Introduction to Mobile Robotics

Graph-Based SLAM

Daniel Büscher

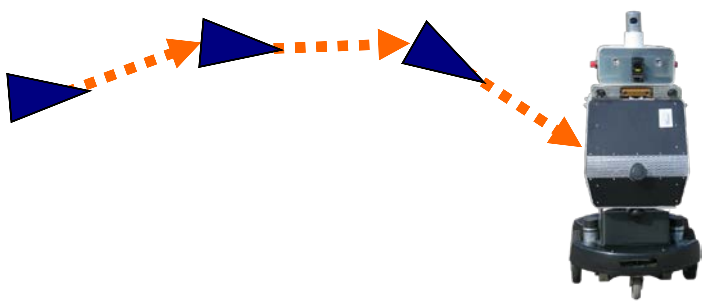
Reminder: Particle Filter Map

- ~~30 particles~~
- 250x250m^2
- 1.75 km (odometry)
- 20cm resolution during scan matching
- 30cm resolution in final map
- **Now:** define graph based on trajectory and **optimize!**

1. Graph-based SLAM

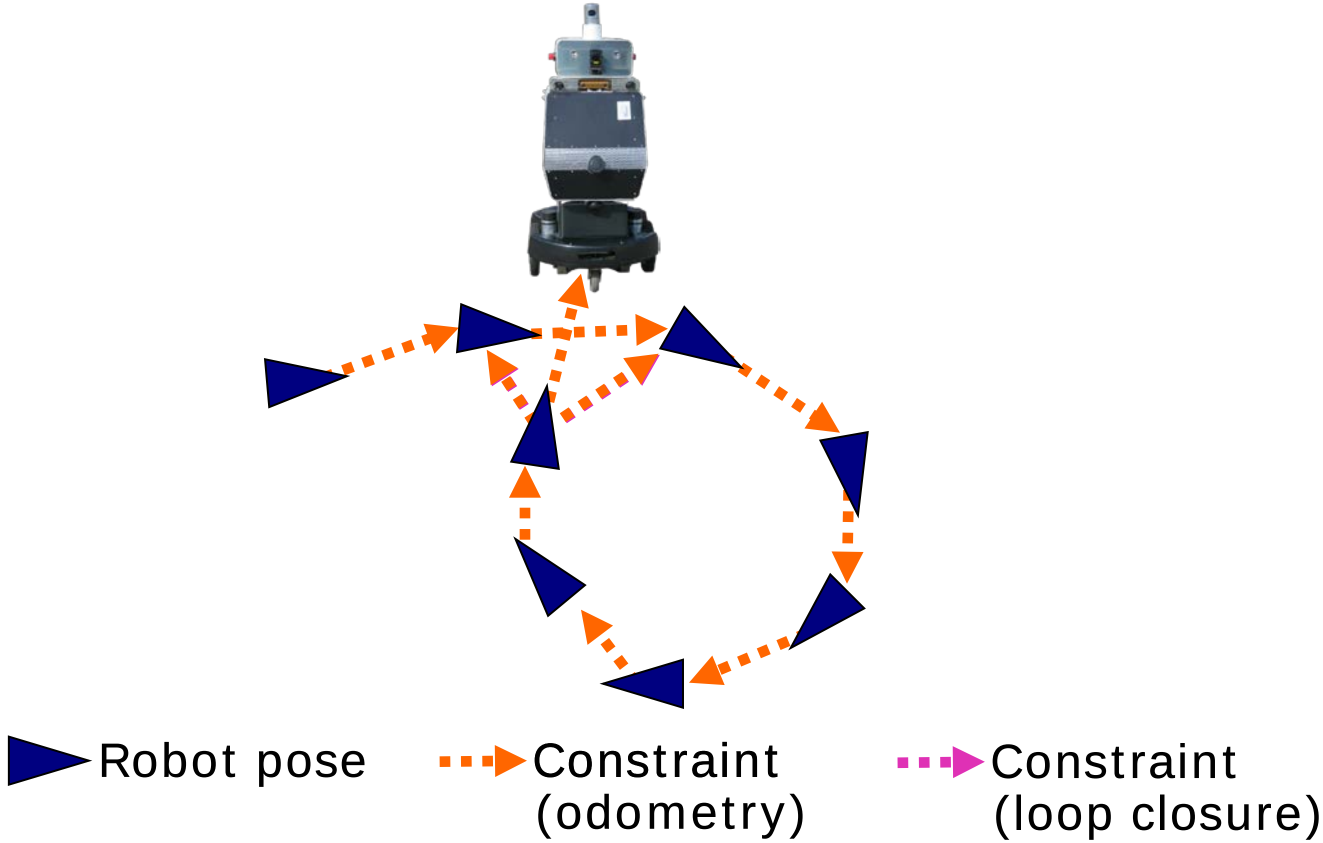
Graph-Based SLAM

- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain

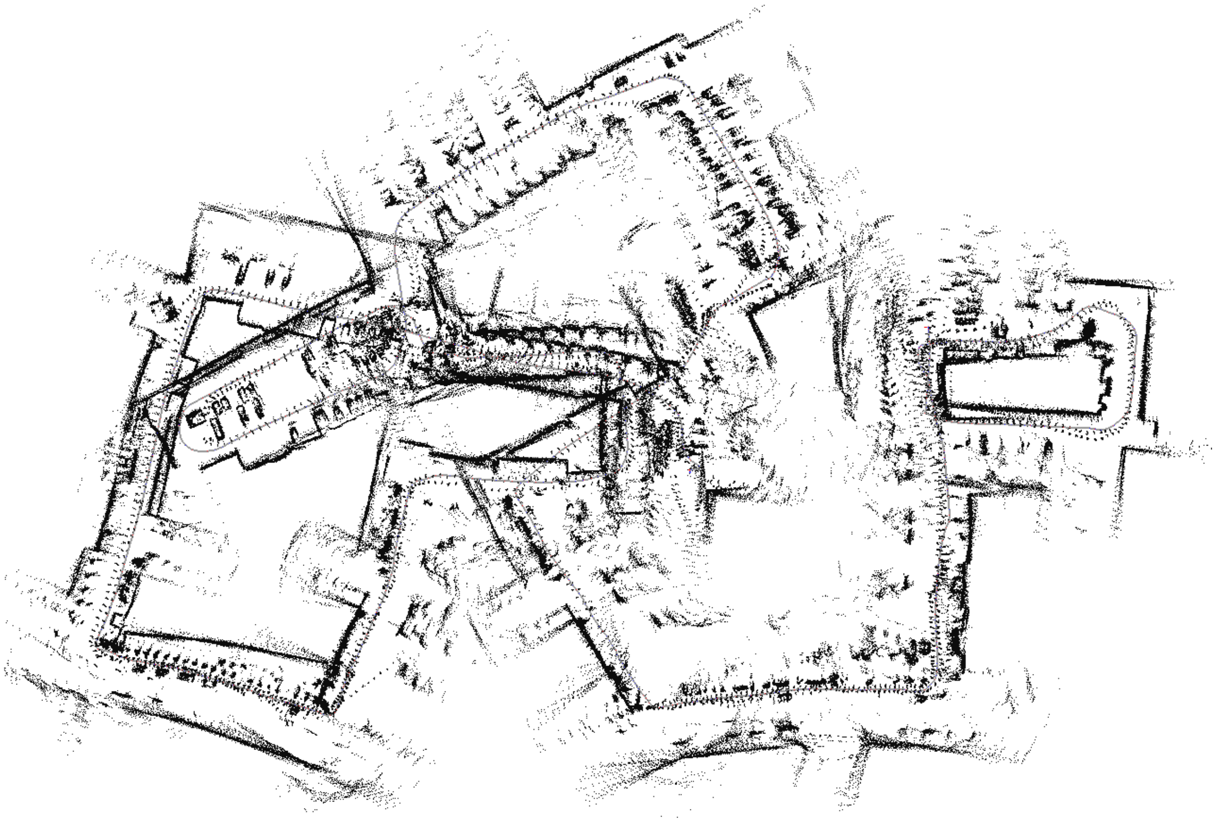


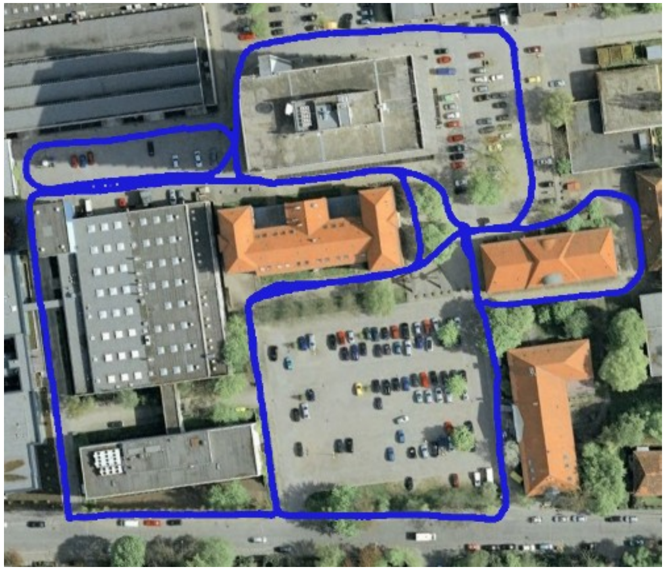
Graph-Based SLAM

- Observing previously seen areas generates constraints between non-successive poses



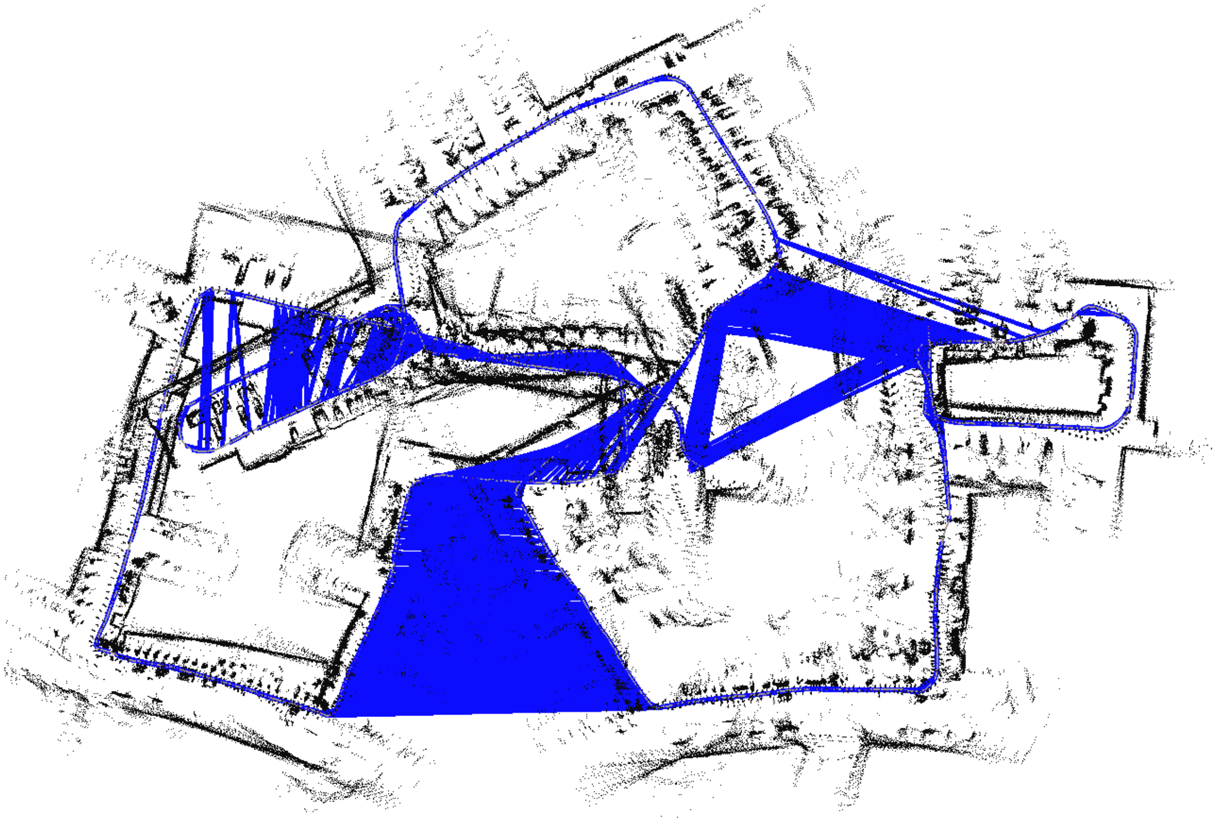
Example: Odometry Map

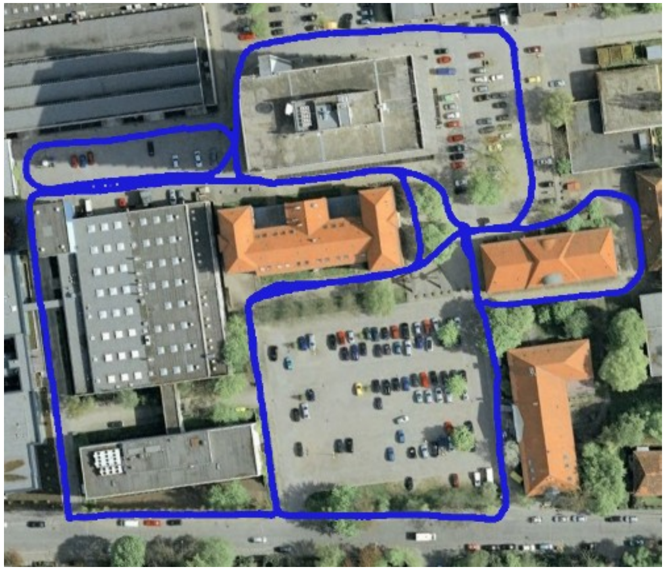




Hanover2 dataset (Courtesy of Oliver Wulf)

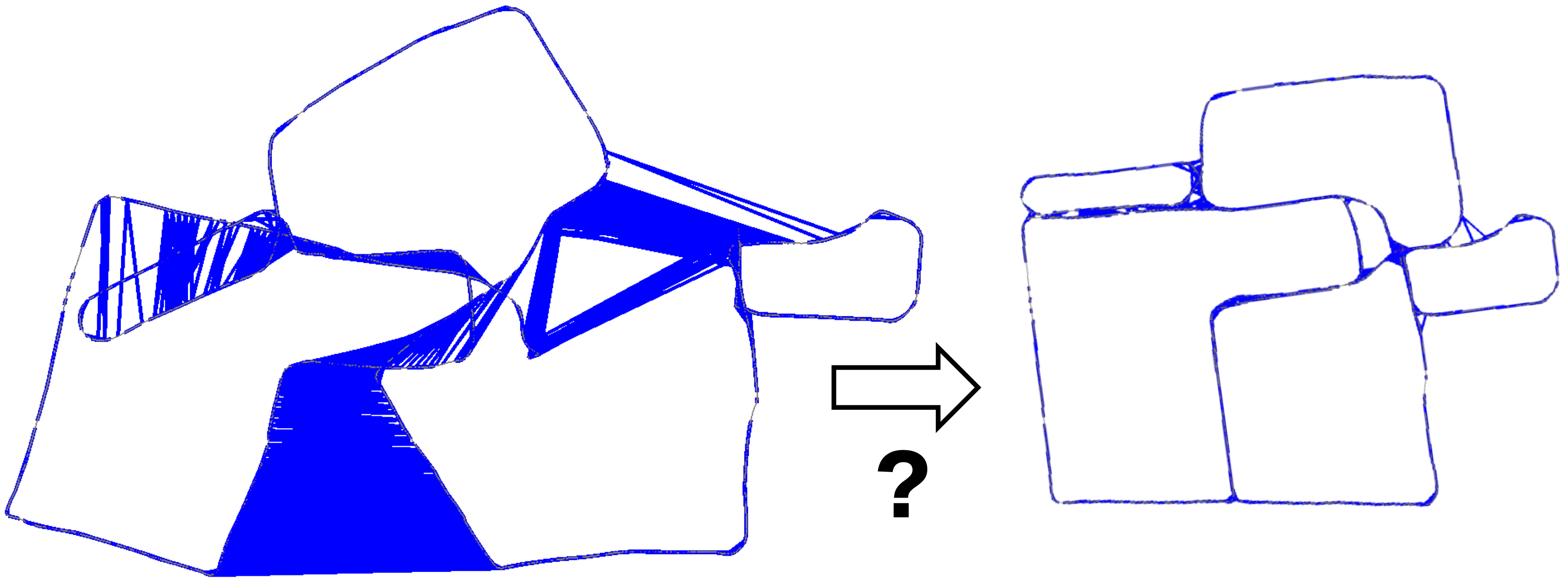
Example: Loop Closures





Hanover2 dataset (Courtesy of Oliver Wulf)

How to correct the trajectory?



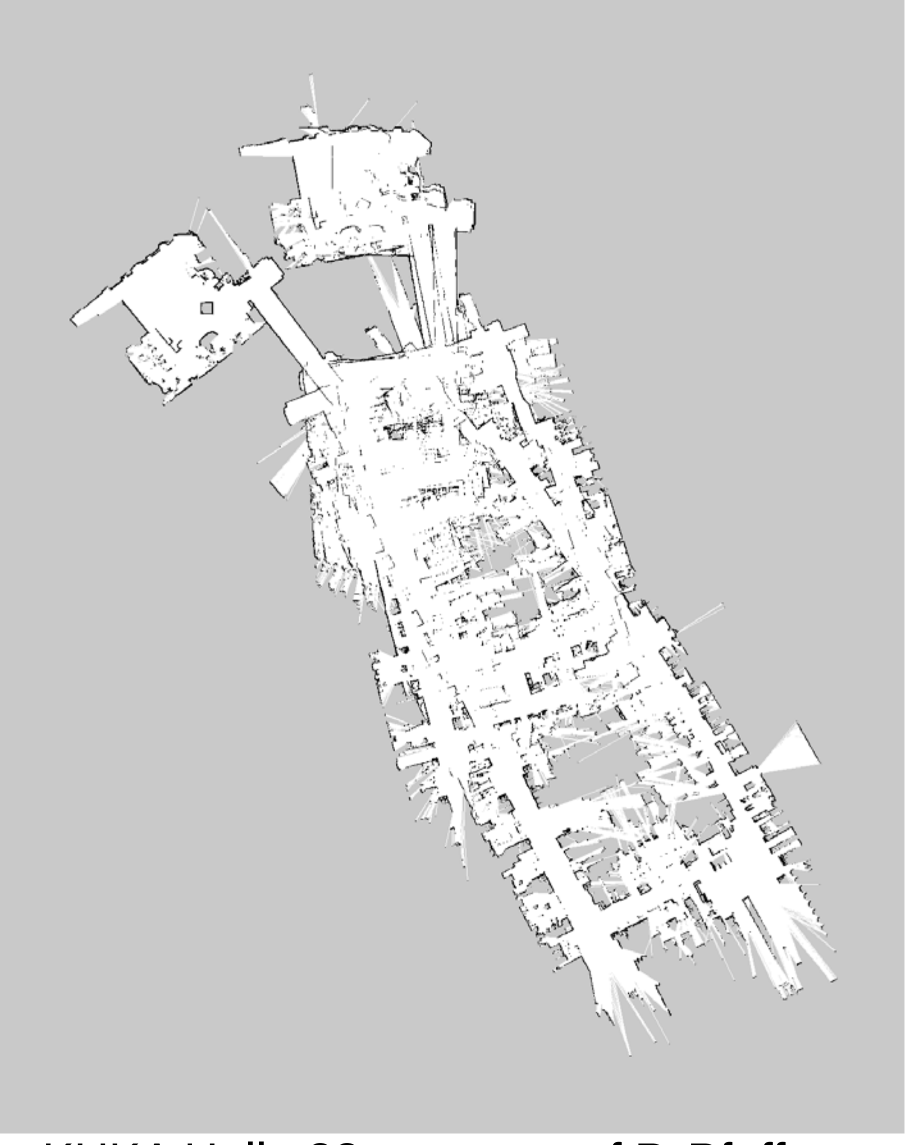
Imagine this to be a system of masses and springs!

Idea of Graph-Based SLAM

- Use a **graph** to represent the problem
- Every **node** in the graph corresponds to a pose of the robot during mapping
- Every **edge** between two nodes corresponds to a spatial constraint between them
- **Graph-Based SLAM:** Build the graph and find a node configuration that minimize the error introduced by the constraints

Graph-Based SLAM in a Nutshell

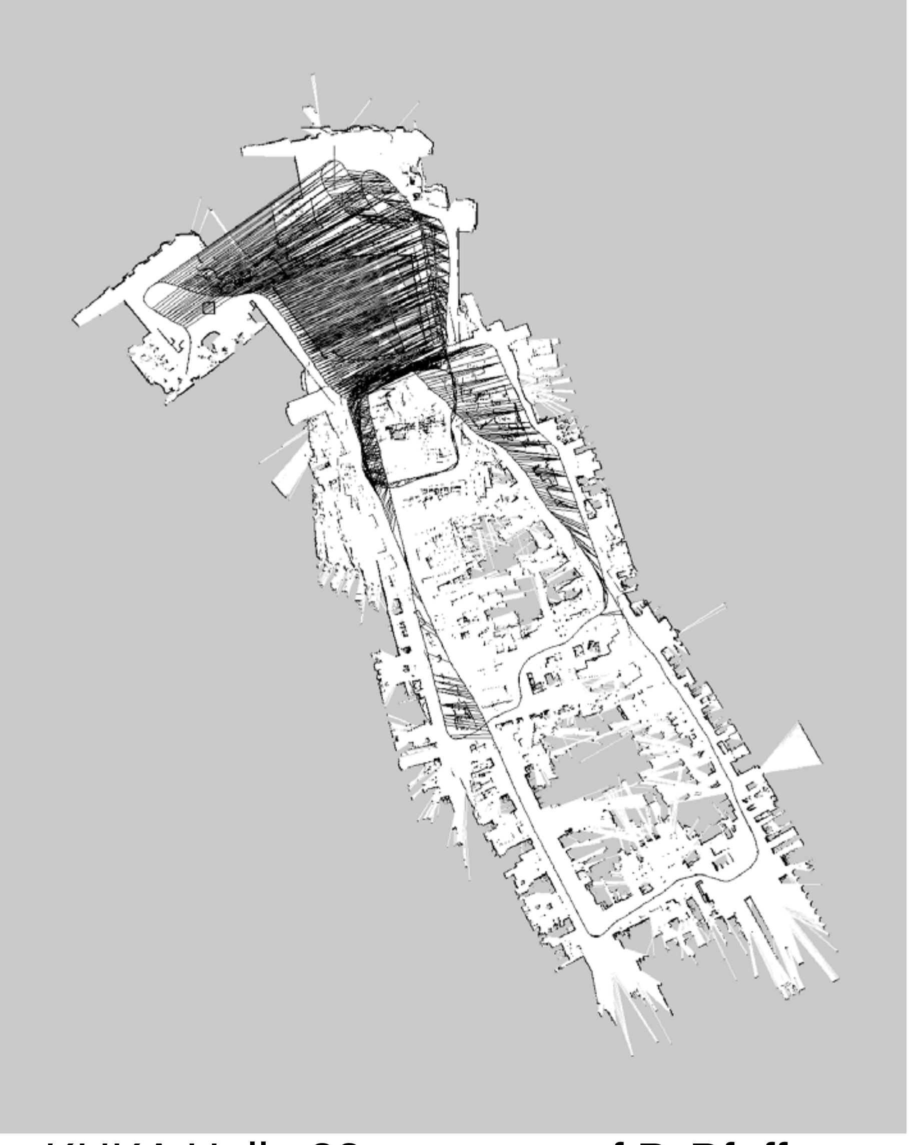
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

Graph-Based SLAM in a Nutshell

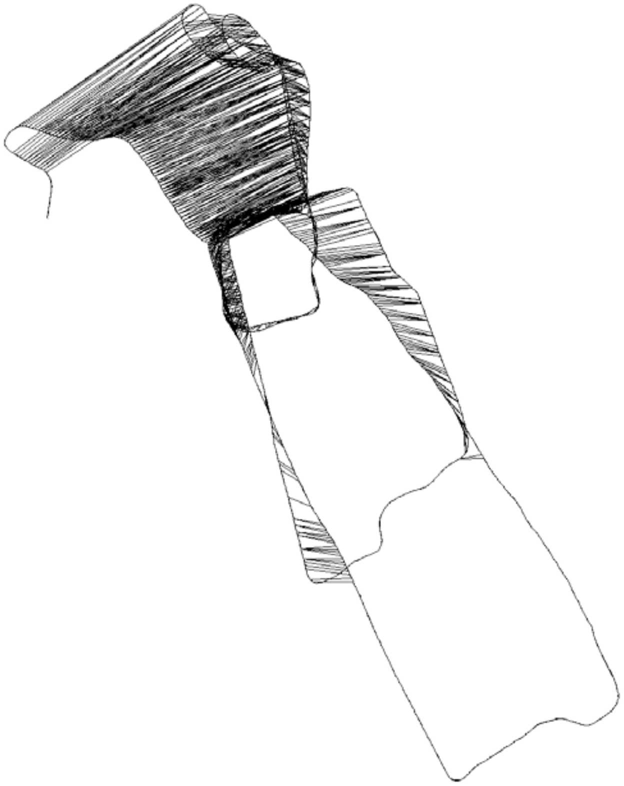
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

Graph-Based SLAM in a Nutshell

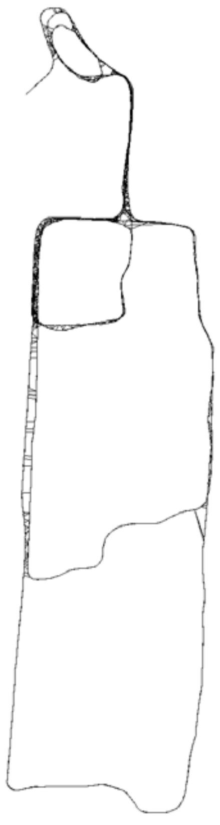
- Once we have the graph, we determine the most likely map by correcting the nodes



Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by correcting the nodes

  ... like this

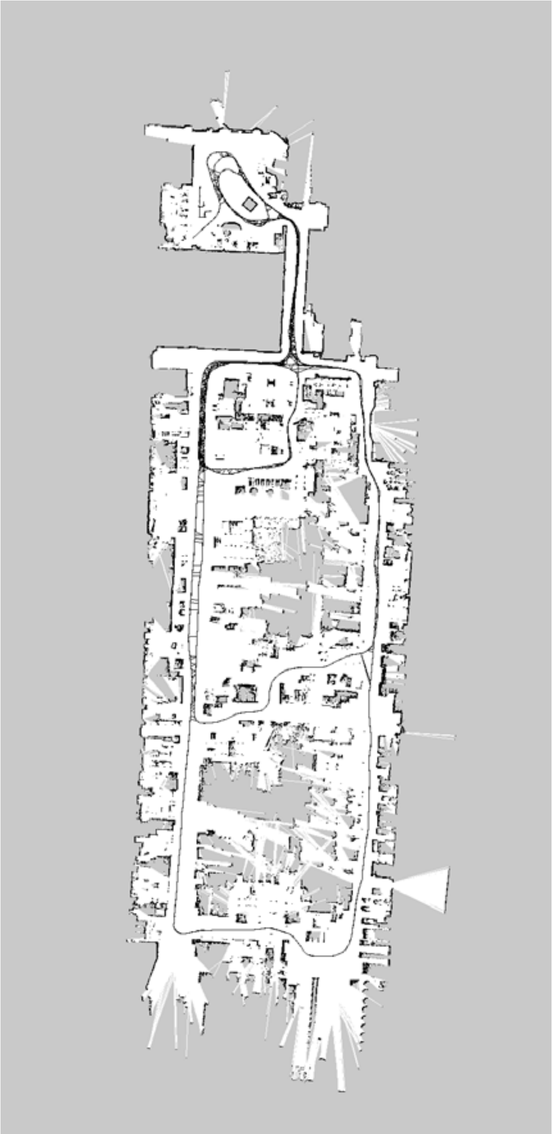


Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by correcting the nodes

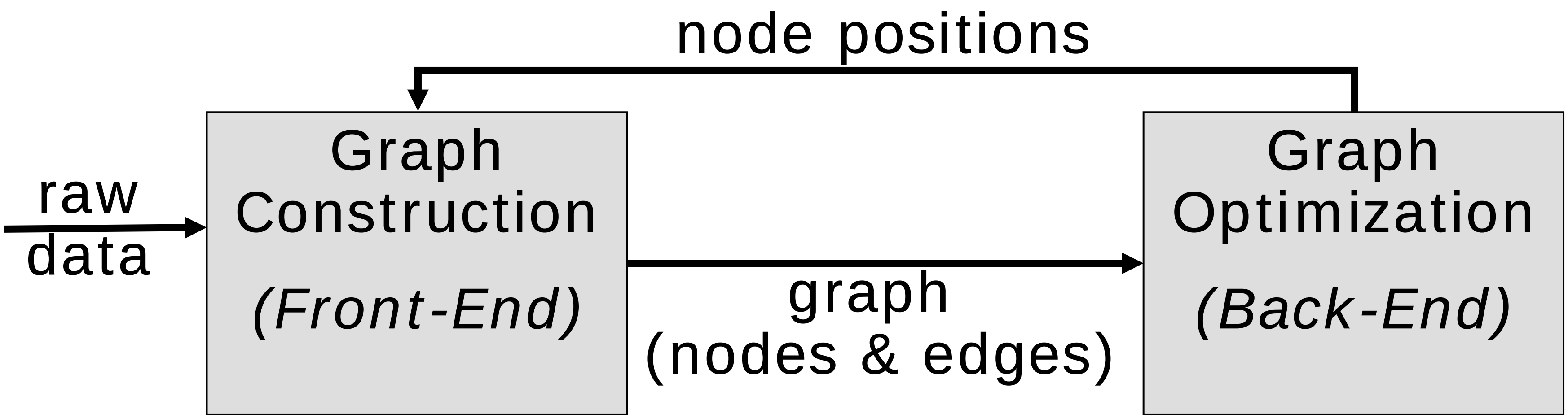
  … like this

- Then, we can render a map based on the known poses



The Overall SLAM System

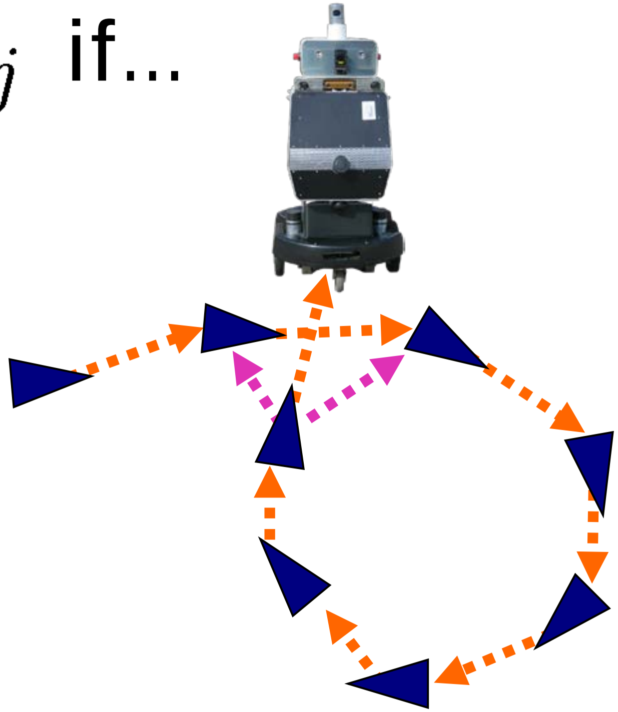
- Interplay of front-end and back-end
- A consistent map helps to determine new constraints by reducing the search space



2. The Pose Graph

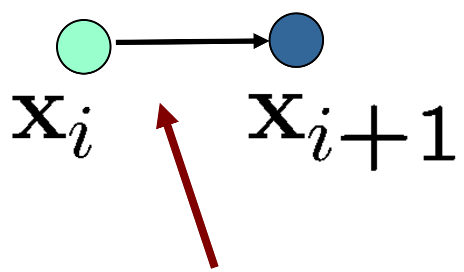
The Pose Graph

- It consists of n nodes $\mathbf{x} = \mathbf{x}_{1:n}$
- Each $\mathbf{x}_i$ is a 2D or 3D transformation (the pose of the robot at time t_i)
- A constraint/edge exists between the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ if...



Create an Edge If… (1)

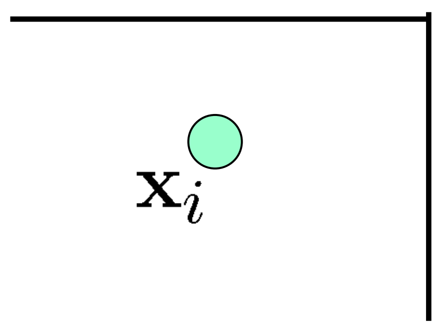
- …the robot moves from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$
- Edge corresponds to odometry



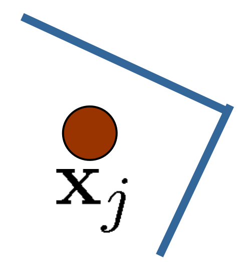
The edge represents the **odometry** measurement

Create an Edge If… (2)

- …the robot observes the same part of the environment from $\mathbf{x}_i$ and from $\mathbf{x}_j$
- Construct a **virtual measurement** about the position of $\mathbf{x}_j$ seen from $\mathbf{x}_i$



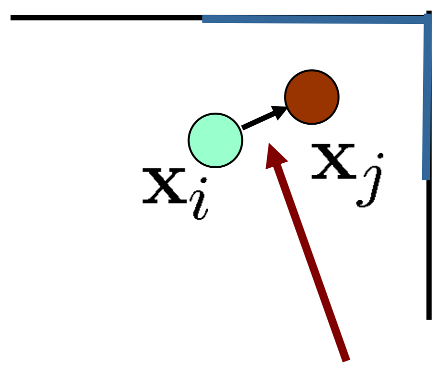
Measurement from $\mathbf{x}_i$



Measurement from $\mathbf{x}_j$

Create an Edge If... (2)

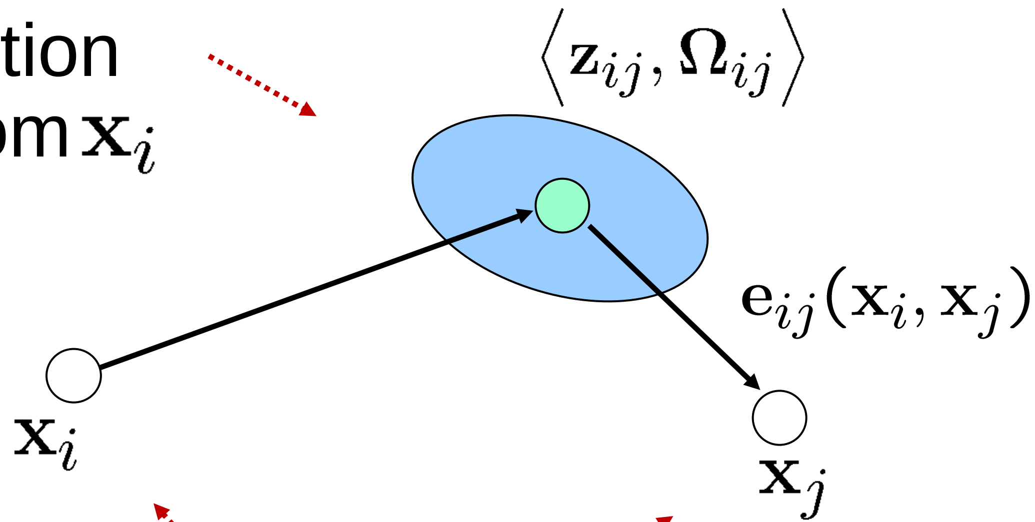
- ...the robot observes the same part of the environment from $\mathbf{x}_i$ and from $\mathbf{x}_j$
- Construct a **virtual measurement** about the position of $\mathbf{x}_j$ seen from $\mathbf{x}_i$



Edge represents the position of $\mathbf{x}_j$ seen from $\mathbf{x}_i$ based on the **observation**

Pose Graph: Loop Closure

virtual measurement:
observation
of $\mathbf{x}_j$ from $\mathbf{x}_i$



error

nodes
according to
odometry

Goal: $$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \sum_{ij} \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}$$

3. Least-squares optimization

Least Squares in General

- Approach for computing a solution for an **overdetermined system**
- “More equations than unknowns”
- Minimizes the **sum of the squared errors** in the equations
- Standard approach to a large set of problems

Problem

- Given a system described by a set of n observation functions $\{f_i(\mathbf{x})\}_{i=1:n}$
- Let
  - $\mathbf{x}$ be the state vector
  - $\mathbf{z}_i$ be a measurement of the state $\mathbf{x}$
  - $\widehat{\mathbf{z}}_i = f_i(\mathbf{x})$ be a function which maps $\mathbf{x}$ to a predicted measurement $\widehat{\mathbf{z}}_i$
- Given n noisy measurements $\mathbf{z}_{1:n}$ about the state $\mathbf{x}$

➡ **Goal:** Estimate the state $\mathbf{x}$ which bests explains the measurements $\mathbf{z}_{1:n}$

Graphical Explanation

$$\mathbf{x}$$

$$f_1(\mathbf{x}) = \hat{\mathbf{z}}_1 \qquad \mathbf{z}_1$$

$$f_2(\mathbf{x}) = \hat{\mathbf{z}}_2 \qquad \mathbf{z}_2$$

$$\cdots$$

$$f_n(\mathbf{x}) = \hat{\mathbf{z}}_n \qquad \mathbf{z}_n$$

state (unknown)	predicted measurements	real measurements

Difference defines error!

Error Function

- Error $\mathbf{e}_i$ is typically the **difference** between the **predicted and actual** measurement

$$\mathbf{e}_i(\mathbf{x}) \quad = \quad \mathbf{z}_i - f_i(\mathbf{x})$$

- We assume that the error has **zero mean** and is **normally distributed**
- Gaussian error with information matrix $\mathbf{\Omega}_i$
- The squared error of a measurement depends only on the state and is a scalar

$$e_i(\mathbf{x}) \quad = \quad \mathbf{e}_i(\mathbf{x})^T \mathbf{\Omega}_i \mathbf{e}_i(\mathbf{x})$$

Gauss-Newton: The Overall Error Minimization Procedure

- Define the error function
- Linearize the error function
- Compute its derivative
- Set the derivative to zero
- Solve the linear system
- Iterate this procedure until convergence

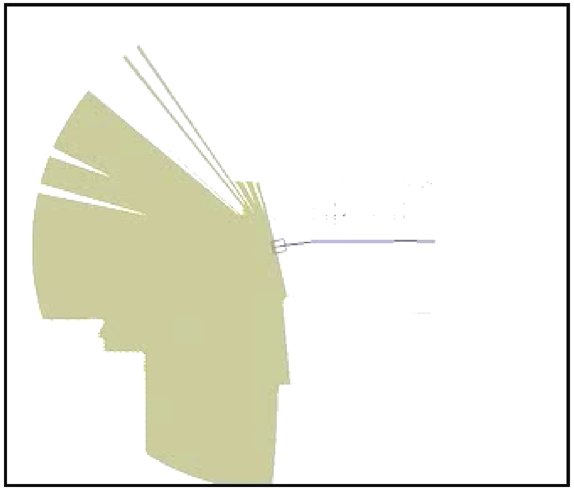
Least Squares for SLAM

- Overdetermined system for estimating the robot's poses given observations
- "More observations than states"
- Minimizes the **sum of the squared errors**

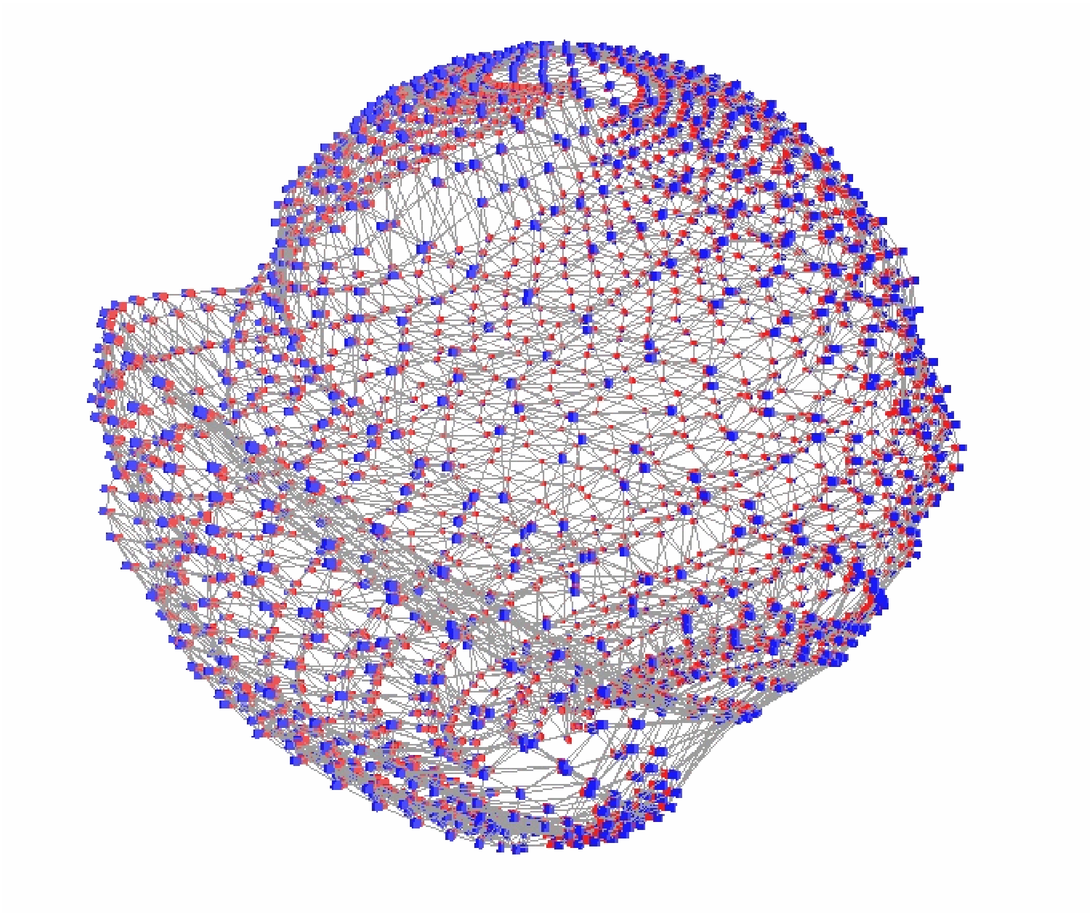
4. Examples

Sparse Pose Adjustment

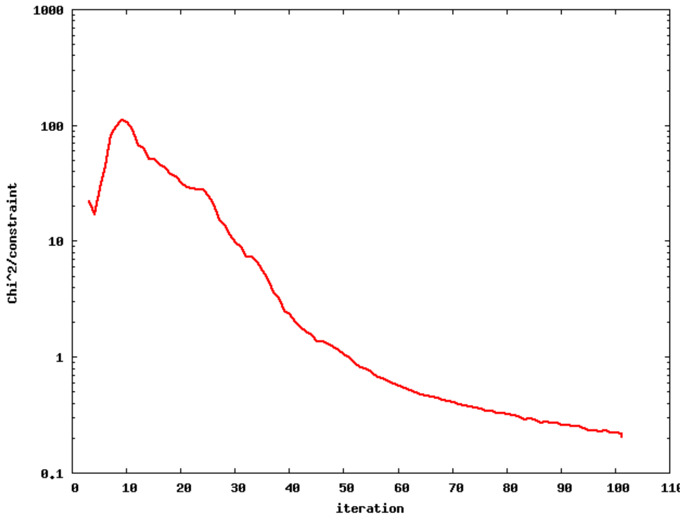
Example: CS Campus Freiburg



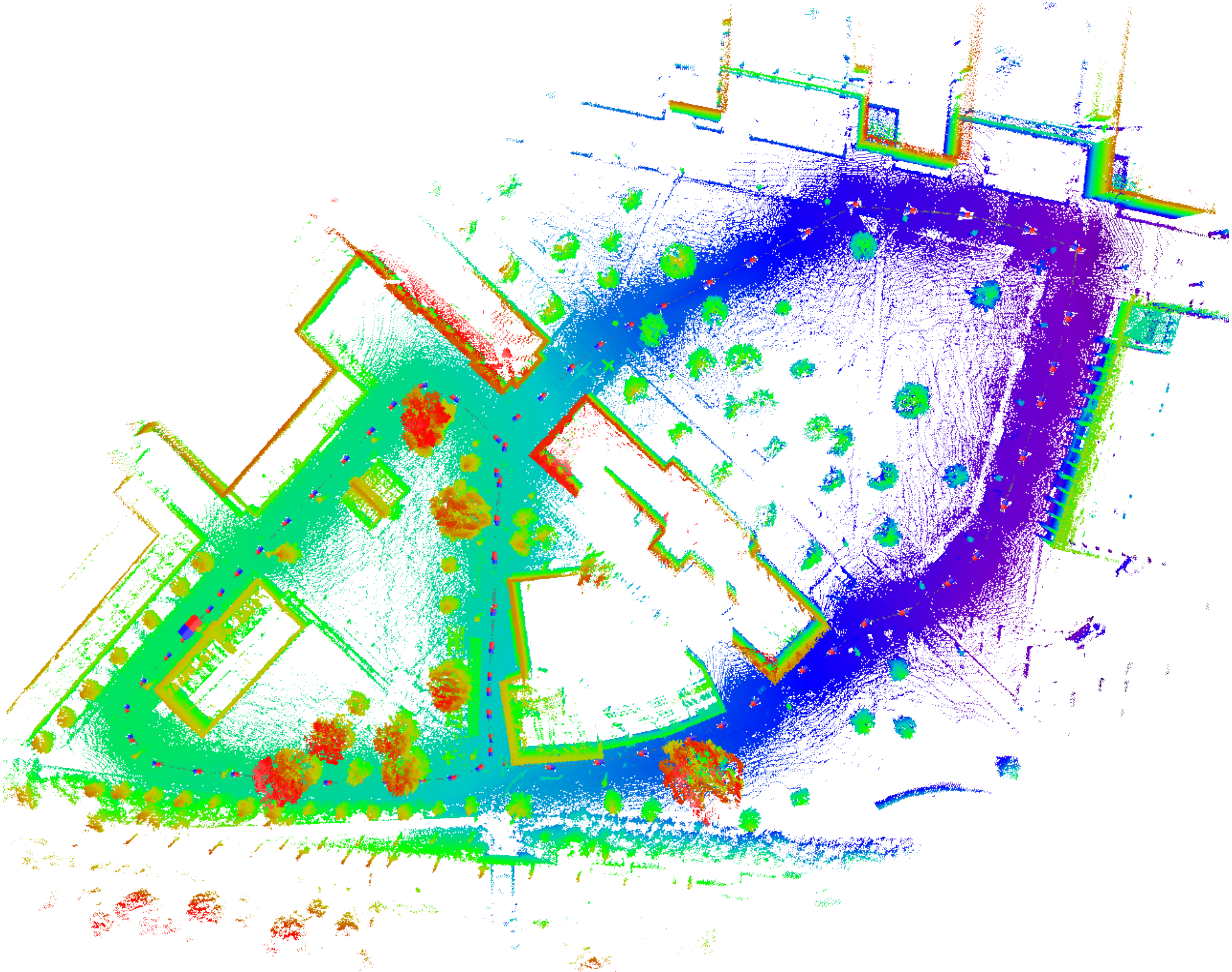
There are Variants for 3D



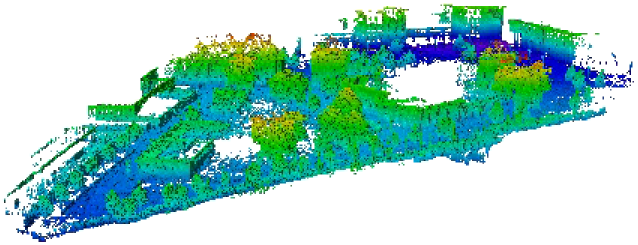
- Highly connected graph
- Poor initial guess
- LU & variants fail
- 2200 nodes
- 8600 constraints



Campus : SLAM Map



Freiburg Campus Octomap



Kolorierte Punktwolke

Conclusions

- Graph SLAM: optimization procedure
- Error functions compute the mismatch between the state and the observations
- The back-end part of the SLAM problem can be effectively solved with Gauss-Newton error minimization
- Currently one of the state-of-the-art solutions for SLAM