Principles of Robot Autonomy I

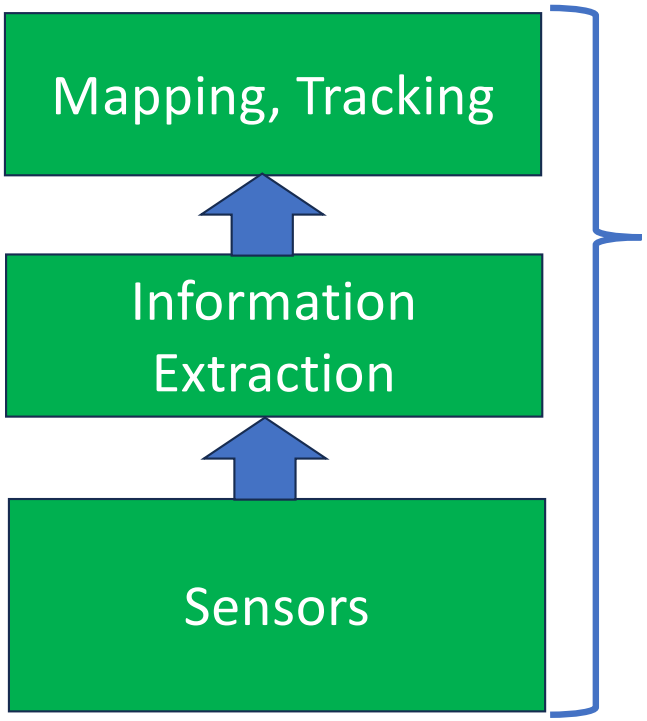
SLAM, Factor graphs, Pose Graphs

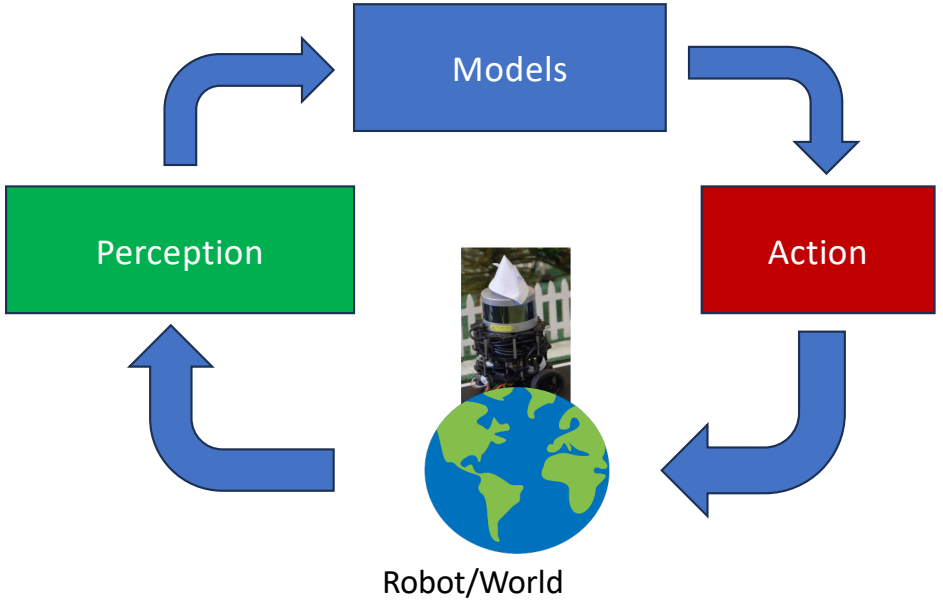
Logistics

- Homework 3: due today, Oct 28, 5pm
- Homework 4: out Tues Nov 4 (After the midterm)
- No labs: Wed–Fri this week (midterm), and Mon–Tues next week (democracy day)
- No lecture: Tues, Nov 4 (Democracy day)
- Video lecture Thurs, Nov 6 (Mac traveling)
- Midterm window: Wed, Oct 29, 5pm – Sat, Nov 1
  - Take home, 72 hour window
  - Check out exam on gradescope
  - You will have personal 5 hour time slot
  - Open notes, book, HW solutions
  - No internet, no GenAI, no working with others
- Lecture 11:
  - Features
  - Image convolutions
  - CNNs and learning-based semantic CV

Robot Perception

Perception Stack



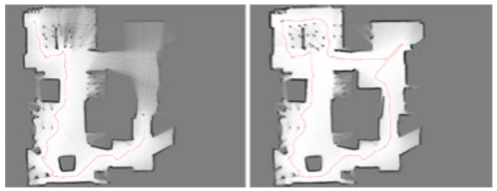


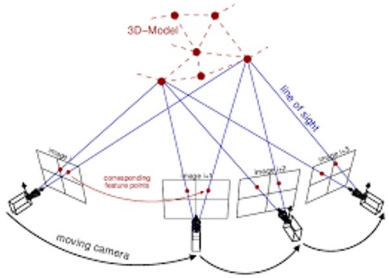
Perception Stack: Computer vision, filtering, SLAM

Use sensor data to update the models

Localization, Mapping, Tracking:

- EKF/Monte Carlo localization
- Occupancy grid mapping
- Factor graphs/SLAM
- Tracking (EKF and Particle Filter)
- **AA273: Filtering (Schwager)**
- **AA275: Navigation (Gao)**



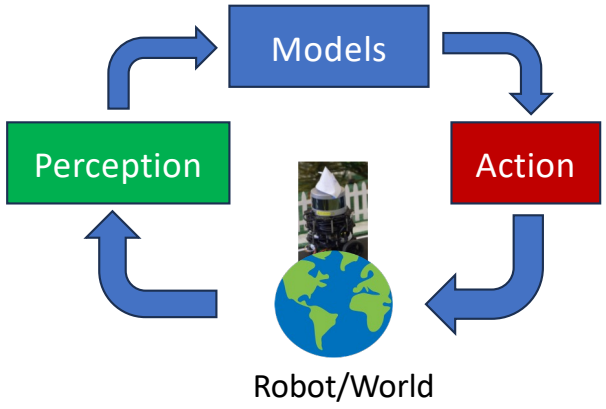


Information extraction

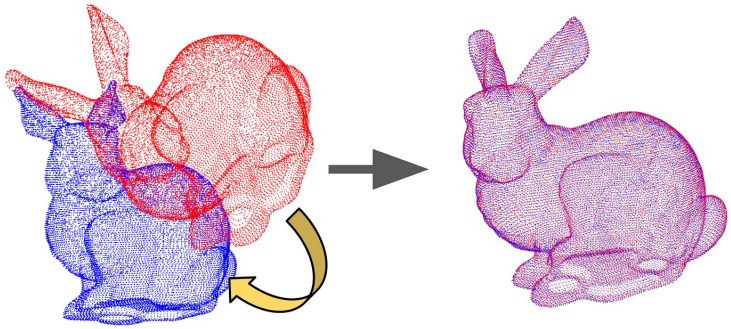
- Computer vision: features, correspondences, Structure from Motion (SfM), depth
- Lidar scan matching, ICP
- **CS231A: Comp Vision**

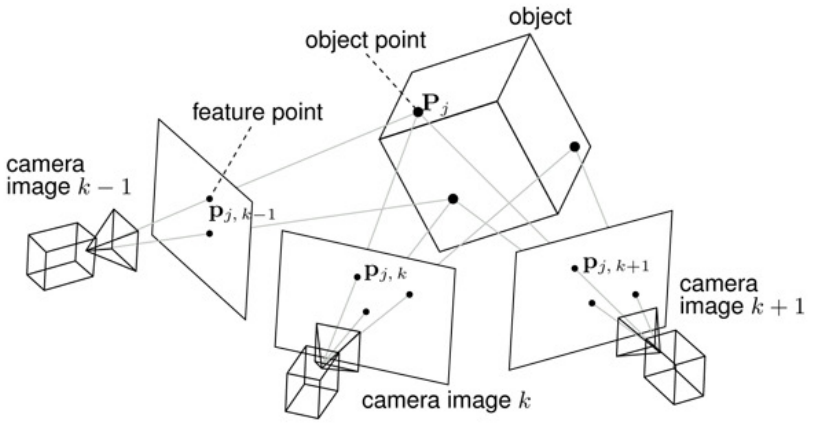
Sensors:

- RGB Cameras, RGB-D/stereo cameras, Lidar
- IMU, GPS, wheel encoders



Point cloud alignment and SfM both give *Pairwise Relative Camera poses*





- Obtain the relative pose: The pose transform required to move P_A to align with P_B

$$(R_{BA}, \tau_{BA})$$

Rotation matrix

Translation vector

Simultaneous Localization and Mapping (SLAM)

- How to reconstruct the robot's **trajectory** and **map feature locations** from sensor measurements?
- Large, highly heterogeneous research field
- Lots of intersection with multi-view SfM in CV. The distinctions are blurry.
  - SLAM is more focused on retrieving robot trajectory in real time with real-world data. The map is often secondary or even absent. Often includes IMU data.
  - SfM is more focused on accuracy of the map reconstruction. Compute time, camera poses, and robot trajectory are often secondary. Rarely includes IMU data.
- Earliest methods (Leonard, Durrant-Whyte 1991) used Extended Kalman Filters (EKF-SLAM).
- Modern methods all use a **Factor Graph Optimization Approach**

NeRFBridge for SLAM with NeRFs

NerfBridge: Bringing Real-time, Online Neural Radiance Field Training to Robotics

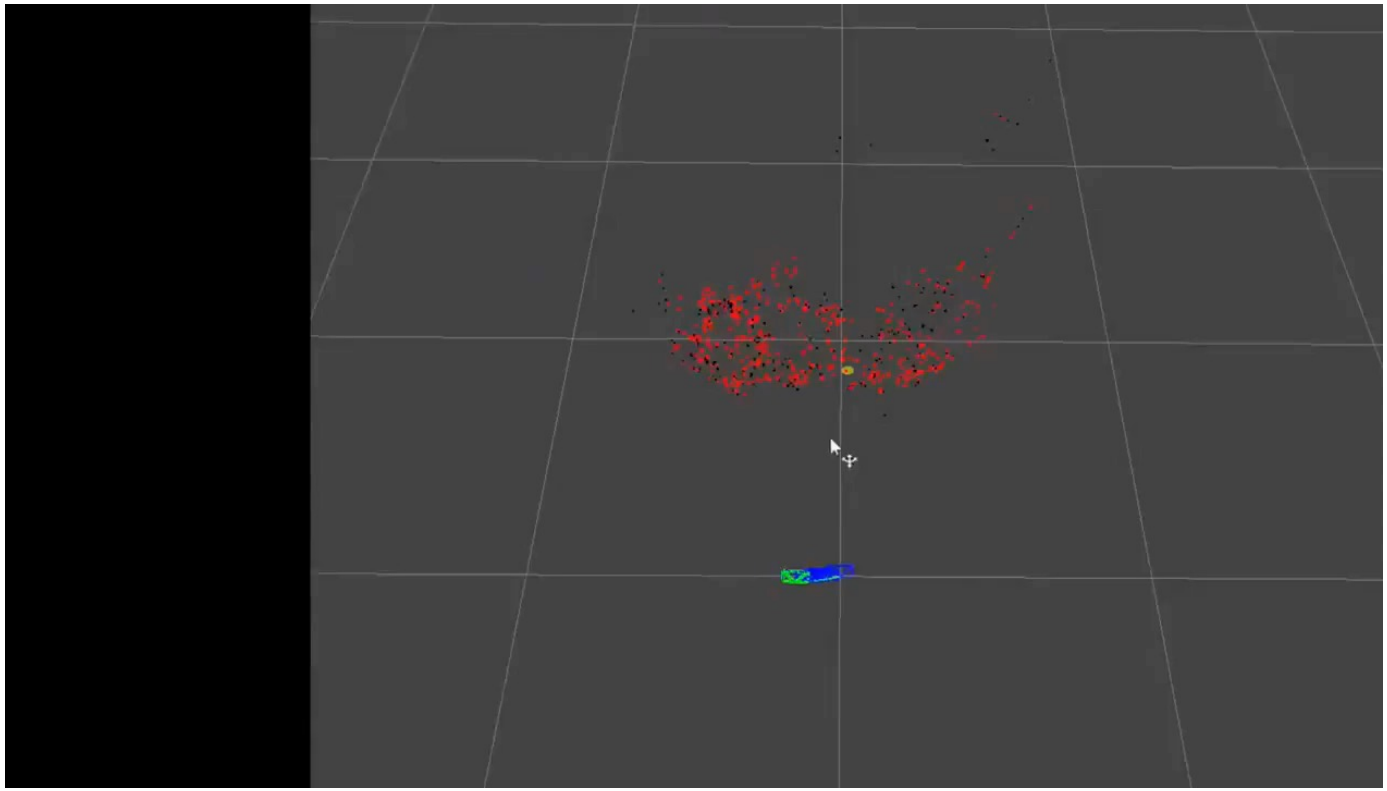
Javier Yu, Jun En Low, Keiko Nagami, and Mac Schwager

Stanford University

Contact: javieryu@stanford.edu
Code: https://github.com/javieryu/nerf_bridge

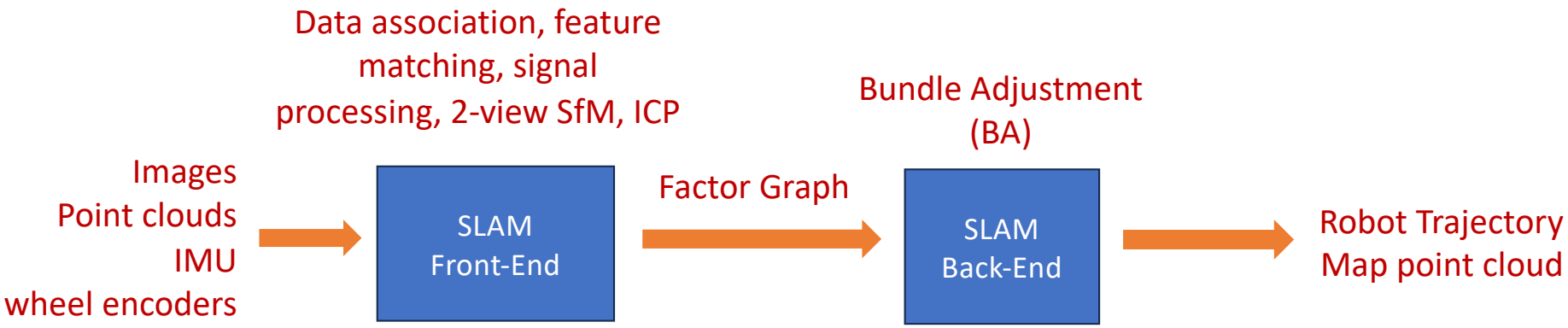
SLAM in ROS NAV2 toolbox

SLAM in Space for Asteroid Reconstruction



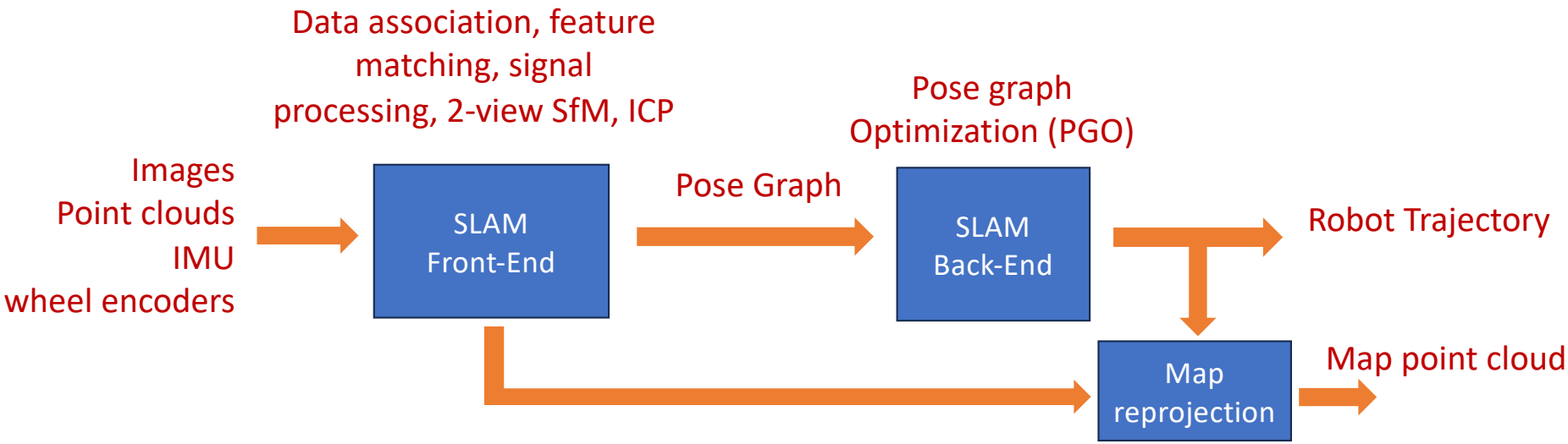
Factor Graphs: A Unifying Perspective for SLAM

(Frank Dellaert, 2012, 2017)



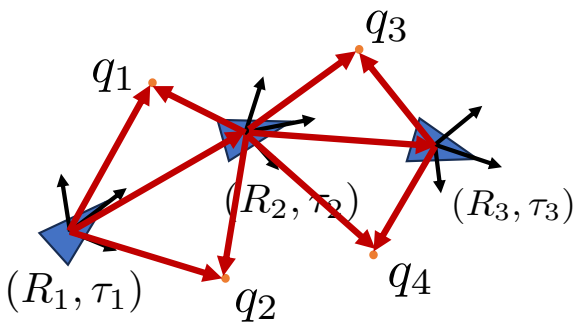
Bundle Adjustment SLAM - Produces both robot traj and map point cloud

Factor Graphs: A Unifying Perspective for SLAM



Pose Graph SLAM - Produces robot traj first, map obtained in post processing.

Factor Graph



(R_i, τ_i)
q_k ○ Unknown quantities

f_{ik} □ "Factors": Costs associated with measurements

Solve for

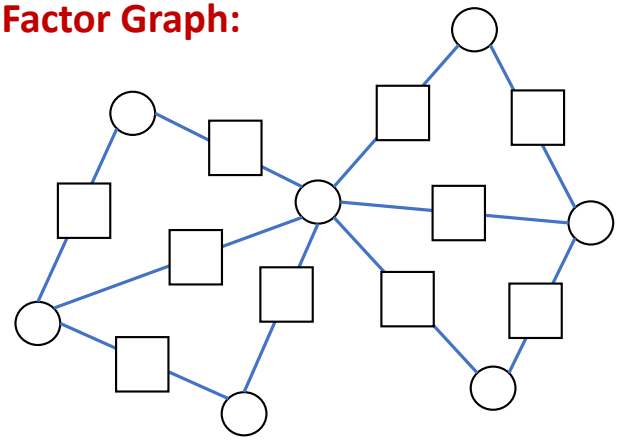
Robot trajectory: $((R_1, \tau_1), \ldots, (R_T, \tau_T))$

Map: $(q_1, \ldots, q_M)$

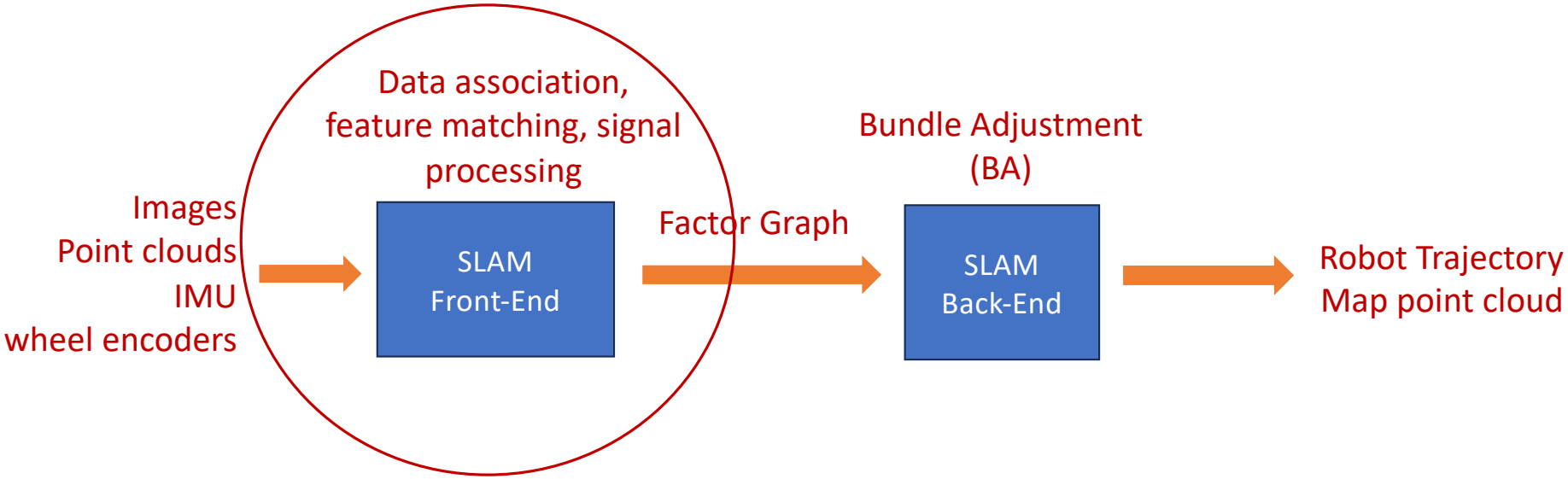
Given: Sensor Measurements

- RGB, Point Cloud, RGB-D
- IMU/Wheel encoder odometry

Factor Graph:

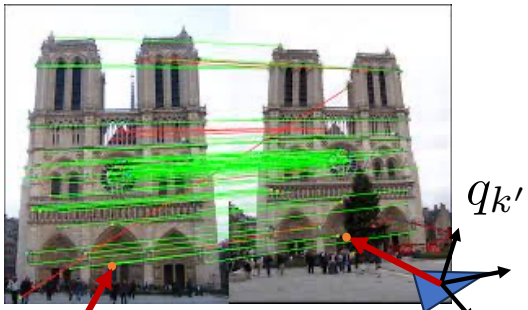


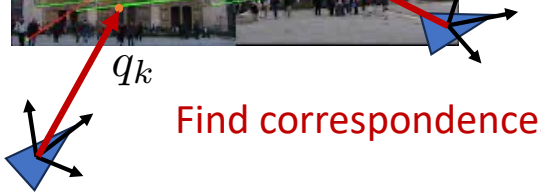
Factor Graphs: A Unifying Perspective for Modern SLAM



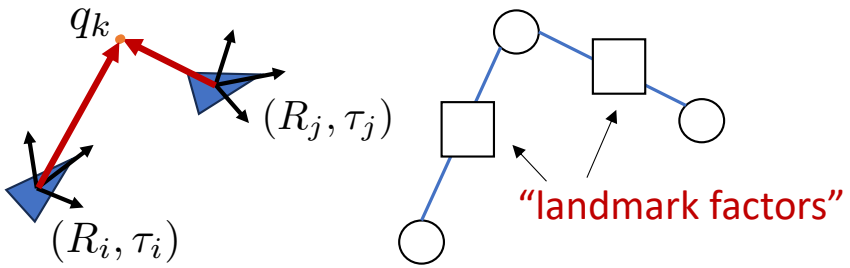
Front End: Building the factor graph

For RGB, RGB-D sensors:

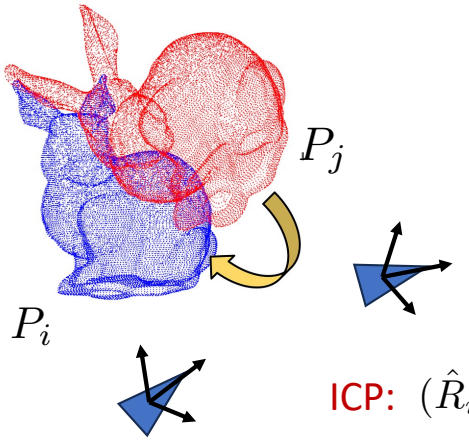




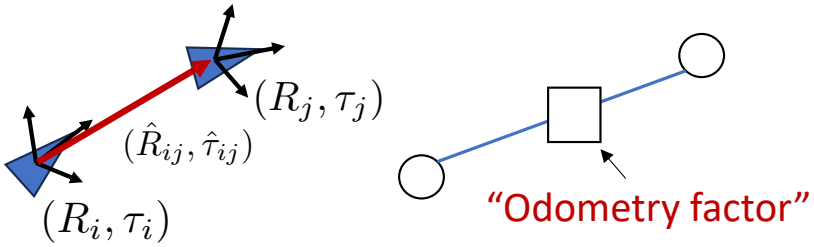
Find correspondences: $q_k = q_{k'}$



For Lidar:

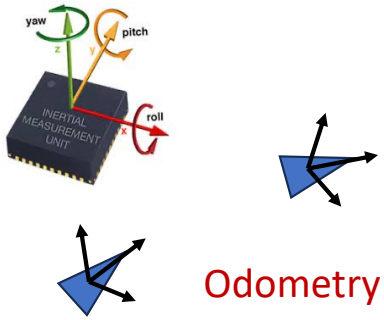


ICP: $(\hat{R}_{ij}, \hat{\tau}_{ij})$



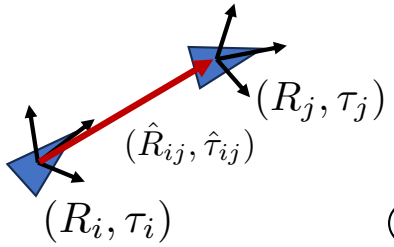
Front End: Building the factor graph

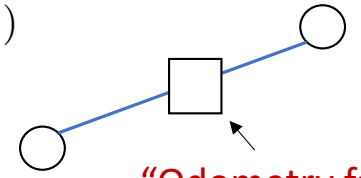
For IMU/encoder odometry:



Odometry (usually with EKF):

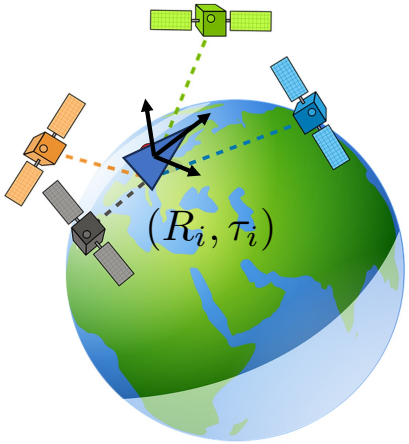
$(\hat{R}_{ij}, \hat{\tau}_{ij})$



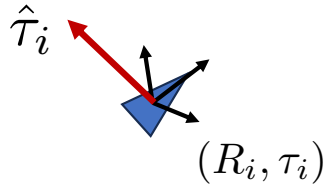


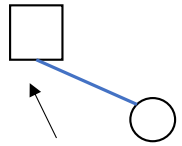
"Odometry factor"

GPS:



GPS: $\hat{\tau}_i$

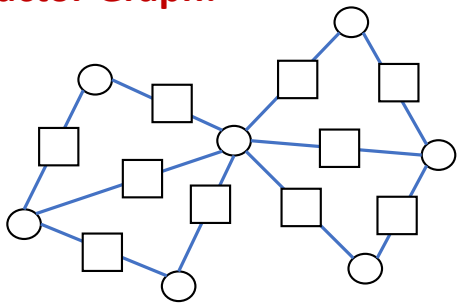


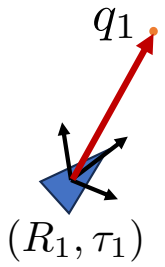


"GPS factor"
"Strapdown factor"

Factors: RGB-D measurements

Factor Graph:





RGB-D measurement:

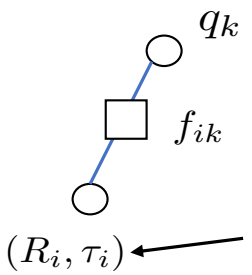
$$\hat{q}_{ik} = R_i^T(q_k - \tau_i) + \nu_{ik}^D$$

3D feature location in robot's frame (measured)

noise

$$\nu_{ik}^D \sim \mathcal{N}(0, \Sigma_{ik}^D)$$

RGB-D factor



3D feature location in global frame (unknown)

Robot pose (unkown)

RGB-D factor:

$$f_{ik} = \lambda_{ik}^D \| R_i \hat{q}_{ik} - (q_k - \tau_i) \|^2$$

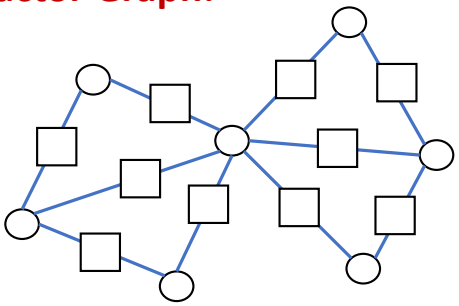
Weighting constant

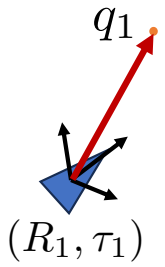
$$\lambda_{ik}^D = \det(\Sigma_{ik}^D)^{-1}$$

Interpretation: Cost of *inconsistency* with measurement.

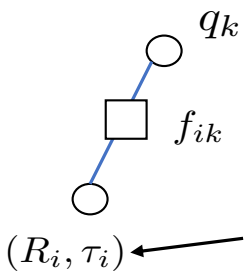
Factors: RGB measurements

Factor Graph:





RGB-D factor



3D feature location in global frame (unknown)

Robot pose (unkown)

Camera calibration matrix

noise

$$\nu_{ik}^V \sim \mathcal{N}(0, \Sigma_{ik}^V)$$

RGB measurement:

$$p_{ik}^h = K R_i^T (q_k - \tau_i) + \nu_{ik}$$

$$p_{ik}^h = (u_{ik}, v_{ik}, 1) z_{ik}^h$$

Homogeneous pixel coords

Measured pixel coords in image

Unknown pixel depth

RGB factor:

$$f_{ik} = \lambda_{ik}^V \| R_i K^{-1} p_{ik}^h - (q_k - \tau_i) \|^2$$

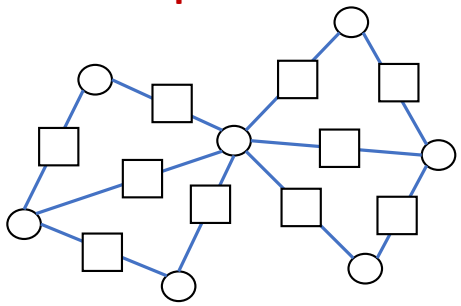
Weighting constant

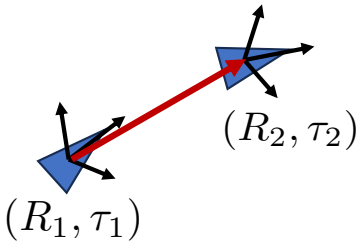
$$\lambda_{ik}^V = \det(\Sigma_{ik}^V)^{-1}$$

Interpretation: Cost of *inconsistency* with measurement.

Factors: Lidar and Odometry measurements

Factor Graph:



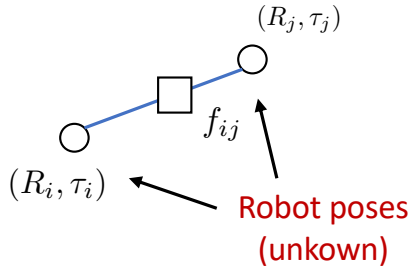


Odometry measurement: $(\hat{R}_{ij}, \hat{\tau}_{ij})$

$$\hat{R}_{ij} = R_i^T R_j + v_{ij}^R$$
$$\nu_{ij}^R \sim \mathcal{N}(0, \Sigma_{ij}^R)$$

$$\hat{\tau}_{ij} = R_i^T (\tau_j - \tau_i) + \nu_{ij}^\tau$$
$$\nu_{ij}^\tau \sim \mathcal{N}(0, \Sigma_{ij}^\tau)$$

Odometry factor



Odometry factor:

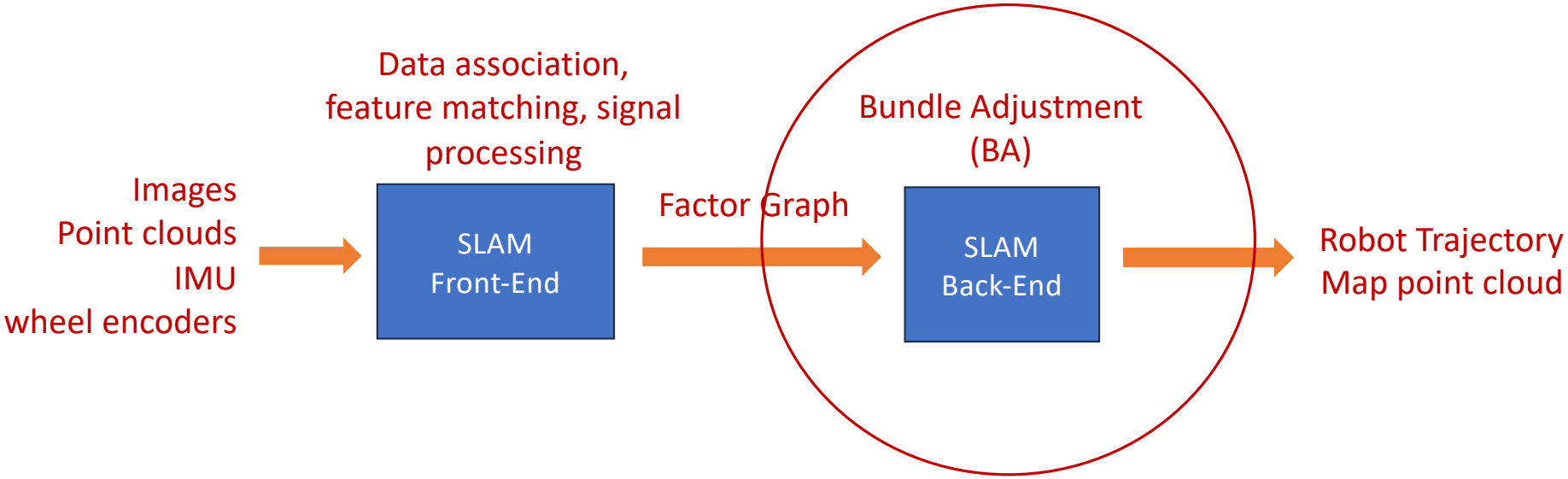
$$f_{ij} = \lambda_{ij}^R \| R_i \hat{R}_{ij} - R_j \|^2 + \lambda_{ij}^\tau \| R_i \hat{\tau}_{ij} - (\tau_j - \tau_i) \|^2$$

$$\lambda_{ij}^R = \det(\Sigma_{ij}^R)^{-1} \qquad \lambda_{ij}^\tau = \det(\Sigma_{ij}^\tau)^{-1}$$

Weighting constants

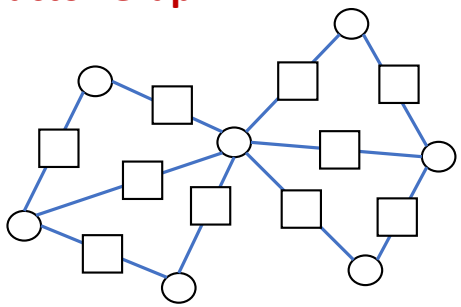
Interpretation: Cost of *inconsistency* with measurement.

Factor Graphs: Bundle Adjustment



Bundle Adjustment Optimization

Factor Graph:



Sum of all factors gives total cost:

$$J((R_i,\tau_i),q_k) = \sum_{(\hat{R}_{ij},\hat{\tau}_{ij})} (\lambda_{ij}^R \|R_i\hat{R}_{ij} - R_j\|^2 + \lambda_{ij}^\tau \|R_i\hat{\tau}_{ij} - (\tau_j - \tau_i)\|^2)$$
$$+ \sum_{\hat{q}_{ik}} \lambda_{ik}^D \|R_i\hat{q}_{ik} - (q_k - \tau_i)\|^2 + \sum_{(u_{ik},v_{ik})} \lambda_{ik}^V \|R_i K^{-1} p_{ik}^h - (q_k - \tau_i)\|^2$$

Optimization Problem (Bundle Adjustment) :

$$\min_{(R_i,\tau_i),q_k} J((R_i,\tau_i),q_k)$$
$$\text{subject to} \quad \left.\begin{array}{r} R_i^T R_i = I \\ \det(R_i) = 1 \end{array}\right\} \; SO(3)$$

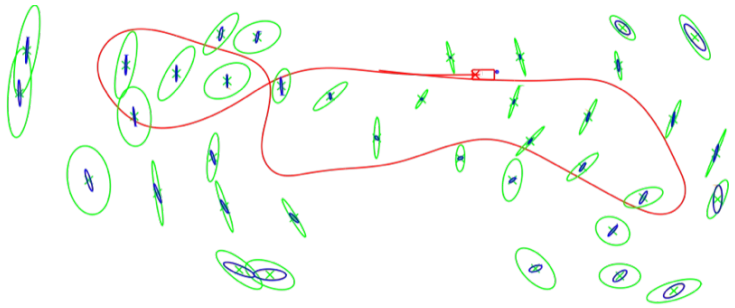
Interpretation: Find poses and feature locations that are *as consistent as possible* with all the measurements.

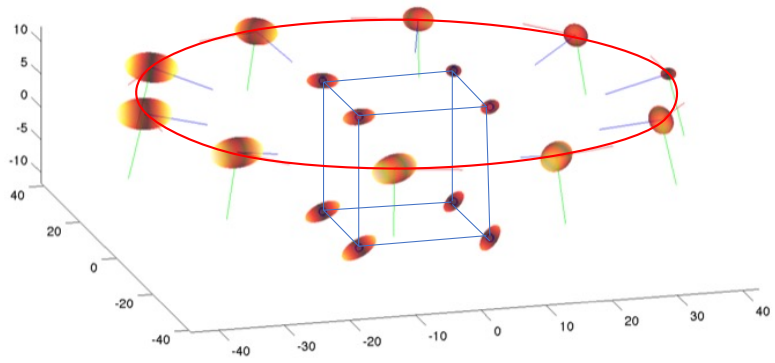
SLAM pose and map estimates:

$$\{(R_i^*,\tau_i^*),q_k^*\} = \arg\min_{(R_i,\tau_i),q_k} J((R_i,\tau_i),q_k)$$

GTSAM: The Standard Open-Source Factor Graph Optimizer

Frank Dellaert, and collaborators, GA Tech

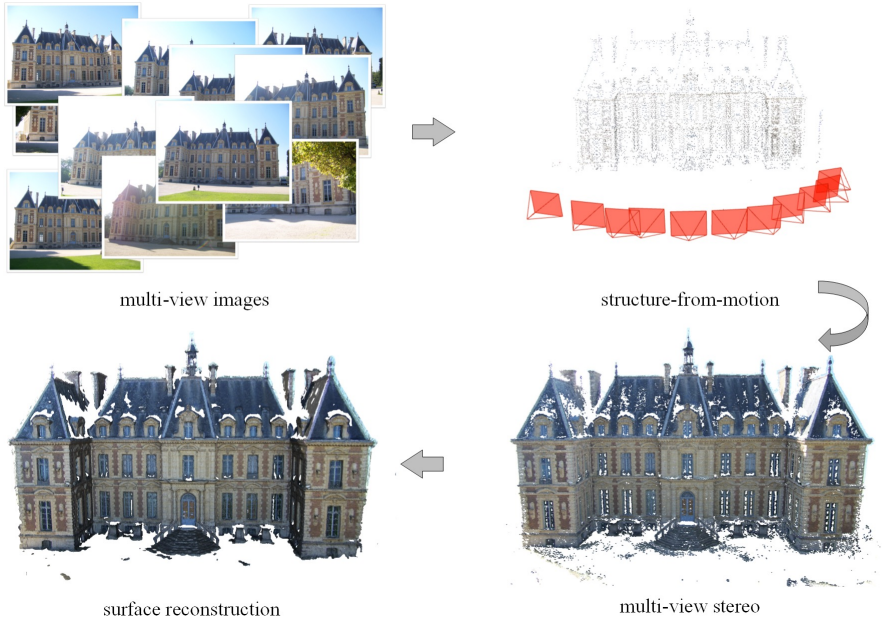




https://gtsam.org/

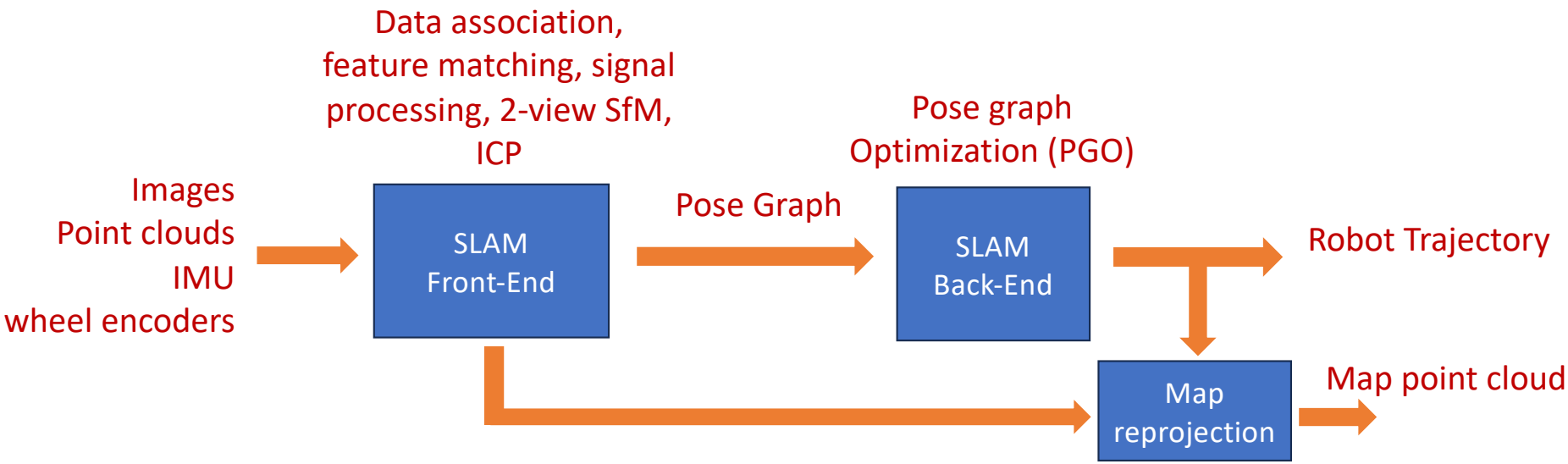
Colmap: The Standard Open-Source CV 3D Photogrammetry Package

Johannes Schönberger, and collaborators, ETH Zurich



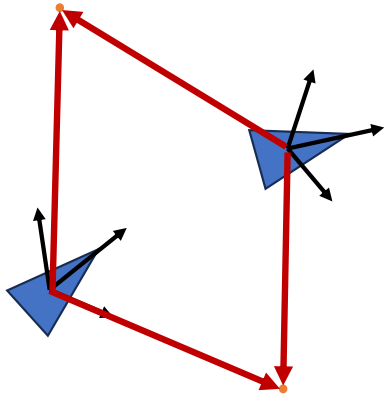
https://github.com/colmap/colmap

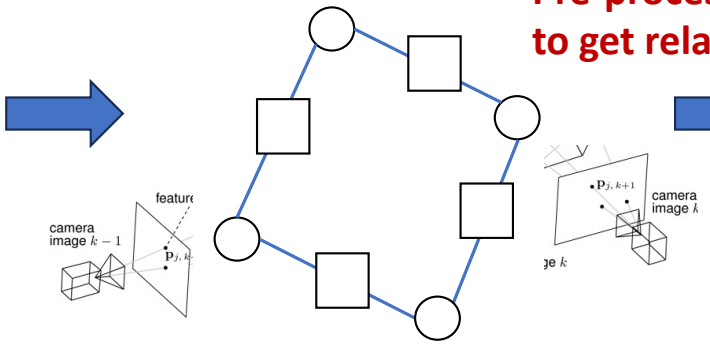
Pose Graph SLAM

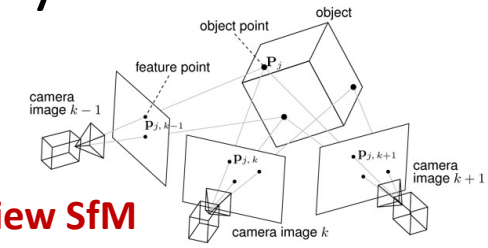


Pose Graph SLAM - Produces robot traj first, map obtained in post processing.

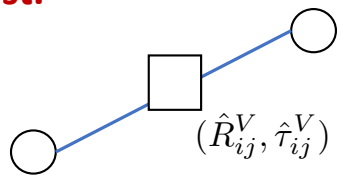
Pose Graph: A Factor Graph with Only Odometry Factors!



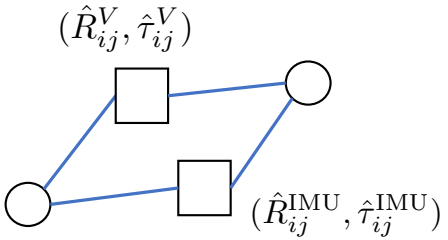




Pre-process with 2 view SfM to get relative pose est.



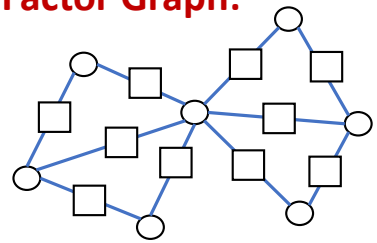
e.g. with lidar and IMU/wheel odometry



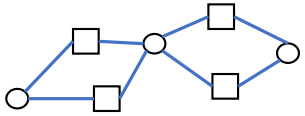
Interpretation: Do more processing in front end to get a simpler optimization in the back end.

Pose Graph Optimization

Factor Graph:



Pose Graph:



Much simpler!

Interpretation: Still looking for poses *most consistent* with all relative pose measurements. Typically **much faster** than full bundle adjustment, but can be **less accurate**.

Pose Graph Cost Function:

Odometry factors (from all sensors)

$$J((R_i,\tau_i)_{i=1}^N) = \sum_{(\hat{R}_{ij},\hat{\tau}_{ij})} (\lambda_{ij}^R \|R_i\hat{R}_{ij} - R_j\|^2 + \lambda_{ij}^\tau \|R_i\hat{\tau}_{ij} - (\tau_j - \tau_i)\|^2)$$

Pose Graph Optimization (PGO):

$$\min_{(R_i,\tau_i)_{i=1}^N} J((R_i,\tau_i)_{i=1}^N)$$

$$\text{subject to} \quad R_i^T R_i = I$$

$$\det(R_i) = 1$$

SLAM pose estimates only (no map!):

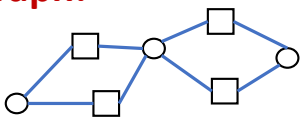
$$\{(R_i^*,\tau_i^*)_{i=1}^N\} = \arg\min_{(R_i,\tau_i)_{i=1}^N} J((R_i,\tau_i)_{i=1}^N)$$

Obtaining the Map from PGO Solution

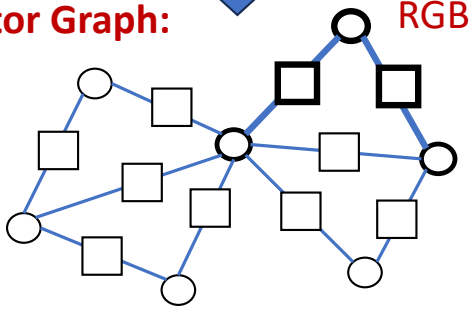
SLAM pose estimates:

$$\{(R_i^*, \tau_i^*)_{i=1}^N\} = \arg \min_{(R_i,\tau_i)_{i=1}^N} J((R_i, \tau_i)_{i=1}^N)$$

Pose Graph:

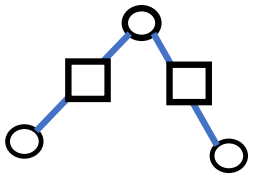


Factor Graph: RGB factors



Point in point cloud

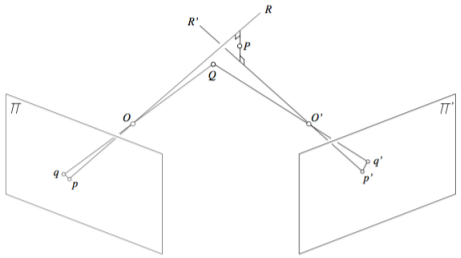
q_k^* unknown



(R_i^*, τ_i^*) (R_j^*, τ_j^*)

known

Stereo Depth: find location of point in 3D from two RGB images with known poses.

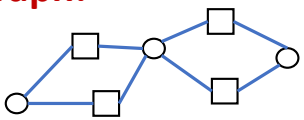


Obtaining the Map from PGO solution

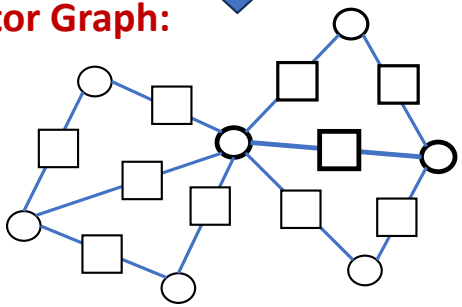
SLAM pose estimates:

$$\{(R_i^*, \tau_i^*)_{i=1}^N\} = \arg \min_{(R_i,\tau_i)_{i=1}^N} J((R_i, \tau_i)_{i=1}^N)$$

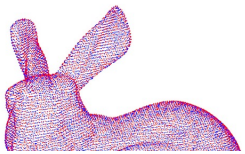
Pose Graph:

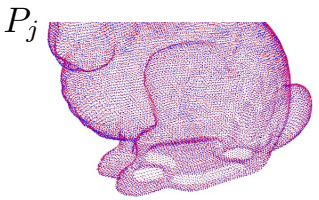


Factor Graph:



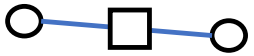
Lidar point clouds in local frames





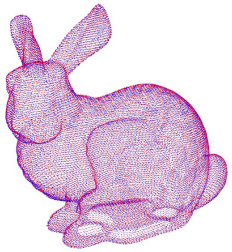
$P_i^* = R_i^* P_i + \tau_i^*$

$P_j^* = R_j^* P_j + \tau_j^*$



(R_i^*, τ_i^*) (R_j^*, τ_j^*)

Lidar point clouds in world frame



Fused point cloud in world frame

$\{P_i^* \cup P_j^*\}$

Next time: Optimization techniques (e.g., iSAM, colmap)

Optimization Problem:

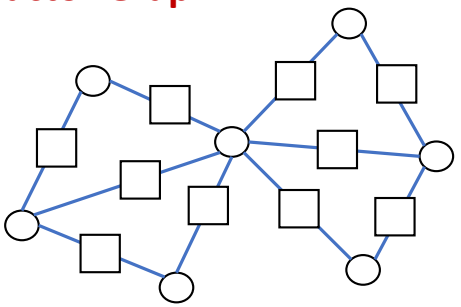
$$\min_{(R_i,\tau_i)_{i=1}^N,(q_k)_{k=1}^M} J((R_i,\tau_i)_{i=1}^N,(q_k)_{k=1}^M)$$

$$\text{subject to} \quad R_i^T R_i = I$$
$$\det(R_i) = 1$$

Large, non-convex, constrained problem.
In practice, many local minima.

Initialization: most optimization solvers require a rough initialization to converge to reasonable solution.

Factor Graph:



Initialization with min spanning tree: