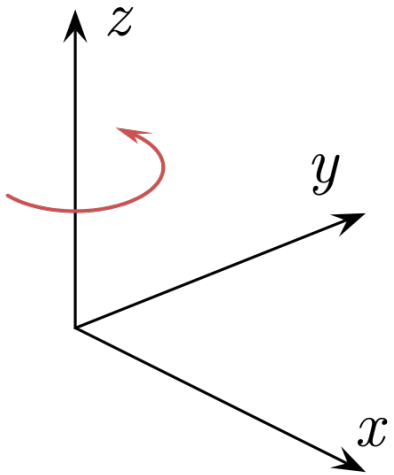
(safely and efficiently)

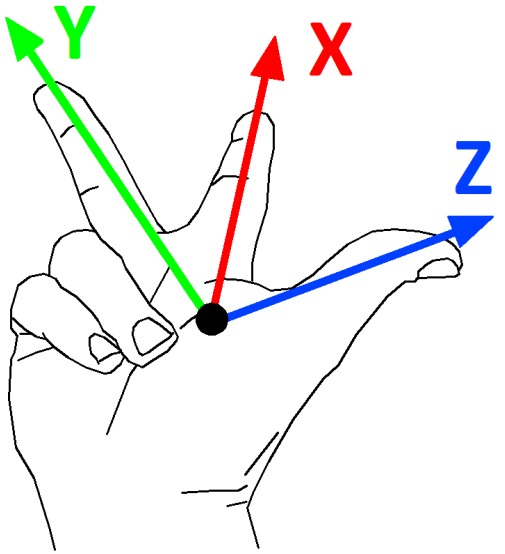
The robot needs a map to do this!

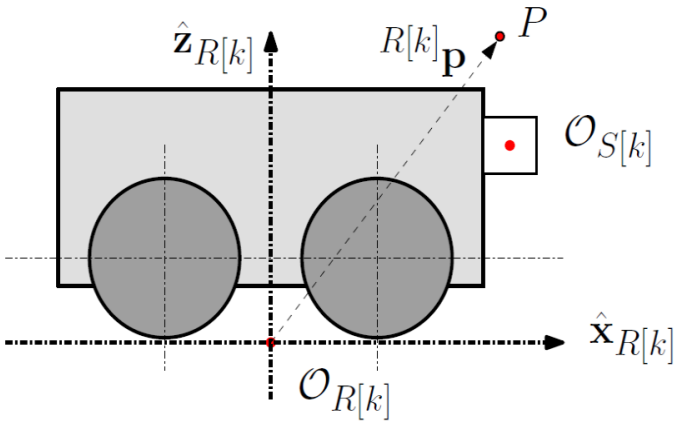
COORDINATE SYSTEM

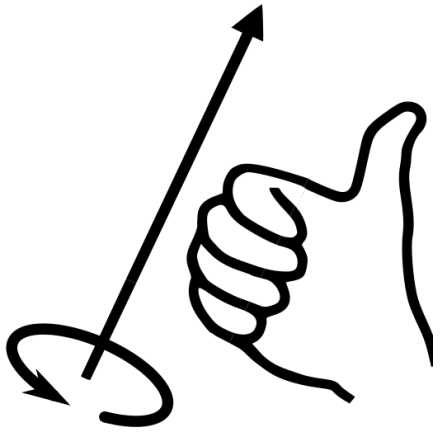
Right Hand Coordinate System

- Standard in Robotics
- Positive rotation around X is anti-clockwise
- Right-hand rule mnemonic:
  - Thumb: z-axis
  - Index finger: x-axis
  - Second finger: y-axis
  - Rotation: Thumb = rotation axis, positive rotation in finger direction
- Robot Coordinate System:
  - X front
  - Z up (Underwater: Z down)
  - Y ???



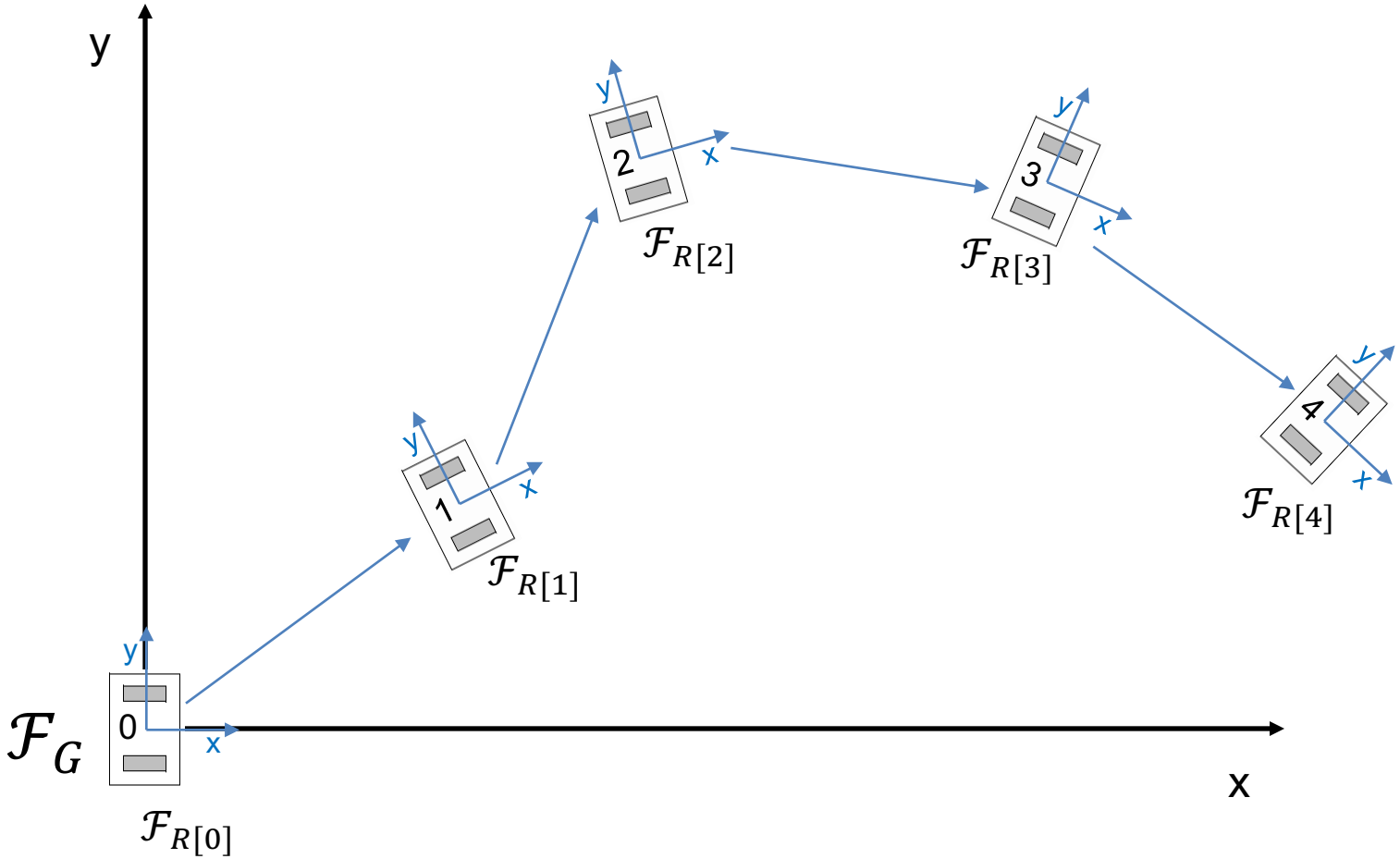






Right Hand Rule http://en.wikipedia.org/wiki/Right-hand_rule

Odometry



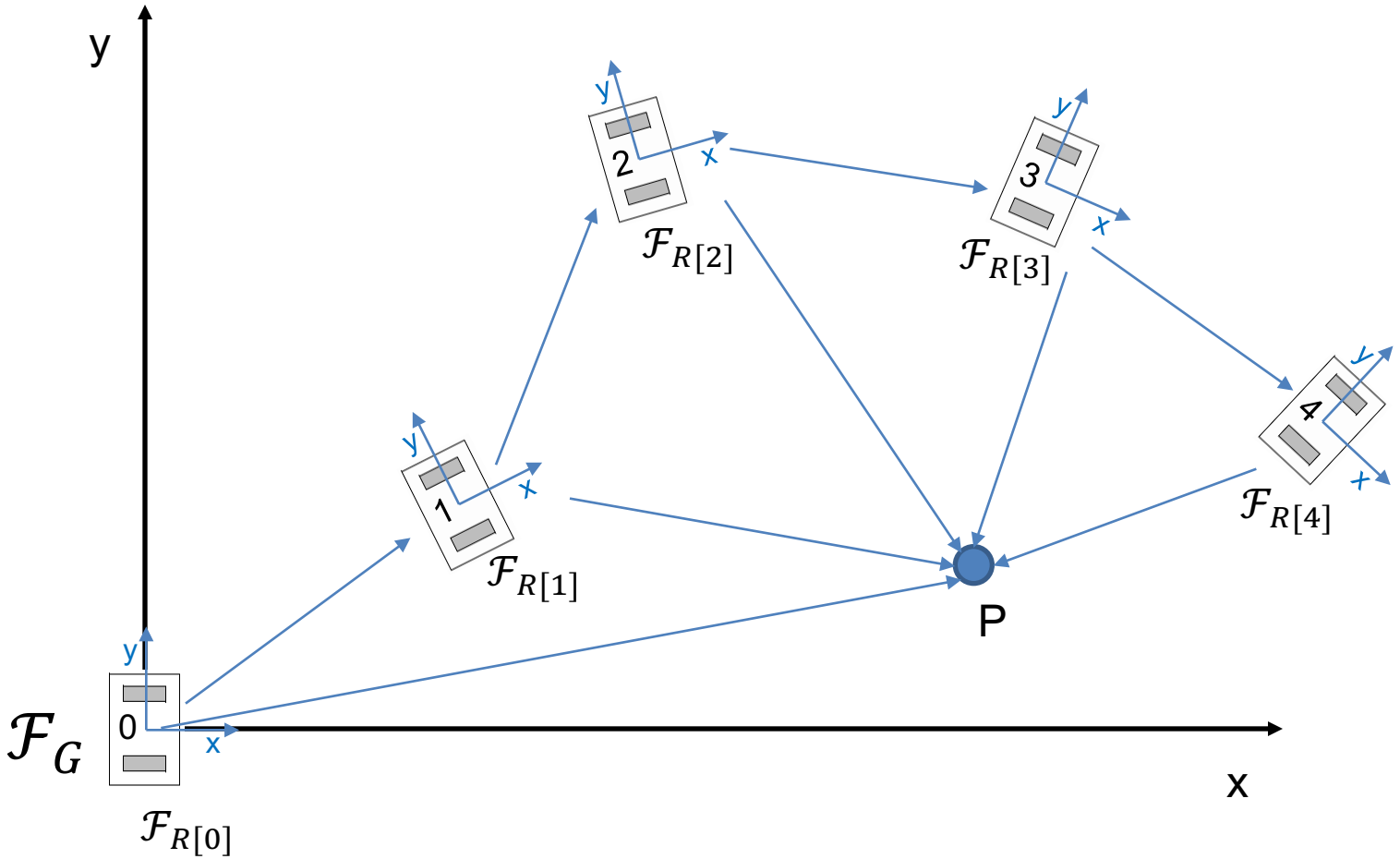
With respect to the robot start pose:
Where is the robot now?

Two approaches – same result:

- Geometry (easy in 2D)
- Transforms (better for 3D)

$\mathcal{F}_{R[X]}$: The ***F***rame of reference (the local coordinate system) of the ***R***obot at the time ***X***

Use of robot frames $\mathcal{F}_{R[X]}$



$\mathcal{O}_{R[X]}$: Origin of $\mathcal{F}_{R[X]}$
(coordinates (0, 0)

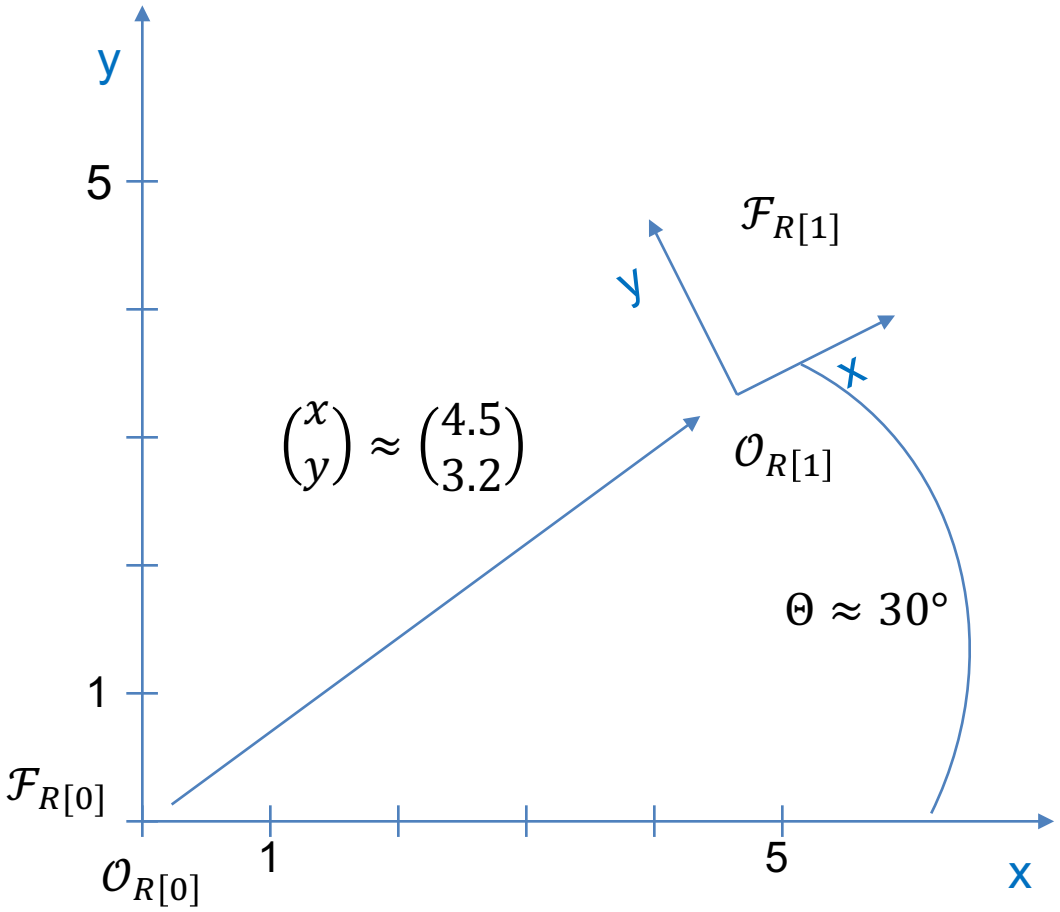
$\overrightarrow{\mathcal{O}_{R[X]}\mathrm{P}}$: position vector from $\mathcal{O}_{R[X]}$ to point P - $\begin{pmatrix} x \\ y \end{pmatrix}$

- Object P is observed at times 0 to 4
- Object P is static (does not move)
- The Robot moves

  (e.g. $\mathcal{F}_{R[0]} \neq \mathcal{F}_{R[1]}$)
- => (x, y) coordinates of P are

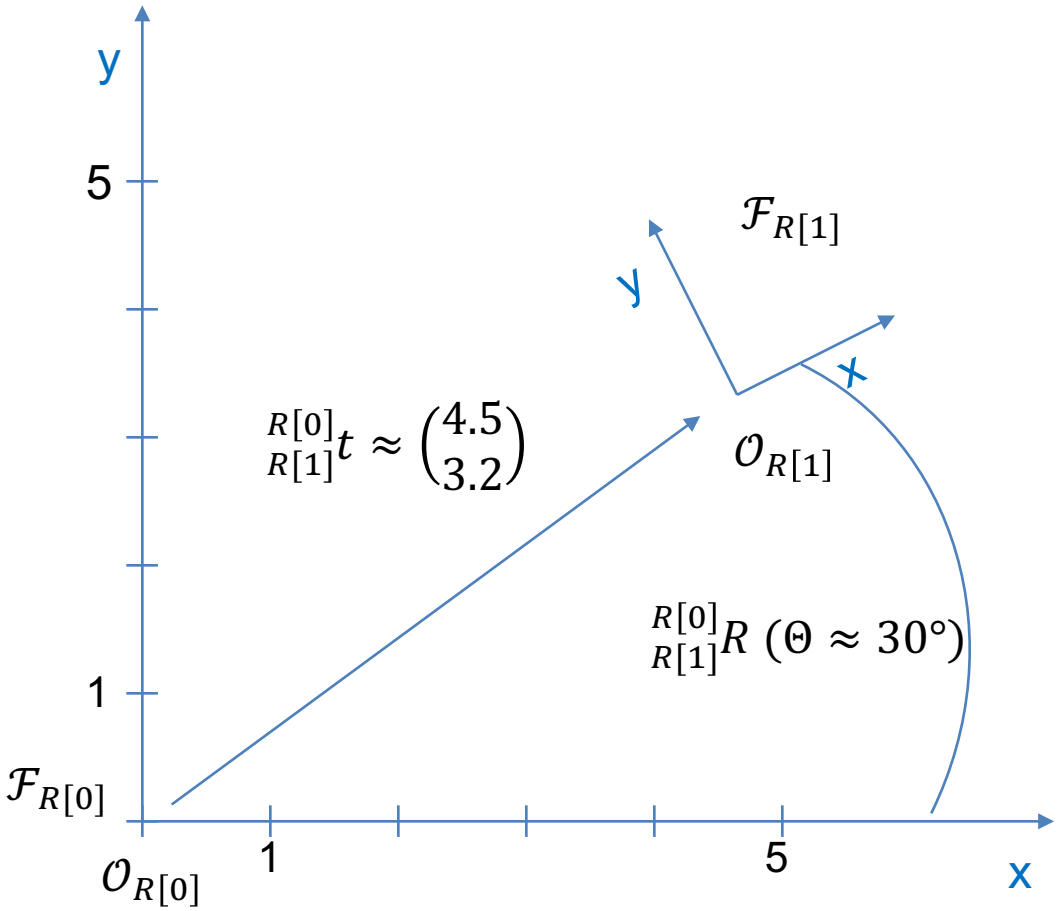
  different in all frames, for example:
  - $\overrightarrow{\mathcal{O}_{R[0]}\mathrm{P}} \neq \overrightarrow{\mathcal{O}_{R[1]}\mathrm{P}}$

Position, Orientation & Pose



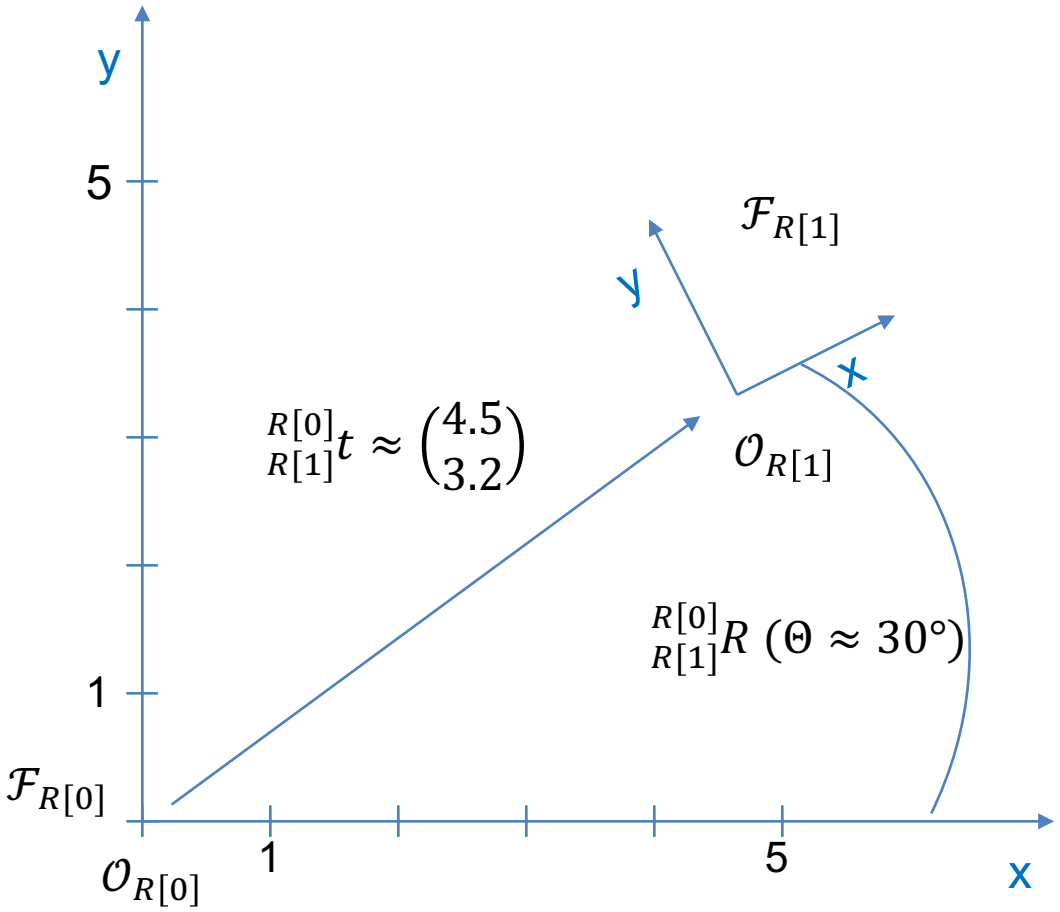
- **Position**:
  - $\begin{pmatrix} x \\ y \end{pmatrix}$ coordinates of any object or point (or another frame)
  - with respect to (wrt.) a specified frame
- **Orientation**:
  - (Θ) angle of any oriented object (or another frame)
  - with respect to (wrt.) a specified frame
- **Pose**:
  - $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$ position and orientation of any oriented object
  - with respect to (wrt.) a specified frame

Translation, Rotation & Transform



- **Translation**:
  - $\begin{pmatrix} x \\ y \end{pmatrix}$ difference, change, motion from one reference frame to another reference frame
- **Rotation**:
  - (Θ) difference in angle, rotation between one reference frame and another reference frame
- **Transform**:
  - $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$ difference, motion between one reference frame and another reference frame

Position & Translation, Orientation & Rotation



- $\mathcal{F}_{R[X]}$: Frame of reference of the robot at time X
- Where is that frame $\mathcal{F}_{R[X]}$?
  - Can only be expressed with respect to (wrt.) another frame (e.g. global Frame $\mathcal{F}_G$) =>
  - Pose of $\mathcal{F}_{R[X]}$ wrt. $\mathcal{F}_G$

- $\mathcal{O}_{R[X]}$: Origin of $\mathcal{F}_{R[X]}$
  - $\overrightarrow{\mathcal{O}_{R[X]}\mathcal{O}_{R[X+1]}}$: **Position** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

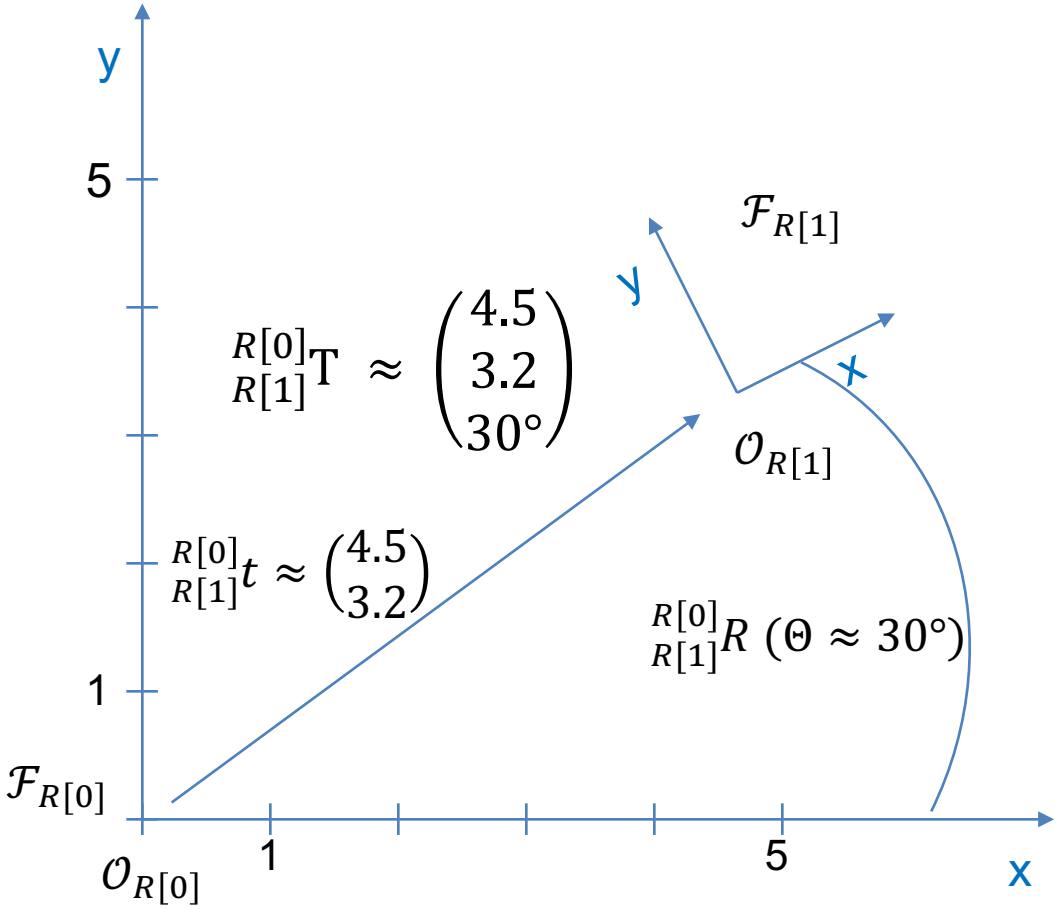
    to $\mathcal{O}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

    $\triangleq {}^{R[X]}_{R[X+1]}t$: **Translation**

- The angle Θ between the x-Axes:
  - **Orientation** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

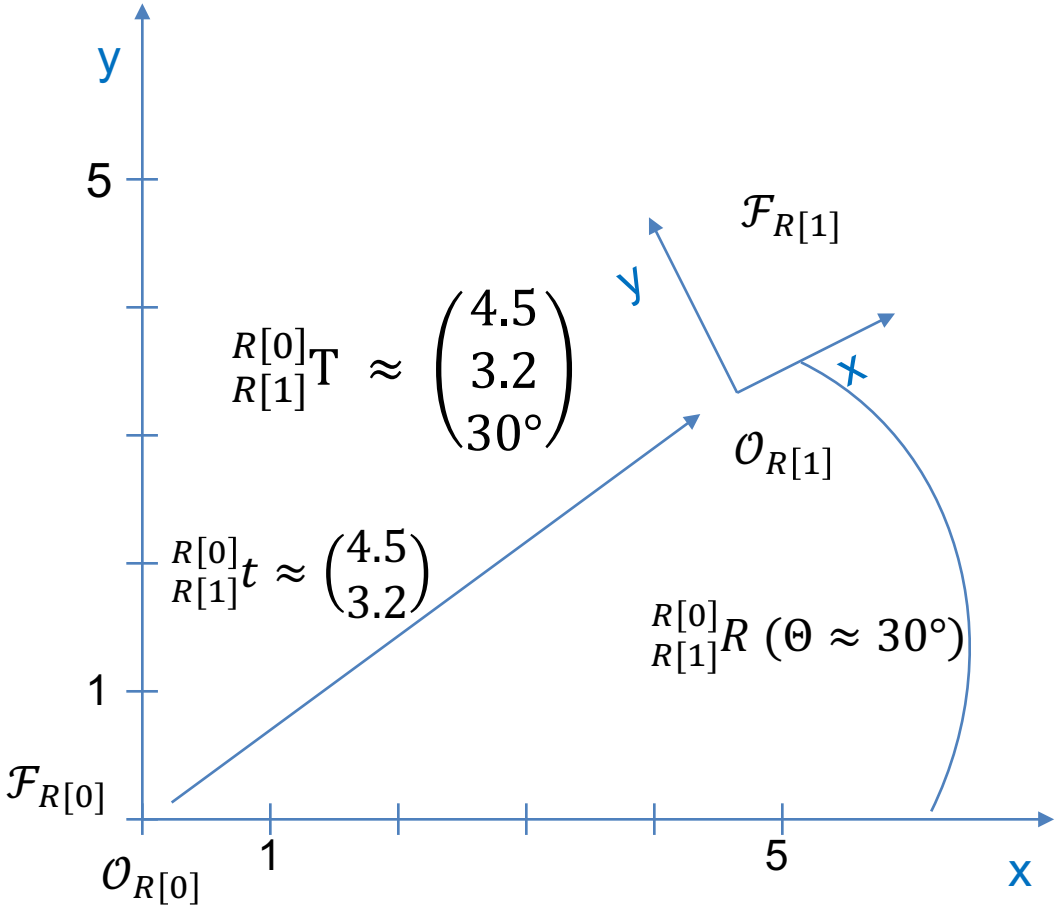
    $\triangleq {}^{R[X]}_{R[X+1]}R$: **Rotation** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

Transform



- ${}^{R[X]}_{R[X+1]}t$: **Translation**
  - Position vector (x, y) of $R[X+1]$ wrt. $R[X]$
- ${}^{R[X]}_{R[X+1]}R$ **: Rotation**
  - Angle (Θ) of $R[X+1]$ wrt. $R[X]$
- **Transfrom:** ${}^{R[X]}_{R[X+1]}\mathrm{T} \equiv \left\{ {}^{R[X]}_{R[X+1]}t,\ {}^{R[X]}_{R[X+1]}R \right\}$

Geometry approach to Odometry



We want to know:

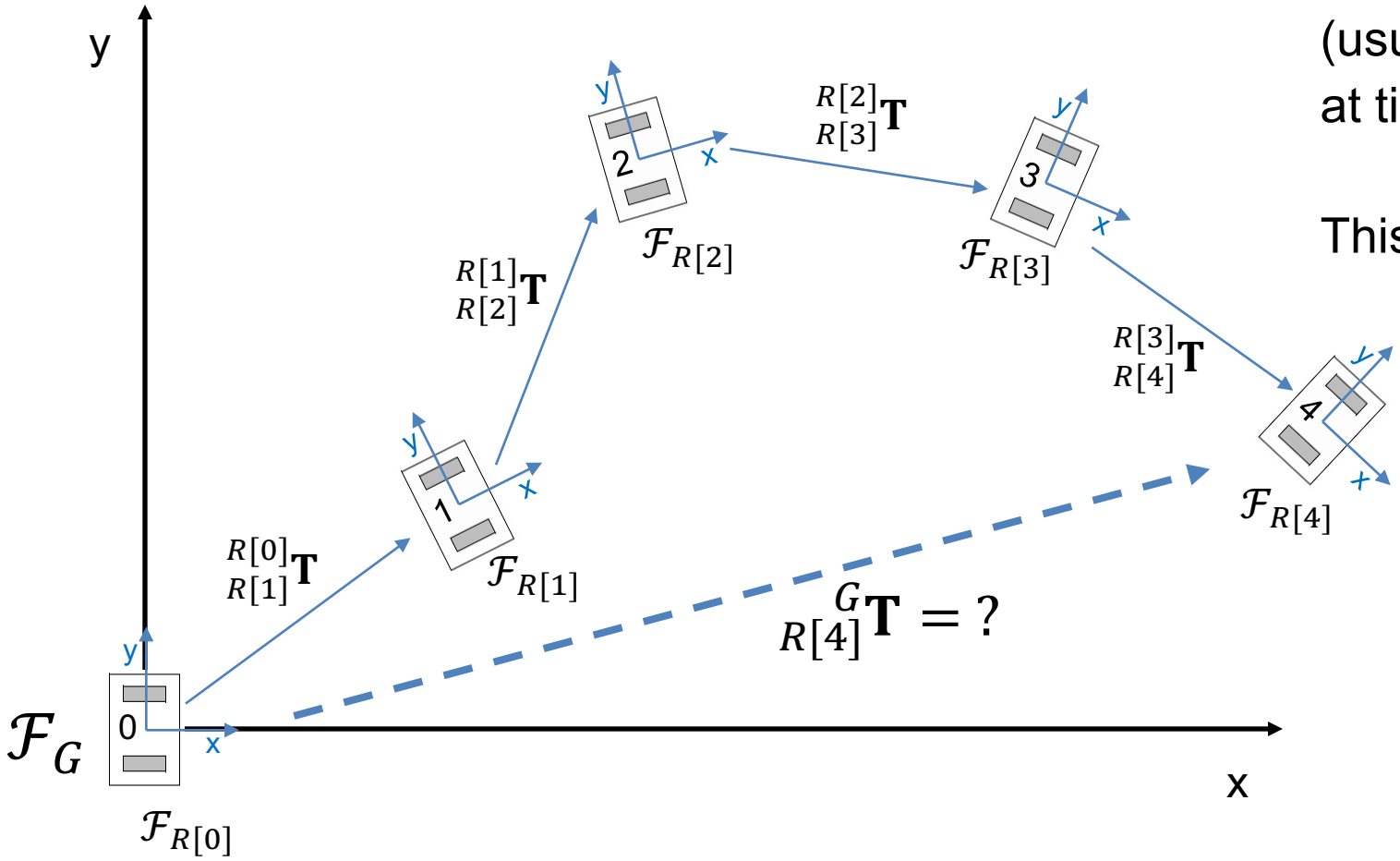
- Position of the robot (x, y)
- Orientation of the robot (Θ)
- => together: Pose $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$

With respect to (wrt.) $\mathcal{F}_G$: The global frame; global coordinate system

$$\mathcal{F}_{R[0]} = \mathcal{F}_G \Rightarrow {}^{G}\mathcal{F}_{R[0]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$${}^{G}\mathcal{F}_{R[1]} = {}^{R[0]}_{R[1]}\mathrm{T} \approx \begin{pmatrix} 4.5 \\ 3.2 \\ 30° \end{pmatrix}$$

Mathematical approach: Transforms



Where is the Robot now?

The pose of $\mathcal{F}_{R[X]}$ with respect to $\mathcal{F}_G$ (usually = $\mathcal{F}_{R[0]}$) is the pose of the robot at time X.

This is equivalent to $^{G}_{R[X]}\mathbf{T}$

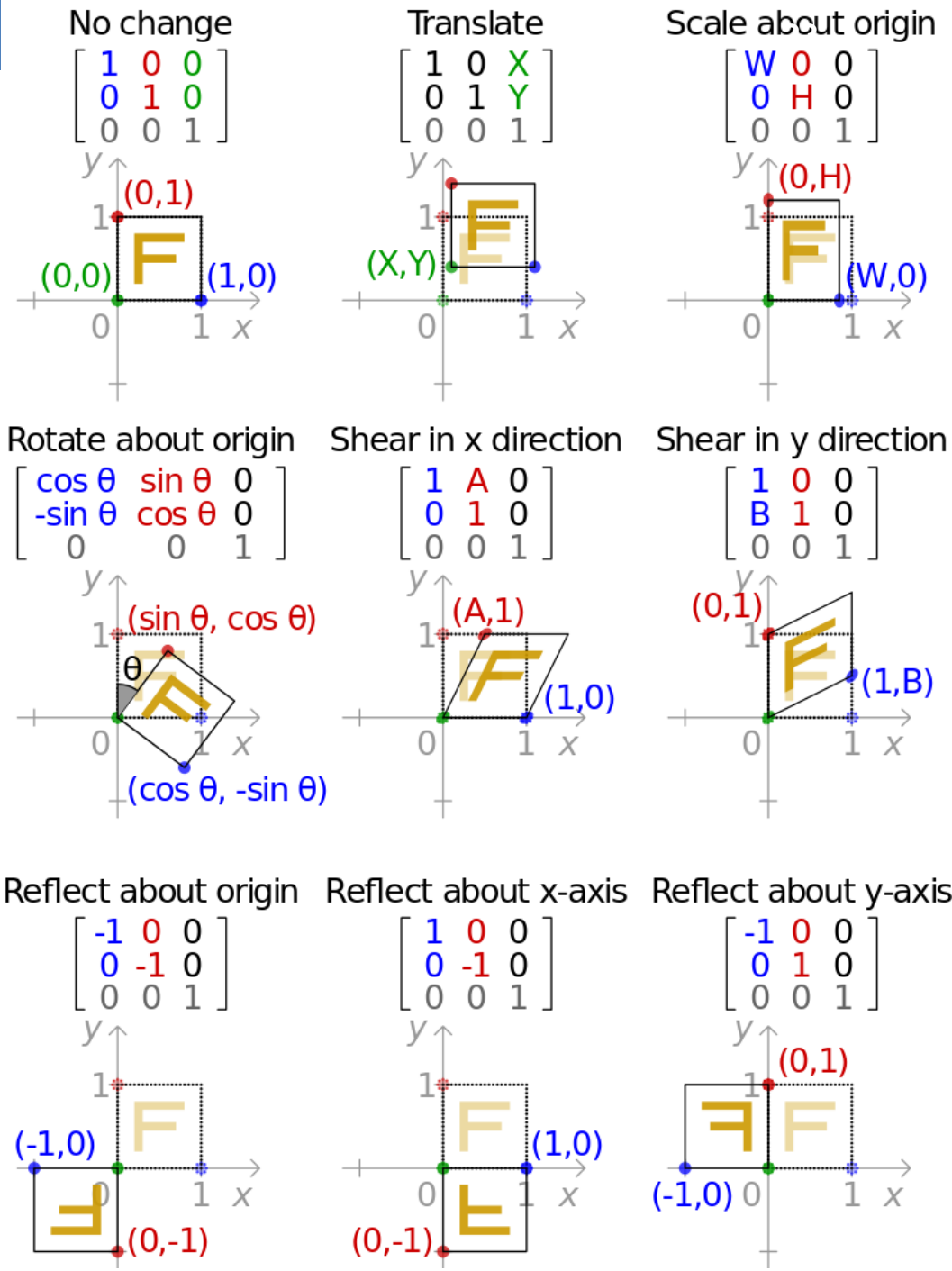
Chaining of Transforms

$$^{G}_{R[X+1]}\mathbf{T} = {}^{G}_{R[X]}\mathbf{T}\ {}^{R[X]}_{R[X+1]}\mathbf{T}$$

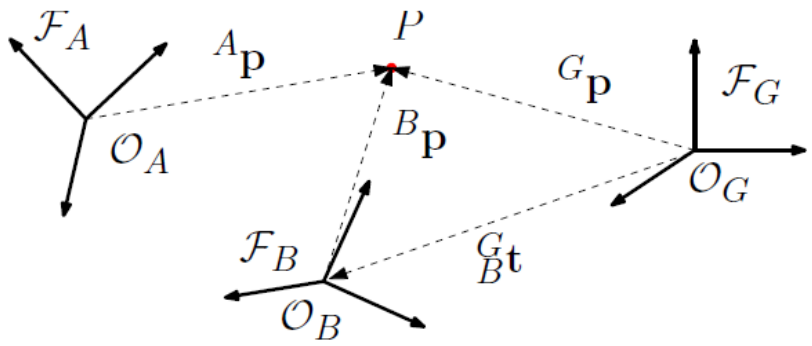
often: $\mathcal{F}_G \equiv \mathcal{F}_{R[0]} \Rightarrow {}^{G}_{R[0]}\mathbf{T} = id$

Affine Transformation

- Function between affine spaces. Preserves:
  - points,
  - straight lines
  - planes
  - sets of parallel lines remain parallel
- Allows:
  - Interesting for Robotics: translation, rotation, (scaling), and chaining of those
  - Not so interesting for Robotics: reflection, shearing, homothetic transforms
- Rotation and Translation: $\begin{bmatrix} \cos\theta & \sin\theta & X \\ -\sin\theta & \cos\theta & Y \\ 0 & 0 & 1 \end{bmatrix}$



Transform



Notation	Meaning
$\mathcal{F}_{\mathrm{R}[k]}$	Coordinate frame attached to object 'R' (usually the robot) at sample time-instant k.
$\mathcal{O}_{\mathrm{R}[k]}$	Origin of $\mathcal{F}_{\mathrm{R}[k]}$.
${}^{\mathrm{R}[k]}\mathbf{p}$	For any general point P, the position vector $\overrightarrow{\mathcal{O}_{\mathrm{R}[k]}P}$ resolved in $\mathcal{F}_{\mathrm{R}[k]}$.
${}^{\mathrm{H}}\hat{\mathbf{x}}_{\mathrm{R}}$	The x-axis direction of $\mathcal{F}_{\mathrm{R}}$ resolved in $\mathcal{F}_{\mathrm{H}}$. Similarly, ${}^{\mathrm{H}}\hat{\mathbf{y}}_{\mathrm{R}}$, ${}^{\mathrm{H}}\hat{\mathbf{z}}_{\mathrm{R}}$ can be defined. Obviously, ${}^{\mathrm{R}}\hat{\mathbf{x}}_{\mathrm{R}} = \hat{\mathbf{e}}_1$. Time indices can be added to the frames, if necessary.
${}^{\mathrm{R}[k]}_{\mathrm{S}[k']}\mathbf{R}$	The rotation-matrix of $\mathcal{F}_{\mathrm{S}[k']}$ with respect to $\mathcal{F}_{\mathrm{R}[k]}$.
${}^{\mathrm{R}}_{\mathrm{S}}\mathbf{t}$	The translation vector $\overrightarrow{\mathcal{O}_{\mathrm{R}}\mathcal{O}_{\mathrm{S}}}$ resolved in $\mathcal{F}_{\mathrm{R}}$.

Transform between two coordinate frames

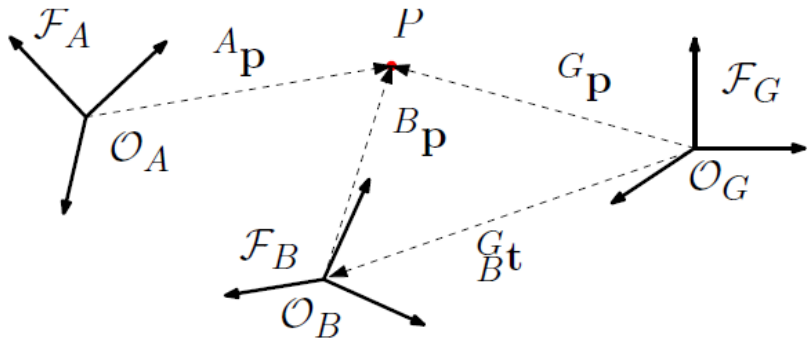
$${}^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \triangleq \overrightarrow{\mathcal{O}_{\mathrm{G}}\mathcal{O}_{\mathrm{A}}} \text{ resolved in } \mathcal{F}_{\mathrm{G}}$$

$$\begin{pmatrix} {}^{\mathrm{G}}\mathbf{p} \\ 1 \end{pmatrix} \equiv \begin{pmatrix} {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{R} & {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \\ \mathbf{0}_{1\times[2,3]} & 1 \end{pmatrix} \begin{pmatrix} {}^{\mathrm{A}}\mathbf{p} \\ 1 \end{pmatrix} \qquad {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{T} \equiv \left\{ \begin{matrix} {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \\ {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{R} \end{matrix} \right\}$$

$$\begin{aligned} {}^{\mathrm{G}}\mathbf{p} &= {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{R}\ {}^{\mathrm{A}}\mathbf{p} + {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \\ &\triangleq {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{T}({}^{\mathrm{A}}\mathbf{p}). \end{aligned}$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{t}_{\mathrm{x}} \\ \sin\theta & \cos\theta & {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{t}_{\mathrm{y}} \\ 0 & 0 & 1 \end{bmatrix}$$

Transform: Operations



Transform between two coordinate frames (chaining, compounding):

$$ {}^G_B\mathbf{T} = {}^G_A\mathbf{T}\,{}^A_B\mathbf{T} \equiv \begin{Bmatrix} {}^G_A\mathbf{R}\,{}^A_B\mathbf{t} + {}^G_A\mathbf{t} \\ {}^G_A\mathbf{R}\,{}^A_B\mathbf{R} \end{Bmatrix} $$

Inverse of a Transform :

$$ {}^B_A\mathbf{T} = {}^A_B\mathbf{T}^{-1} \equiv \begin{Bmatrix} -{}^A_B\mathbf{R}^\mathsf{T}\,{}^A_B\mathbf{t} \\ {}^A_B\mathbf{R}^\mathsf{T} \end{Bmatrix} $$

Relative (Difference) Transform :

$$ {}^B_A\mathbf{T} = {}^G_B\mathbf{T}^{-1}\,{}^G_A\mathbf{T} $$

See: **Quick Reference to Geometric Transforms in Robotics** by Kaustubh Pathak on the webpage!

Chaining :

$$ {}^{G}_{R[X+1]}\mathbf{T} = {}^{G}_{R[X]}\mathbf{T}\ {}^{R[X]}_{R[X+1]}\mathbf{T} \equiv \begin{Bmatrix} {}^{G}_{R[X]}\mathrm{R} & {}^{R[X]}_{R[X+1]}t + {}^{G}_{R[X]}t \\ & {}^{G}_{R[X]}\mathrm{R}\ {}^{R[X]}_{R[X+1]}\mathrm{R} \end{Bmatrix} = \begin{Bmatrix} {}^{G}_{R[X+1]}t \\ {}^{G}_{R[X+1]}\mathrm{R} \end{Bmatrix} $$

In 2D Translation:

$$ \begin{bmatrix} {}^{G}_{R[X+1]}t_x \\ {}^{G}_{R[X+1]}t_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos {}^{G}_{R[X]}\theta & -\sin {}^{G}_{R[X]}\theta & {}^{G}_{R[X]}\mathrm{t_x} \\ \sin {}^{G}_{R[X]}\theta & \cos {}^{G}_{R[X]}\theta & {}^{G}_{R[X]}\mathrm{t_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{R[X]}_{R[X+1]}\mathrm{t_x} \\ {}^{R[X]}_{R[X+1]}\mathrm{t}_y \\ 1 \end{bmatrix} $$

In 2D Rotation:

$$ {}^{G}_{R[X+1]}\mathrm{R} = \begin{bmatrix} \cos {}^{G}_{R[X+1]}\theta & -\sin {}^{G}_{R[X+1]}\theta \\ \sin {}^{G}_{R[X+1]}\theta & \cos {}^{G}_{R[X+1]}\theta \end{bmatrix} = \begin{bmatrix} \cos {}^{G}_{R[X]}\theta & -\sin {}^{G}_{R[X]}\theta \\ \sin {}^{G}_{R[X]}\theta & \cos {}^{G}_{R[X]}\theta \end{bmatrix} \begin{bmatrix} \cos {}^{R[X]}_{R[X+1]}\theta & -\sin {}^{R[X]}_{R[X+1]}\theta \\ \sin {}^{R[X]}_{R[X+1]}\theta & \cos {}^{R[X]}_{R[X+1]}\theta \end{bmatrix} $$

In 2D Rotation (simple):

$$ {}^{G}_{R[X+1]}\theta = {}^{G}_{R[X]}\theta + {}^{R[X]}_{R[X+1]}\theta $$

In ROS

- First Message at time 97 : G
- Message at time 103 : X
- Next Message at time 107 : X+1

$$ {}^{G}_{R[X+1]}\mathbf{T} = {}^{G}_{R[X]}\mathbf{T} \; {}^{R[X]}_{R[X+1]}\mathbf{T} $$

$$ {}^{R[X]}_{R[X+1]}t_x $$

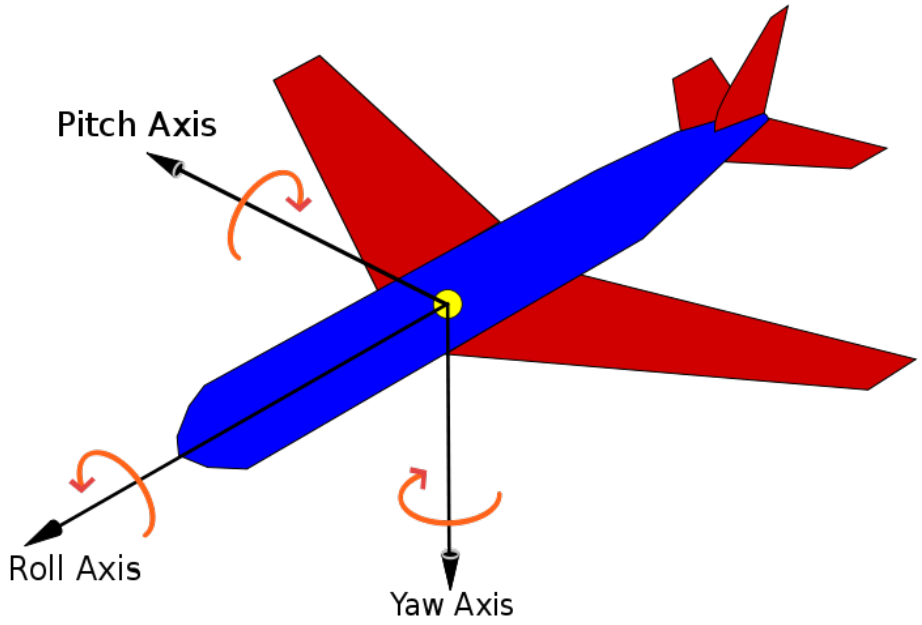
$$ {}^{R[X]}_{R[X+1]}t_y $$

$$ {}^{R[X]}_{R[X+1]}\Theta $$

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose2D pose2D
  float64 x
  float64 y
  float64 theta
```

3D Rotation

- Euler angles: Roll, Pitch, Yaw
  - ☹ Singularities
- Quaternions:
  - Concatenating rotations is computationally faster and numerically more stable
  - Extracting the angle and axis of rotation is simpler
  - Interpolation is more straightforward
  - Unit Quaternion: norm = 1
  - Scalar (real) part: q_0 , sometimes q_w
  - Vector (imaginary) part: $\mathbf{q}$
  - Over determined: 4 variables for 3 DoF



$$\check{\mathbf{p}} \equiv p_0 + p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}$$

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

$$\check{\mathbf{q}} = \begin{pmatrix} q_0 & q_x & q_y & q_z \end{pmatrix}^\mathsf{T} \equiv \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix}$$

Transform in 3D

Matrix Euler Quaternion

$$ {}^{G}_{A}\mathbf{T} = \begin{bmatrix} {}^{G}_{A}\mathrm{R} & {}^{G}_{A}\mathrm{t} \\ 0_{1x3} & 1 \end{bmatrix} = \begin{pmatrix} {}^{G}_{A}\mathrm{t} \\ {}^{G}_{A}\Theta \end{pmatrix} = \begin{pmatrix} {}^{G}_{A}\mathrm{t} \\ {}^{G}_{A}\check{q} \end{pmatrix} $$

$$ {}^{G}_{A}\Theta \triangleq (\theta_r, \theta_p, \theta_y)^T $$

In ROS: Quaternions! (w, x, y, z)

Rotation Matrix 3x3

$$ R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} $$

$$ R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} $$

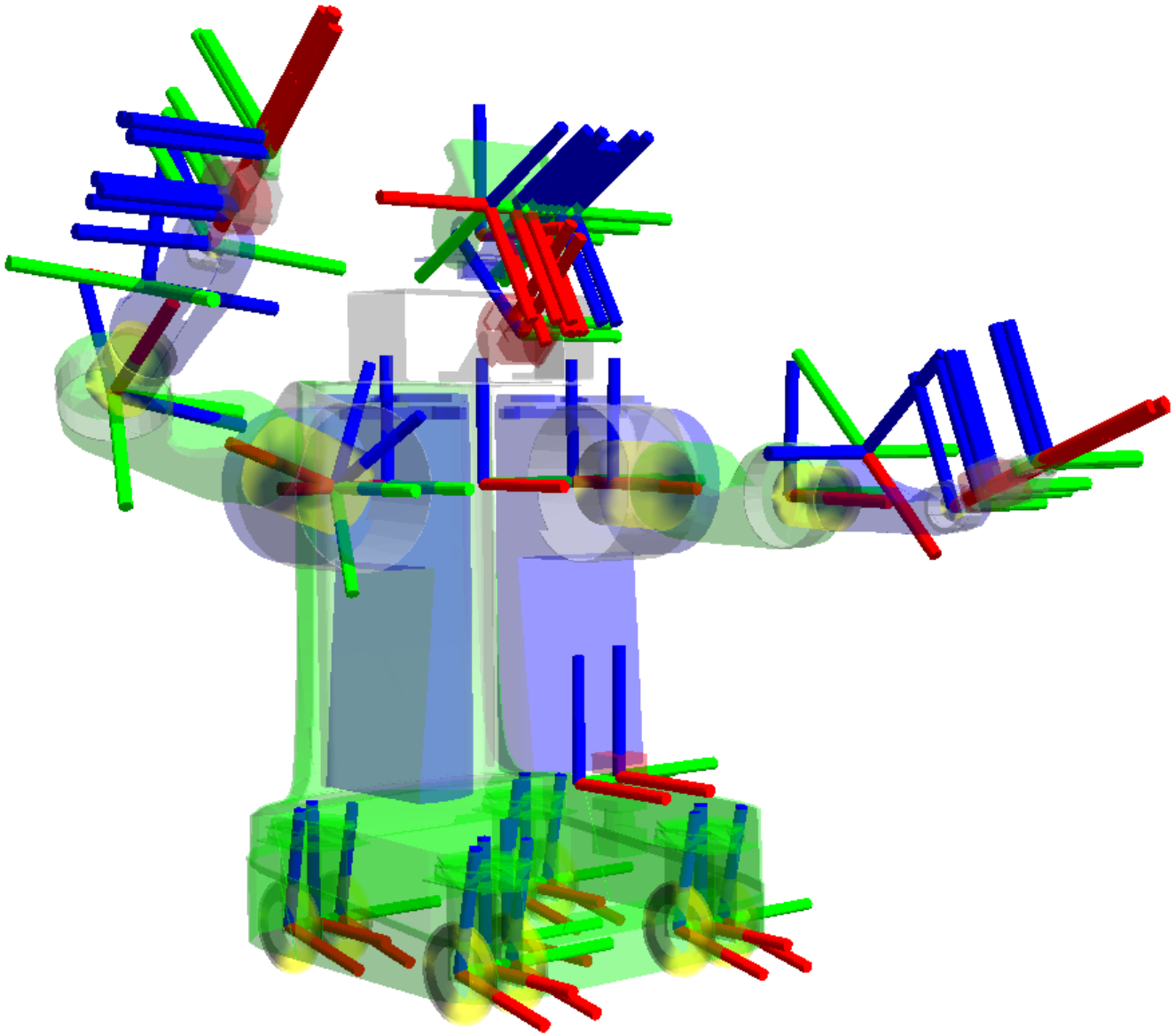
$$ R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ R = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma) $$

yaw = α, pitch = β, roll = γ

ROS: Transforms : TF

- http://wiki.ros.org/tf
- http://wiki.ros.org/tf/Tutorials



ROS geometry_msgs/TransformStamped

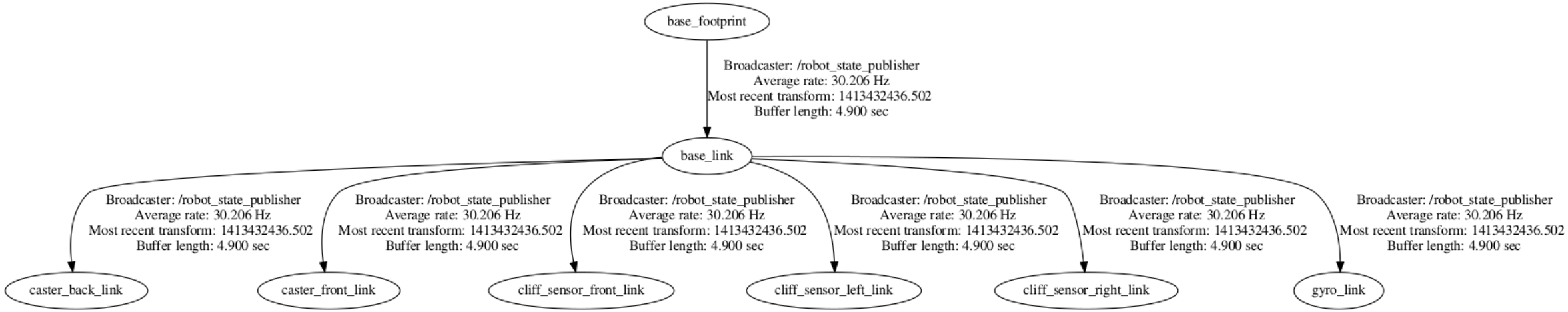
- $\mathrm{header.frame\_id[header.stamp]}_{\mathbf{T}}\ \mathrm{child\_frame\_id[header.stamp]}$
- Transform between header (time and reference frame) and child_frame
- 3D Transform representation:
  - geometry_msgs/Transform:
    - Vector3 for translation (position)
    - Quaternion for rotation (orientation)

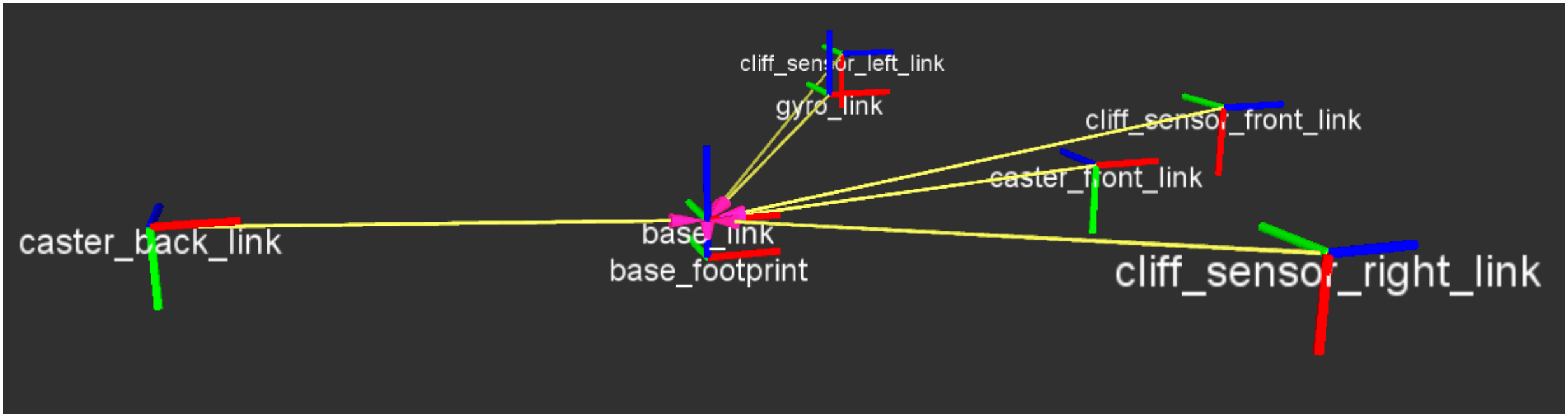
```
rosmsg show geometry_msgs/TransformStamped

std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/Transform transform
  geometry_msgs/Vector3 translation
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion rotation
    float64 x
    float64 y
    float64 z
    float64 w
```

ROS tf2_msgs/TFMessage

- An array of TransformStamped
- Transforms form a tree
- Transform listener: traverse the tree
  - tf::TransformListener listener;
- Get transform:
  - tf::StampedTransform transform;
  - listener.lookupTransform("/base_link", "/camera1", ros::Time(0), transform);
  - ros::Time(0): get the latest transform
  - Will calculate transform by chaining intermediate transforms, if needed

```
rosmsg show tf2_msgs/TFMessage

geometry_msgs/TransformStamped[] transforms
  std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
  string child_frame_id
  geometry_msgs/Transform transform
    geometry_msgs/Vector3 translation
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion rotation
      float64 x
      float64 y
      float64 z
      float64 w
```


base_footprint
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
base_link
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
caster_back_link
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
caster_front_link
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
cliff_sensor_front_link
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
cliff_sensor_left_link
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
cliff_sensor_right_link
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
gyro_link


cliff_sensor_left_link
gyro_link
cliff_sensor_front_link
caster_front_link
caster_back_link
base_link
base_footprint
cliff_sensor_right_link

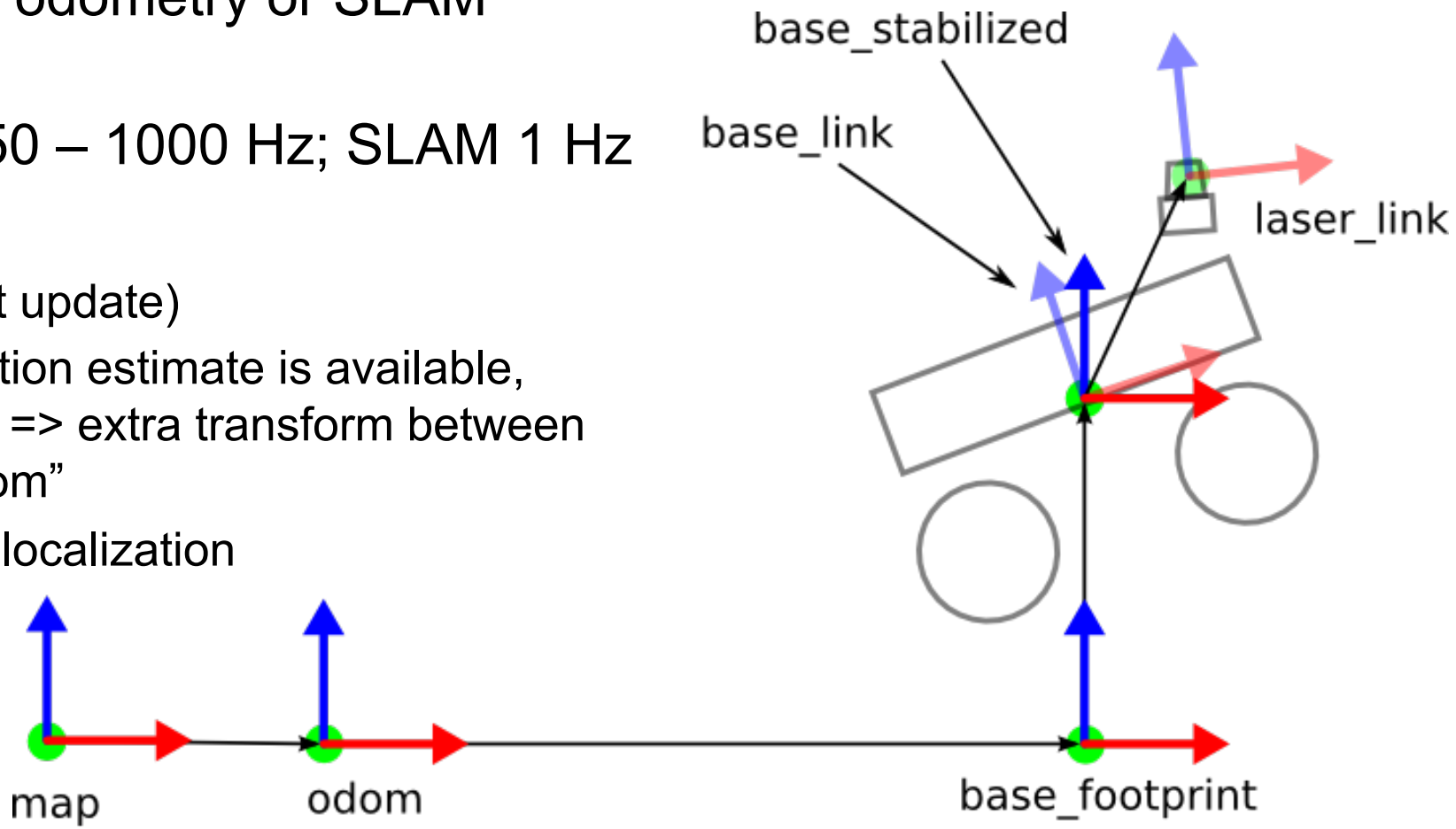
Transforms in ROS

- Imagine: Object recognition took 3 seconds – it found an object with:
  - tf::Transform object_transform_camera; // ${}^{Cam[X]}_{Obj}\mathbf{T}$ (has tf::Vector3 and tf::Quaternion)
  - and header with: ros::Time stamp; // Timestamp of the camera image (== X)
    - and std::string frame_id; // Name of the frame (“Cam”)
- Where is the object in the global frame (= odom frame) “odom” ${}^{G}_{Obj}\mathbf{T}$?
  - tf::StampedTransform object_transform_global; // the resulting frame
  - listener.lookupTransform(header.frame_id, “/odom", header.stamp, object_transform_global);
- tf::TransformListener keeps a history of transforms – by default 10 seconds

TF and SLAM

http://wiki.ros.org/hector_slam/Tutorials/SettingUpForYourRobot

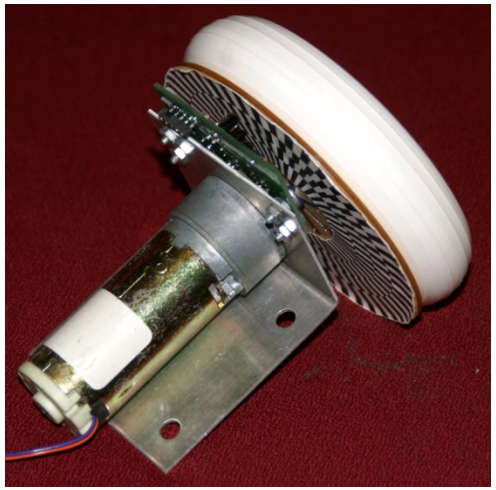
- How is the global frame named?
- Which transform to use – odometry or SLAM localization?
- Frequencies: Odometry 50 – 1000 Hz; SLAM 1 Hz
- Solution: use both! :
  - Basic: Odometry frame (fast update)
  - SLAM: Once a new localization estimate is available, CORRECT the odom frame => extra transform between frame ”map” and frame ”odom”
  - => fast update AND correct localization

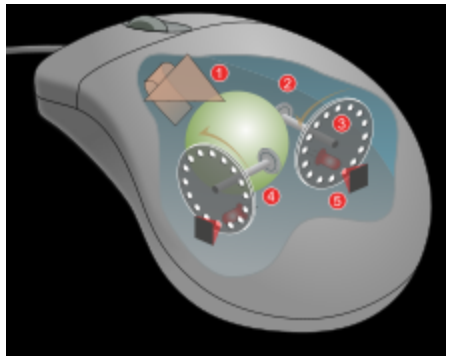


SENSORS FOR MAPPING

Encoders

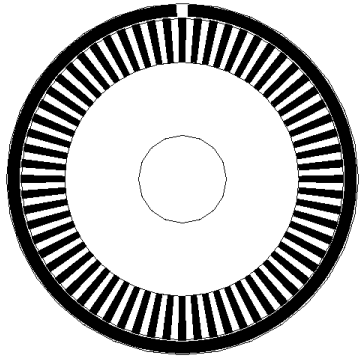
An encoder is an electro-mechanical device that converts the angular position of a shaft to an analog or digital signal, making it an angle transducer

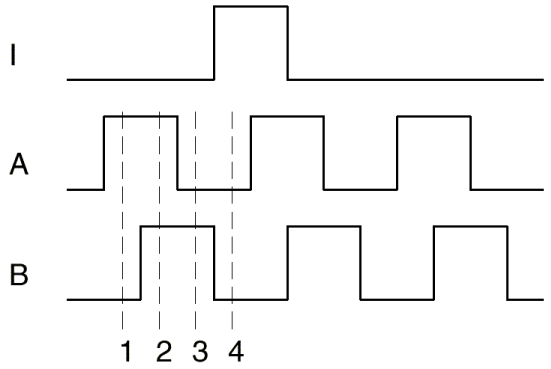




Wheel / Motor Encoders

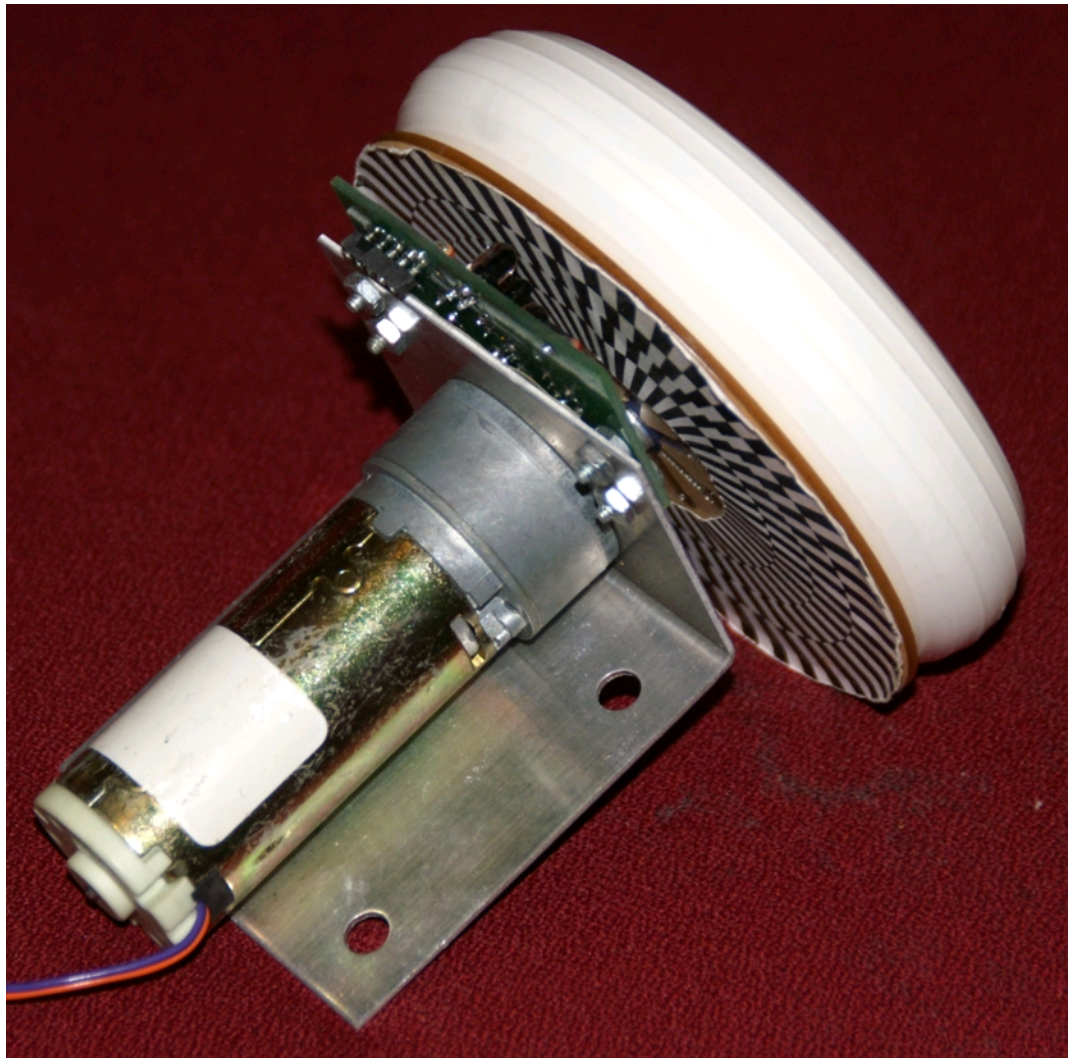
- measure position or speed of the wheels or steering
- integrate wheel movements to get an estimate of the position -> odometry
- optical encoders are proprioceptive sensors
- typical resolutions: 64 - 2048 increments per revolution.
  - for high resolution: interpolation
- optical encoders
  - regular: counts the number of transitions but cannot tell the direction of motion
  - quadrature: uses two sensors in quadrature-phase shift. The ordering of which wave produces a rising edge first tells the direction of motion. Additionally, resolution is 4 times bigger
  - a single slot in the outer track generates a reference pulse per revolution





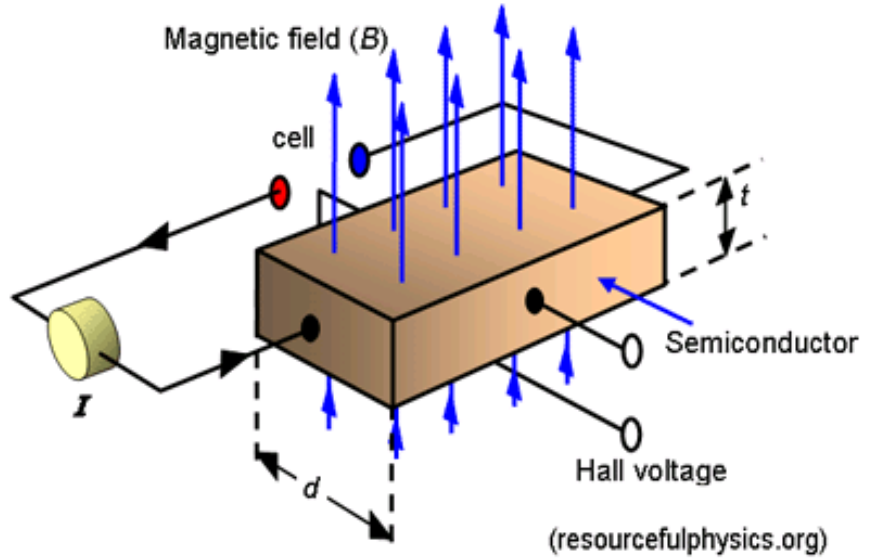
State	Ch A	Ch B
S_1	High	Low
S_2	High	High
S_3	Low	High
S_4	Low	Low

A custom made optical encoder



Heading Sensor: Compass

- Since over 2000 B.C.
  - China: suspended a piece of naturally magnetite from a silk thread to guide a chariot over land.
- Magnetic field on earth
  - absolute measure for orientation (even birds use it for migrations (2001 discovery))
- Large variety of solutions to measure the earth magnetic field
  - mechanical magnetic compass
  - direct measure of the magnetic field (Hall-effect, magneto-resistive sensors)
- Major drawback
  - weakness of the earth field (30 µTesla)
  - easily disturbed by magnetic objects or other sources
  - bandwidth limitations (0.5 Hz) and susceptible to vibrations
  - not feasible for indoor environments for absolute orientation
  - useful indoor (only locally)

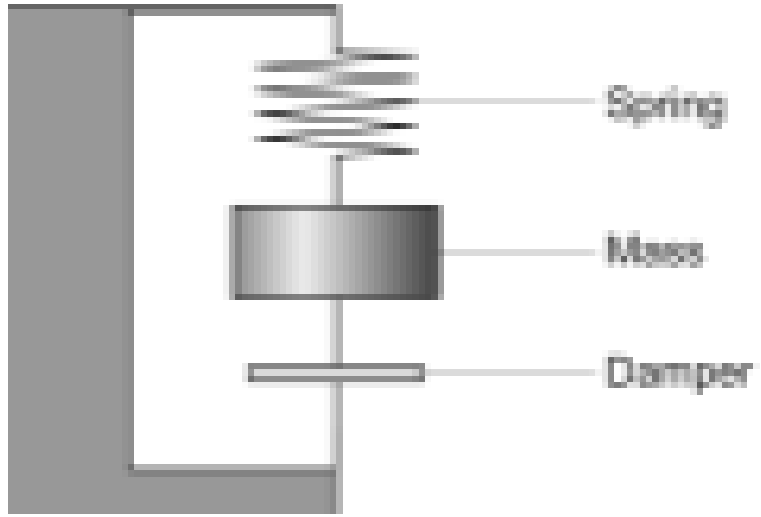


Heading Sensor: Gyroscope

- Heading sensors that preserve their orientation in relation to a fixed reference frame
  - absolute measure for the heading of a mobile system.
- Two categories, the mechanical and the optical gyroscopes
  - Mechanical Gyroscopes
    - Standard gyro (angle)
    - Rate gyro (speed)
  - Optical Gyroscopes
    - Rate gyro (speed)

Heading Sensor: Mechanical Accelerometer

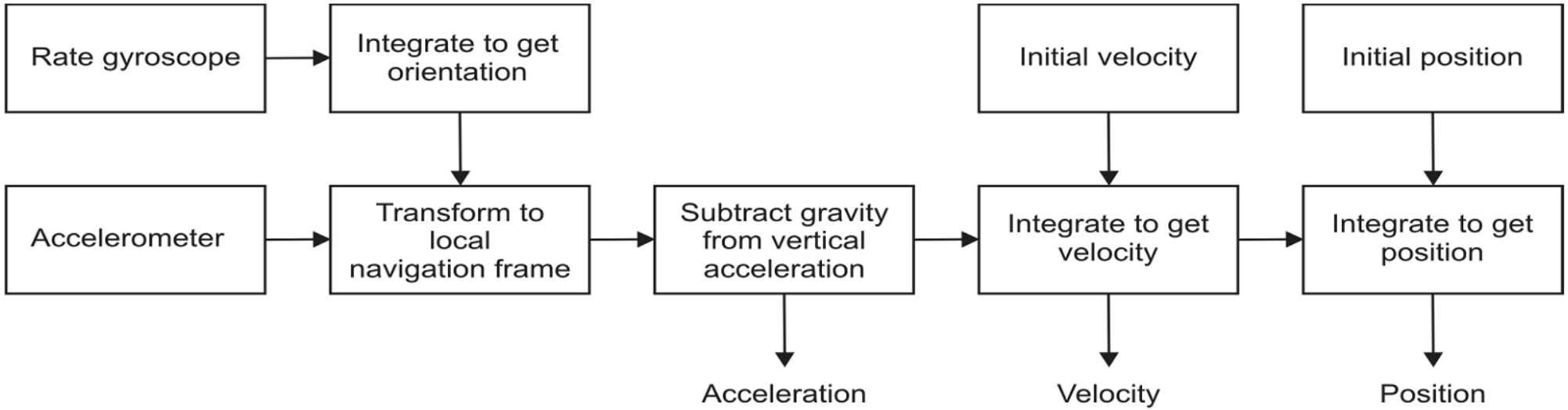
- Accelerometers measure all external forces acting upon them, including gravity
- Accelerometer acts like a spring–mass–damper system
- On the Earth's surface, the accelerometer always indicates 1g along the vertical axis
- To obtain the inertial acceleration (due to motion alone), the gravity must be subtracted.
- Bandwidth up to 50 KHz
- An accelerometer measures acceleration only along a single axis
- => mount 3 accelerometers orthogonally => three-axis accelerometer



Inertial Measurement Unit (IMU)

- Device combining different measurement systems:
  - Gyroscopes, Accelerometers, Compass
- Estimate relative position (x, y, z), orientation (roll, pitch, yaw), velocity, and acceleration
- Gravity vector is subtracted to estimate motion
  - Initial velocity has to be known

Xsens MTI



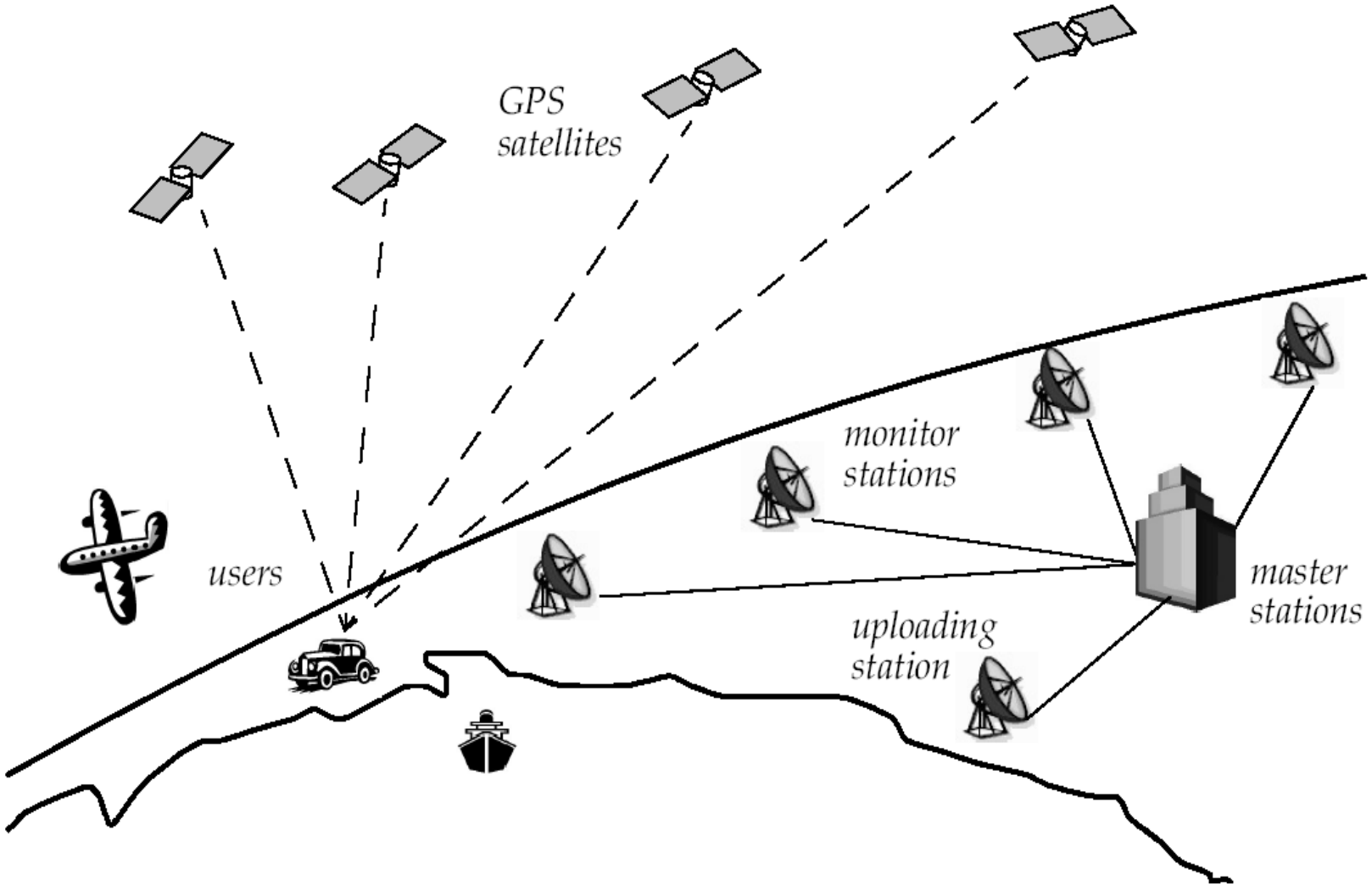
IMU Error and Drift

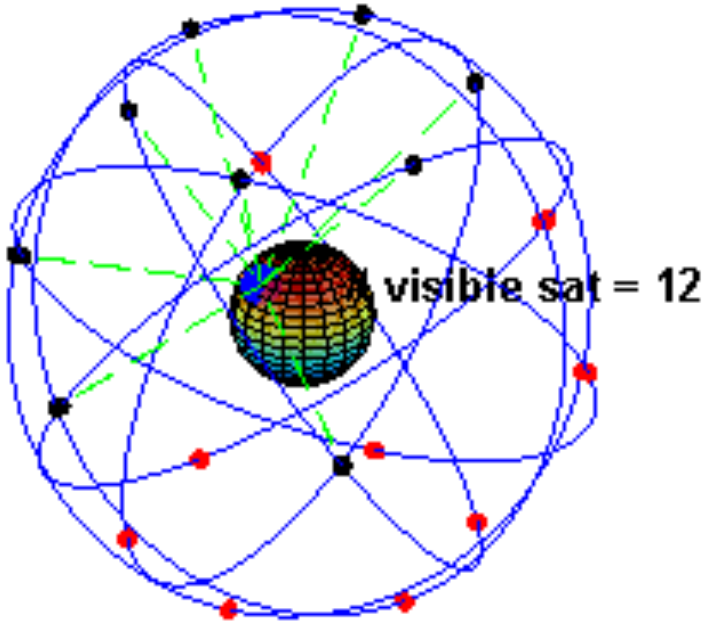
- Extremely sensitive to measurement errors in gyroscopes and accelerometers:
  - drift in the gyroscope unavoidably =>
  - error in orientation relative to gravity =>
  - incorrect cancellation of the gravity vector.

- Accelerometer data is integrated twice to obtain the position => gravity vector error leads to quadratic error in position.

- All IMUs drift after some time
  - Use of external reference for correction:
  - compass, GPS, cameras, localization

Global Positioning System (GPS)



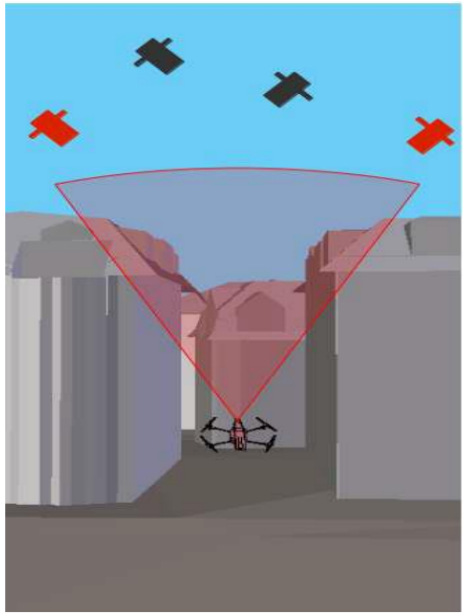


GPS Error Sources

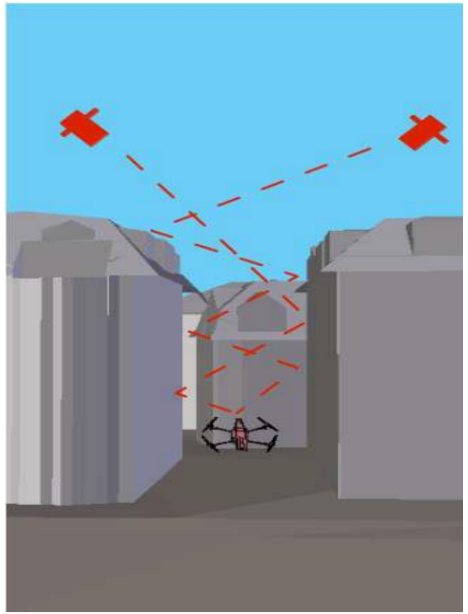
Satellite clock errors uncorrected by monitor stations may result in one meter errors:

- Ephemeris data errors: 1 meter
- Tropospheric delays: 1 meter.
  - Troposphere (8 - 13 km) changes in temperature, pressure, and humidity associated with weather changes. Complex models of tropospheric delay require estimates or measurements of these parameters.
- Un-modeled ionosphere delays: 10 meters.
  - Ionosphere (50 to 500 km): ionized air. The transmitted model can only remove about half of the possible 70 ns of delay leaving a ten meter un-modeled residual.
- Number of satellites under line of sight
- Multipath: 0.5 meters

Satellite coverage

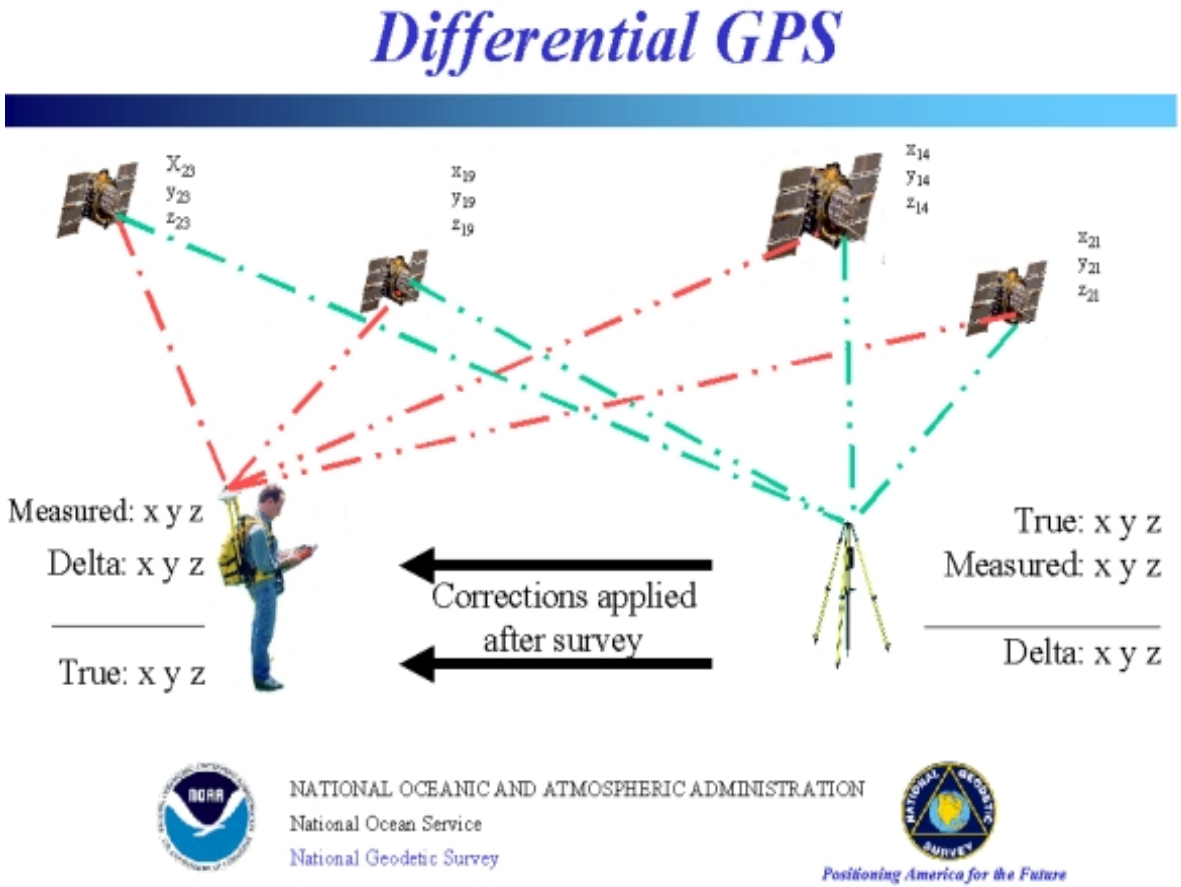


Multipath problem



Differential Global Positioning System (dGPS)

- Base station GPS receiver: set up on a precisely known location
- Base station receiver calculates its position based on satellite signals
- Compares this location to the known location
- Difference is applied to the GPS data recorded by the mobile GPS receivers
- Position accuracies in sub-meter to cm range



- Accuracy: up to 2cm

Laser Range Sensor (time of flight, electromagnetic)

- Is called Laser Range Finder or Lidar (LIght Detection And Ranging)

SICK

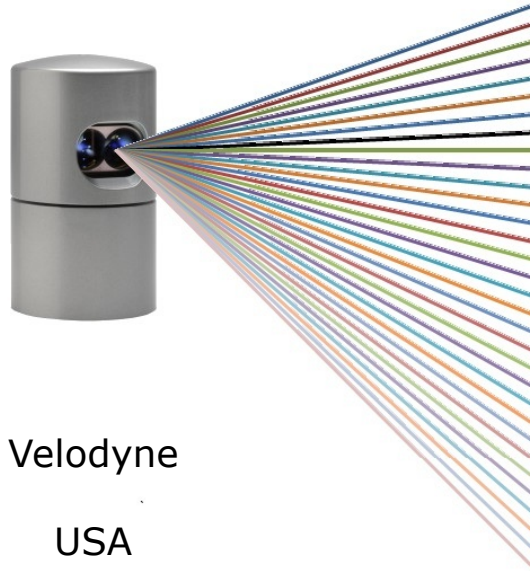
Germany

High Quality,
Safety Certification
Expensive

Hokuyo

Japan

Small sizes,
Popular for Robotics

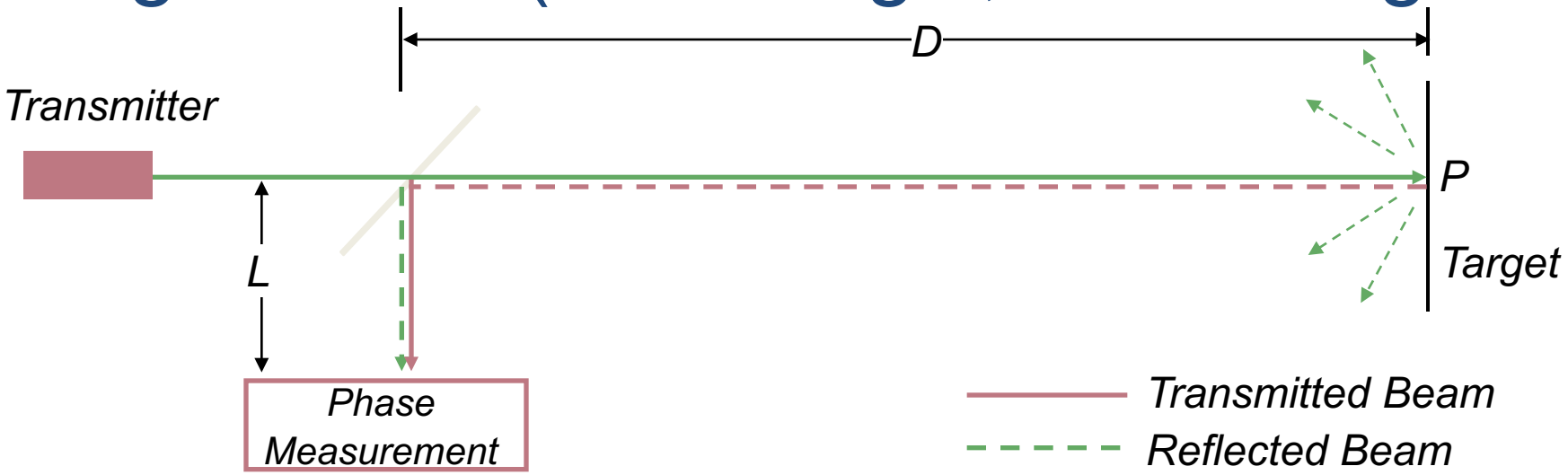


Velodyne

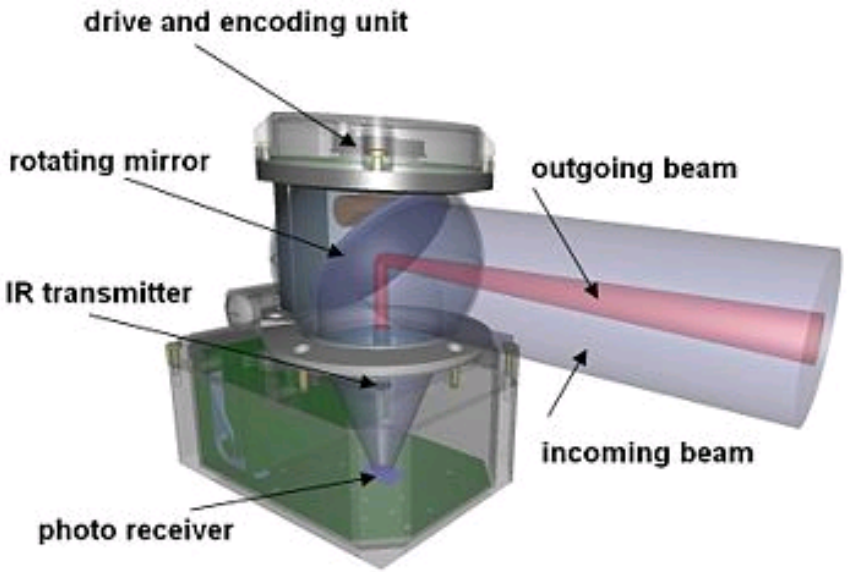
USA

3D Sensor (32 beams),
For autonomous cars,
Extremely Expensive

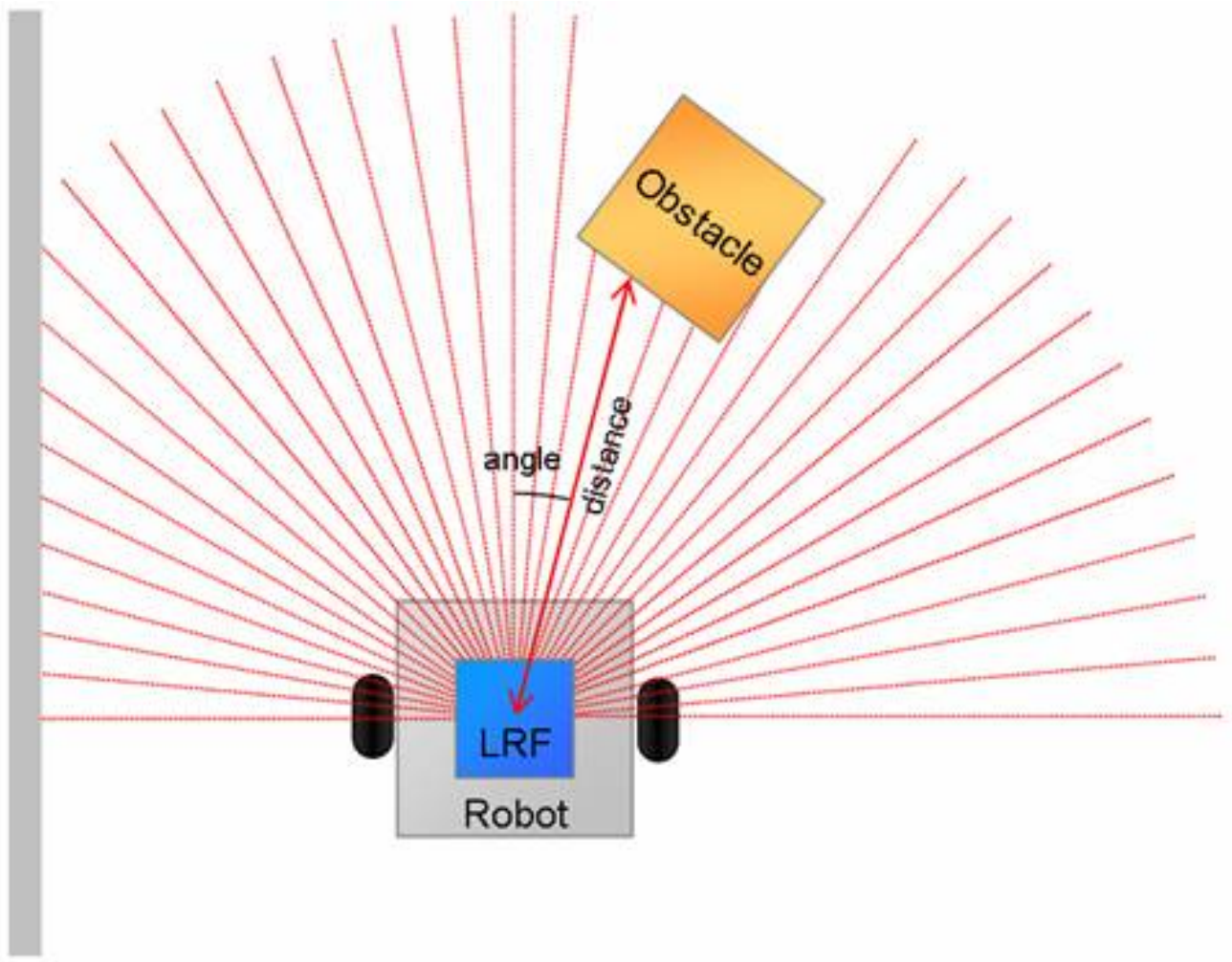
Laser Range Sensor (time of flight, electromagnetic)



- Transmitted and received beams coaxial
- Transmitter illuminates a target with a collimated laser beam
- Received detects the time needed for round-trip
- A mechanical mechanism with a mirror sweeps
  - 2D or 3D measurement



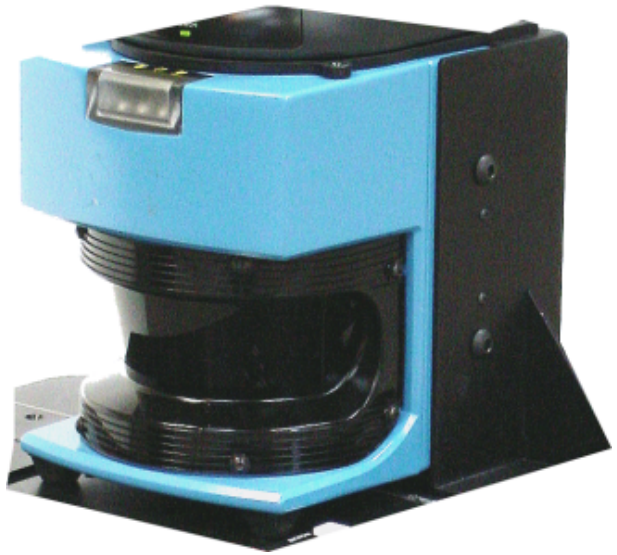
2D Laser Range Finder Scan



The SICK LMS 200 Laser Scanner

- Angular resolution 0.25 deg
- Depth resolution ranges between 10 and 15 mm and the typical accuracy is 35 mm, over a range from 5 cm up to 20 m or more (up to 80 m)
  - depending on the reflectivity of the object being ranged.
- This device performs seventy five 180-degrees per sec.
- Dimensions: W155 x D155 x H210mm,
- Weight: 1,2 kg

- Cost: about RMB 22,000



Hokuyo UTM-30LX

- Long Detection range: 30m
- 0.1 to 10m: ± 30mm, 1
- 0 to 30m: ± 50mm*1
- Field of View: 270°
- 40Hz
- Outdoor Environment
- Dimensions: 60 x 60 x 87 mm
- Weight: 370g

- Cost: about 35,000 RMB

URG-04LX-UG01

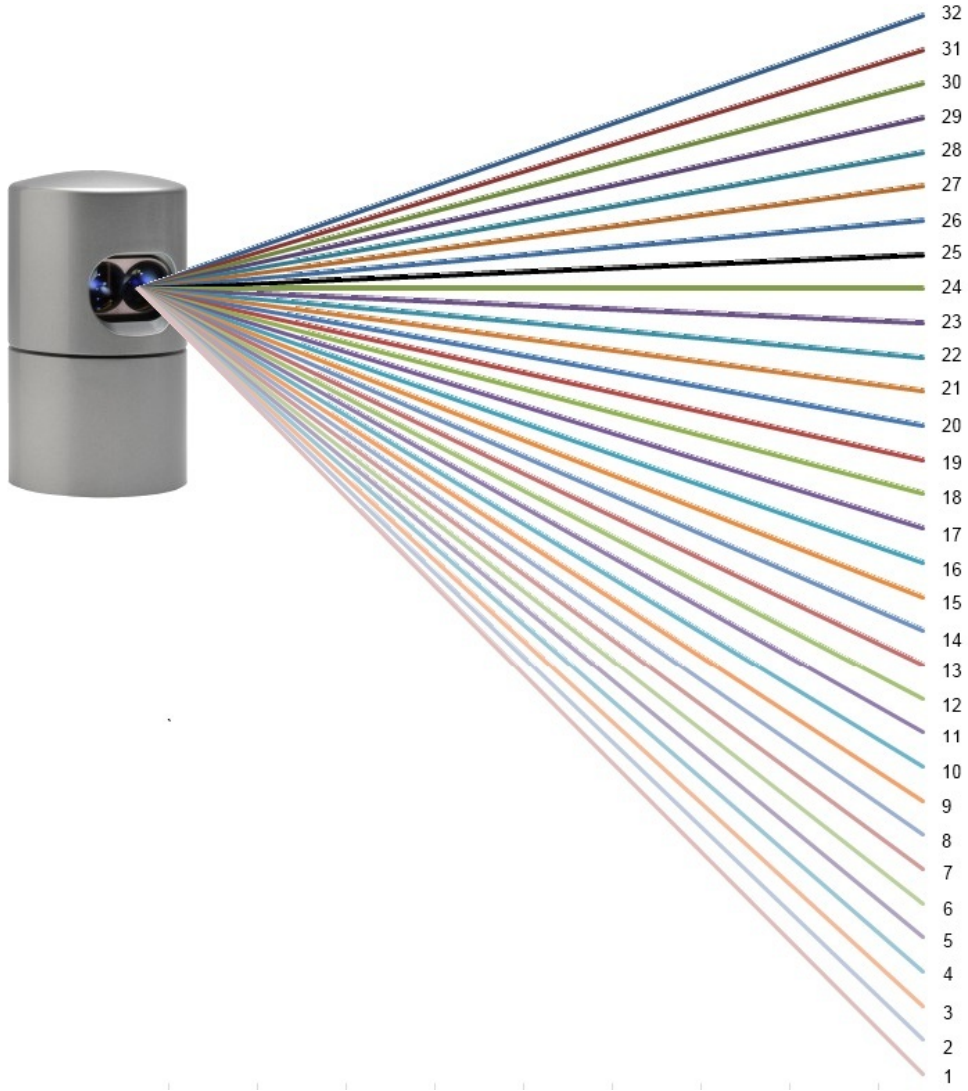
- Low-power consumption (2.5W)
- Wide-range (5600mm×240°).
- 60 to 1,000mm : ±30mm,
- 1,000 to 4,095mm : ±3% of measurement
- 10Hz
- Dimensions: 50 x 50 x 70 mm
- Weight: 160g

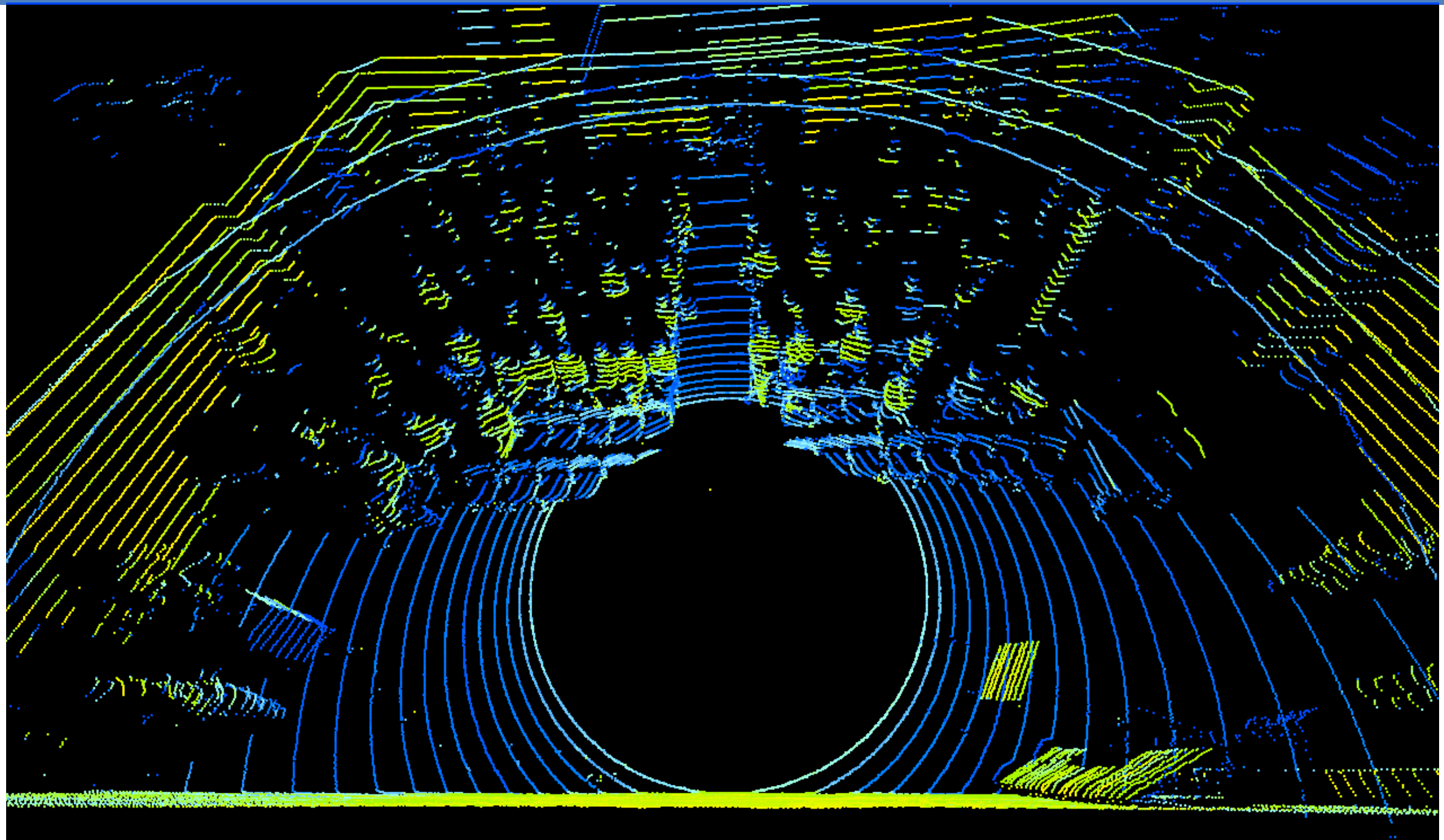
- Cost: about 6,500 RMB

Velodyne hdl-32e

- Range: up to 80 – 100 m
- +10.67 to -30.67 degrees field of view (vertical)
- 360° field of view (horizontal)
- 10 Hz frame rate
- Accuracy: <2 cm (one sigma at 25 m)
- Angular resolution (vertical) 1.33°
- 700,000 points per second
- internal MEMS accelerometers and gyros for six-axis motion correction

- Dimensions:
  - Diameter: 85mm,
  - Deight: 144 mm
- Weight: 1kg
- Cost: about 220,000 RMB





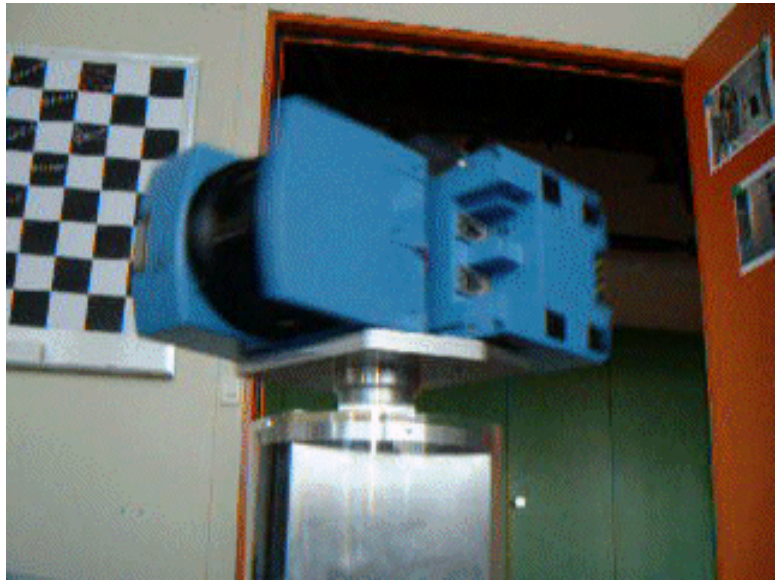
Laser Range Finder: Applications

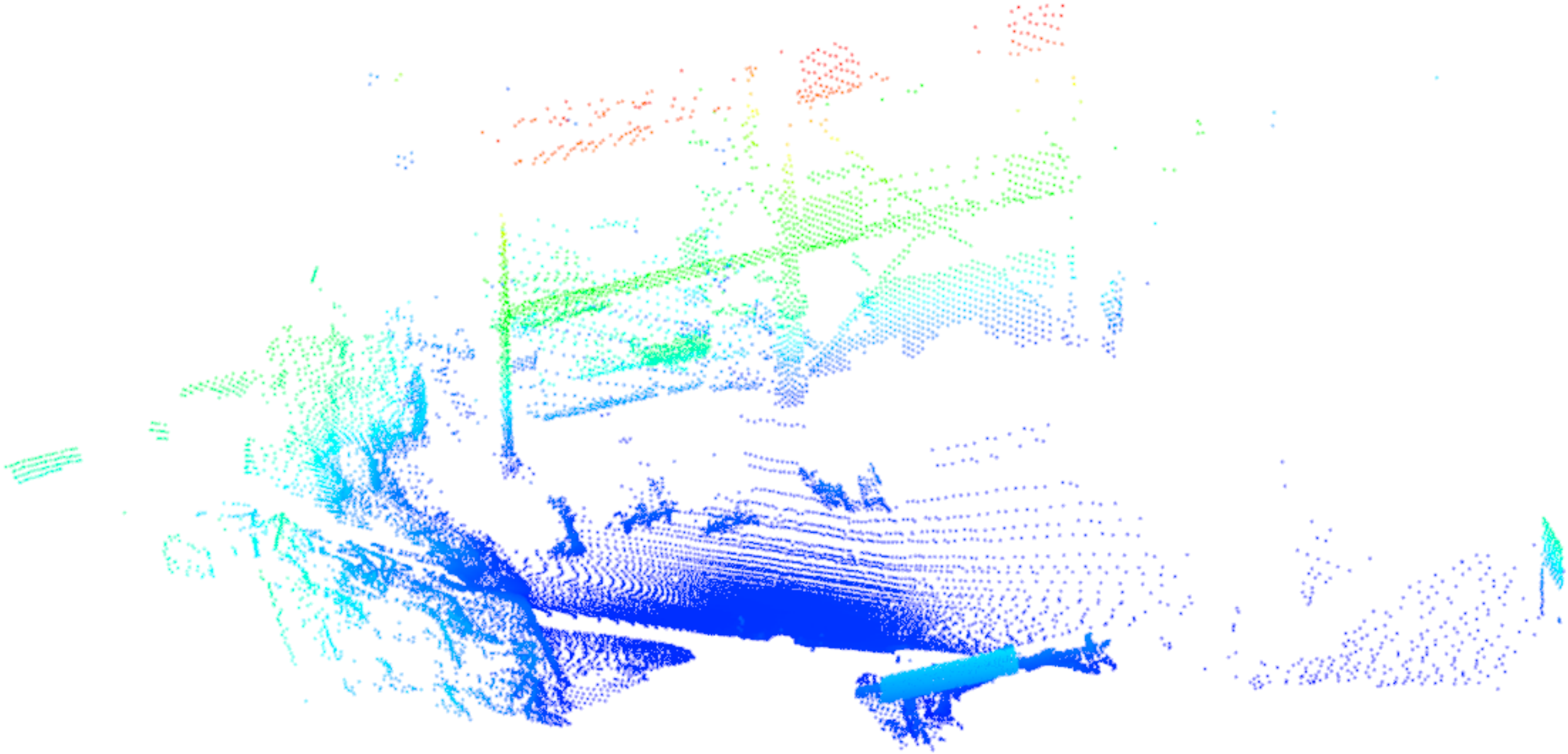
Stanley: Stanford
(winner of the 2005 Darpa Grand Challenge)

Jack the Gripper
Jacobs University – DLR SpaceBot Cup

3D Laser Range Finder (LRF)

- A 3D laser range finder is a laser scanner that acquires scan data in more than a single plane.
- Custom-made 3D scanners are typically built by nodding or rotating a 2D scanner in a stepwise or continuous manner around an axis parallel to the scanning plane.
- By lowering the rotational speed of the turn-table, the angular resolution in the horizontal direction can be made as small as desired.
- A full spherical field of view can be covered (360° in azimuth and 90° in elevation).
- However, acquisition takes up to some seconds!

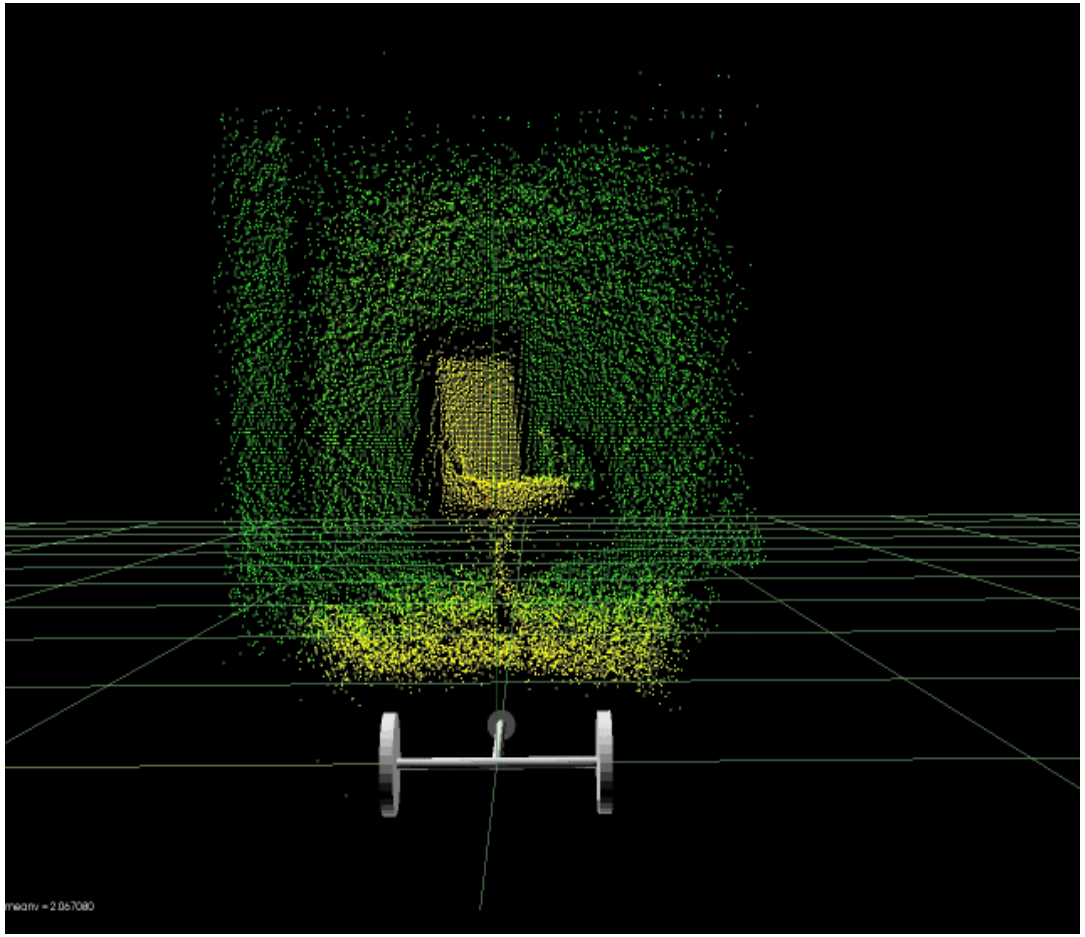




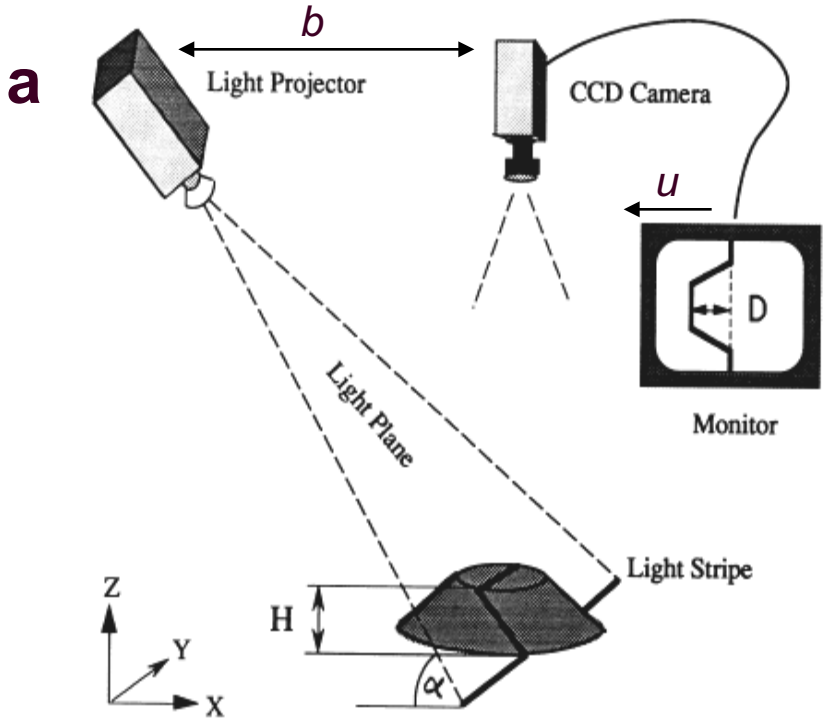
3D Range Sensor: Time Of Flight (TOF) camera

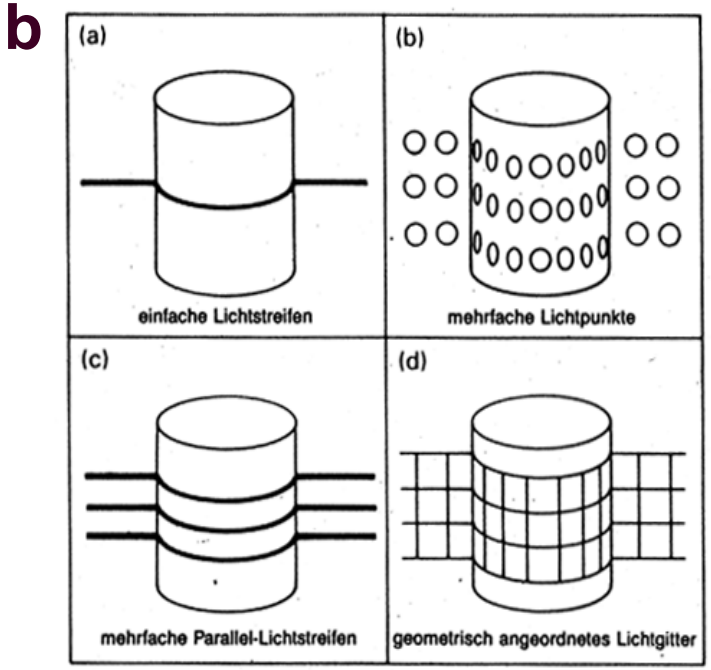
- Range Camera
  - 3D information with high data rate (100 Hz)
  - Compact and easy to manage
  - High, non-uniform measurement noise
  - High outlier rate at jump edges
  - However very low resolution (174x144 pixels)
  - ZCAM achieves 320x240 pixels

  - Sensitive to ambient infrared light!



Structured Light (vision, 2 or 3D): Structured Light

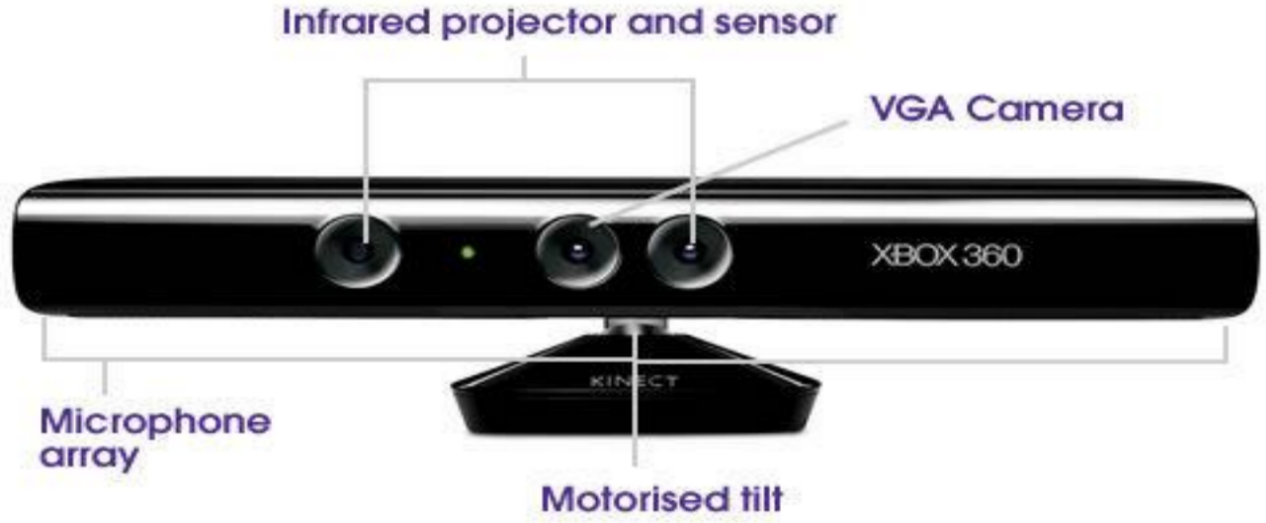




- Eliminate the correspondence problem by projecting structured light on the scene.
- Slits of light or emit collimated light (possibly laser) by means of a rotating mirror.
- Light perceived by camera
- Range to an illuminated point can then be determined from simple geometry.

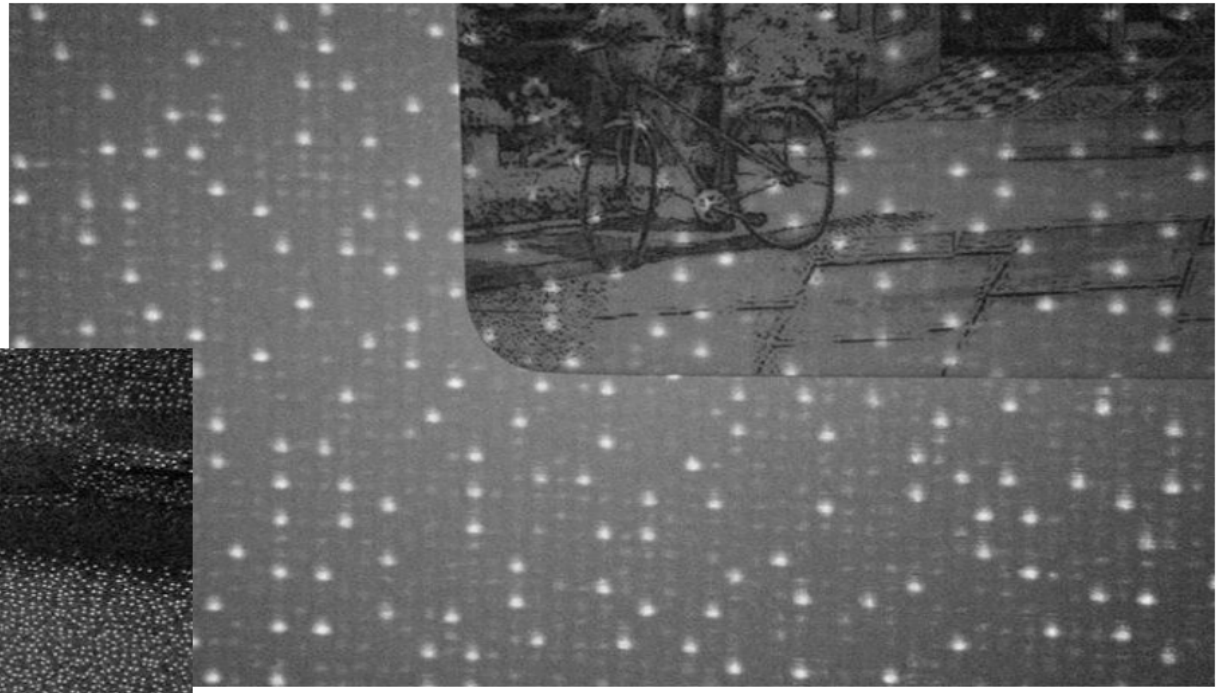
PrimeSense Cameras

- Devices: Microsoft Kinect and Asus Xtion
- Developed by Israeli company PrimeSense in 2010



- Components:
  - IR camera (640 x 480 pixel)
  - IR Laser projector
  - RGB camera (640 x 480 or 1280 x 1024)
  - Field of View (FoV):
    - 57.5 degrees horizontally,
    - 43.5 degrees vertically

IR Pattern

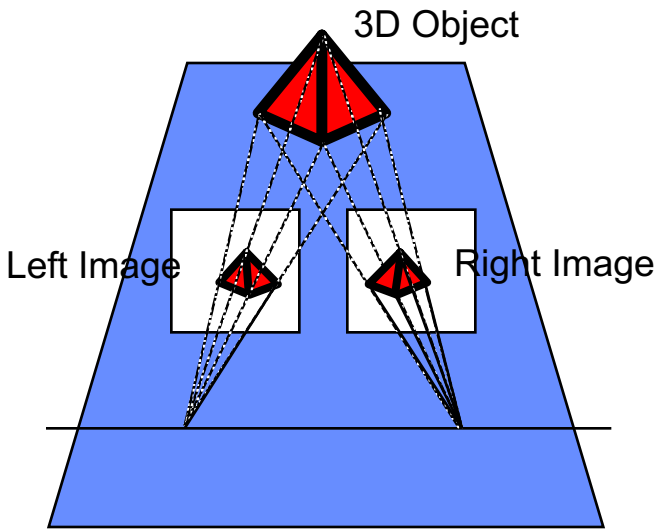


Depth Map

Mapping with Kinect: Challenges

- Kinect & PrimeSense: RGBD camera: RGB color image + Depth
- 640 x 480 x 30 Hz x 5 bytes (3 bytes RGB + 2 Depth):
  - more than 9 million points per second => 44 MB per second
- Small field of view (FoV):
  - 57.5 degrees horizontally,
  - 43.5 degrees vertically
  - => easy to get too little overlap between frames (scans)
- Does not work in sunlight (outdoors!)
- Limited range: nor more than a few meters
- Error goes up with higher ranges

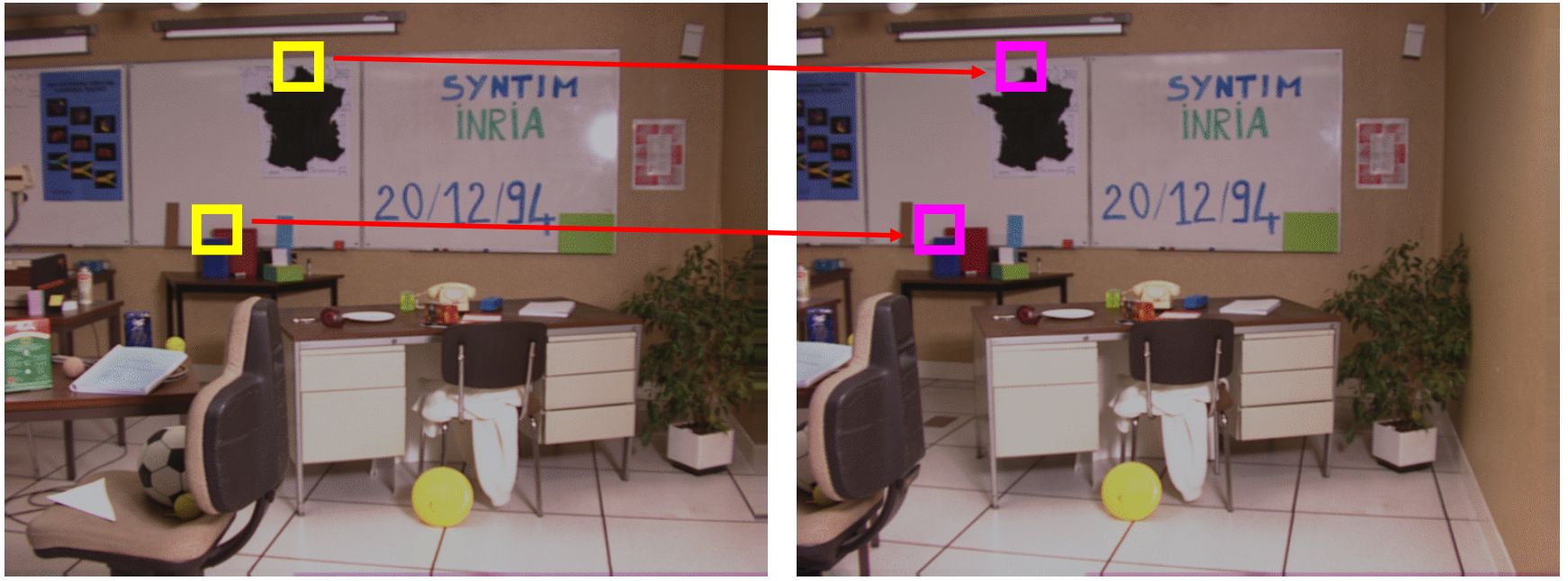
3D Vision Sensor: Stereo Vision



1. Stereo camera calibration -> compute camera relative pose
2. Epipolar rectification -> align images
3. Search correspondences
4. Output: compute stereo triangulation or disparity map
5. Consider baseline and image resolution to compute accuracy!

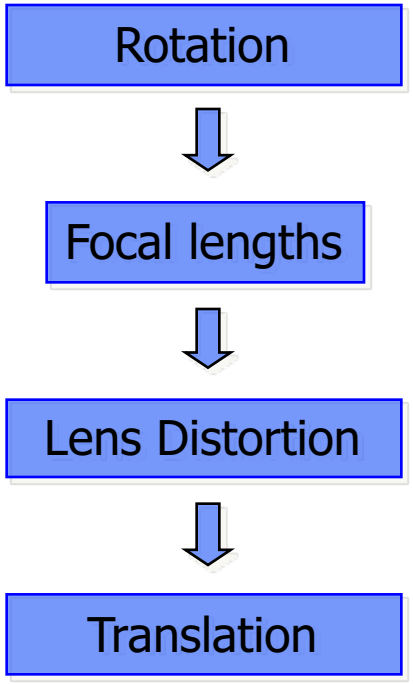
Stereo Vision: Correspondence Problem

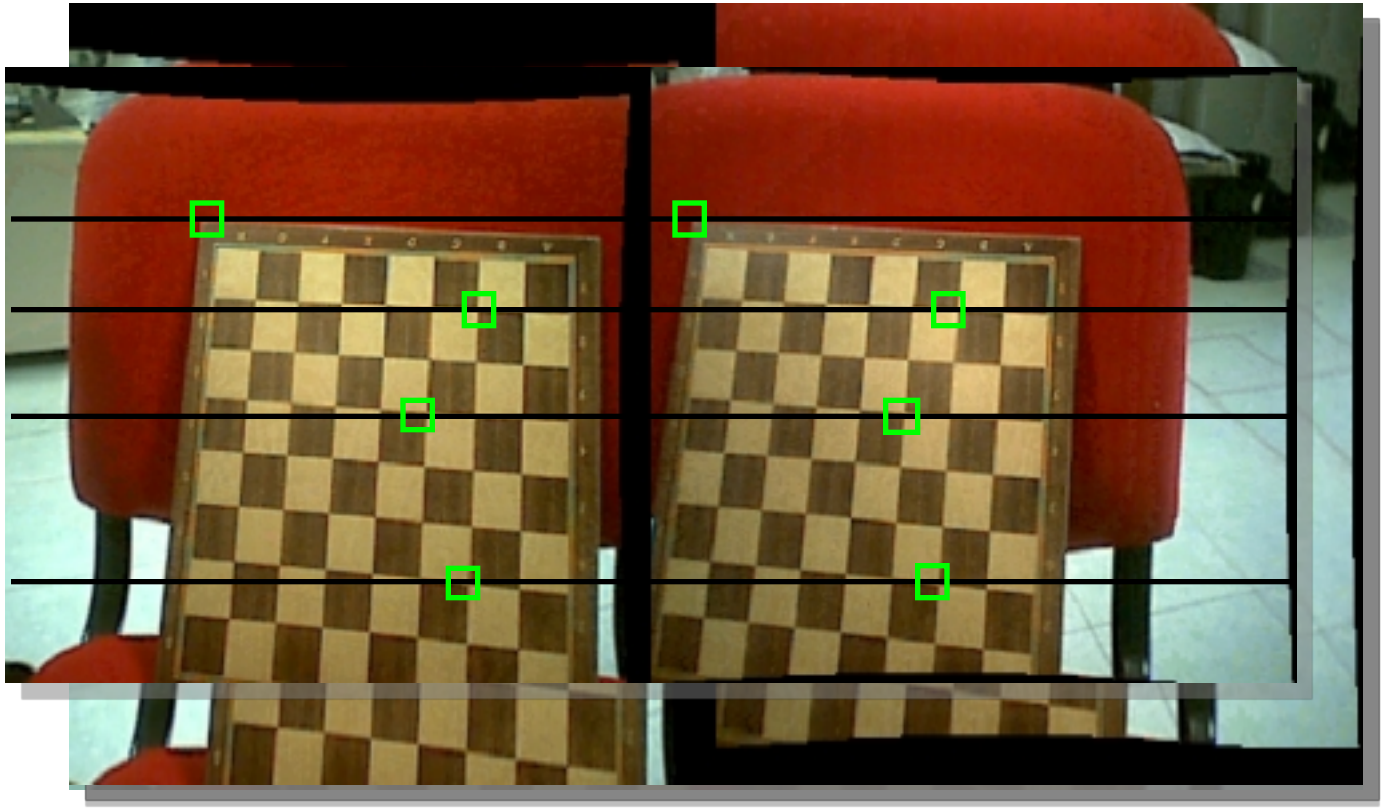
- Matching between points in the two images which are projection of the same 3D real point
- Correspondence search could be done by comparing the observed points with all other points in the other image. Typical similarity measures are the Correlation and image Difference.
- This image search can be computationally very expensive!



Epipolar Rectification

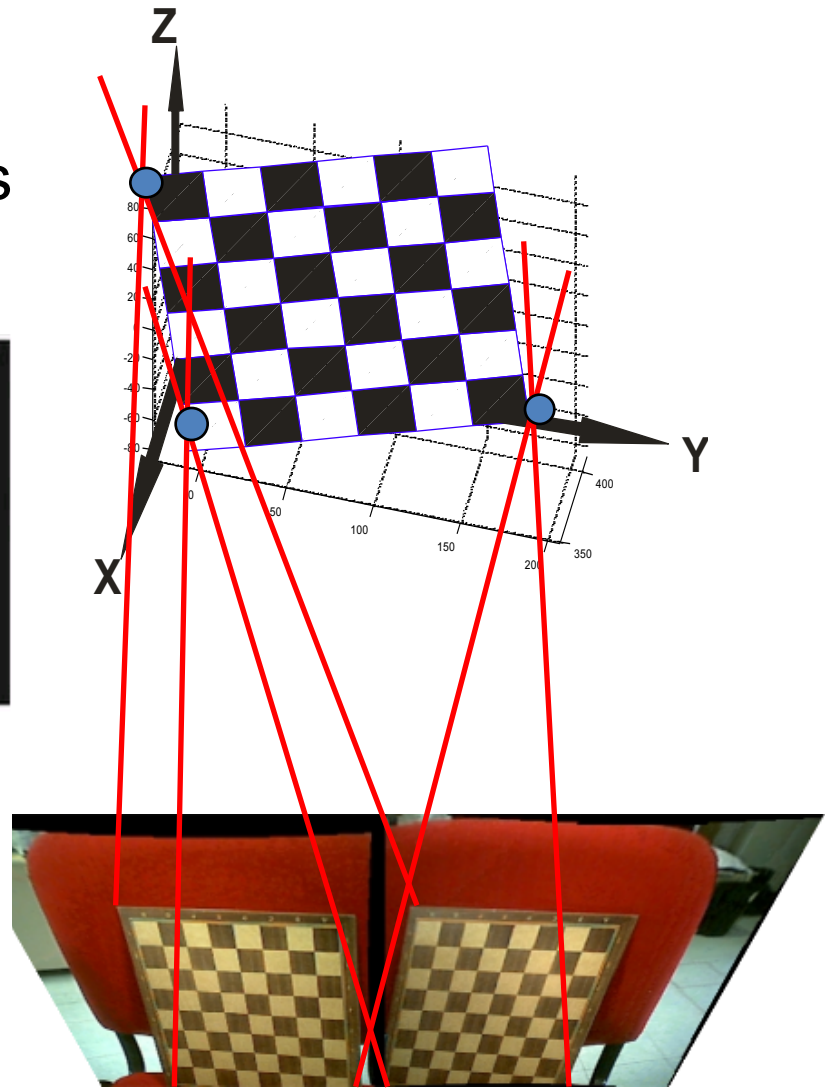
- Determines a transformation of each image plane so that pairs of conjugate epipolar lines become collinear and parallel to one of the image axes (usually the horizontal one)





Stereo Vision Output - 3D Reconstruction via triangulation

- Problems for Robotics:
  - High computation costs
  - Needs to find matching features/ correspondences
  - Only works for short ranges (baseline!)



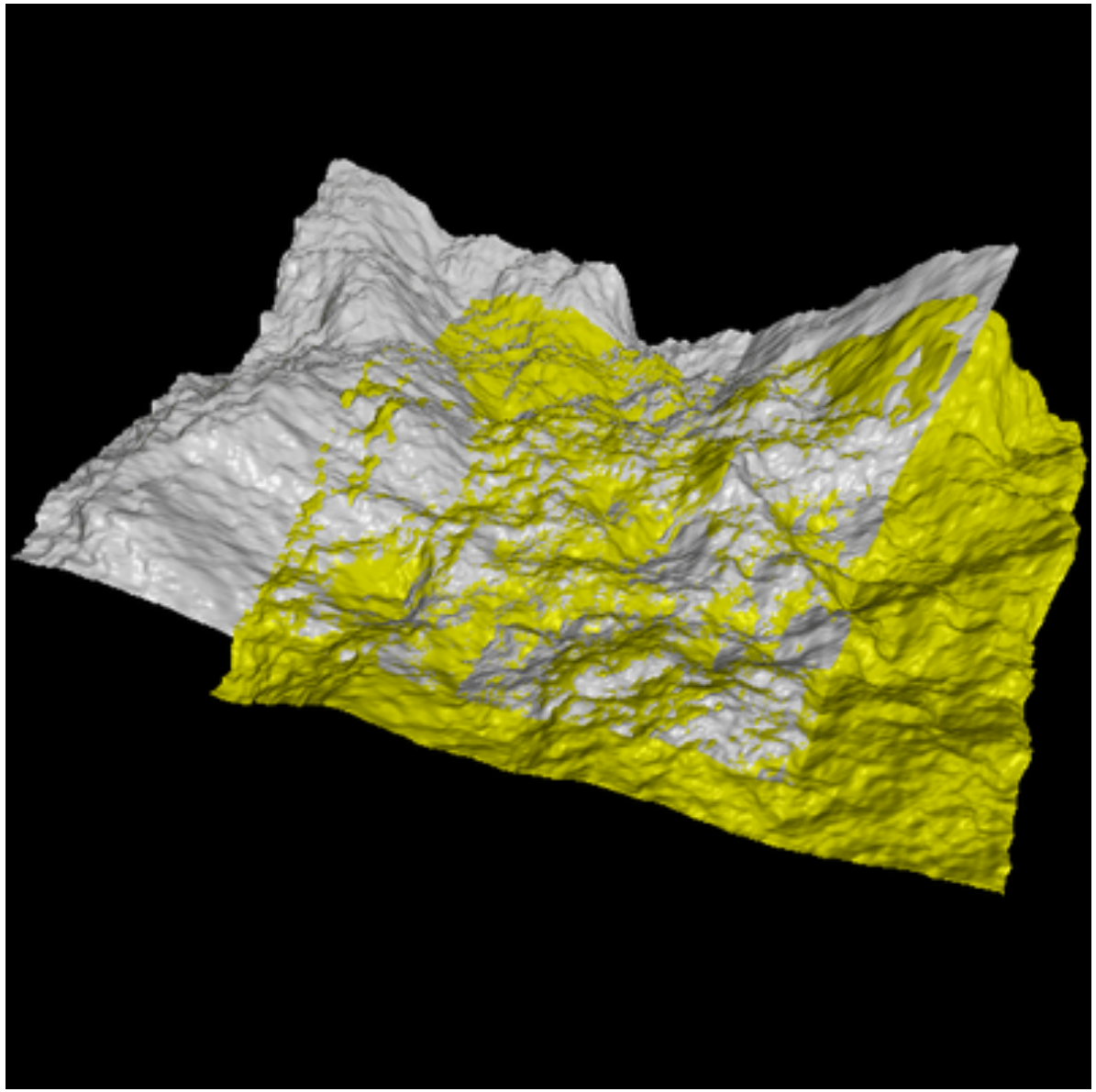
- http://wiki.ros.org/stereo_image_proc
- http://wiki.ros.org/viso2_ros

SCAN MATCHING

Iterative Closest Point - ICP

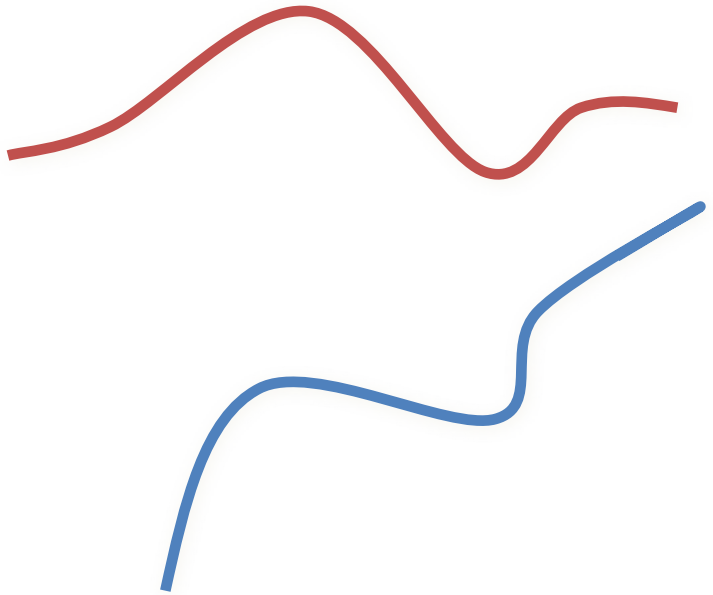
The Problem

- Align two partially overlapping meshes (or point clouds) given initial guess of relative transform



Material from Ronen Gvili

Aligning Scans => Scan Matching

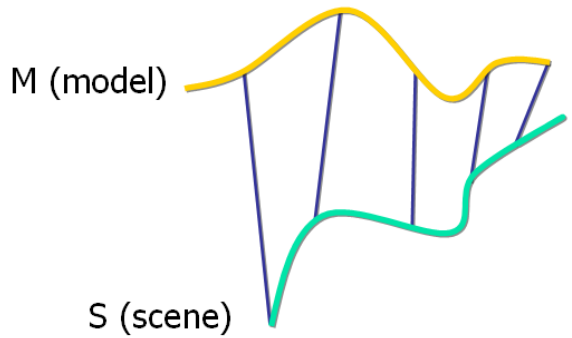


Corresponding Point Set Alignment

- Let M be a model point set.
- Let S be a scene point set.

We assume :

1. $N_M = N_S$.
2. Each point S_i correspond to M_i .



Corresponding Point Set Alignment

The MSE objective function :

$$f(R,T) = \frac{1}{N_S} \sum_{i=1}^{N_S} \left\| m_i \quad Rot(s_i) \quad Trans \right\|^2$$

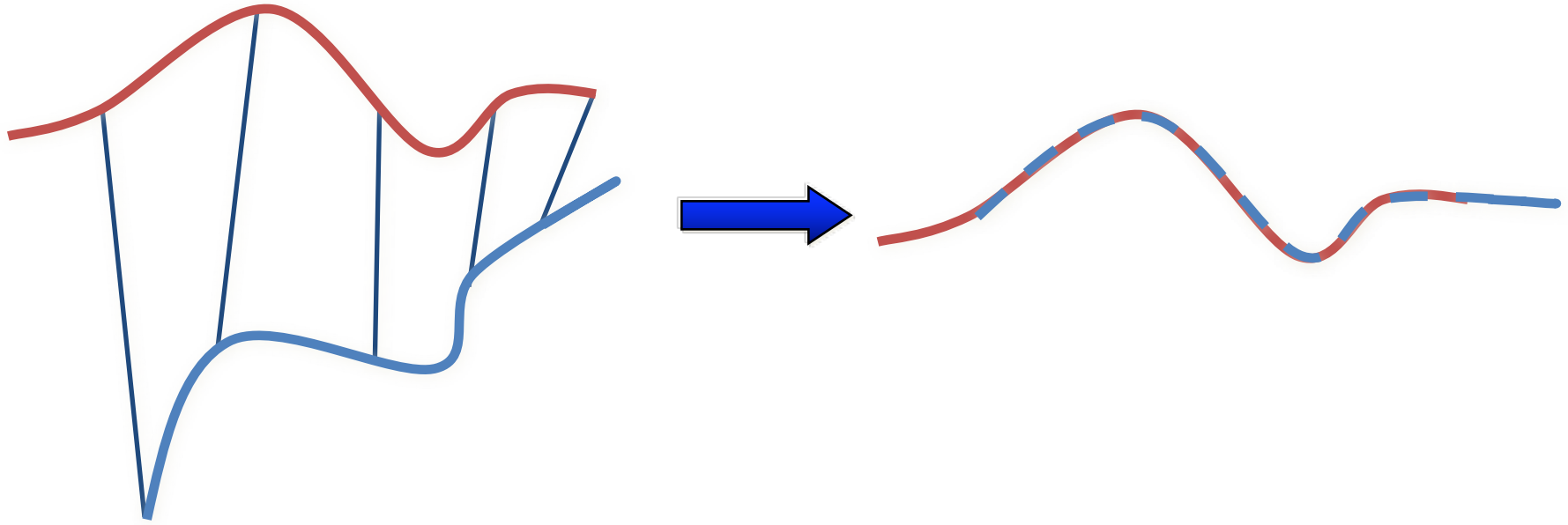
$$f(q) = \frac{1}{N_S} \sum_{i=1}^{N_S} \left\| m_i \quad R(q_R)s_i \quad q_T \right\|^2$$

The alignment is :

$$(rot, trans, d_{mse}) = \quad (M, S)$$

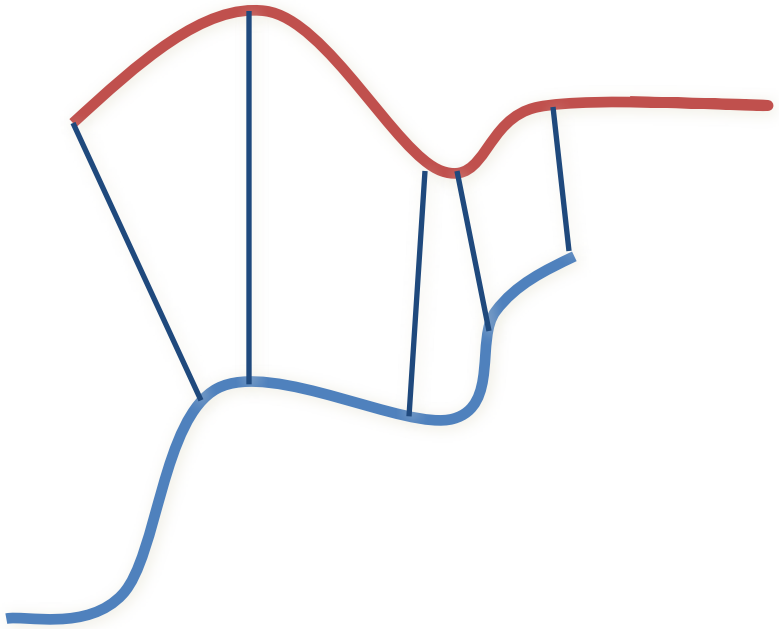
Aligning Scans

- If correct correspondences are known, can find correct relative rotation/translation



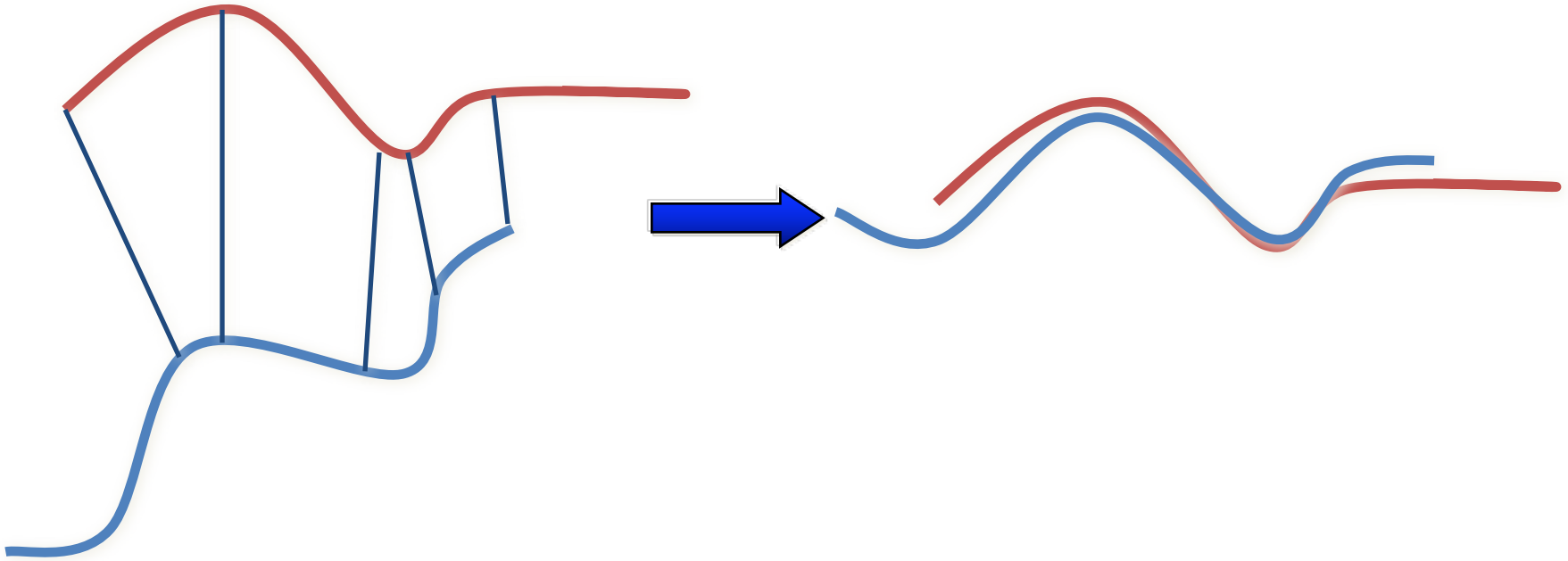
Aligning Scans

- How to find correspondences: User input? Feature detection? Signatures?
- Alternative: assume closest points correspond



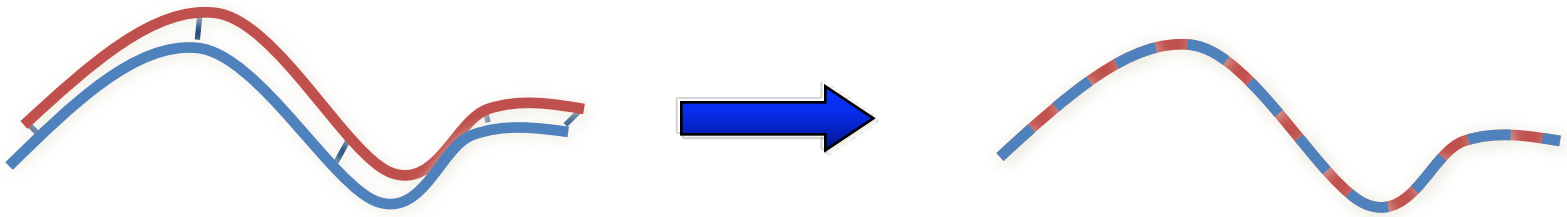
Aligning Scans

- How to find correspondences: User input? Feature detection? Signatures?
- Alternative: assume closest points correspond



Aligning Scans

- Converges if starting position “close enough“



Closest Point

- Given 2 points r_1 and r_2 , the Euclidean distance is:

$$d(r_1, r_2) = \|r_1 \quad r_2\| = \sqrt{(x_1 \quad x_2)^2 + (y_1 \quad y_2)^2 + (z_1 \quad z_2)^2}$$

- Given a point r_1 and set of points A , the Euclidean distance is:

$$d(r_1, A) = \min_{i \ 1..n} d(r_1, a_i)$$

Finding Matches

- The scene shape S is aligned to be in the best alignment with the model shape M.
- The distance of each point s of the scene from the model is :

$$d(s,M) = \min_{m \quad M} d\|m \quad s\|$$

Finding Matches

$$d(s,M) = \min_{m \quad M} d\|m \quad s\| = d(s,y)$$

$$y \quad M$$

$$Y = C(S,M)$$

$$Y \quad M$$

C – the closest point operator

Y – the set of closest points to S

Finding Matches

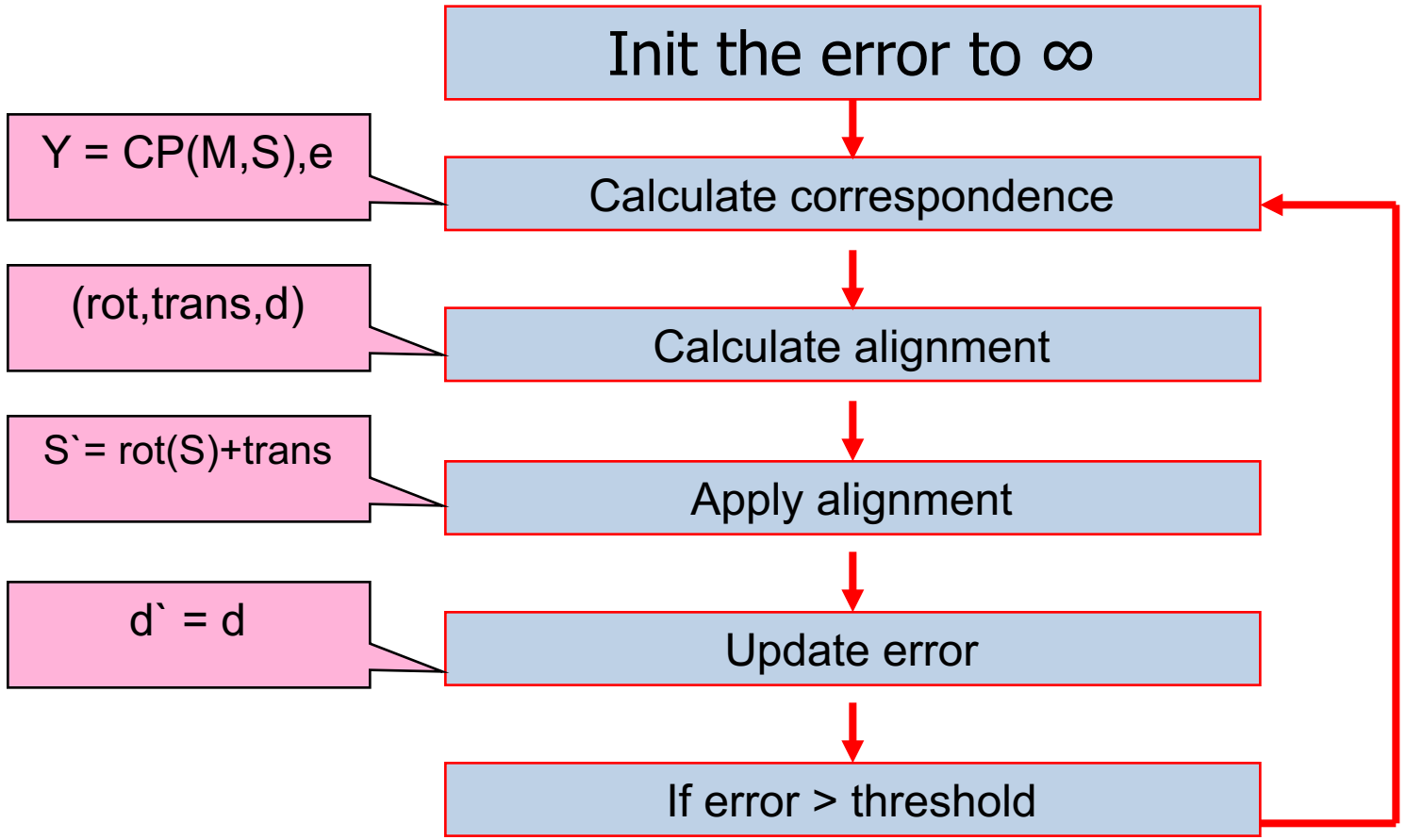
- Finding each match is performed in $O(N_M)$ worst case.
- Given Y we can calculate alignment

$$(rot, trans, d) = \quad (S, Y)$$

- S is updated to be :

$$S_{new} = rot(S) + trans$$

The Algorithm

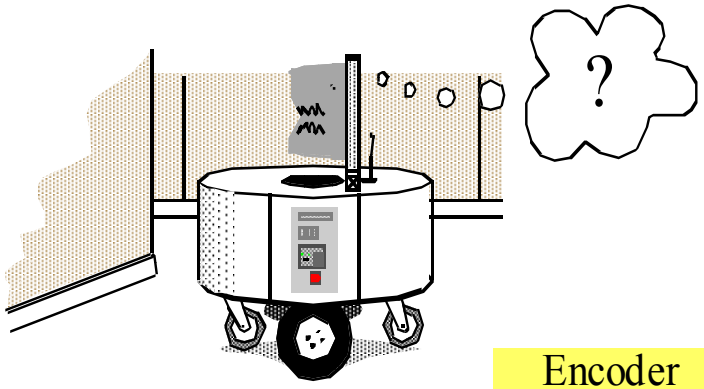


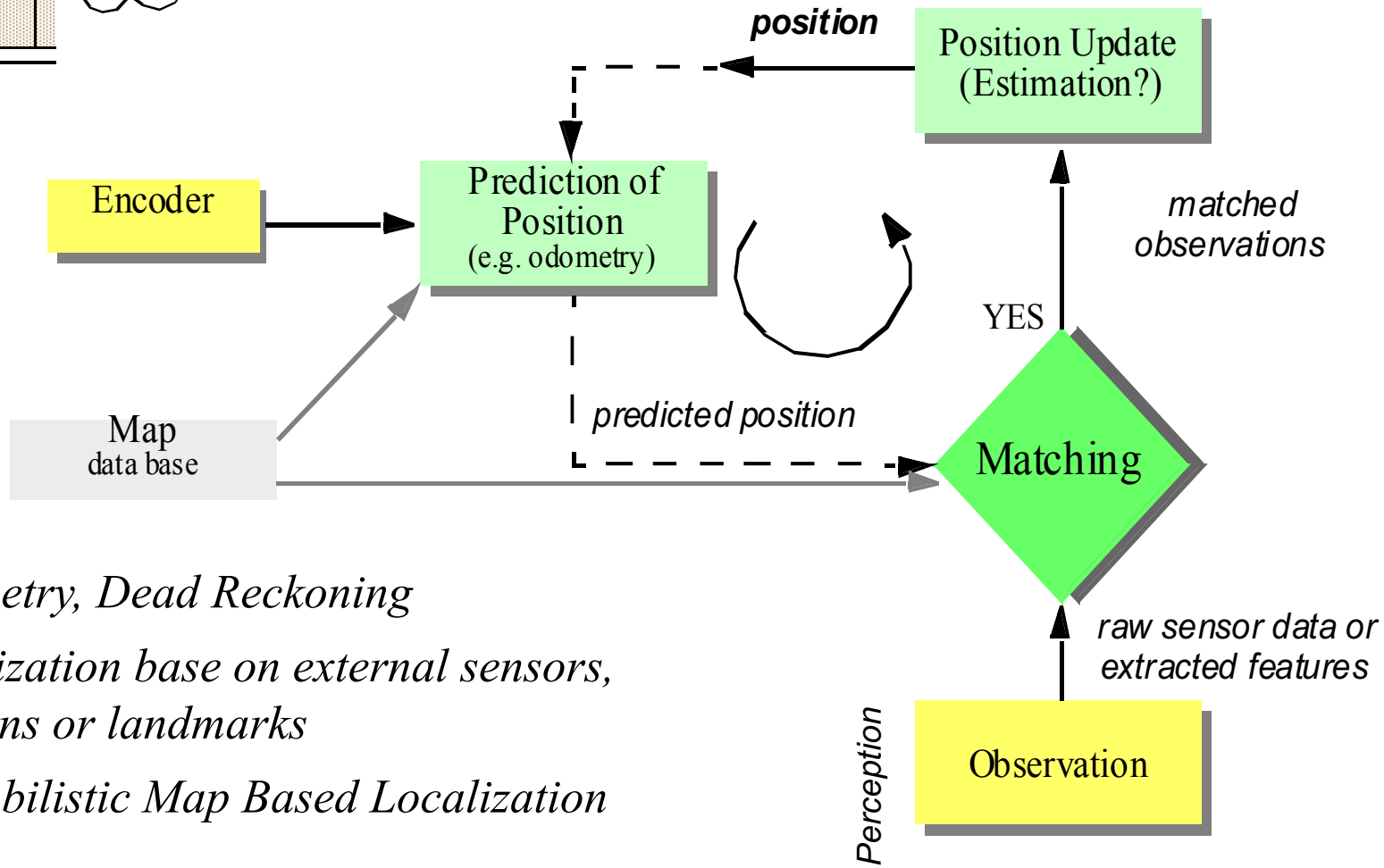
The Algorithm

```
function ICP(Scene,Model)
begin
E` ← + ∞;
(Rot,Trans) ← In Initialize-Alignment(Scene,Model);
repeat
        E ← E`;
        Aligned-Scene ← Apply-Alignment(Scene,Rot,Trans);
        Pairs ← Return-Closest-Pairs(Aligned-Scene,Model);
        (Rot,Trans,E`) ← Update-Alignment(Scene,Model,Pairs,Rot,Trans);
Until |E`- E|  < Threshold
return (Rot,Trans);
end
```

LOCALIZATION

Map based localization





- *Odometry, Dead Reckoning*
- *Localization base on external sensors, beacons or landmarks*
- *Probabilistic Map Based Localization*

Exteroceptive Sensor Noise

- Sensor noise is mainly influenced by environment
  e.g. surface, illumination …
- and by the measurement principle itself
  e.g. interference two Kinects
- Sensor noise drastically reduces the useful information of sensor readings.
  The solution is:
  - to model sensor noise appropriately
  - to take multiple readings into account
  - employ temporal and/or multi-sensor fusion

Effector Noise: Odometry, Deduced Reckoning

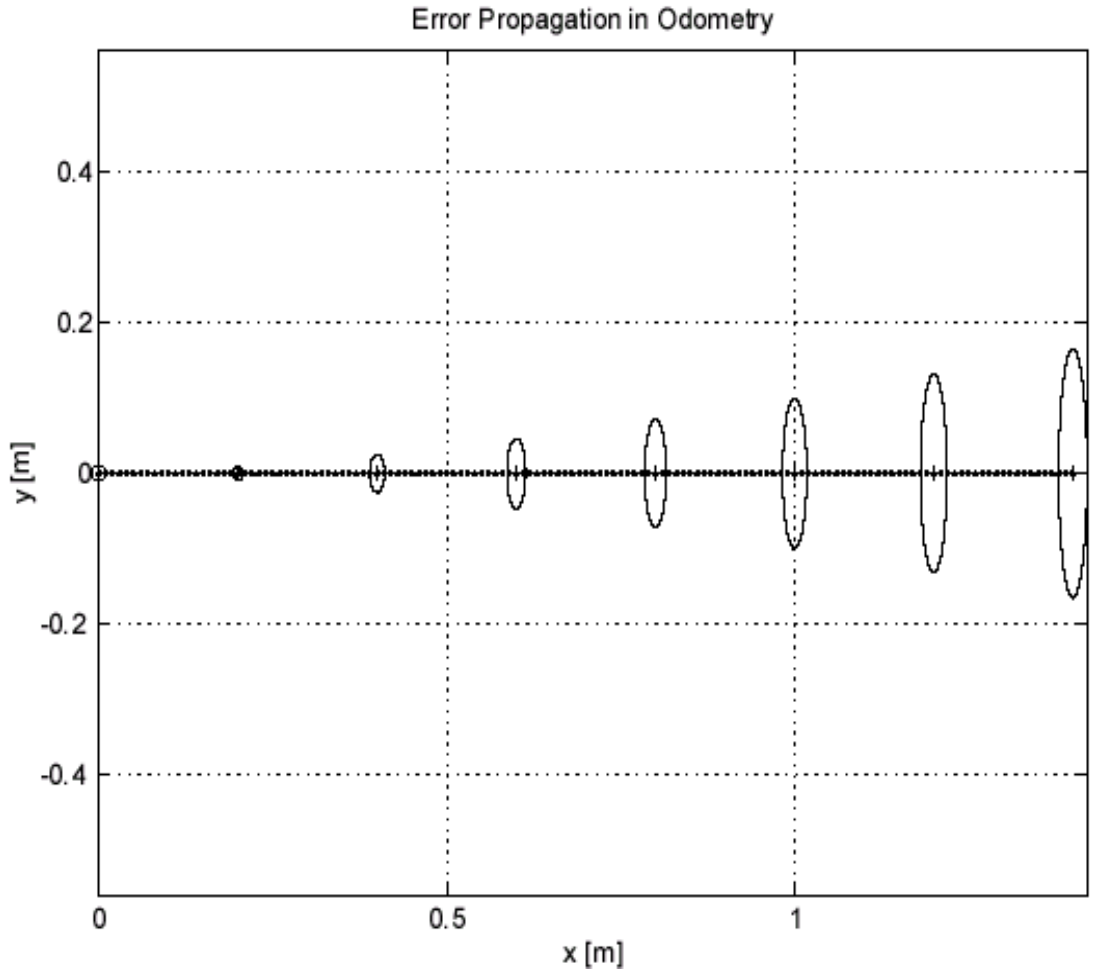
- Odometry and dead reckoning:
  Position update is based on proprioceptive sensors
  - Odometry: wheel sensors only
  - Dead reckoning: also heading sensors

- The movement of the robot, sensed with wheel encoders and/or heading sensors is integrated to the position.
  - Pros: Straight forward, easy
  - Cons: Errors are integrated -> unbound

- Using additional heading sensors (e.g. gyroscope) might help to reduce the cumulated errors, but the main problems remain the same.

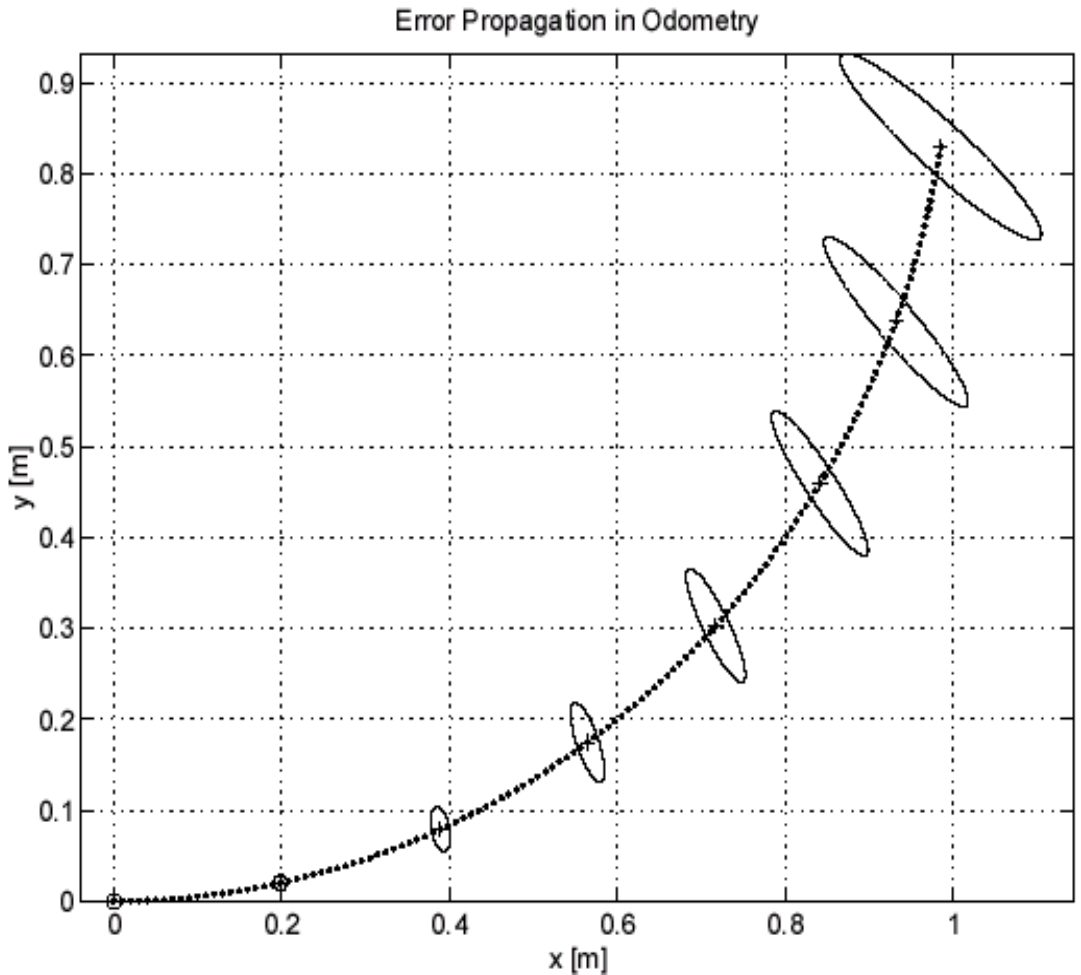
Odometry: Growth of Pose uncertainty for Straight Line Movement

- Note: Errors perpendicular to the direction of movement are growing much faster!



Odometry: Growth of Pose uncertainty for Movement on a Circle

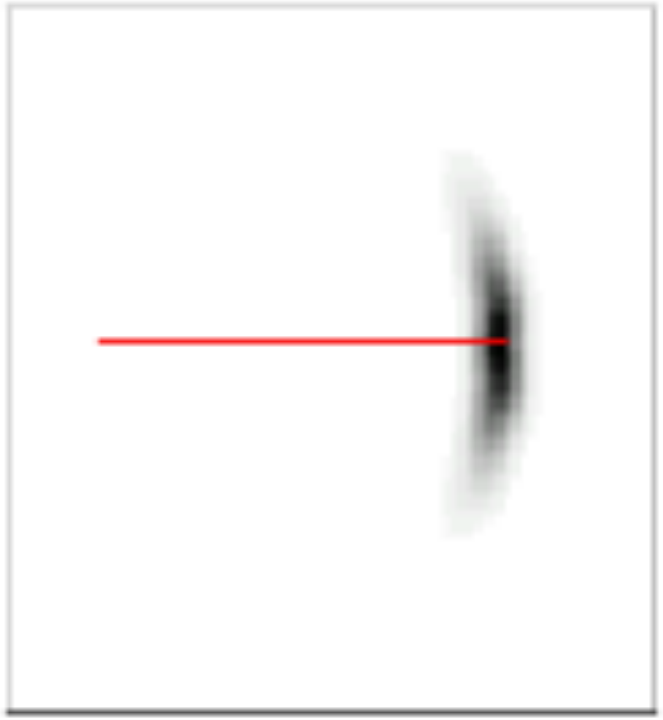
- Note: Errors ellipse in does not remain perpendicular to the direction of movement!

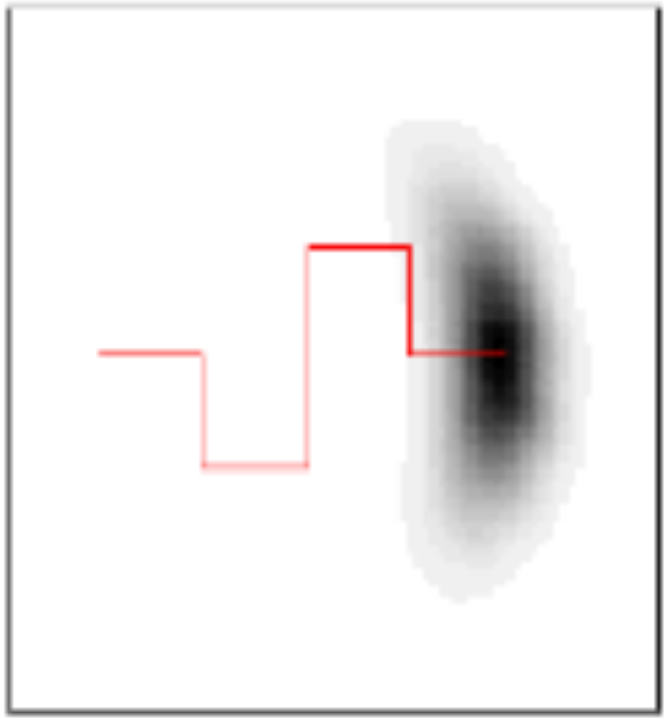


Odometry: example of non-Gaussian error model

- Note: Errors are not shaped like ellipses!

Courtesy AI Lab, Stanford

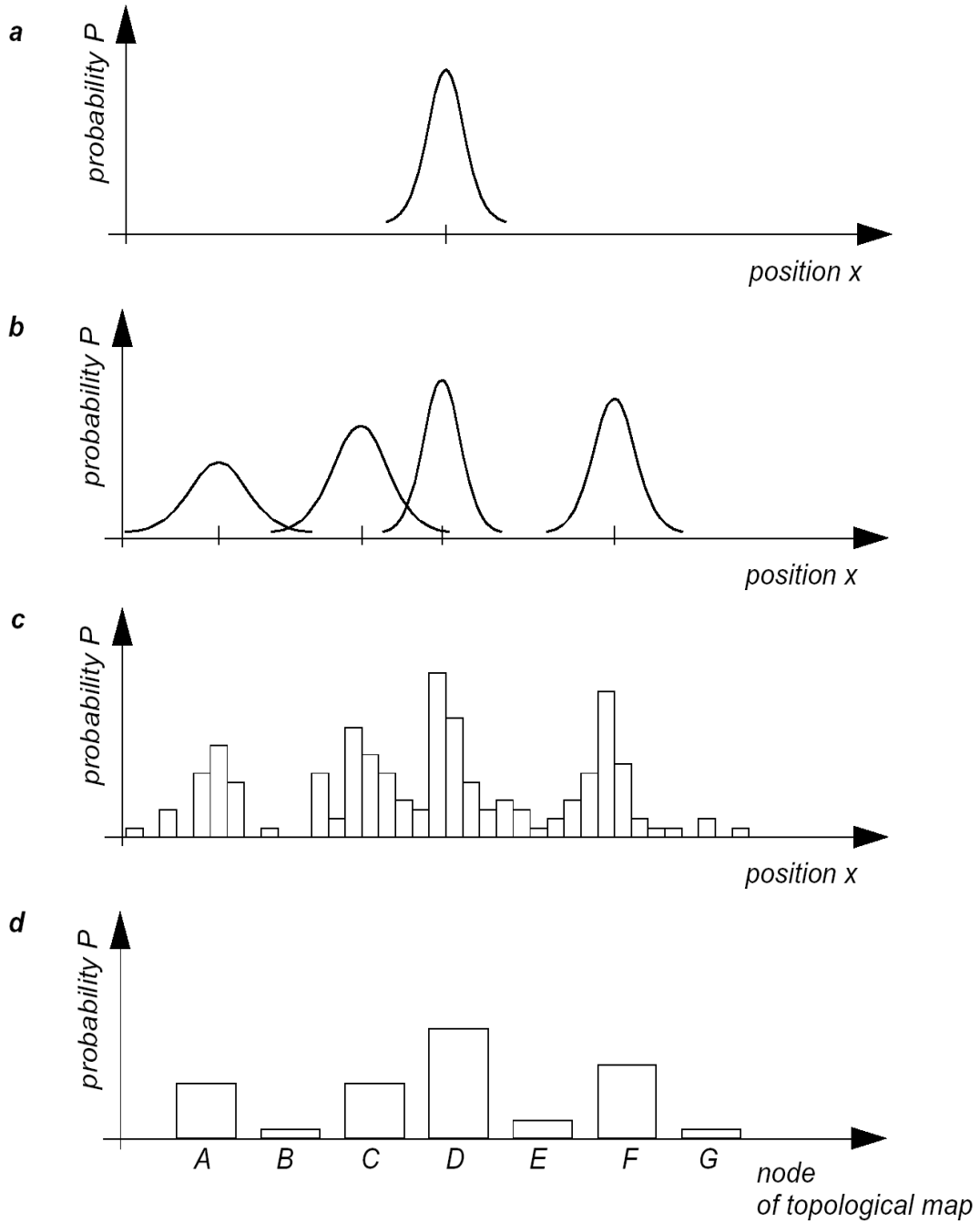




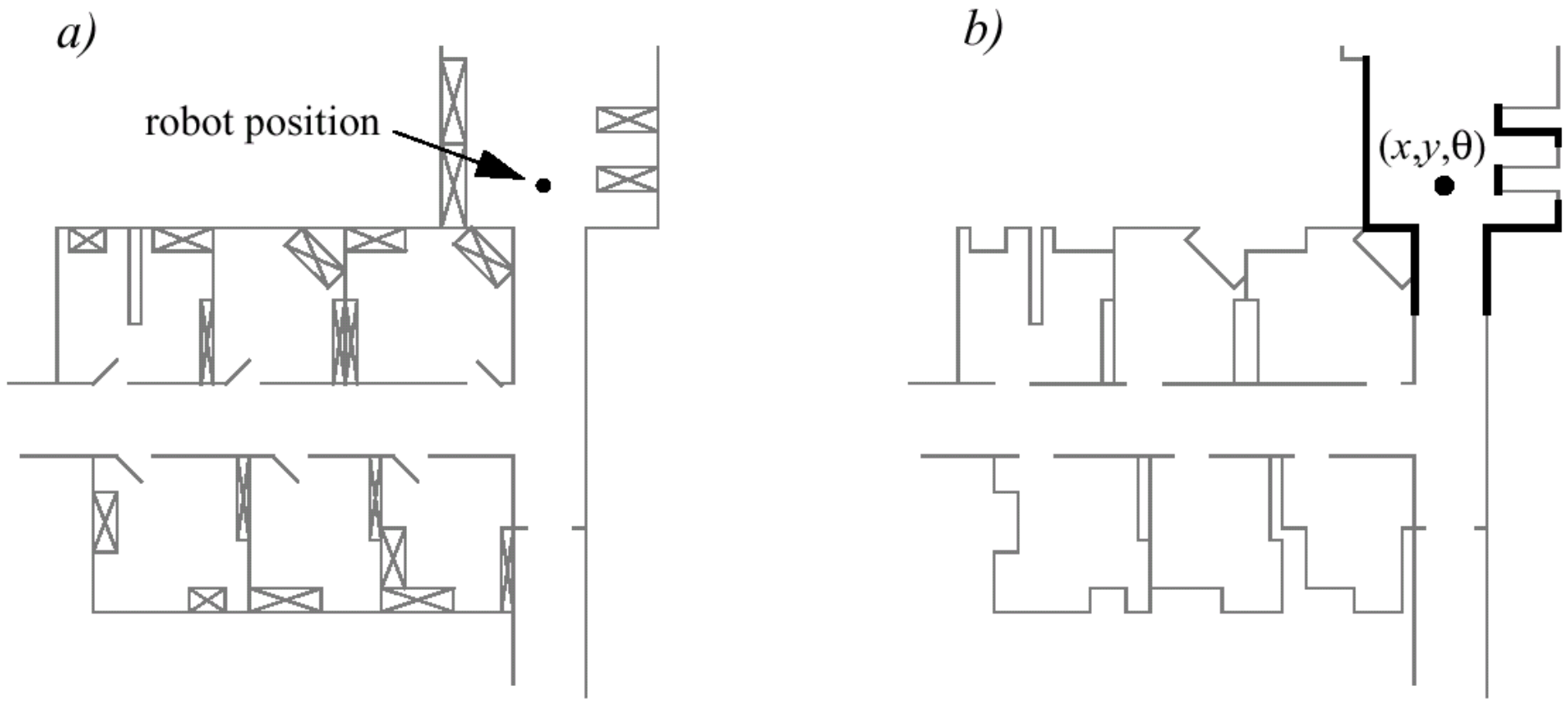
[Fox, Thrun, Burgard, Dellaert, 2000]

Belief Representation

- How do we represent the robot position, where the robot “believes” to be?
- a) Continuous map with single hypothesis probability distribution
- b) Continuous map with multiple hypothesis probability distribution
- c) Discretized map with probability distribution
- d) Discretized topological map with probability distribution

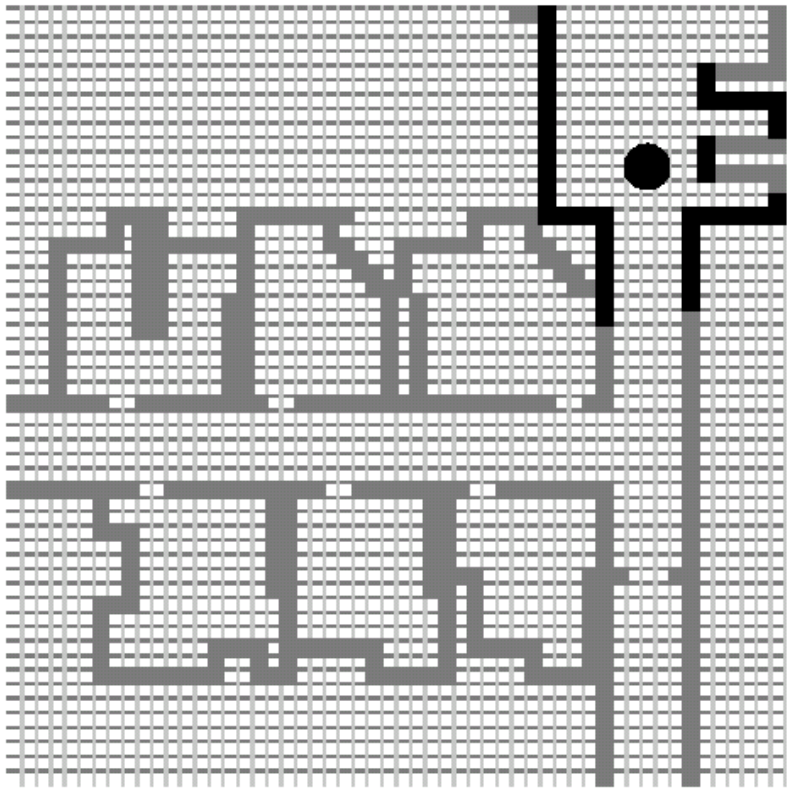


Single-hypothesis Belief – Continuous Line-Map

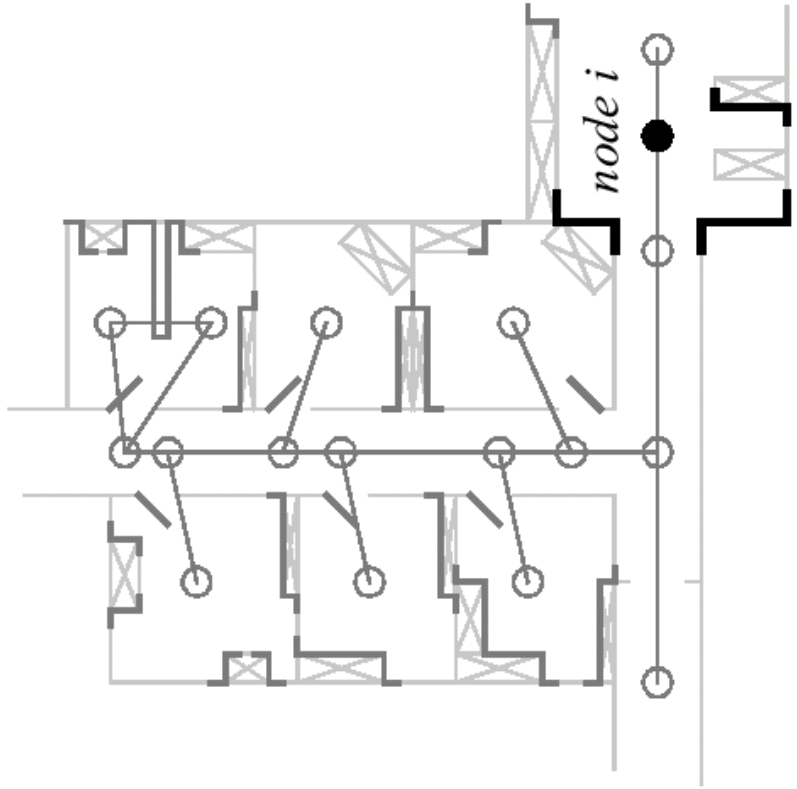


Single-hypothesis Belief – 2D Grid and Topological Map

c)

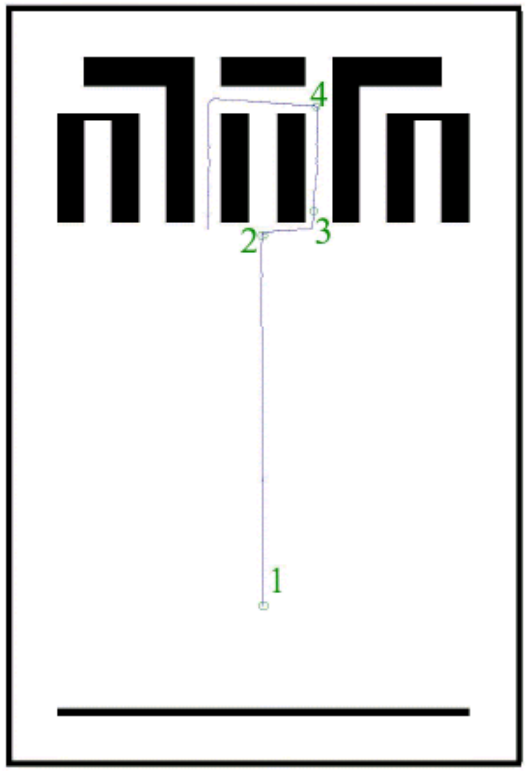


d)

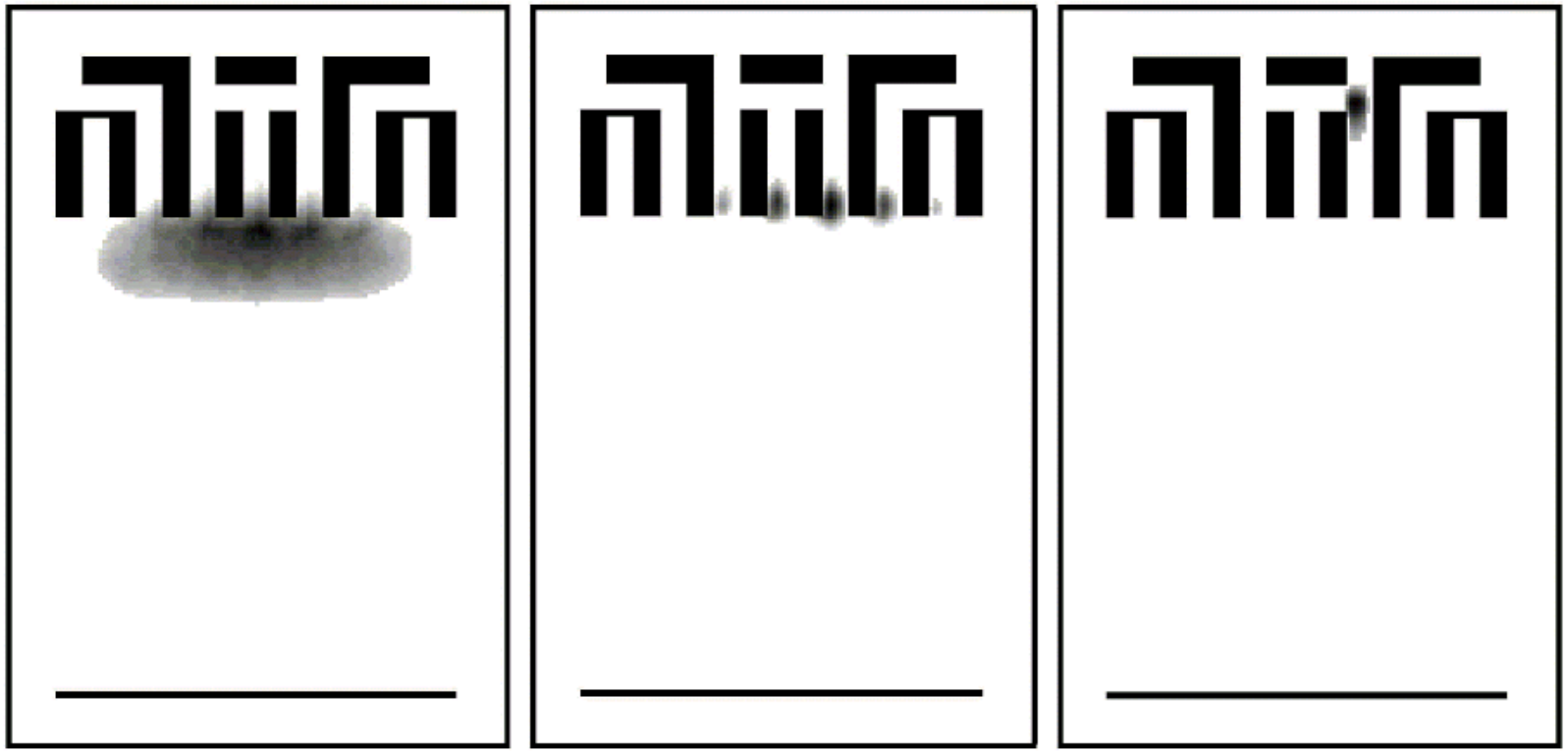


Grid-based Representation - Multi Hypothesis

Courtesy of W. Burgard



Path of the robot



Belief states at positions 2, 3 and 4

Representation of the Environment

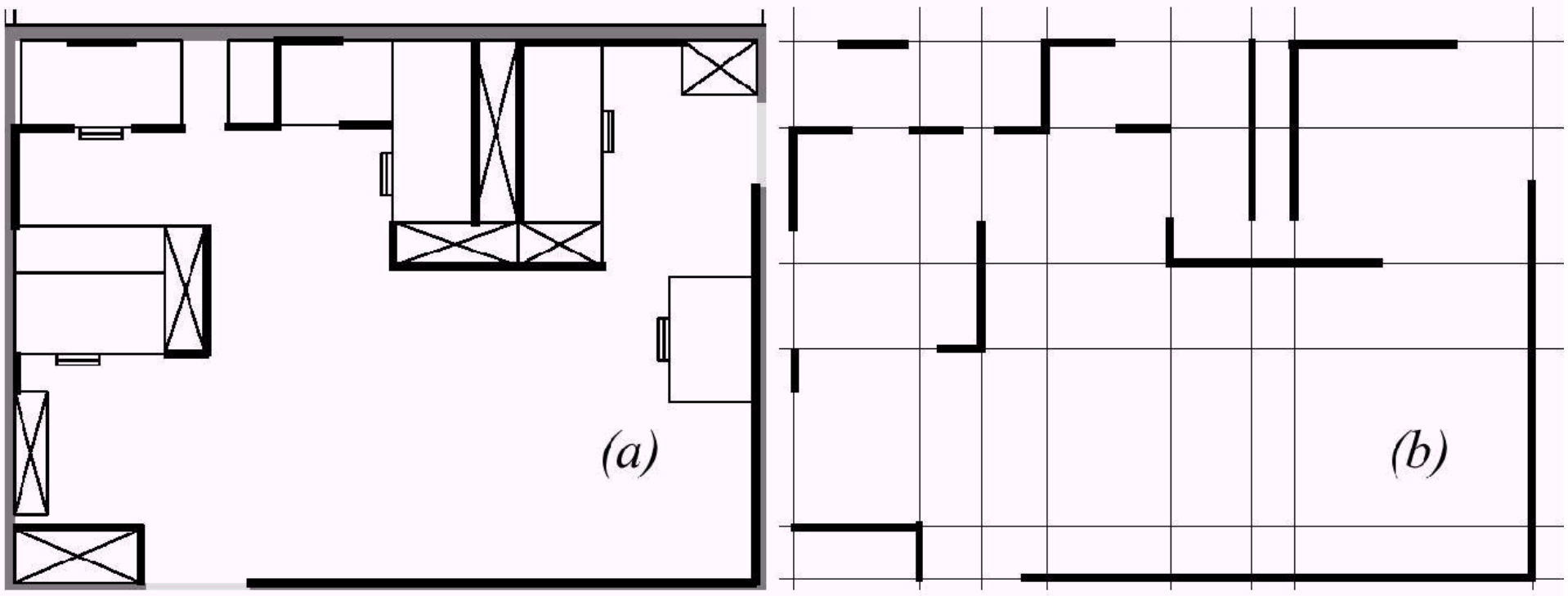
- Environment Representation
  - Continuous Metric → x, y, θ
  - Discrete Metric → metric grid
  - Discrete Topological → topological grid

- Environment Modeling
  - Raw sensor data, e.g. laser range data, grayscale images
    - large volume of data, low distinctiveness on the level of individual values
    - makes use of all acquired information
  - Low level features, e.g. line other geometric features
    - medium volume of data, average distinctiveness
    - filters out the useful information, still ambiguities
  - High level features, e.g. doors, a car, the Eiffel tower
    - low volume of data, high distinctiveness
    - filters out the useful information, few/no ambiguities, not enough information

Map Representation: Continuous Line-Based

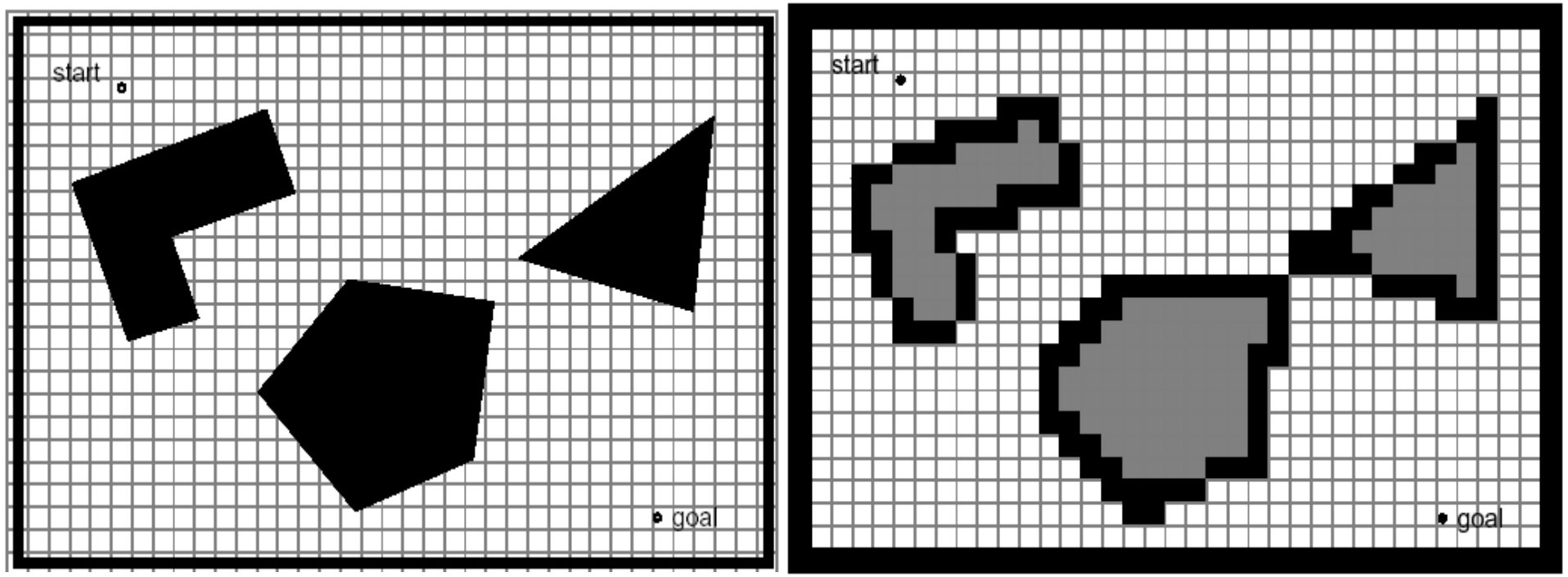
a) Architecture map

b) Representation with set of finite or infinite lines



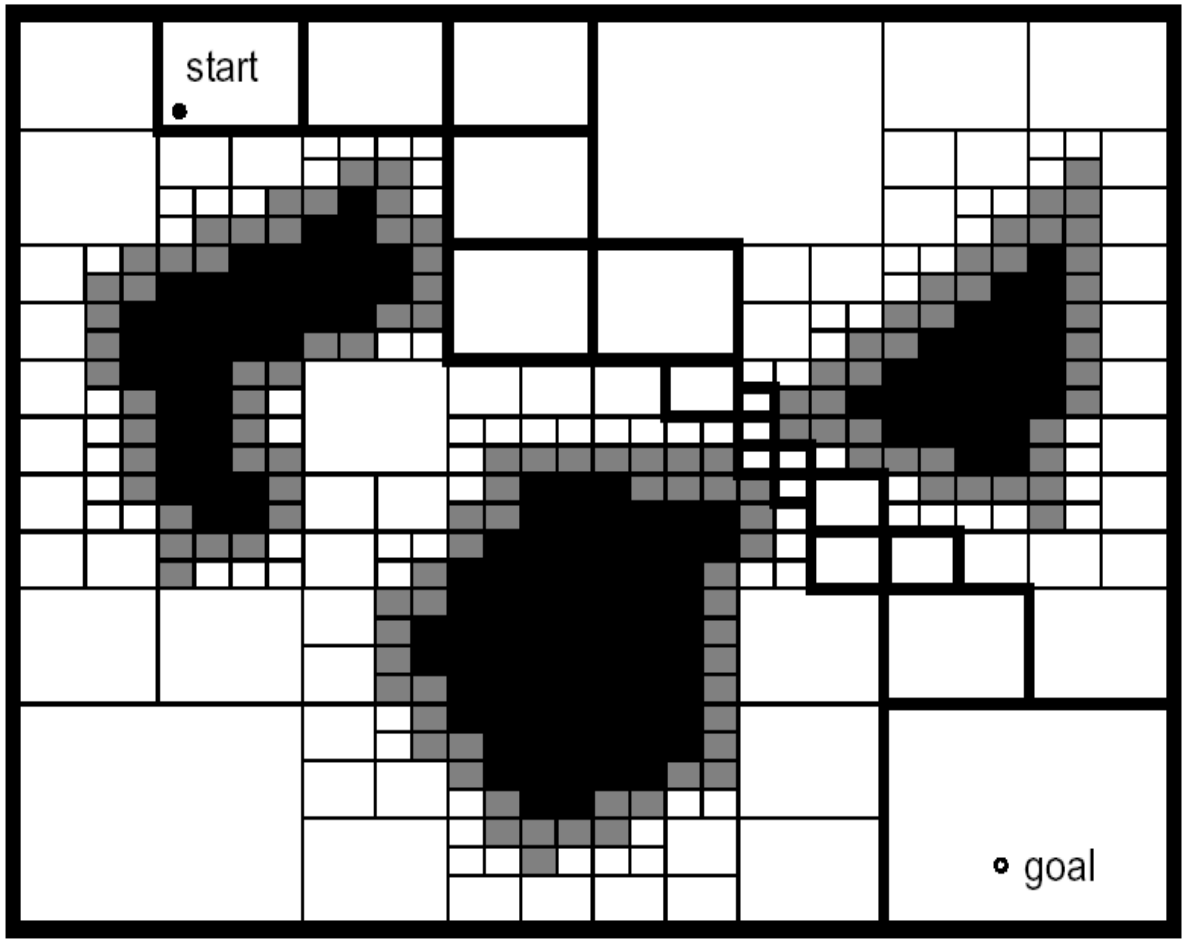
Map Representation: Approximate cell decomposition (1)

- Fixed cell decomposition
  - Narrow passages disappear



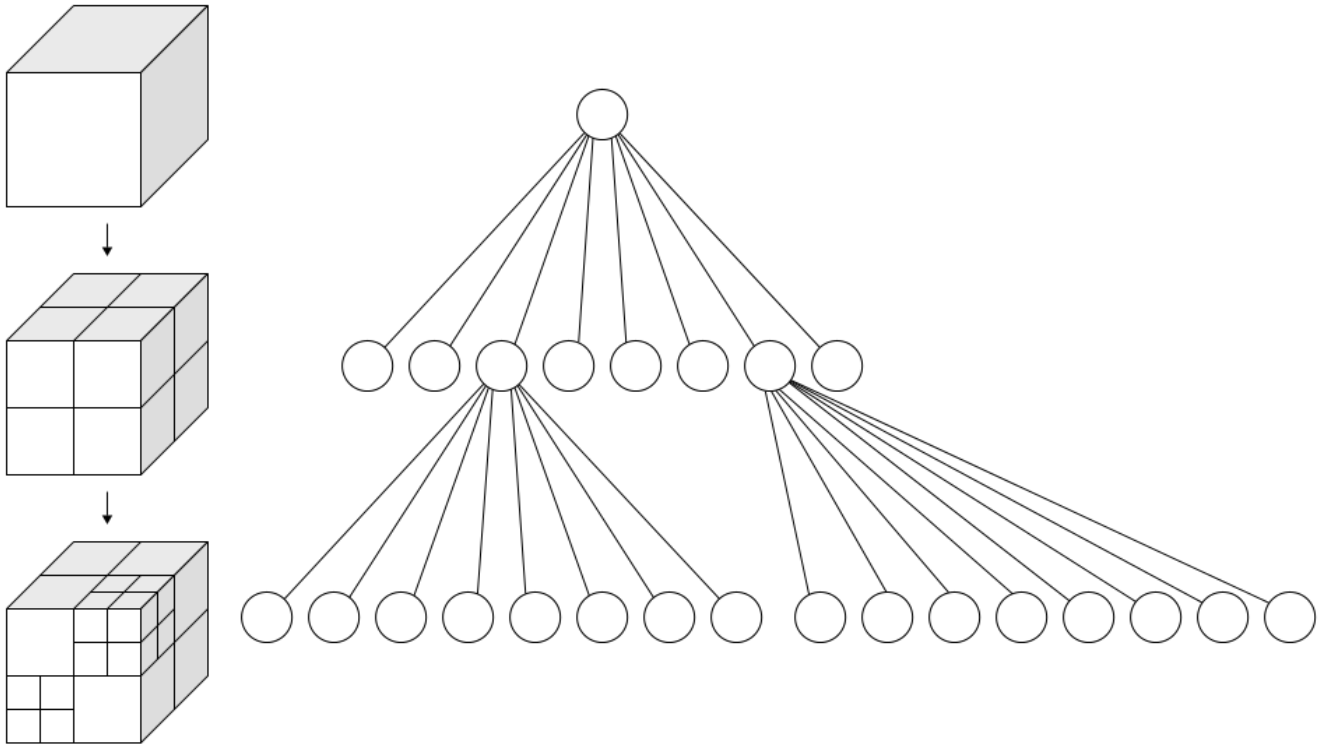
Map Representation: Adaptive cell decomposition (2)

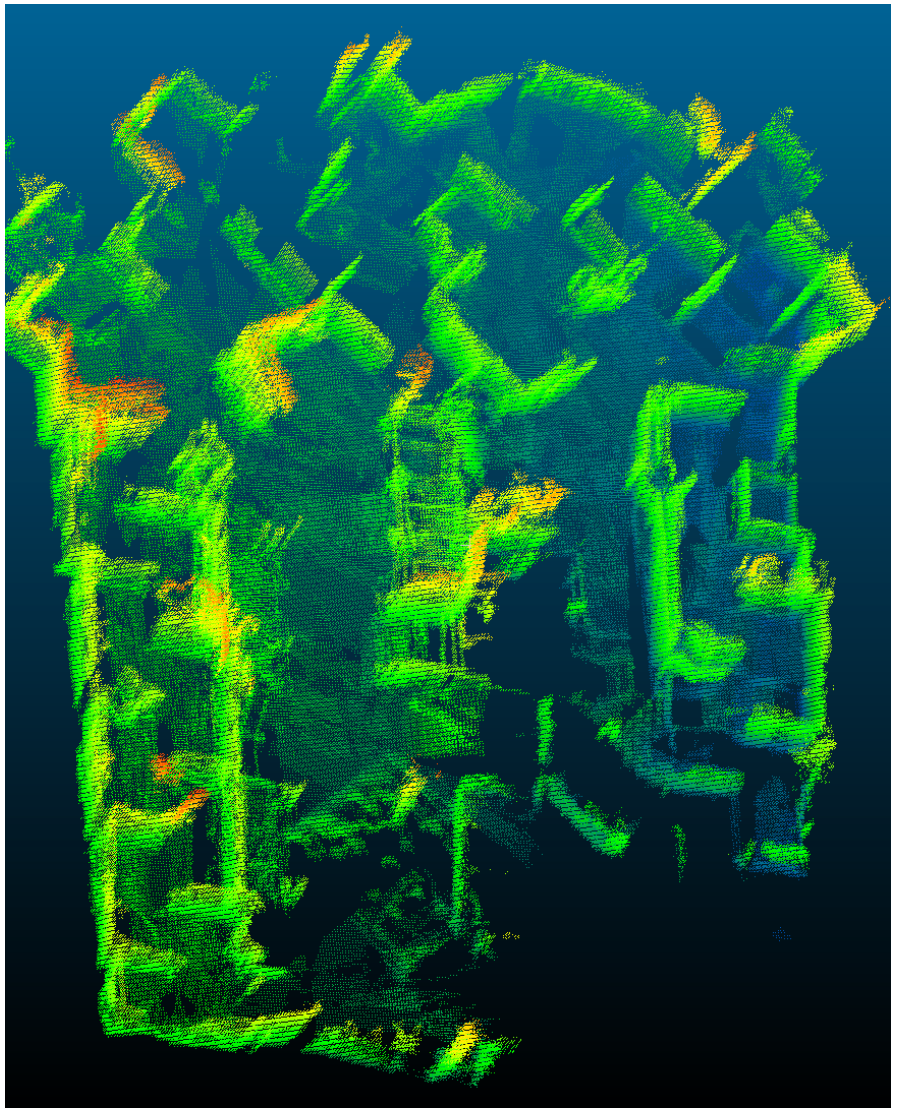
- For example: Quadtree



Octomap – 3D decomposition space

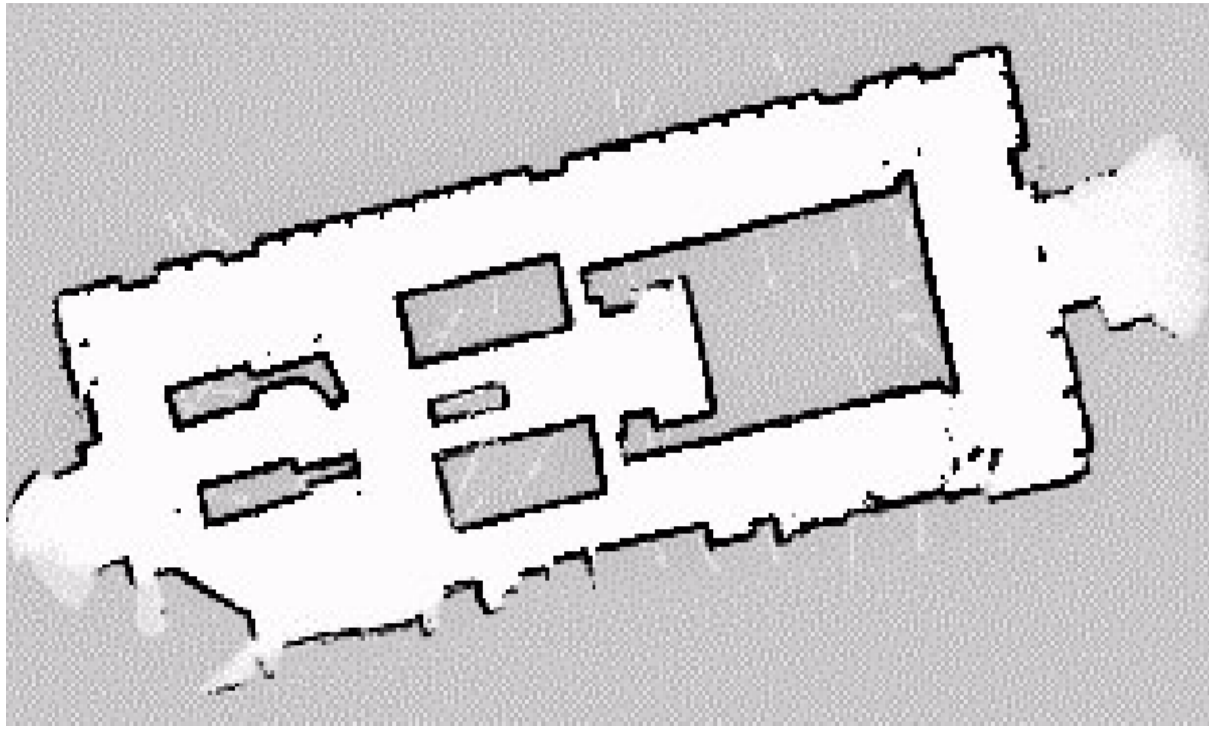
- cube can be devided into 8 parts
- http://wiki.ros.org/octomap





Map Representation: Occupancy grid

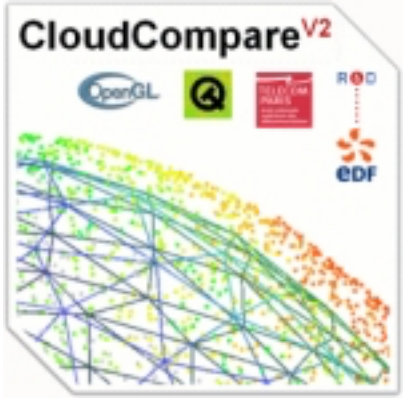
- Fixed cell decomposition: occupancy grid example
  - In occupancy grids, each cell may have a counter where 0 indicates that the cell has not been hit by any ranging measurements and therefore it is likely free-space. As the number of ranging strikes increases, the cell value is incremented and, above a certain threshold, the cell is deemed to be an obstacle
  - The values of the cells are discounted when a ranging strike travels through the cell. This allows us to represent “transient” (dynamic) obstacles

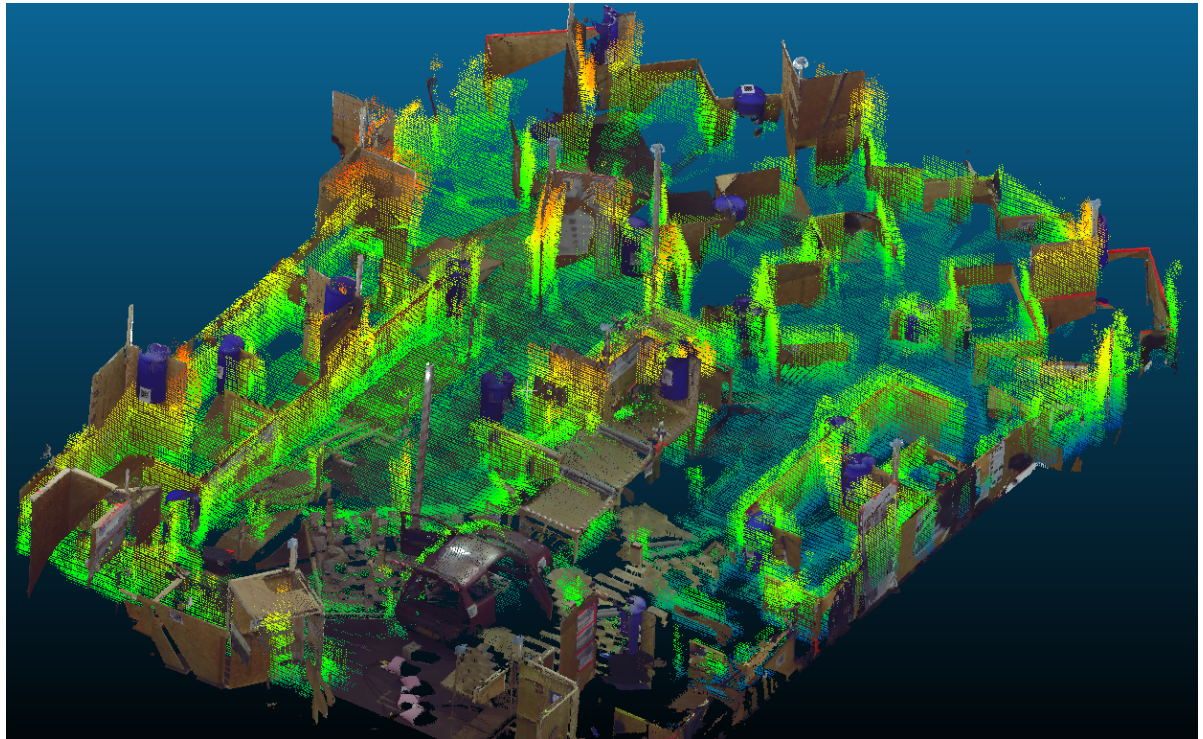


Courtesy of S. Thrun

Tools to view 3D Data

- ROS rviz: http://wiki.ros.org/rviz
- Octovis: http://wiki.ros.org/octovis (Visualizer for Octomap)
- Point Cloud Library Visualization: http://docs.pointclouds.org/trunk/group__visualization.html
- MeshLab: http://meshlab.sourceforge.net/
- CloudCompare: http://cloudcompare.org/

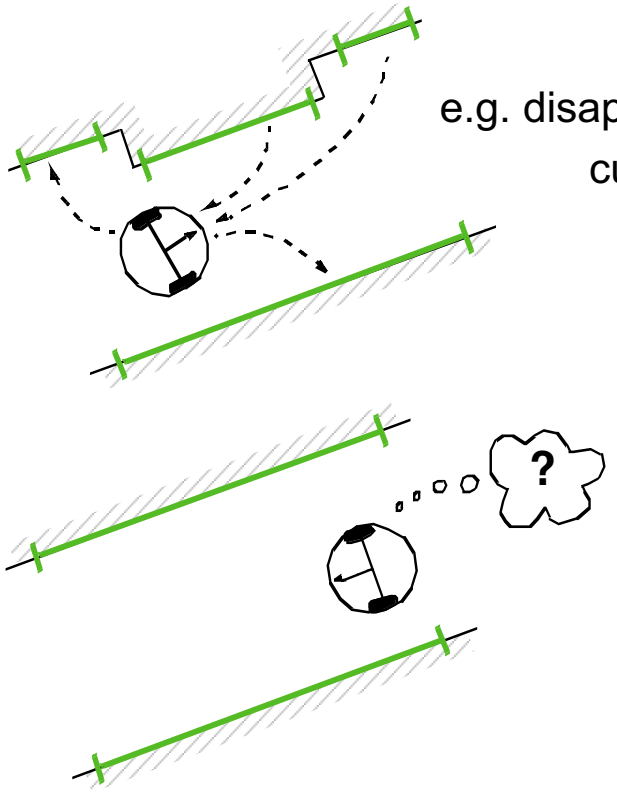




SIMULTANEOUS LOCALIZATION AND MAPPING SLAM

Map Building: The Problems

1. **Map Building:** Creating the map and keeping track of changes in the environment

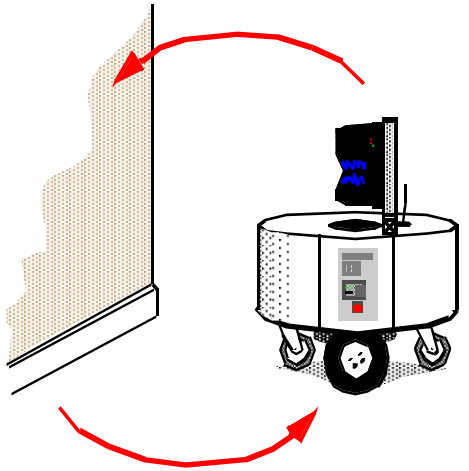


e.g. disappearing cupboard

- e.g. measure of belief of each environment feature

2. **Representation and Reduction of Uncertainty**

position of robot -> position of wall

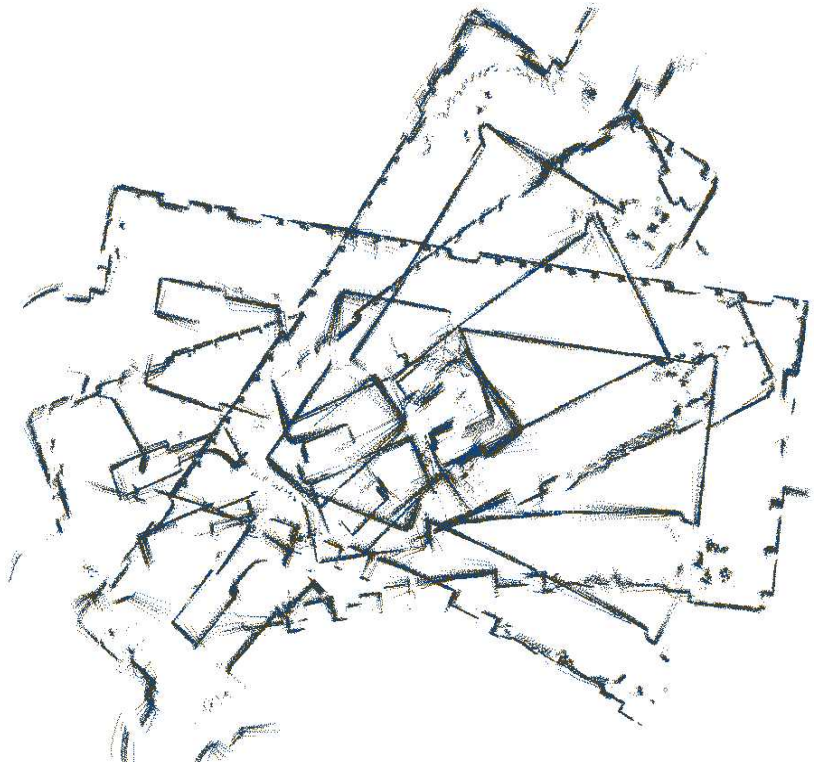


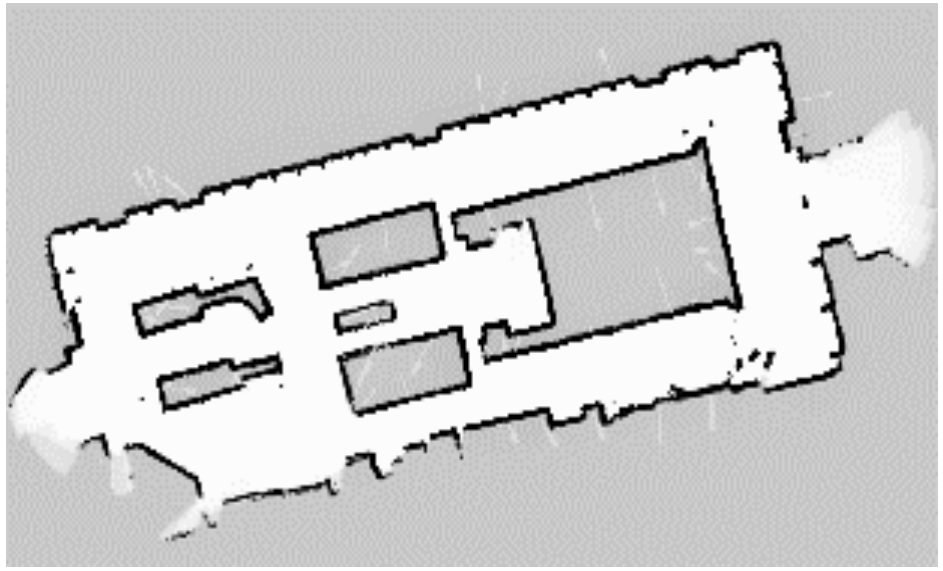
position of wall -> position of robot

- probability densities for feature positions
- Inconsistent map due to motion drift

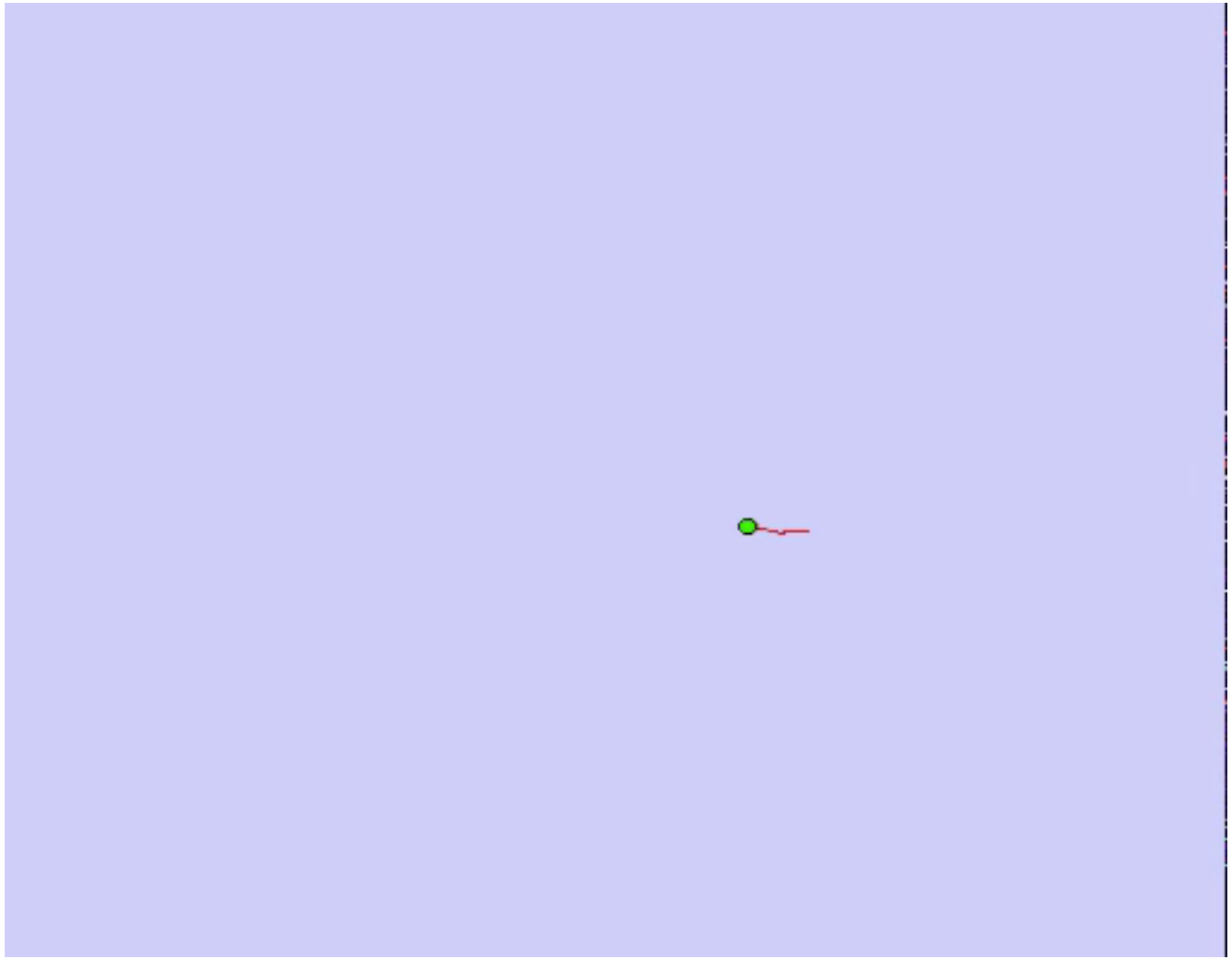
Cyclic Environments

- Small local error accumulate to arbitrary large global errors!
- This is usually irrelevant for navigation
- However, when closing loops, global error does matter





Raw Odometry …



Courtesy of S. Thrun

Scan Matching (use LRF scan from previous scan)

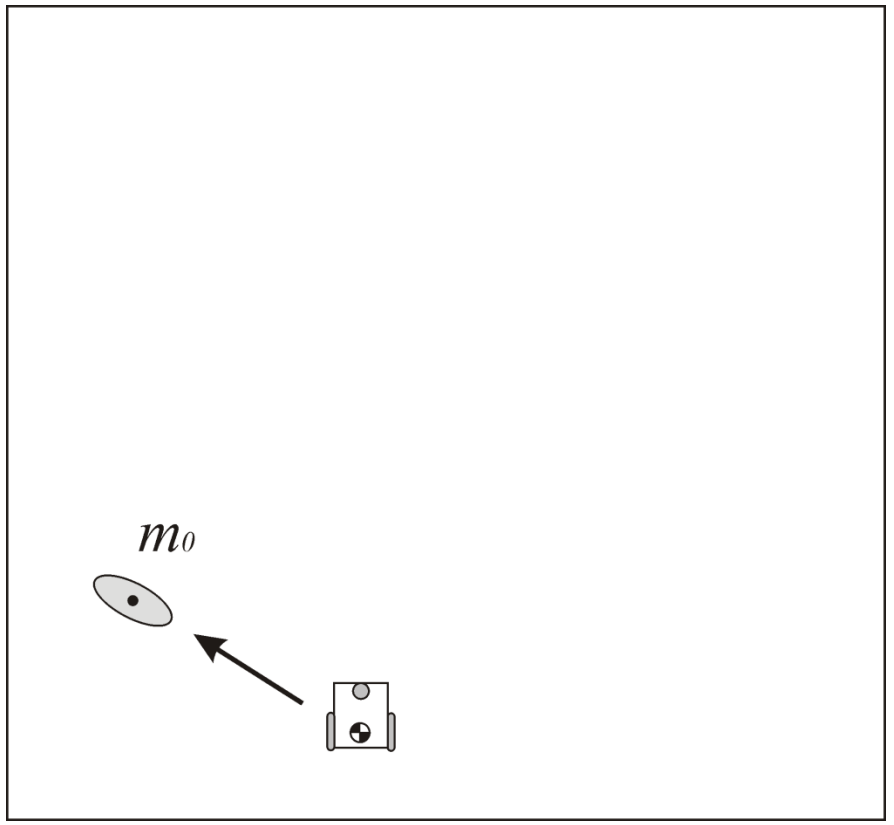
Courtesy of S. Thrun

The Three SLAM paradigms

- Most of the SLAM algorithms are based on the following three different approaches:
  - Extended Kalman Filter SLAM: (called EKF SLAM)
  - Particle Filter SLAM: (called FAST SLAM)
  - Graph-Based SLAM

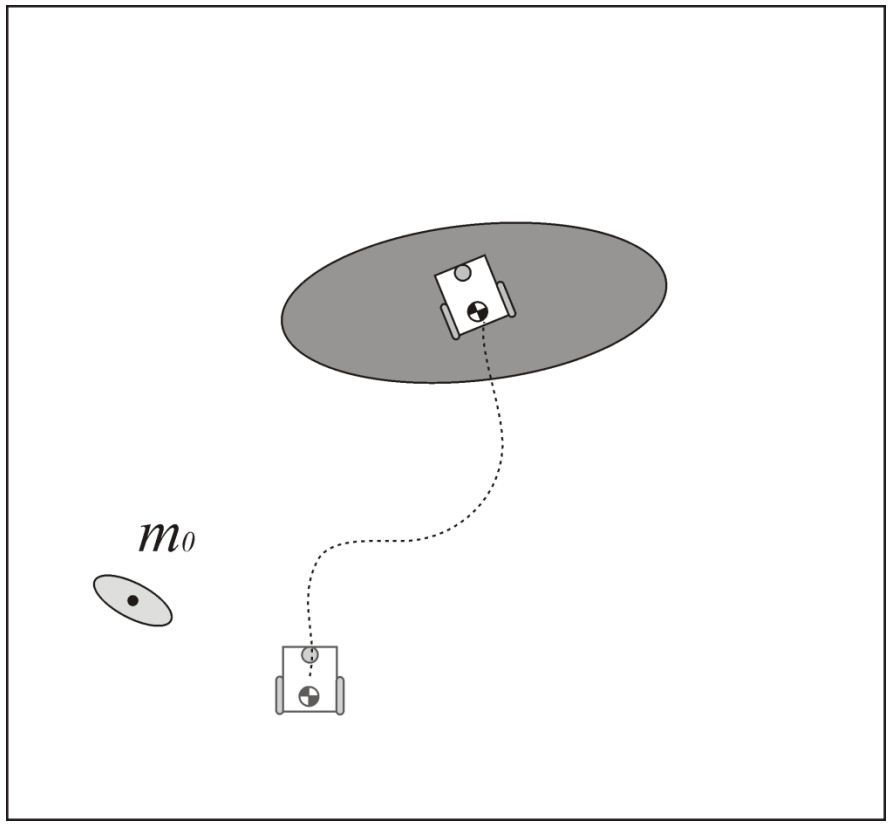
SLAM overview

- Let us assume that the robot uncertainty at its initial location is zero.
- From this position, the robot observes a feature which is mapped with an uncertainty related to the exteroceptive sensor error model



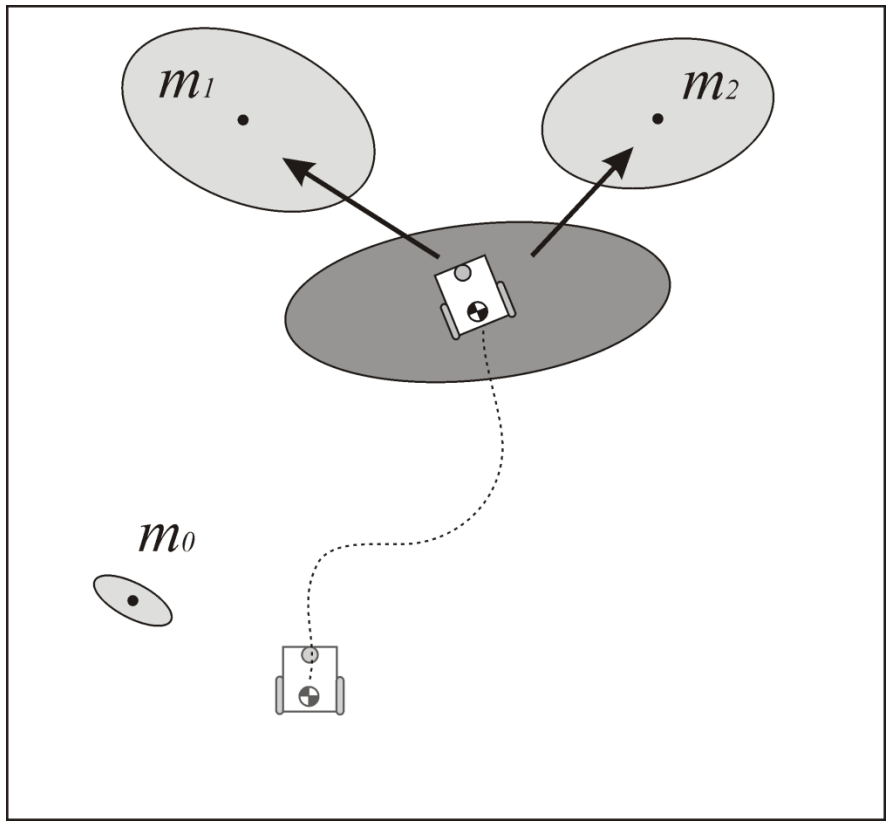
SLAM overview

- As the robot moves, its pose uncertainty increases under the effect of the errors introduced by the odometry



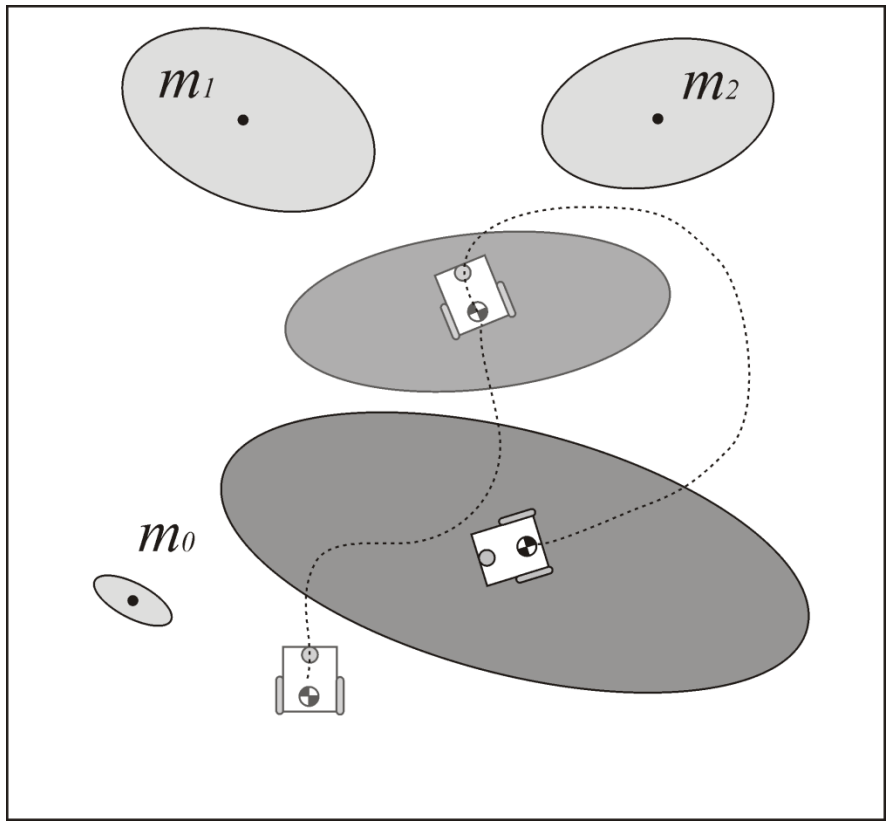
SLAM overview

- At this point, the robot observes two features and maps them with an uncertainty which results from the combination of the measurement error with the robot pose uncertainty
- From this, we can notice that the map becomes correlated with the robot position estimate. Similarly, if the robot updates its position based on an observation of an imprecisely known feature in the map, the resulting position estimate becomes correlated with the feature location estimate.



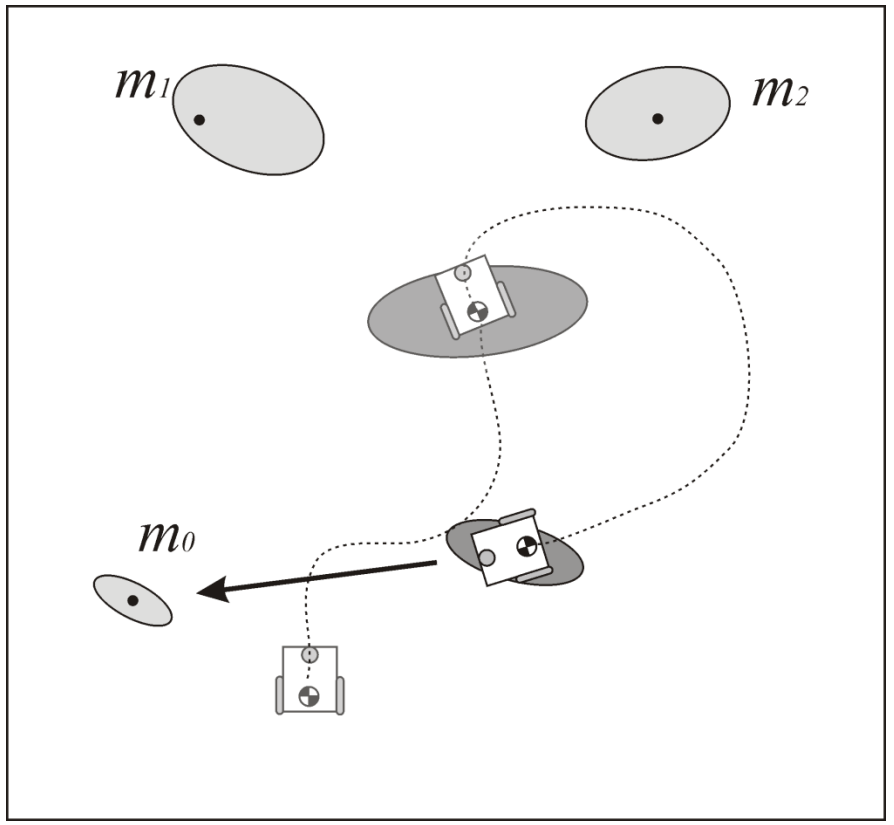
SLAM overview

- The robot moves again and its uncertainty increases under the effect of the errors introduced by the odometry



SLAM overview

- In order to reduce its uncertainty, the robot must observe features whose location is relatively well known. These features can for instance be landmarks that the robot has already observed before.
- In this case, the observation is called *loop closure detection.*
- When a loop closure is detected, the robot pose uncertainty shrinks.
- At the same time, the map is updated and the uncertainty of other observed features and all previous robot poses also reduce



EKF SLAM

- **extended state vector** y_t : robot pose x_t + position of all the features m_i in the map:

$$y_t = [x_t, m_0, \ldots, m_{n-1}]^T$$

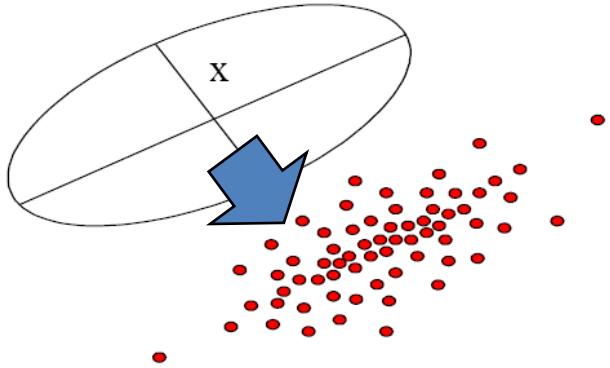
- Example: 2D line-landmarks, size of y_t = $3+2n$: three variables to represent the robot pose + $2n$ variables for the n line-landmarks having vector components $(\ _i, r_i)$

$$y_t = [x_t, y_t, \ _t, \ \ _0, r_0, \ldots, \ \ _{n-1}, r_{n-1}]^T$$

- As the robot moves and takes measurements, the state vector and covariance matrix are updated using the standard equations of the extended Kalman filter
- Drawback: EKF SLAM is computationally very expensive.

Particle Filter SLAM: FastSLAM

- **FastSLAM approach**
  - Using particle filters.
  - Particle filters: mathematical models that represent probability distribution as a set of discrete particles that occupy the state space.



probability distribution (ellipse) as particle set (red dots)

- Particle filter update
  - Generate new particle distribution using motion model and controls

a) For each particle:

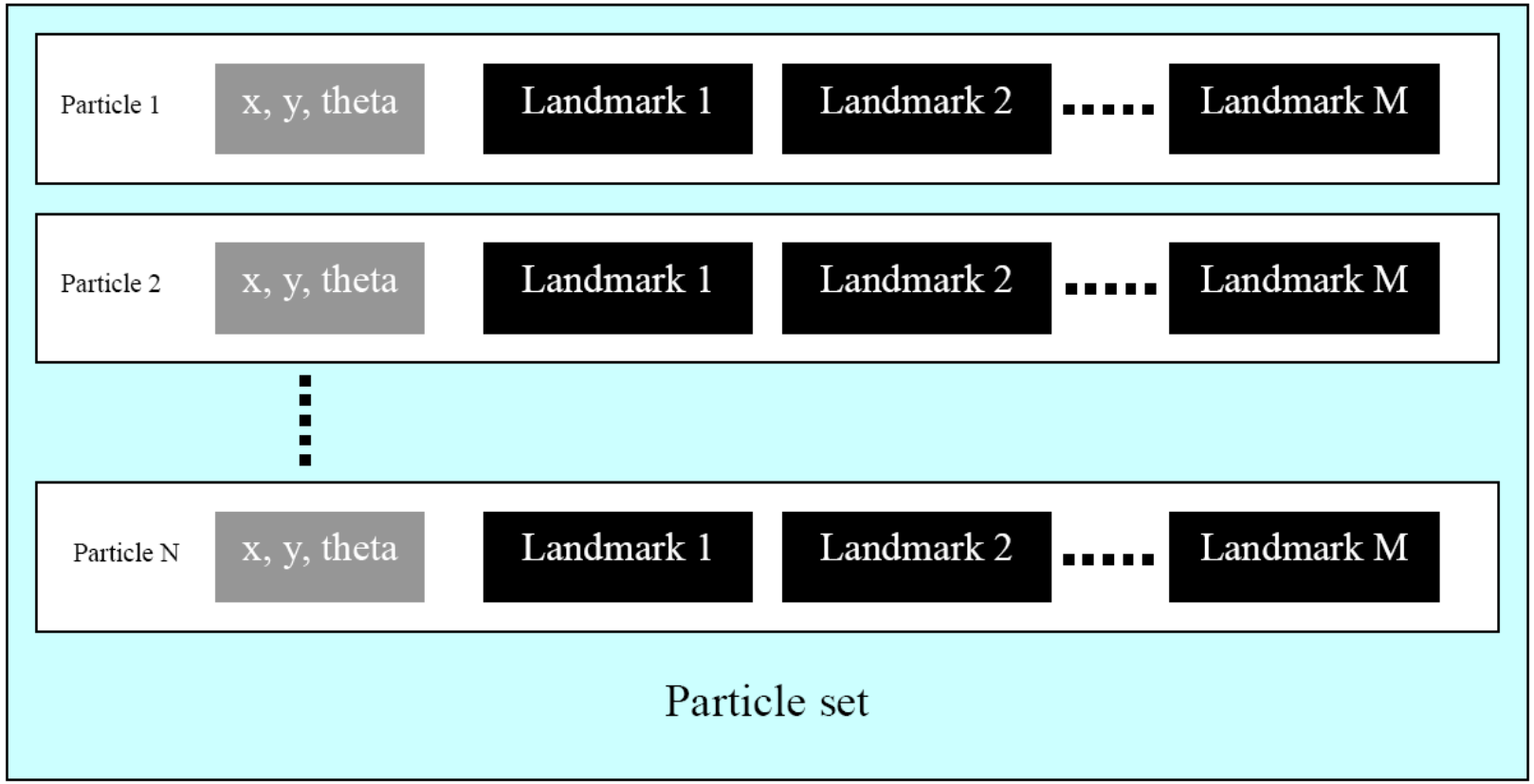
1. Compare particle's prediction of measurements with actual measurements
2. Particles whose predictions match the measurements are given a high weight

b) Filter resample:

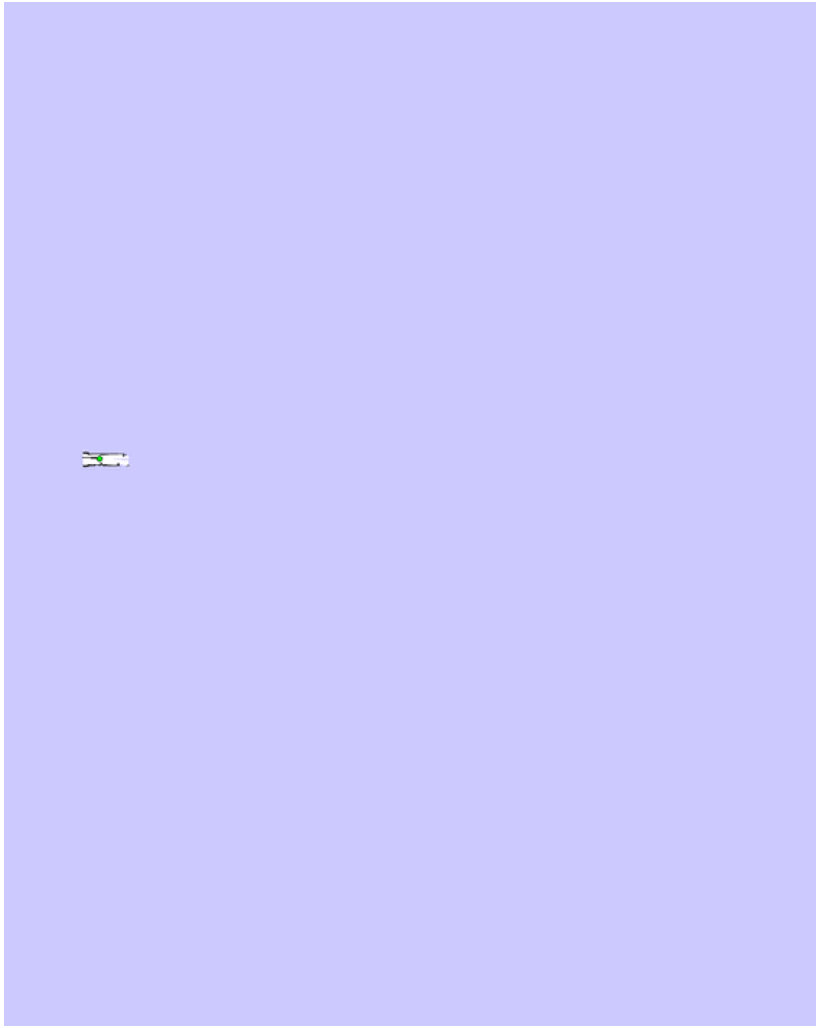
- Resample particles based on weight
- Filter resample
  - Assign each particle a weight depending on how well its estimate of the state agrees with the measurements and randomly draw particles from previous distribution based on weights creating a new distribution.

Particle Filter SLAM

- FastSLAM approach
  - Particle set:

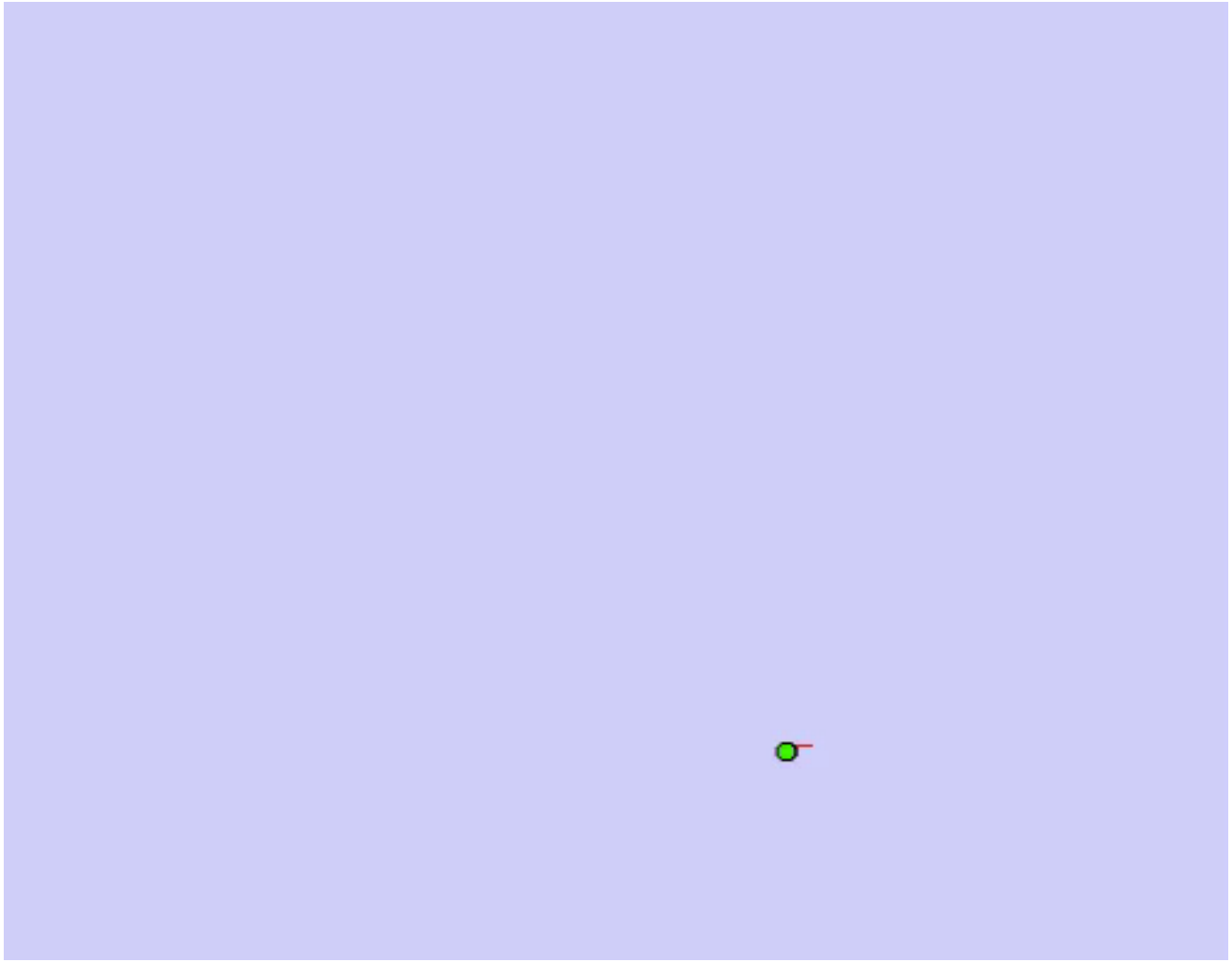


FAST SLAM example



Courtesy of S. Thrun

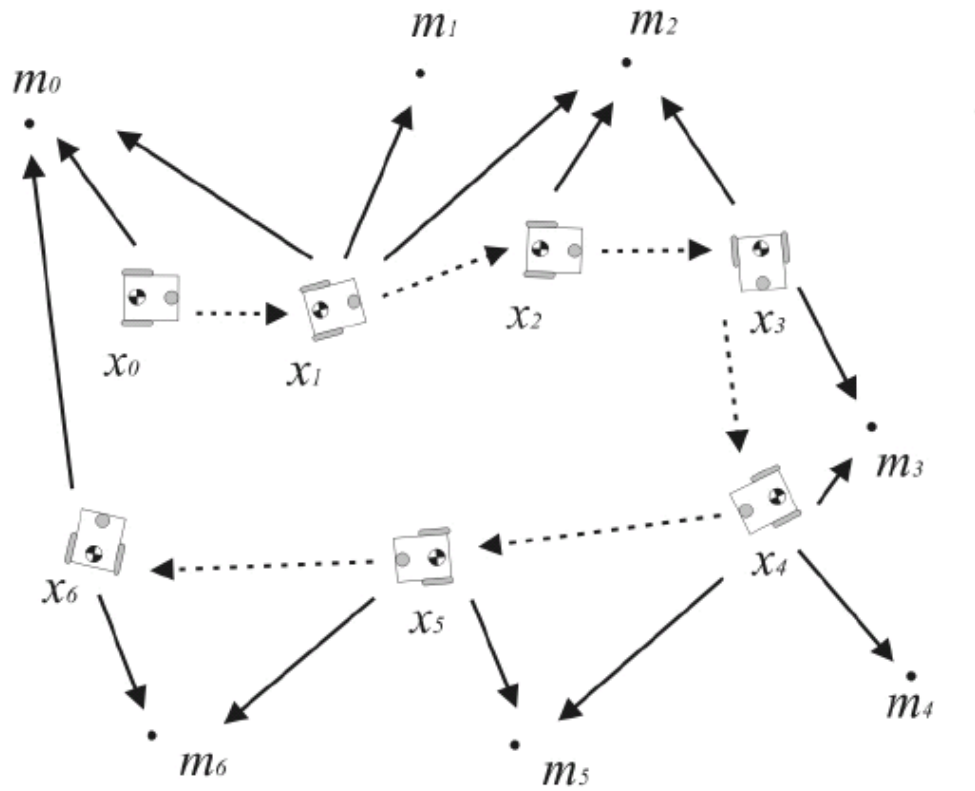
FAST SLAM example 2



Courtesy of S. Thrun

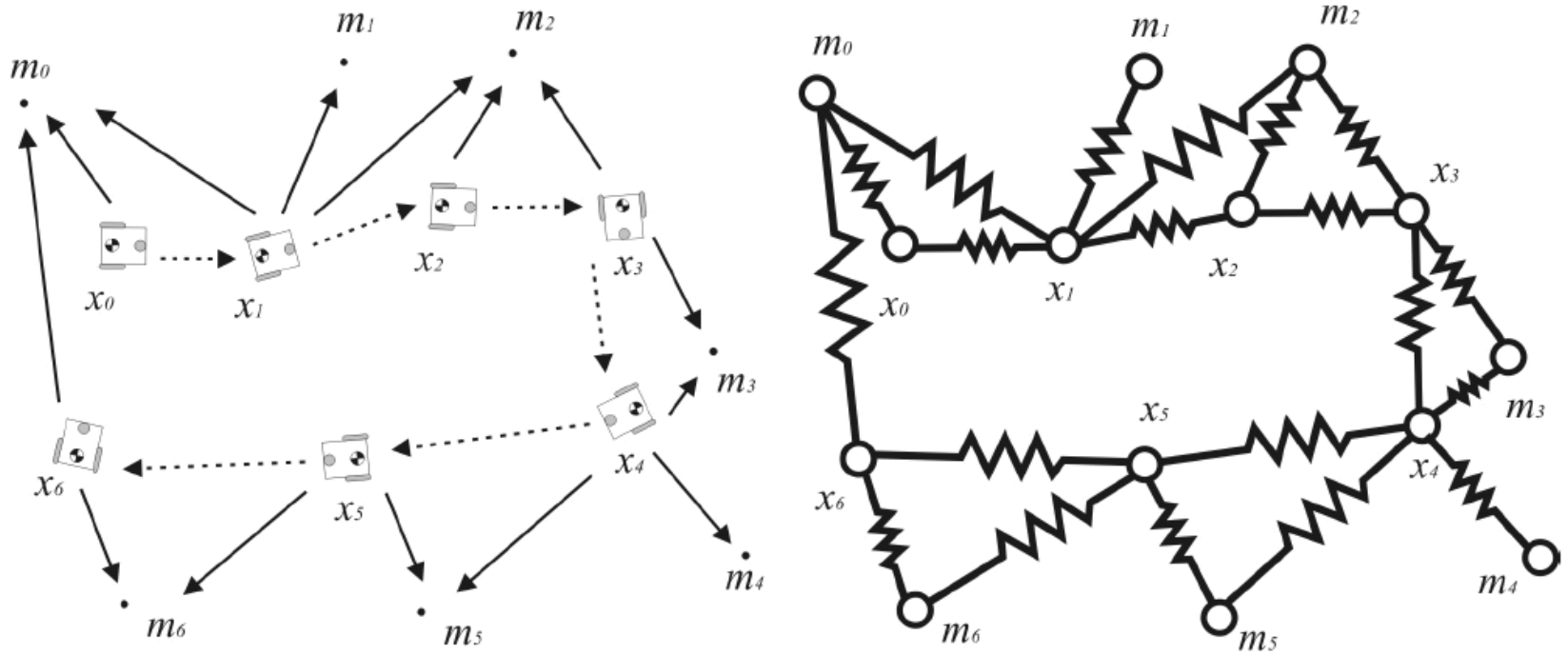
Graph-Based SLAM (1/3)

- SLAM problem can be interpreted as a sparse graph of nodes and constraints between nodes.
- The nodes of the graph are the robot locations and the features in the map.
- Constraints: relative position between consecutive robot poses , (given by the odometry input ***u***) and the relative position between the robot locations and the features observed from those locations.



Graph-Based SLAM (2/3)

- Constraints are not rigid but soft constraints!
- Relaxation: compute the solution to the full SLAM problem =>
  - Compute best estimate of the robot path and the environment map.
  - Graph-based SLAM represents robot locations and features as the nodes of an elastic net. The SLAM solution can then be found by computing the state of minimal energy of this net



Graph-Based SLAM (3/3)

- Significant advantage of graph-based SLAM techniques over EKF SLAM:
  - EKF SLAM: computation and memory for to update and store the covariance matrix is quadratic with the number of features.
  - Graph-based SLAM: update time of the graph is constant and the required memory is linear in the number of features.
- However, the final graph optimization can become computationally costly if the robot path is long.

- Software:
  - G2O: http://www.openslam.org/g2o.html
  - TORO: http://www.openslam.org/

SLAM and ROS

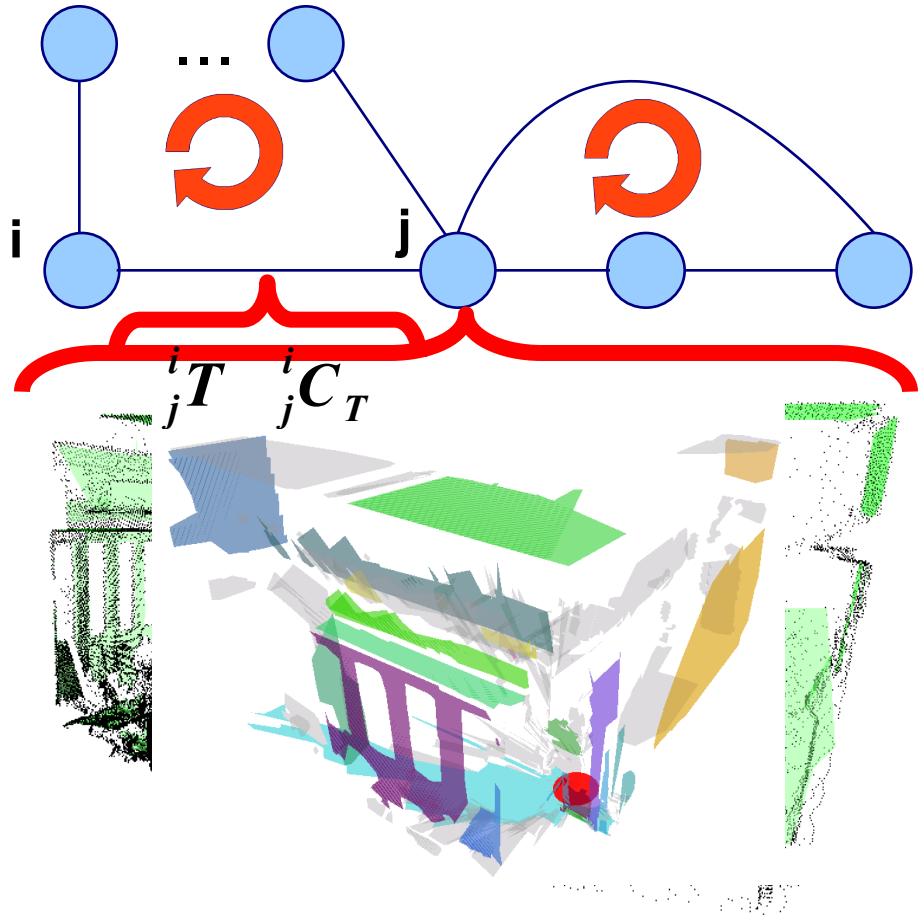
- Hector SLAM: http://wiki.ros.org/hector_slam mapping and localization 2D
  Tutorial: http://wiki.ros.org/hector_slam/Tutorials/SettingUpForYourRobot
  - Very robust – use by many teams of RoboCup Rescue
- Gmapping: http://wiki.ros.org/gmapping
  - Old solution based on particle filters
- ETH Zürich ASL ICP Mapping: http://wiki.ros.org/ethzasl_icp_mapping
  - real-time 2D and 3D ICP-based SLAM system
- PCL: Point Cloud Library – e.g. ICP implementation: http://pointclouds.org/

- RGBD SLAM: http://wiki.ros.org/rgbdslam
  (Use Kinect camera) - can make use of GPU
- Normal ICP (not integrated in ROS): http://jacoposerafin.com/nicp/
- Many more (often no ROS): http://www.openslam.org/

EXAMPLE: 3D PLANE SLAM

Jacobs 3D Mapping – Plane Mapping

- Pose Graph
- 3D Range Sensing
- Plane Extraction
- Planar Scan Matching
- Relax Loop-Closing Errors



Pathak, K., A. Birk, N. Vaskevicius, and J. Poppinga, "Fast Registration Based on Noisy Planes with Unknown Correspondences for 3D Mapping", IEEE Transactions on Robotics, vol. 26, no. 3, pp. 424-441, 2010.

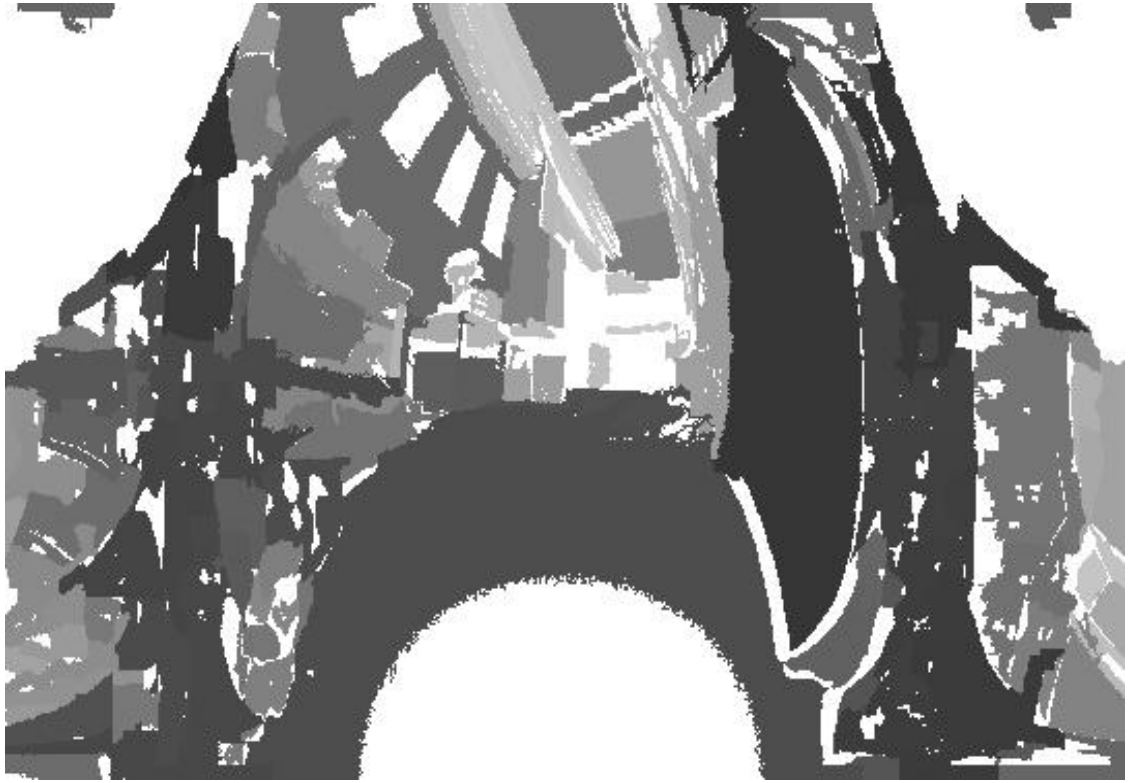
Plane Extraction from 3D Point Clouds

- **Plane Fitting**
  - Assumes 3D sensor has radial Gaussian noise dependent on range
  - Uses Approximate Least Squares solution to find the best fit.
  - Estimates covariance matrix of the plane parameters

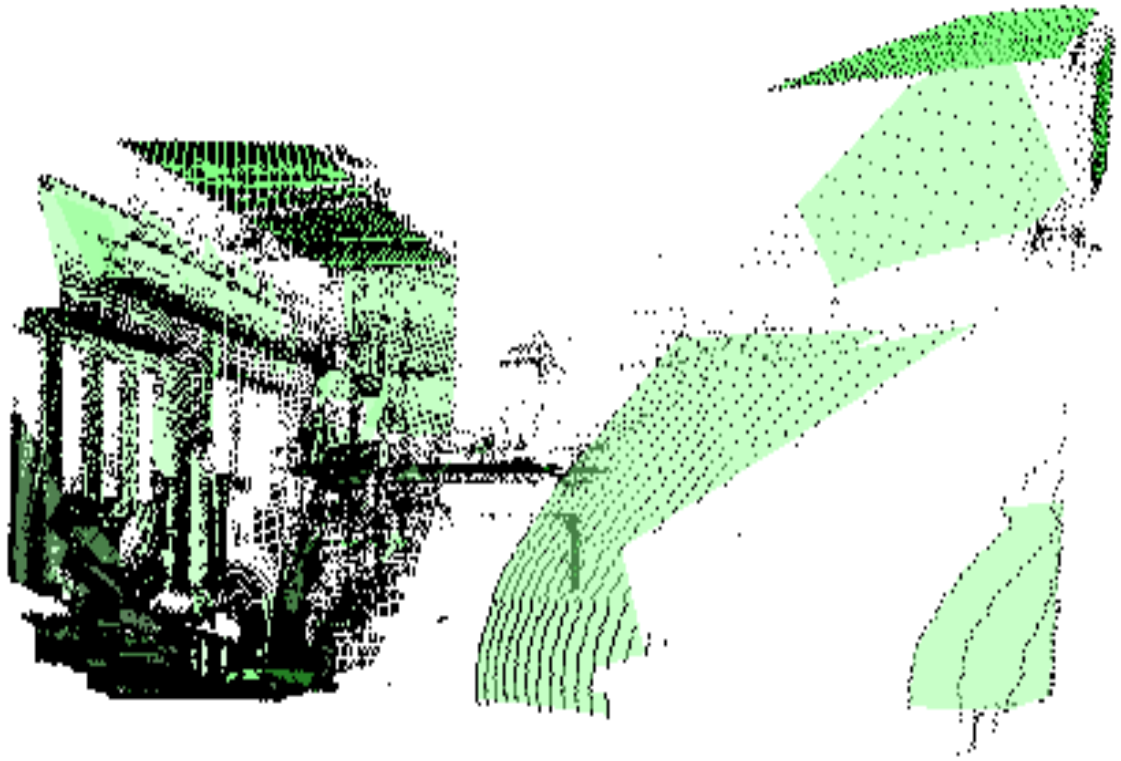
- **Range Image Segmentation**
  - Is based on region growing algorithm
  - Uses incremental formulas, therefore is fast
  - Has linear computational complexity

Given a range image, returns a polygonal model i.e. a set of planar features and boundaries.

An Example



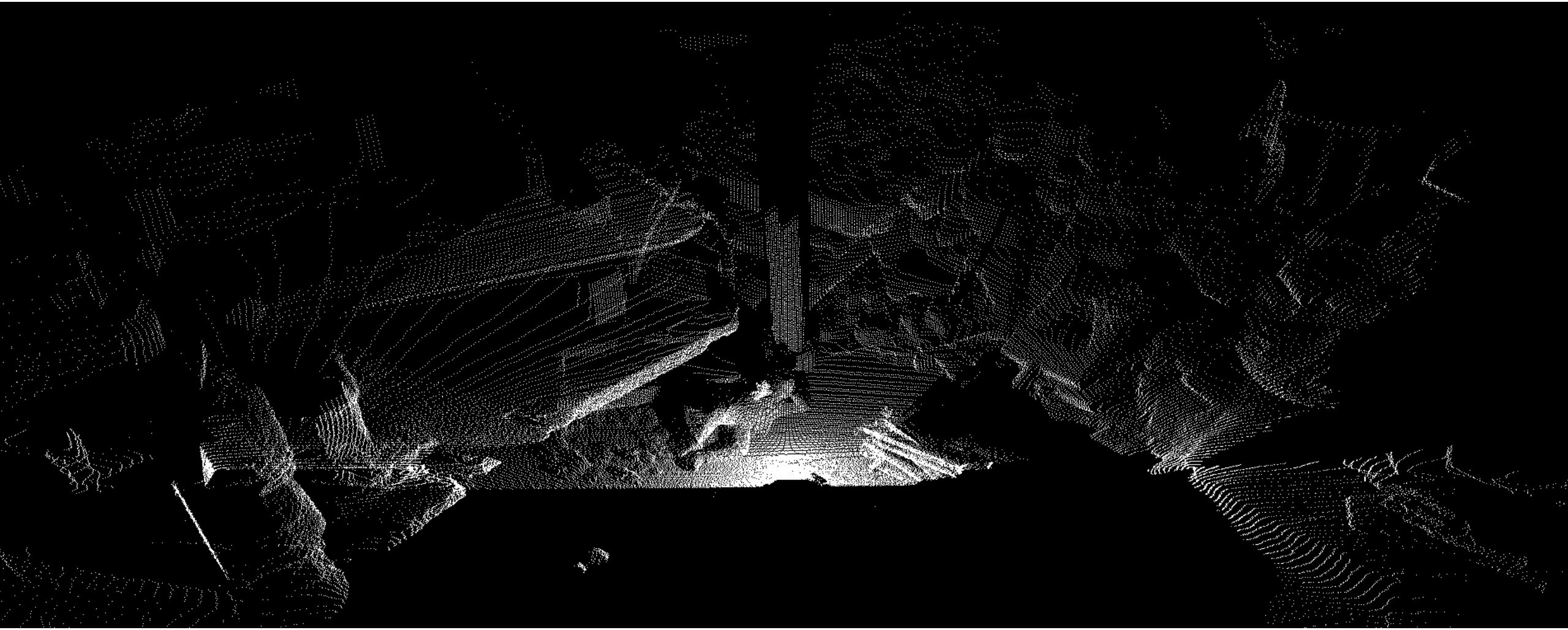
Segmentation Image



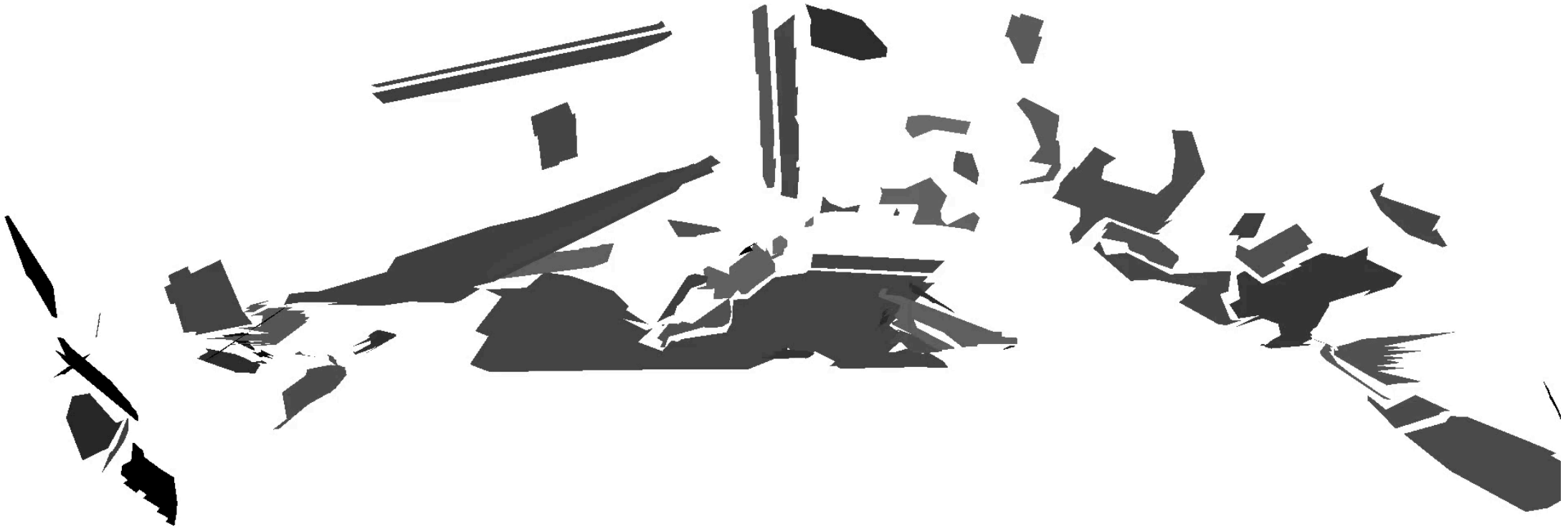
Polygonal 3D Model

Segments range image consisting of ~$2 \cdot 10^5$ pixels in ~3s. On 1.6 Ghz machine.

One Point Cloud with human



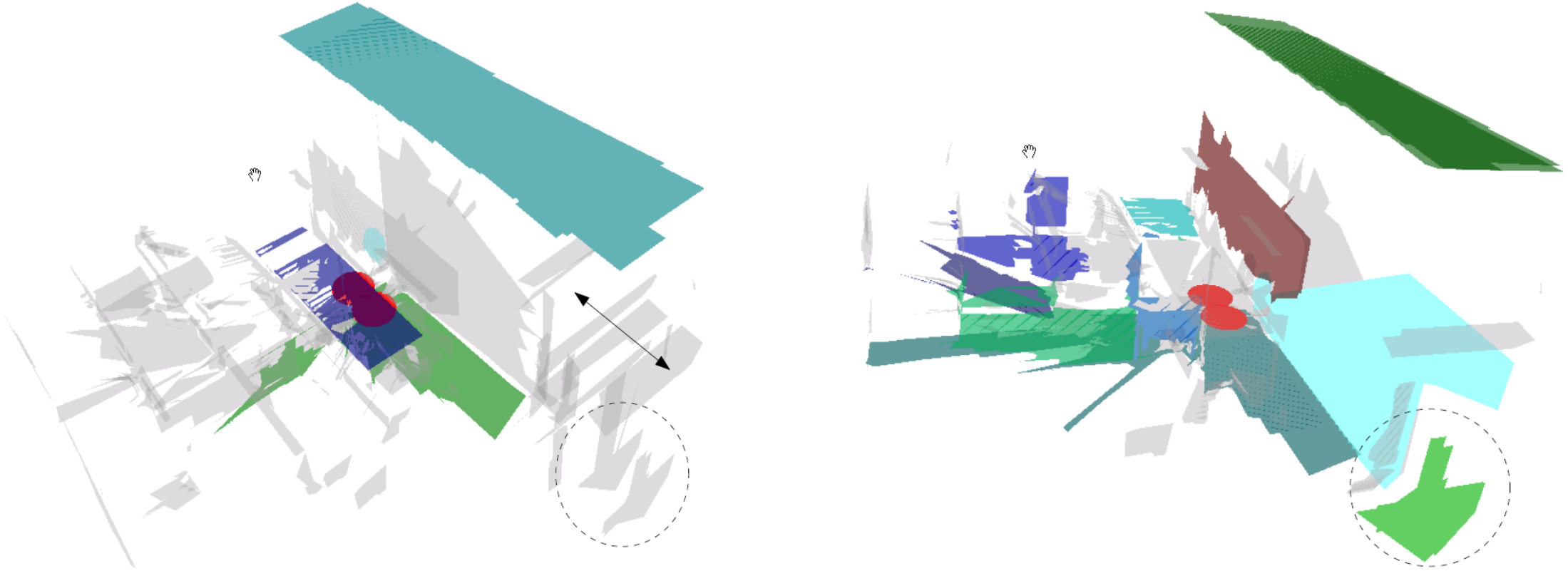
One Plane Cloud with human



Plane Registration (Scan Matching)

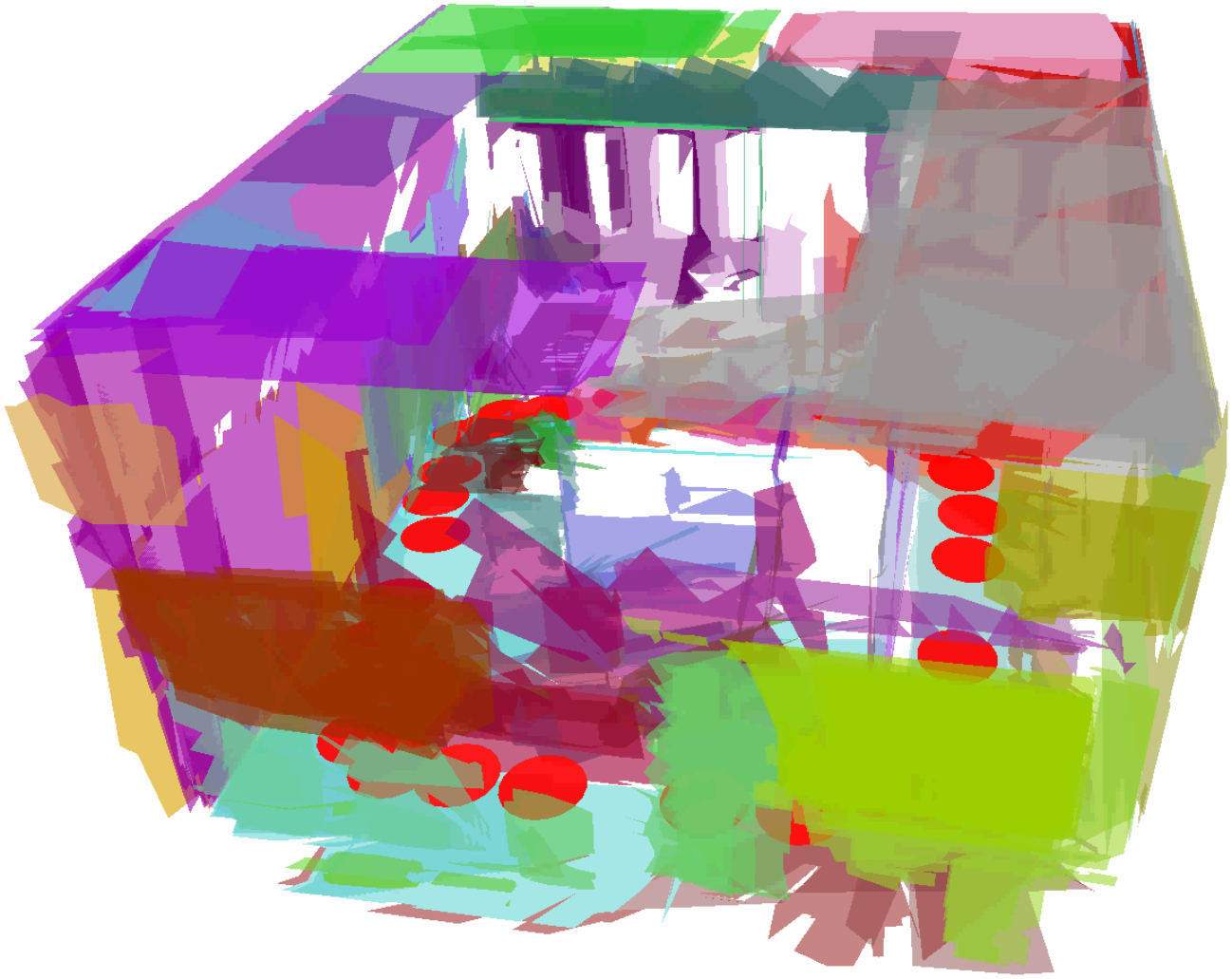
- Determining the correspondence set maximizing the global rigid body motion constraint.
- Finding the optimal decoupled rotations (Wahba's problem) and translations (closed form least squares) with related uncertainties.
- No motion estimates from any other source are needed.
- Very fast
- MUMC: Finding Minimally Uncertain Maximal Consensus
  - Of matched planes
- Idea: select two non-parallel plane matches => fixes rotation and only leaves one degree of translation!

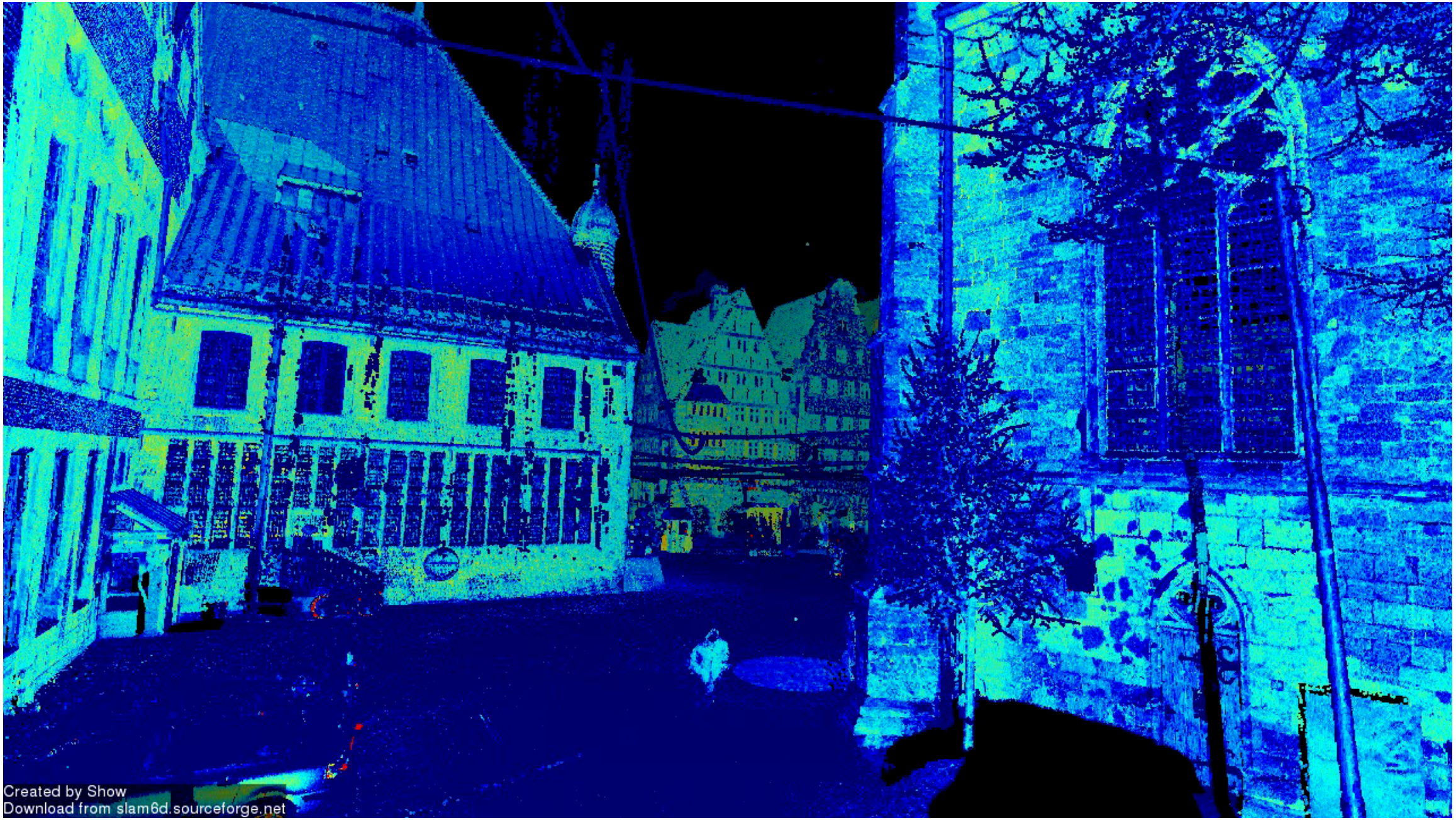
Relaxation of Errors (Translation)



- **Only translation errors are relaxed**
  - Good rotation estimates from the plane matching
  - Non-linear optimization can be exchanged with linear if rotation is assumed to be known precisely.
  - This leads to a fast relaxation method

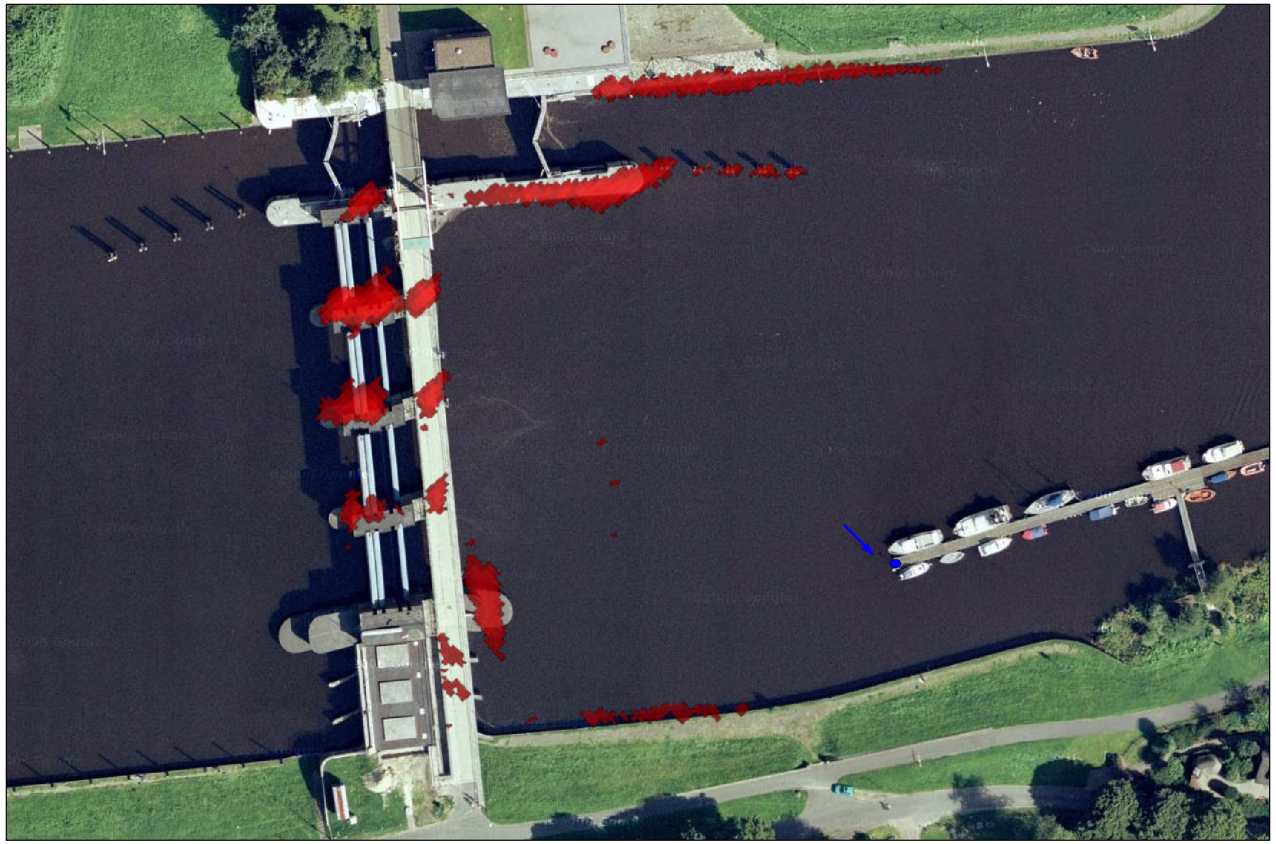
Experiment Lab Run: 29 3D point-clouds; size of each: 541 x 361 = 195,301

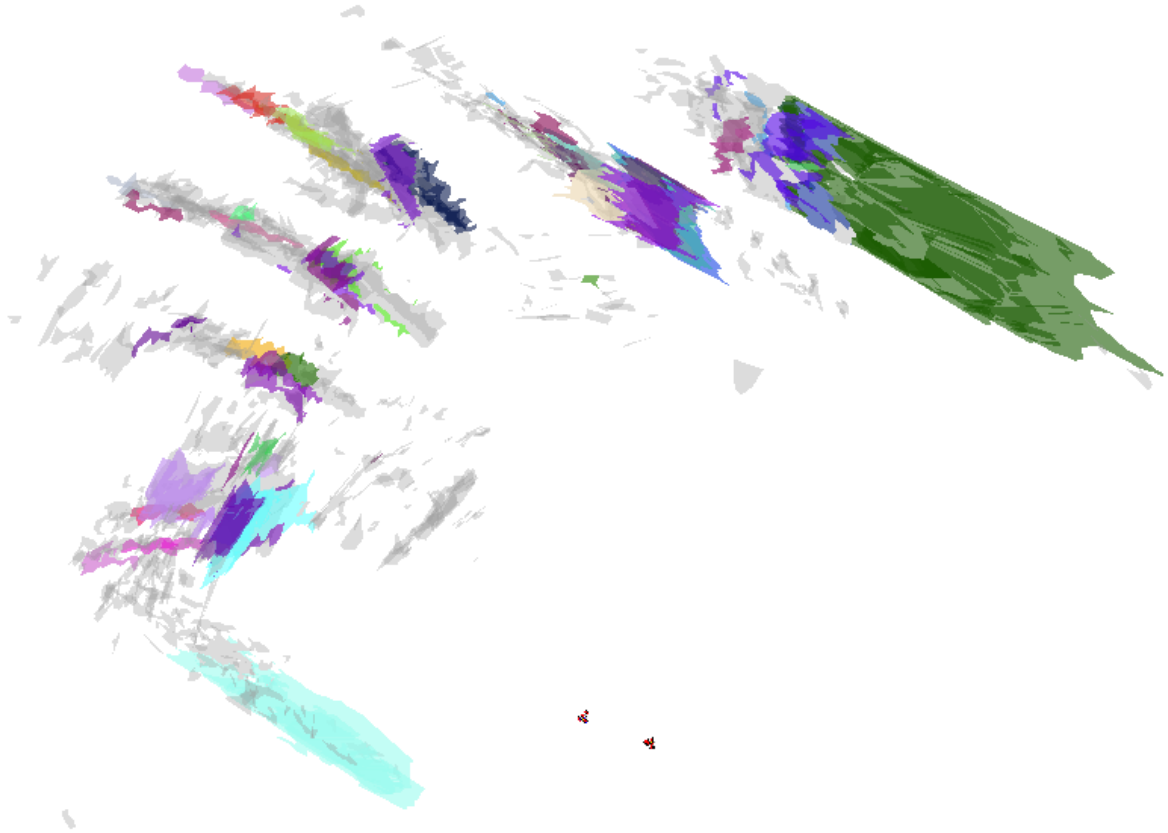



Created by Show
Download from slam6d.sourceforge.net

3D Sonar result

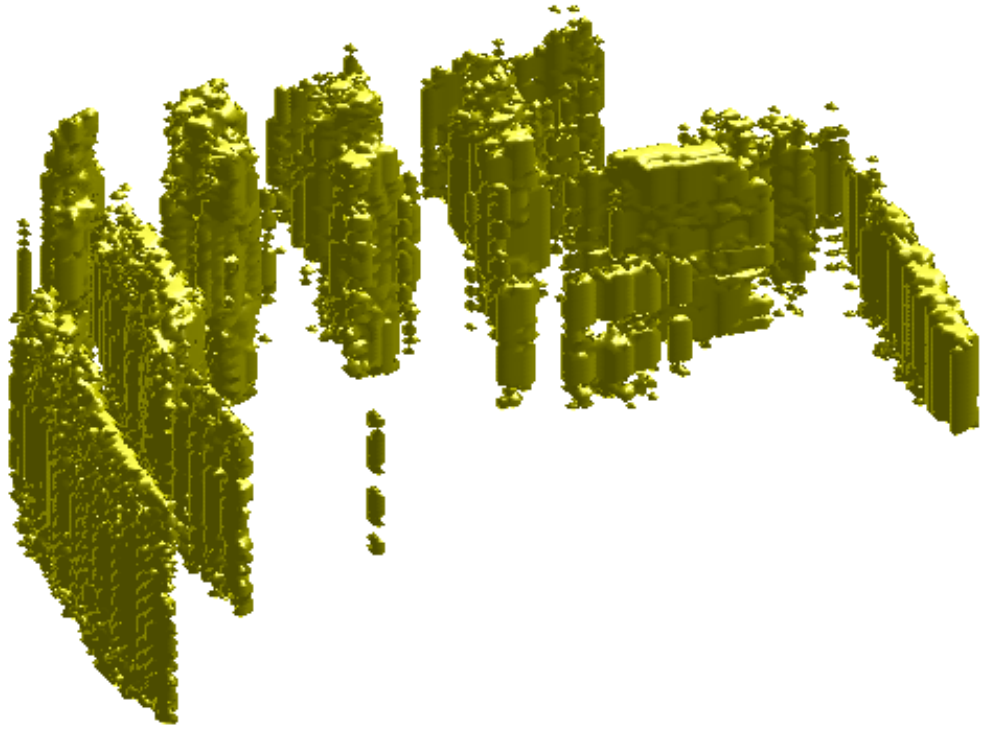
- Flood barrier on river in Bremen
- 18 scans with 3D sonar





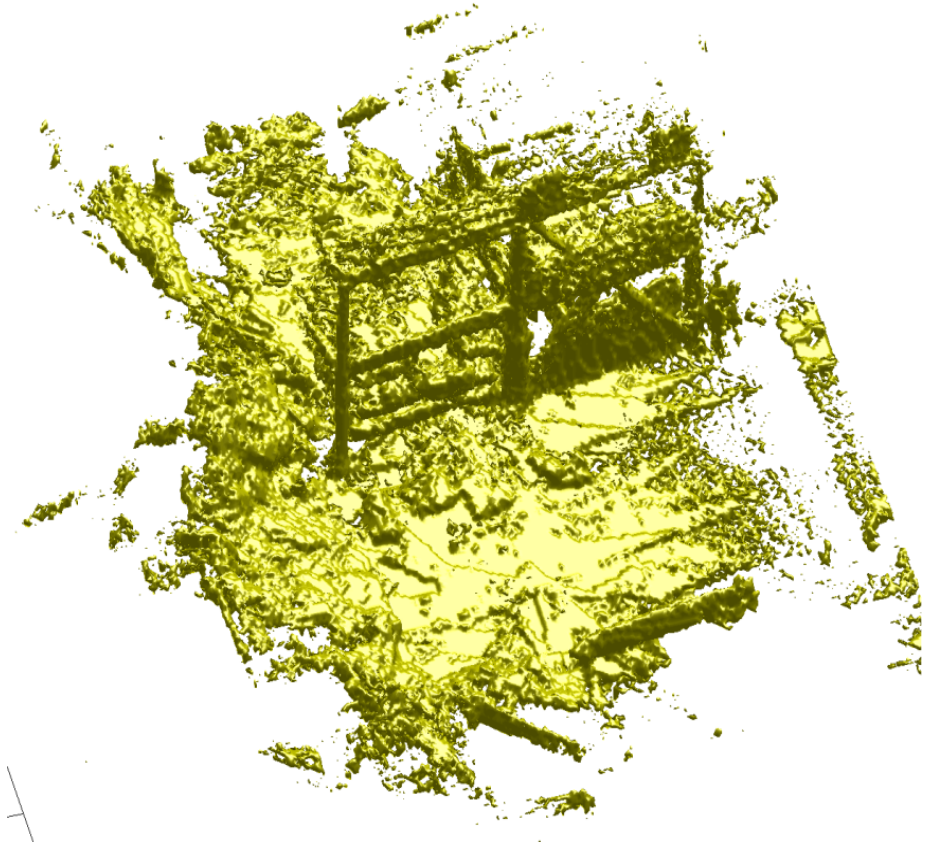
3D Scan matching using Spectral Method FMI

Flood Gate (sonar)



Crashed car park

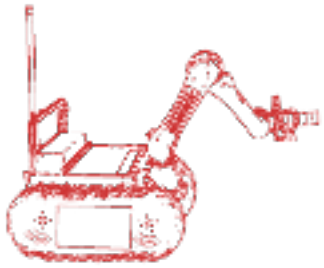
Disaster City (3D LRF)



Join my Robotics Lab!

上海科技大学
ShanghaiTech University

- Visit: http://robotics.shanghaitech.edu.cn
- Take a look at: http://sist.shanghaitech.edu.cn/cn/gsr/
- I am looking to admit one or two excellent students into my group. Prerequisites:
  - Computer Science or related major
  - Good English
  - Very good grades
  - Open and creative mind
  - **Love for (playing with) robots**



ShanghaiTech Advanced Robotics Lab - STAR-Lab

School of Information Science and Technology (SIST) / ShanghaiTech University

信息学院 (SIST) / 上海科技大学