DD2410

Lecture slides

Mapping and SLAM

Localization problem a la Bayes

Prediction based on control input / odometry, u_k:

$p(x_{k+1}|Z_k,U_{k+1}) = \int p(x_{k+1}|u_{k+1,}x_k)\ p(x_k|Z_k,U_k)\ dx_k$

where $\mathbf{p(x_{k+1}|u_{k+1,}x_k)}$ is the motion model often given by odometry

→ distribution smeared out (uncertainty increases)

Update with new measurement z_{k+1}:

$p(x_{k+1}|Z_{k+1},U_{k+1}) = \eta\ p(z_{k+1}|x_{k+1})p(x_{k+1}|Z_k,U_{k+1})$

where $\mathbf{p(z_{k+1}|x_{k+1})}$ is the measurement model

→distribution more peaked (uncertainty decreases)

Extended Kalman Filter (EKF)

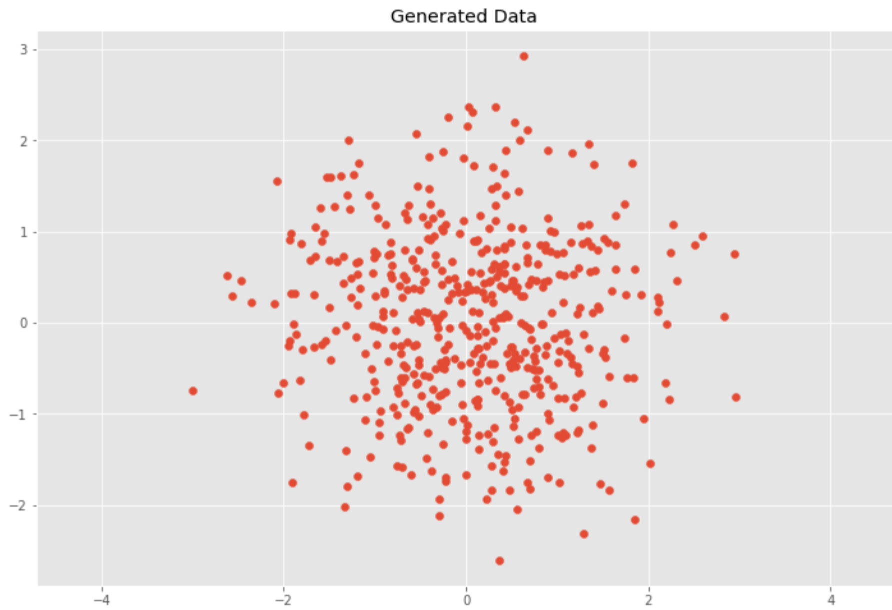
- Initialize
  - $x_{0|0}$ = best guess about state (here pose)
  - $P_{0|0}$ = covariance ("uncertainty") in initial state guess
- Repeat
  - Predict
    - $x_{k+1|k} = f(x_{k|k}, u_{k+1})$
    - $P_{k+1|k} = A_k P_{k|k} A_k^T + W_k Q_k W_k^T$
  - Update
    - $K_k = P_{k+1|k} H_k (H_k P_{k+1|k} H_k^T + V_k R_k V_k^T)^{-1}$
    - $x_{k+1|k+1} = x_{k+1|k} + K_k(z_{k+1} - h(x_{k+1|k}))$
    - $P_{k+1|k+1} = (I - K_k H_k) P_{k+1|k}$

Covariance matrix (P on last slide)

- What is it?
- What does it tell us?
- Give examples

Covariance matrix

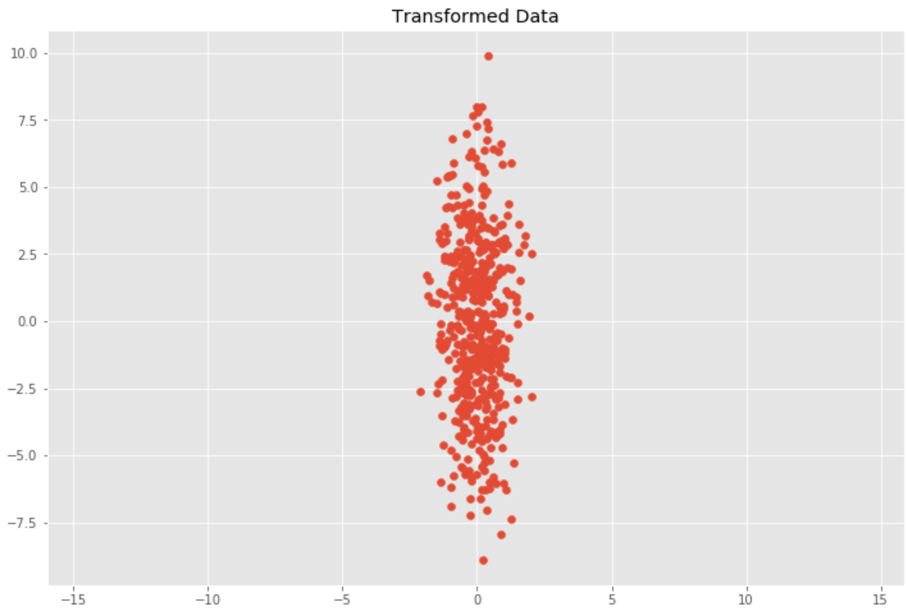
- Covariance matrix for this data?



https://datascienceplus.com/understanding-the-covariance-matrix/

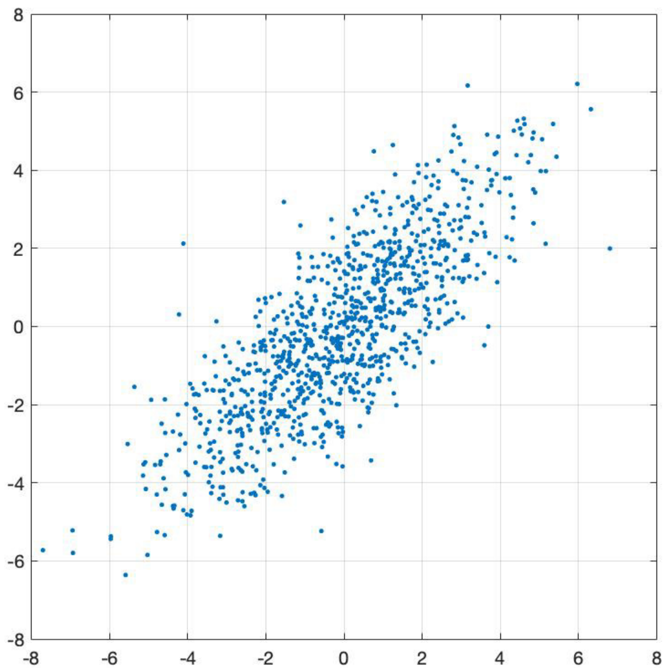
Covariance matrix

- Covariance matrix for this data?



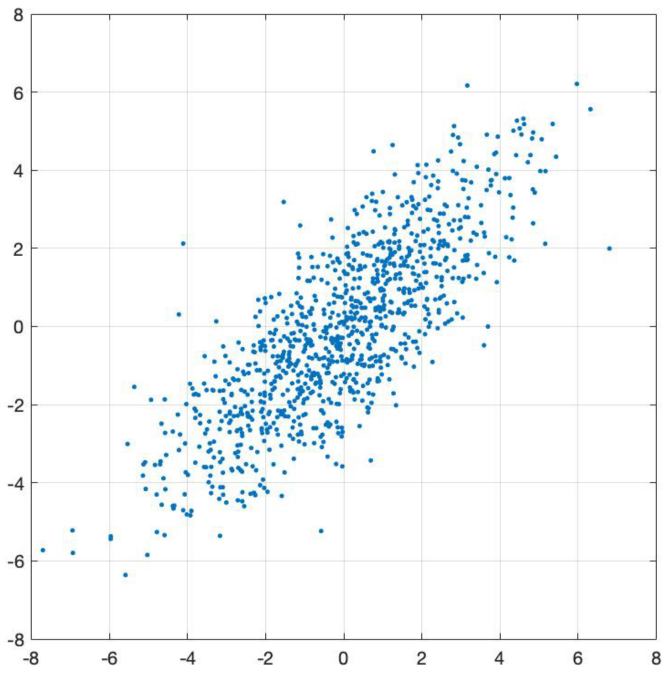
Covariance matrix

- Covariance matrix for this data?

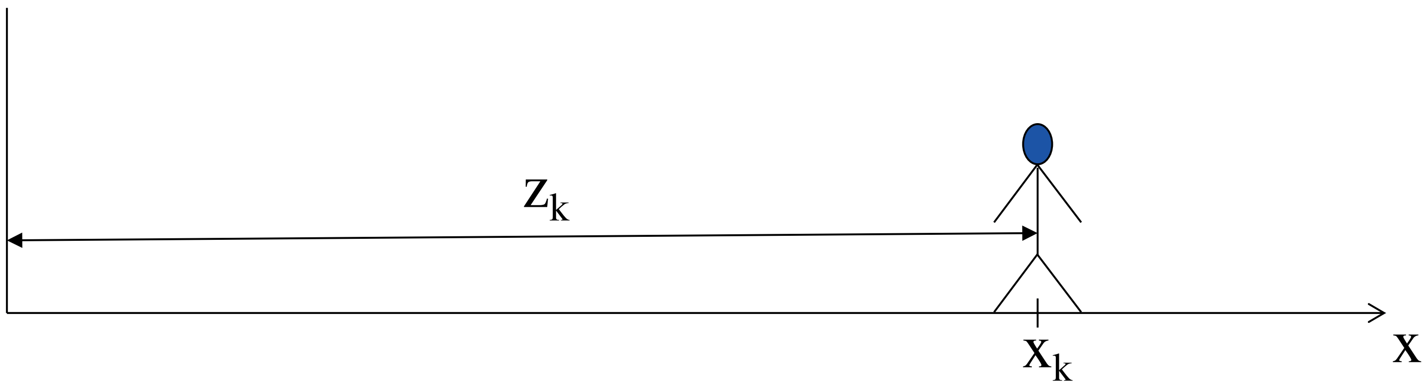


Covariance matrix

- Covariance matrix for this data?
- What does it say about how new information about x will influence y?



Ex: EKF in 1D

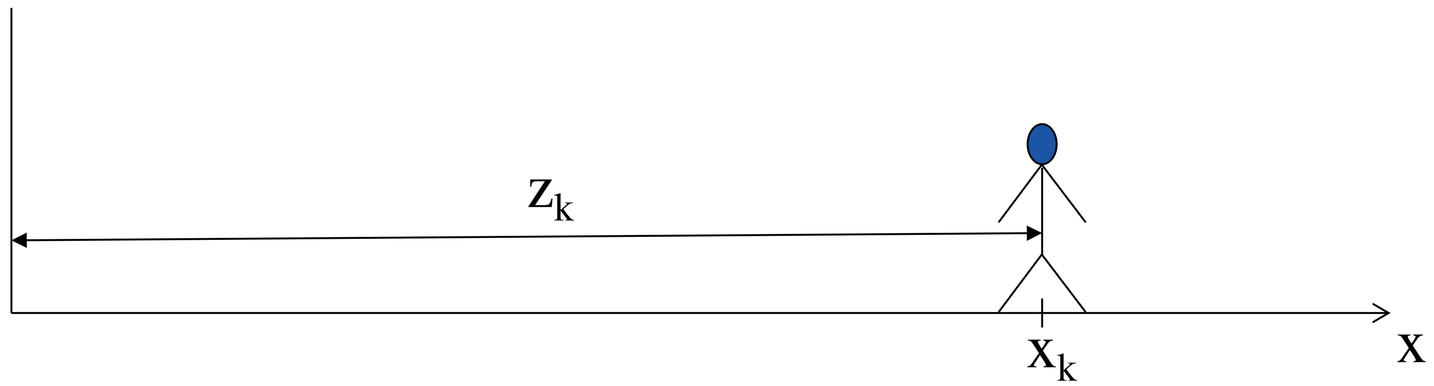


- $x_{k+1} = f(x_k, u_{k+1}, w_{k+1}) = x_k + v_k dT + w_{k+1}$ $(u_k = v_k = \text{speed})$
- $z_k = h(x_k, v_k) = x_k + v_k$

noise

- $A = df/dx = ?$ $\quad H = dh/dx = ?$ $\quad w_k \sim N(0, Q_k)$
- $W = df/dw = ?$ $\quad V = dh/dv = ?$ $\quad v_k \sim N(0, R_k)$

Ex: EKF in 1D

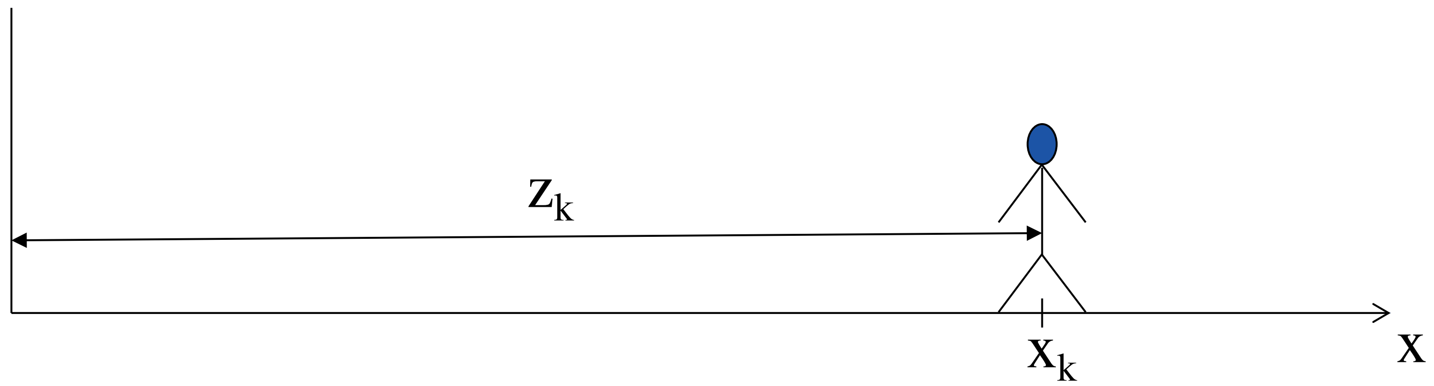


- $x_{k+1} = f(x_k, u_{k+1}, w_{k+1}) = x_k + v_k dT + w_{k+1}$ ($u_k = v_k =$ speed)
- $z_k = h(x_k, v_k) = x_k + v_k$

noise

- $A = df/dx = 1$ $H = dh/dx = 1$ $w_k \sim N(0, Q_k)$
- $W = df/dw = 1$ $V = dh/dv = 1$ $v_k \sim N(0, R_k)$

Ex: EKF in 1D



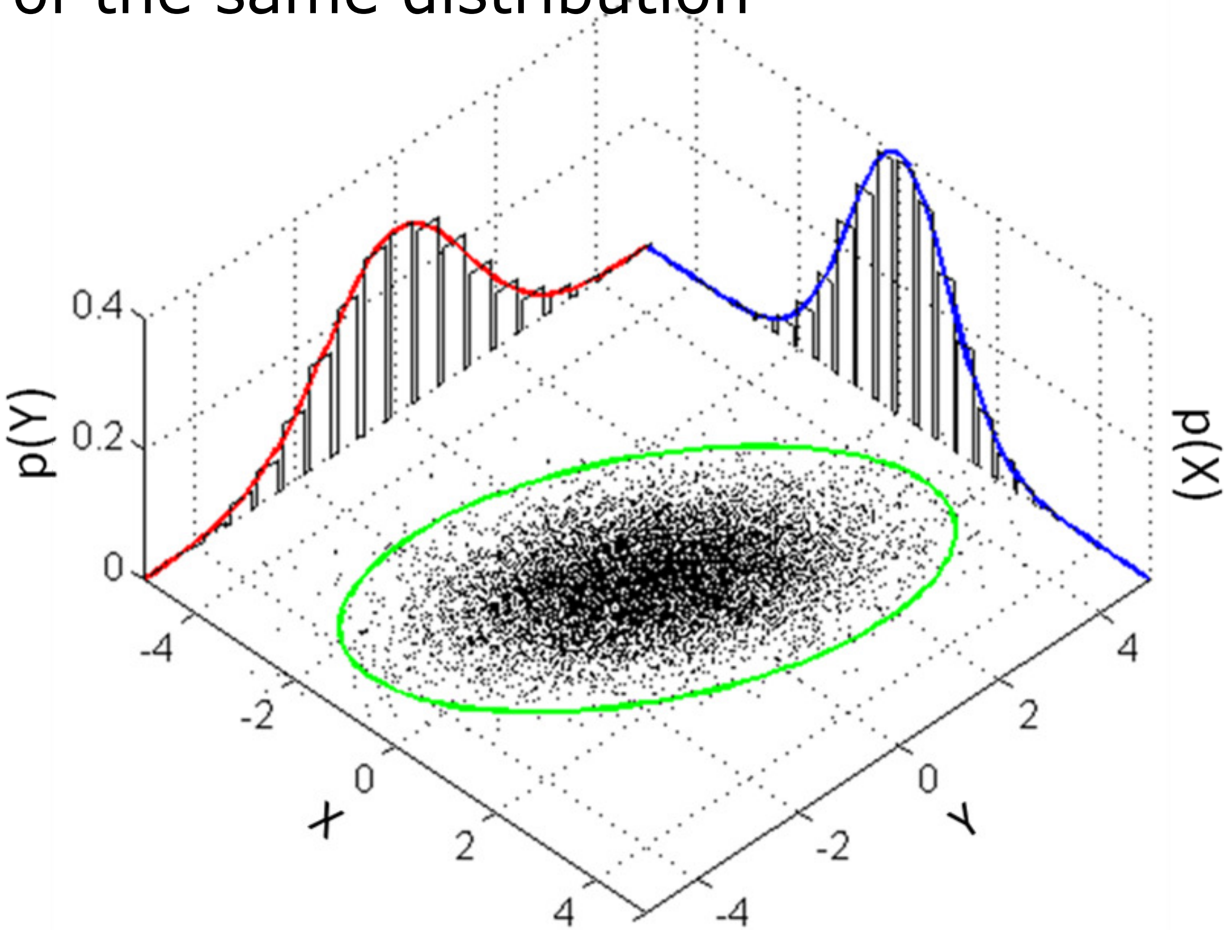
- Initialize: $x_{0|0}$ = x0, $P_{0|0}$ = small (if we know the init pos)
- Predict
  - $x_{k+1|k} = x_k + v_k dT$
  - $P_{k+1|k} = P_{k|k} + Q_k$

The noise increases the uncertainty but does not effect the mean as we assume zero-mean noise

- Update
  - $K_k = P_{k+1|k}(P_{k+1|k} + R_k)^{-1} = P_{k+1|k} / (P_{k+1|k} + R_k)$
  - $x_{k+1|k+1} = x_{k+1|k} + K_k(z_{k+1} - x_{k+1|k})$
  - $P_{k+1|k+1} = (1 - K_k)P_{k+1|k}$

Gauss vs particle set

- Green ellipse: 2D Gaussian
- Black dots: Samples of the same distribution



https://en.wikipedia.org/wiki/Multivariate_normal_distribution

Particle filter

The particle filter represents probability distributions using a set of particles, p_k, sampled from the distribution.

Each particle represents one "hypothesis" about the state.

Each particle also has a weight, initialized as $\pi=1/N$.

$$\underbrace{p_k}_{\text{particle}}=\{\underbrace{x_k}_{\text{state}}, \underbrace{\pi_k}_{\text{weight}}\}$$

Monte Carlo Localizatio (MCL)

1. Initialize the particles given what you know to start with (nothing→uniform, a lot→ very small spread) and with weight 1/N.
2. Use odometry to update all poses of particles and perturb each particle according to odomety noise (different realization of noise for each particle).
3. Use measurements and multiply the weight of each particle, i, with $p(z_k|x^i_k)$
4. Re-sample “if needed” and then return to 1.

Mapping and SLAM

- Difference?

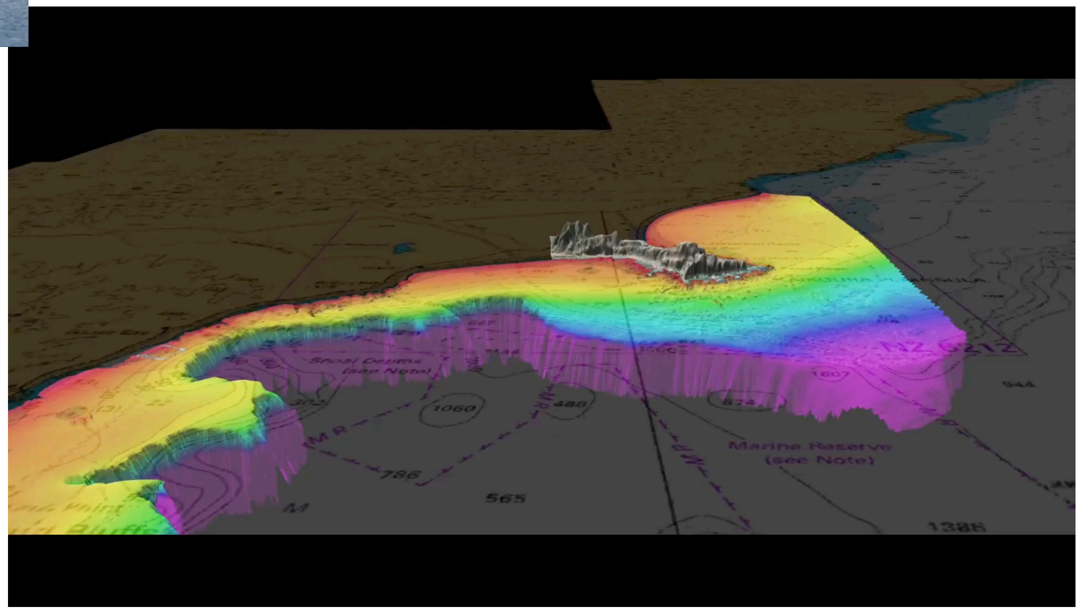
Mapping and SLAM

- Mapping
  - Position is assumed know and the focus is on building the map
- SLAM = Simultaneous Localization and Mapping
  - Neither position nor map known, need to estimate both at the same time

Example: mapping

- Assignment 4!

Example: mapping

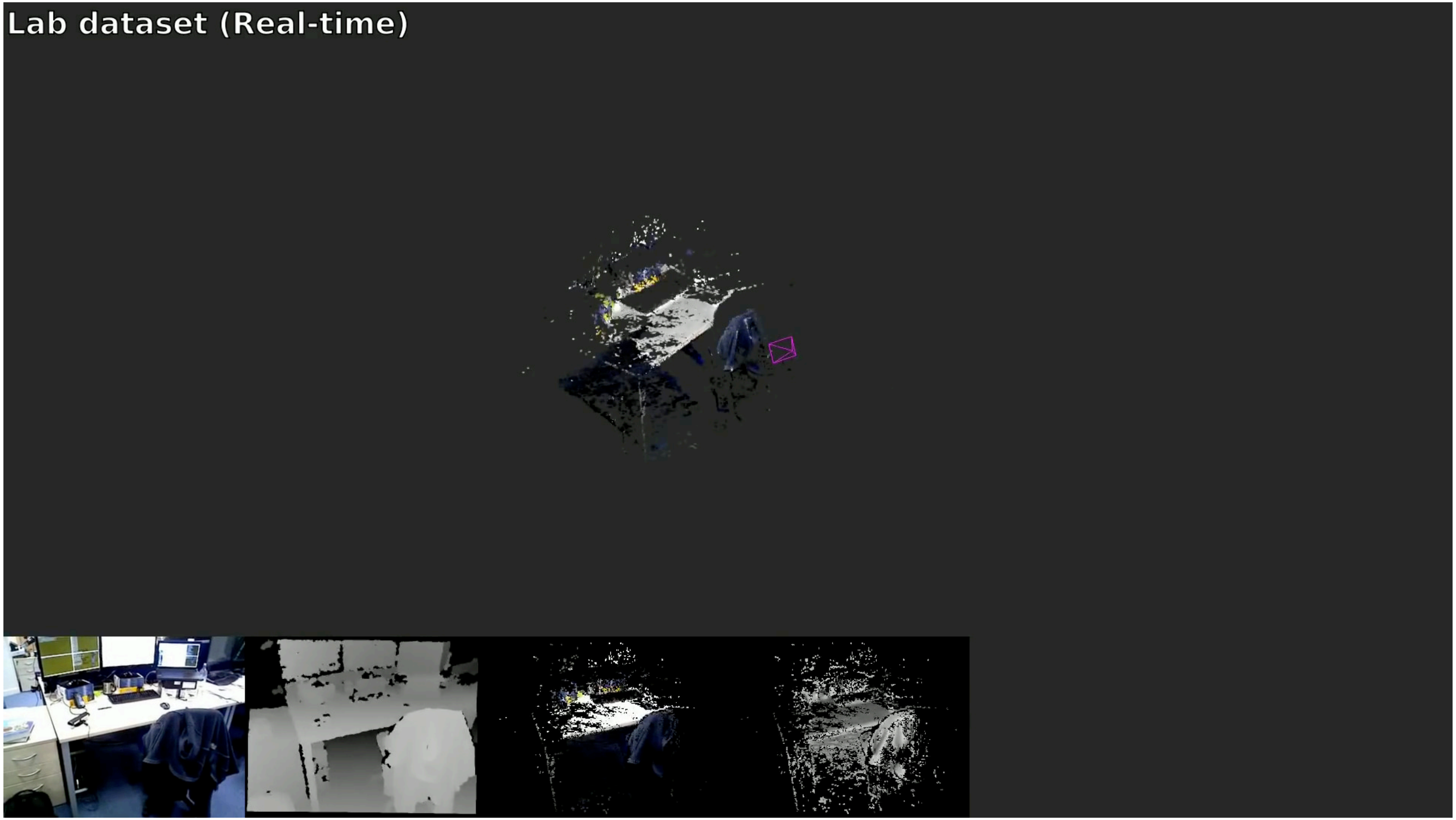


SLAM
Simultaneous Localization and Mapping

- Chicken and egg problem!
- Need map to localize
- Need posiotion to build map
- Do localization and mapping in parallell!

Example SLAM: Elastic Fusion

Notice the loop closures!



https://www.youtube.com/watch?v=XySrhZpODYs

From Localization to SLAM

- How to extend EKF and PF?

Study EKFSLAM

- Extend state
- Correlation between robot pos and landmark (suggest experiment)
- Loop closure (suggest experiment)

Practical issues with EKF SLAM

- Computational complexity $O(N^2)$ (N=nbr landmarks)
- Sensitive to outliers
- How to define landmarks? Need to know that I can find them in the environment I operate in
- …

Particile filter SLAM

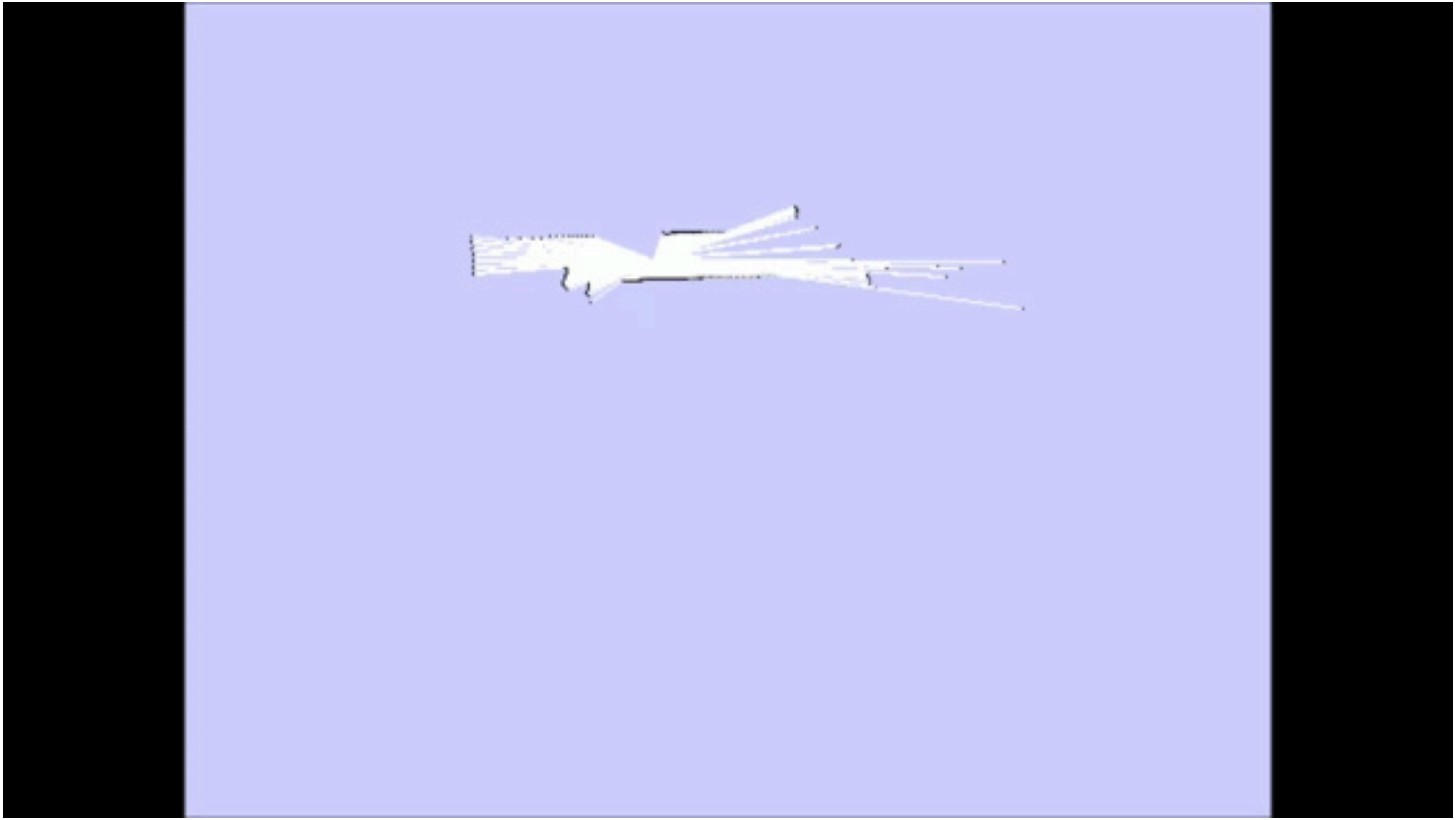
- Particle filter cannot handle the number states needed for SLAM
- Use Rao-Blackwellization!
- Factor distribution into position part and map part
  - Particle filter tracks position
  - EKF or other map technique handles map

Example: Particle filter SLAM

Notice the loop closure!

https://www.youtube.com/watch?v=jBPZIU6AIS0

2D SLAM in ROS: gMapping

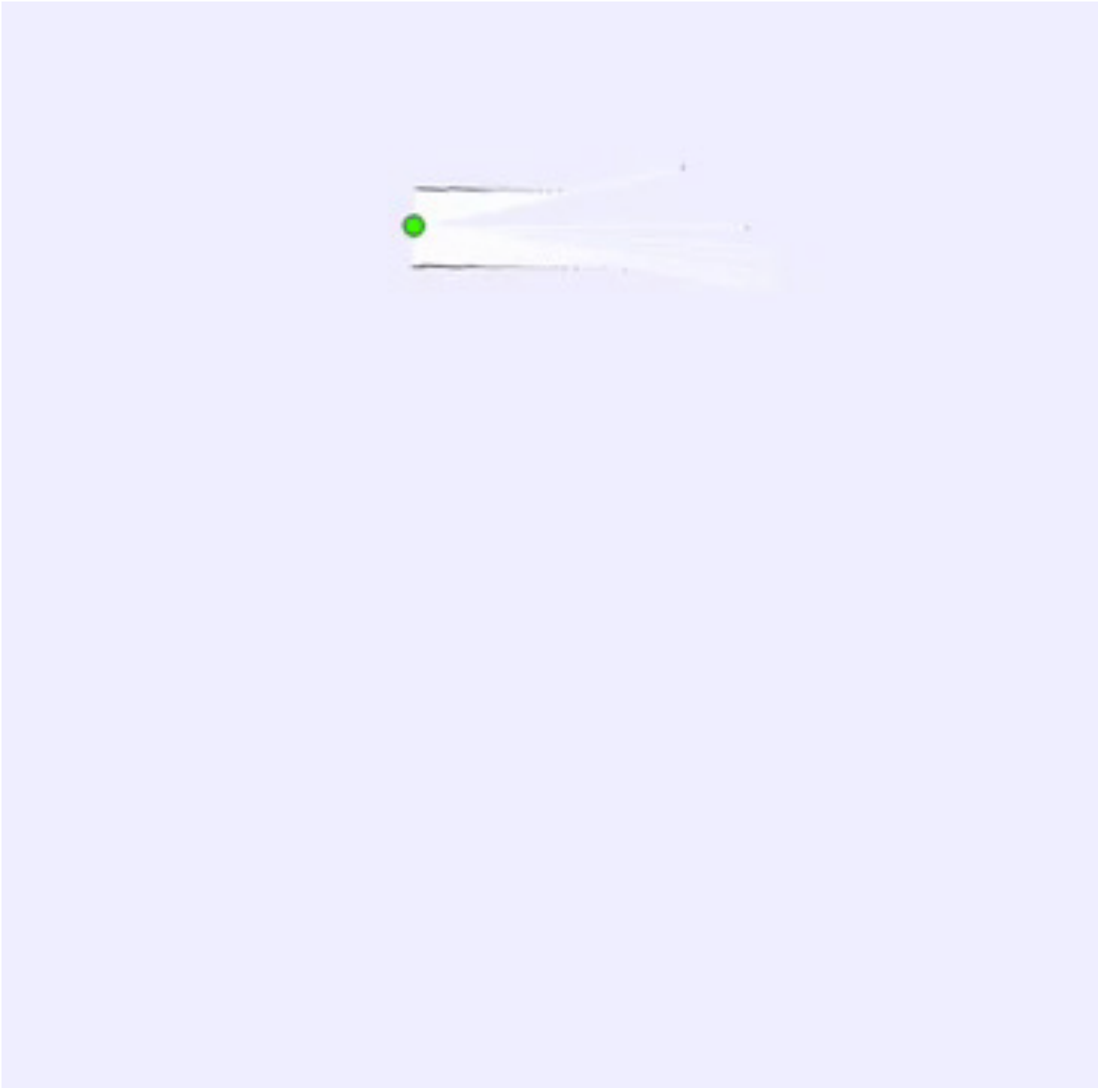


https://www.youtube.com/watch?v=uXDYc5jziag

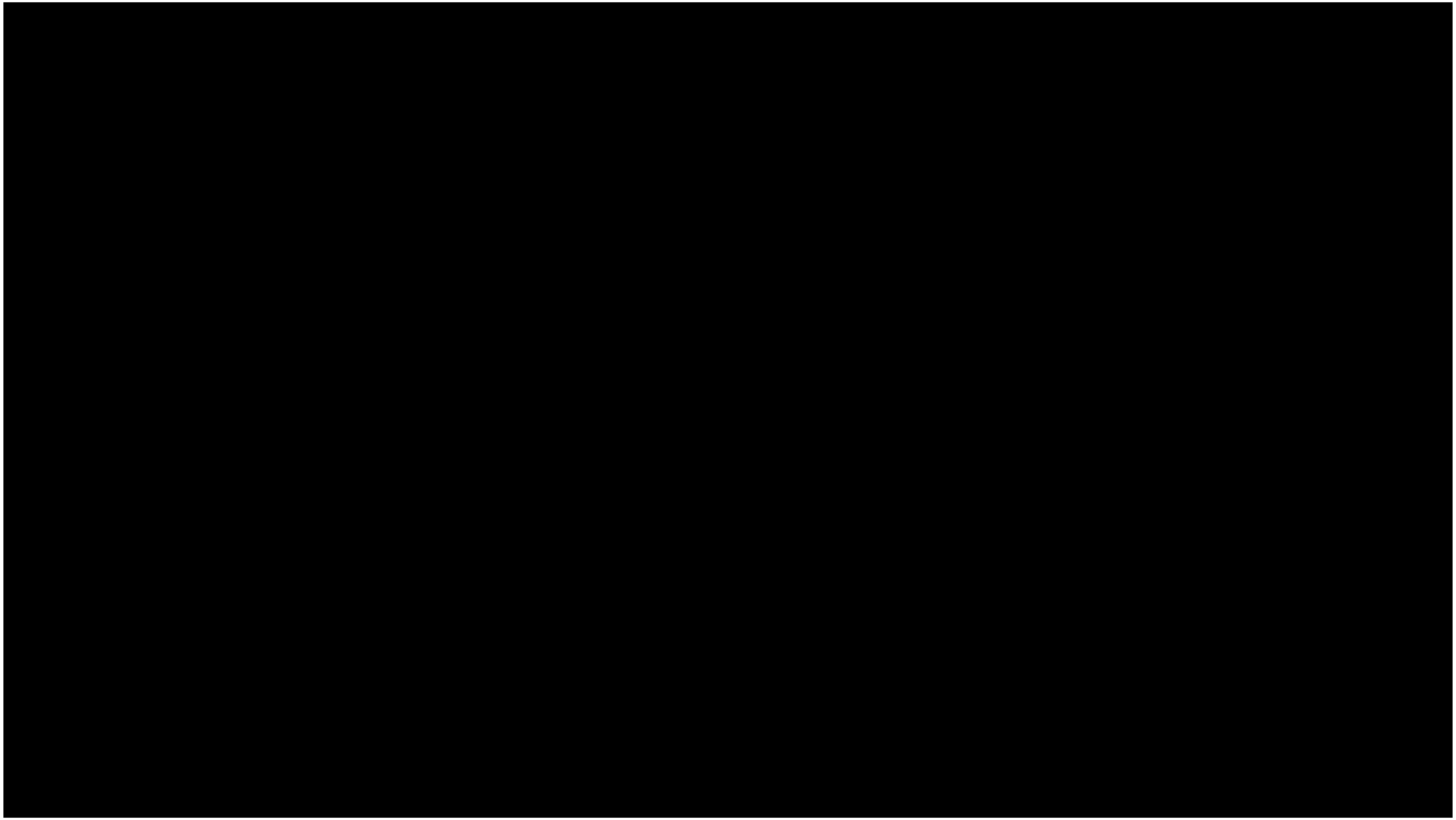
Optimization / Graph-based SLAM methods

- Used in almost all approaches now
- Idea:
  - Build a graph where
    - the nodes corresponds to robot poses and map entities and
    - edges are relative pose estimates (constraints) between nodes
  - Optimize the graph so as to minimize the ”energy” in the graph
- The constraints can be seen as springs whose zero energy state corresponds to the relative pose estimate.
  - Changing the relative pose between nodes costs energy
  - The more certain the relative pose is, the stiffer the spring

Example: Graph based SLAM



Example: Graph based 3D SLAM



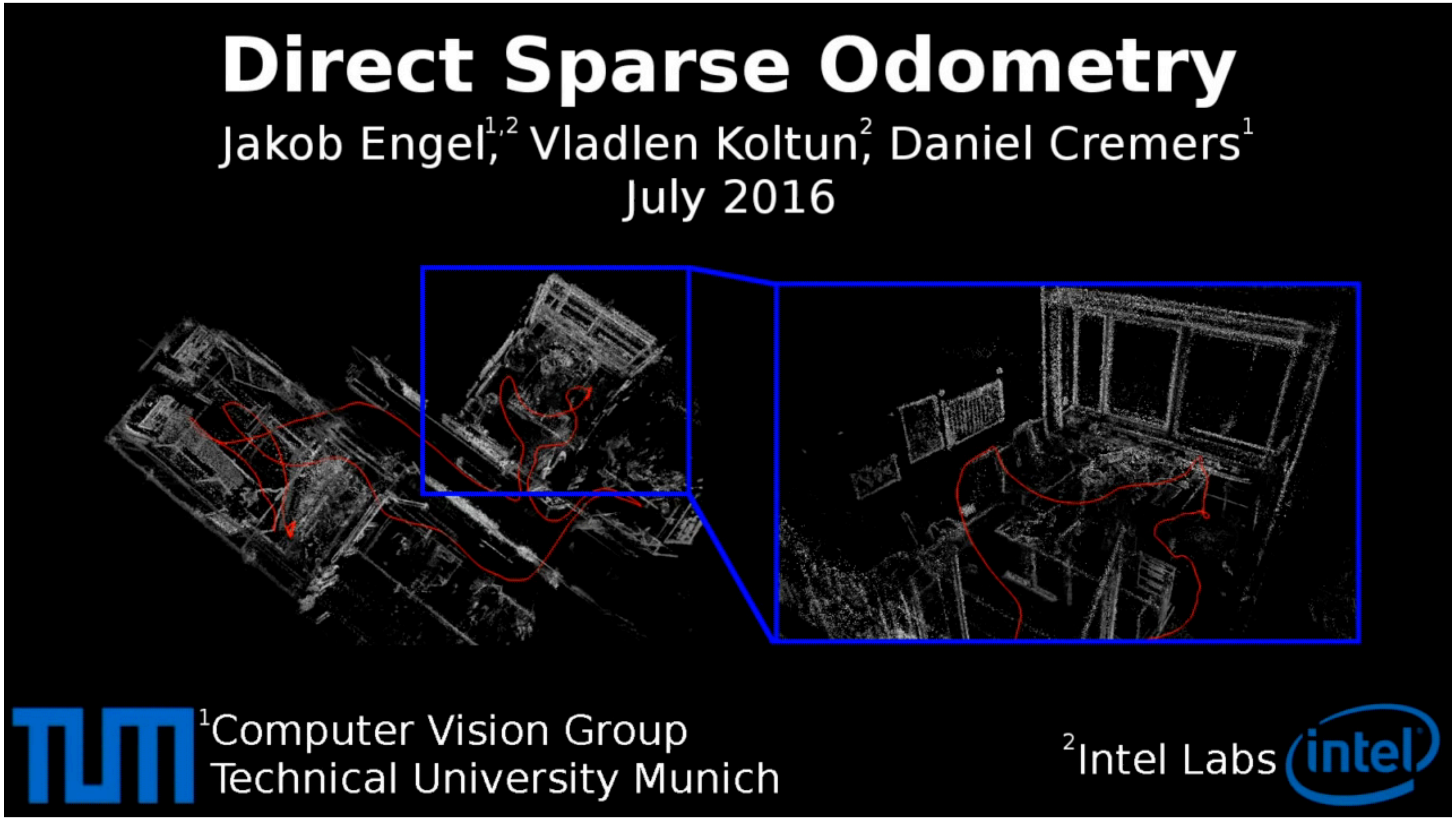
https://www.youtube.com/watch?v=08GTGfNneCI

Typical setup

- Front-end
  - Produces new nodes and edges
  - Does its best to reduce drift
  - Runs in real-time
- Back-end
  - Optimizes the graph in the background
  - Slower than real-time but online

Front-end

- Today in research often vision based - visual odometry

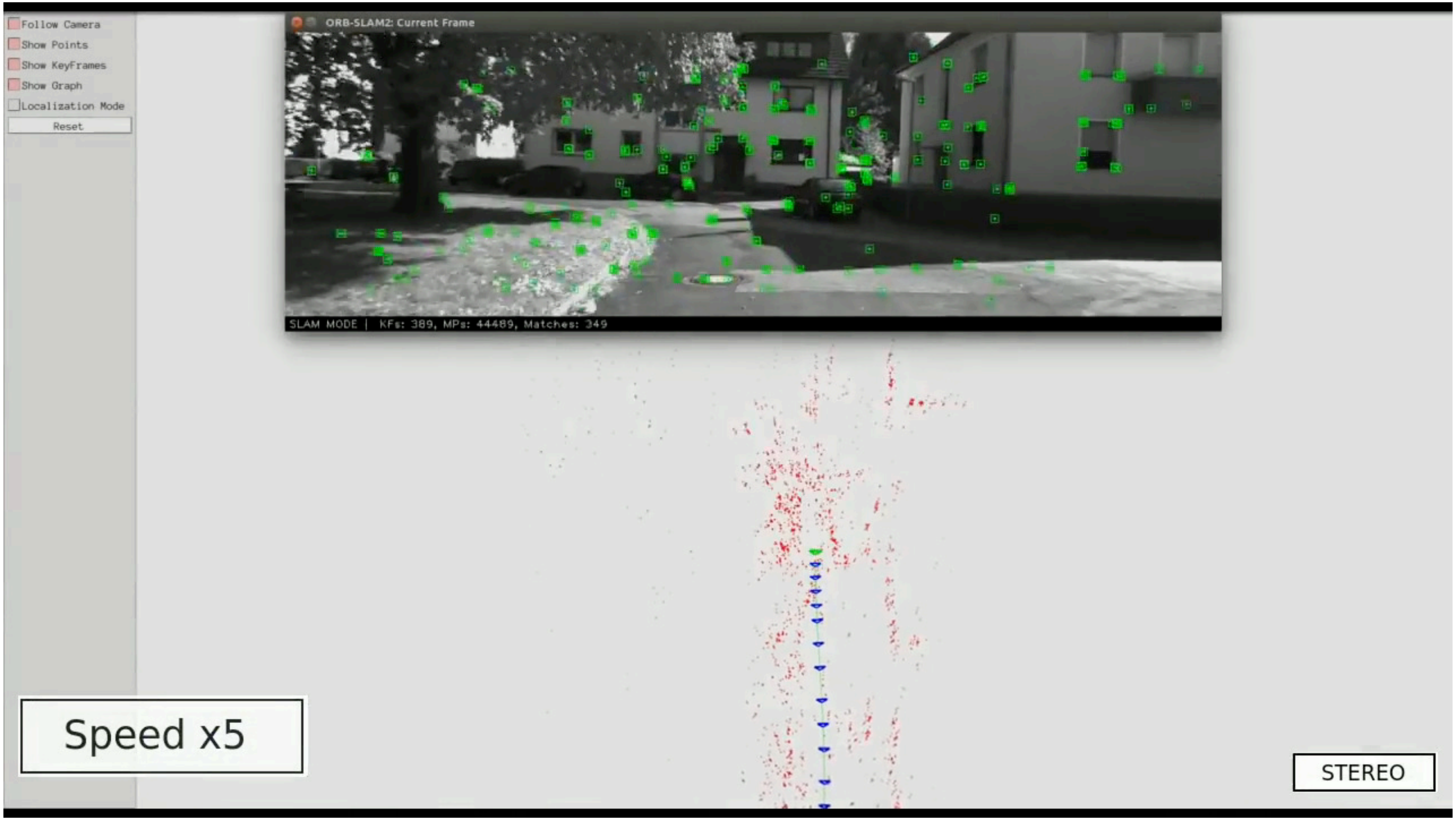


https://www.youtube.com/watch?v=C6-xwSOOdqQ

Back-end

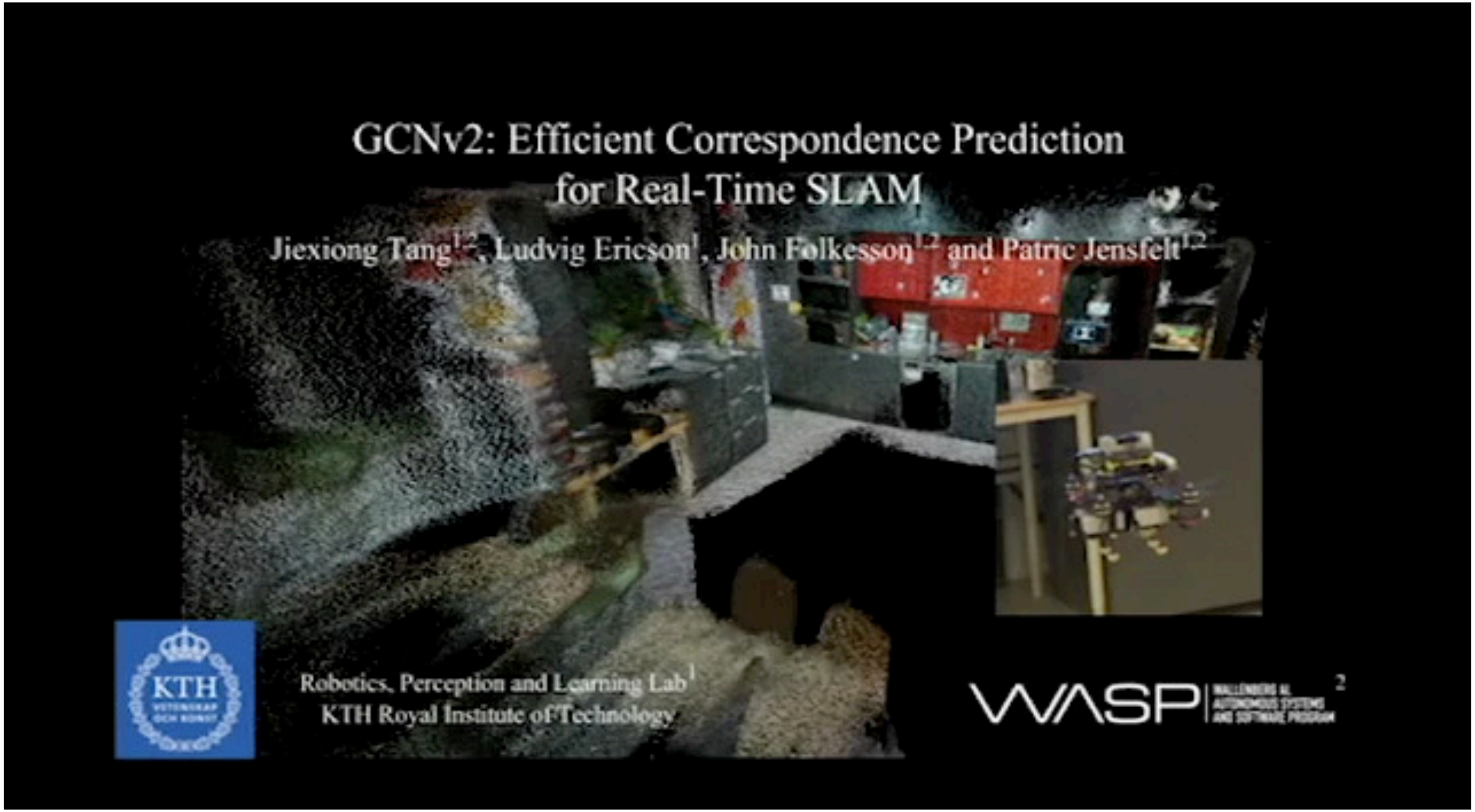
- Many packages to chose from
  - g2o (https://github.com/RainerKuemmerle/g2o)
  - gtsam (https://bitbucket.org/gtborg/gtsam)
  - Google ceres (http://ceres-solver.org)

Example: ORBSLAM2



https://www.youtube.com/watch?v=GByPKZDnG3Y

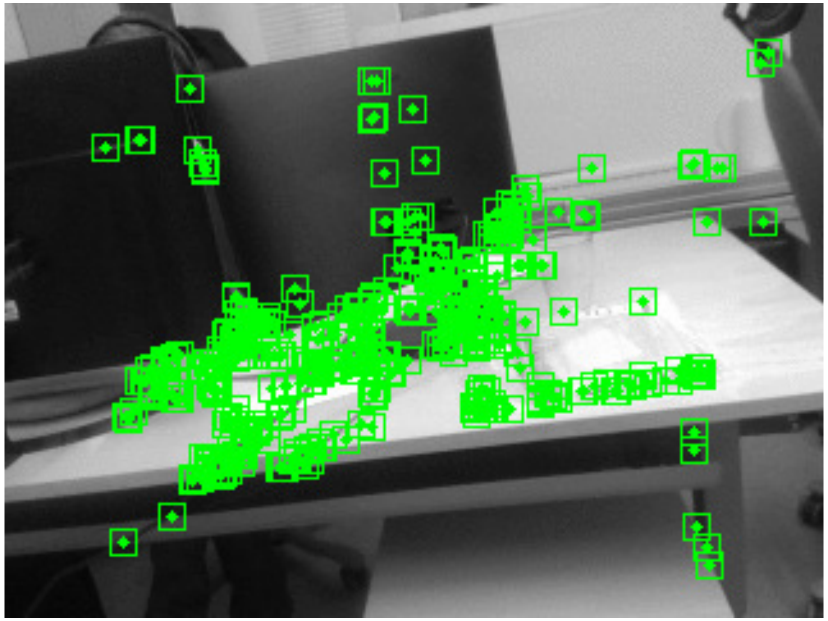
Example: GCNv2



https://github.com/jiexiong2016/GCNv2_SLAM

https://www.youtube.com/watch?v=pz-gdnR9tAM

Example: GCNv2




Notice how the GCNv2 features have a more even distribution of features than ORB
Can run ORBSLAM2 with GCNv2 on Jetson TX2
https://github.com/jiexiong2016/GCNv2_SLAM

A word on monocular vision

- The ”holy grail” of SLAM
- With a single camera one cannot determine the scale of the scene (distances)