Feature-Based Visual Odometry

Graduate Course INTR-6000P

Week 4 - Lecture 8

Changhao Chen

Assistant Professor

HKUST (GZ)

VO and SLAM

Visual Odometry (VO)

Goal: Estimate the ego-motion of a vehicle incrementally from camera images.

Focus: Local consistency. Output: A locally accurate trajectory.

Weakness: Drift. Small errors accumulate over time, making the estimated path diverge from the true path.

Visual SLAM (Simultaneous Localization and Mapping)

Goal: Build a consistent global map of the environment while simultaneously localizing within it.

Focus: Global consistency. Output: A globally consistent map and trajectory.

Solution to Drift: Loop Closing - detecting previously visited locations and correcting the entire map.

Feature-Based SLAM

Visual Pose Estimation in SLAM System

1. Initialization: Use 2D-2D (Epipolar) to bootstrap the map.

2. For every new frame:

3. Track features from previous frame or map.

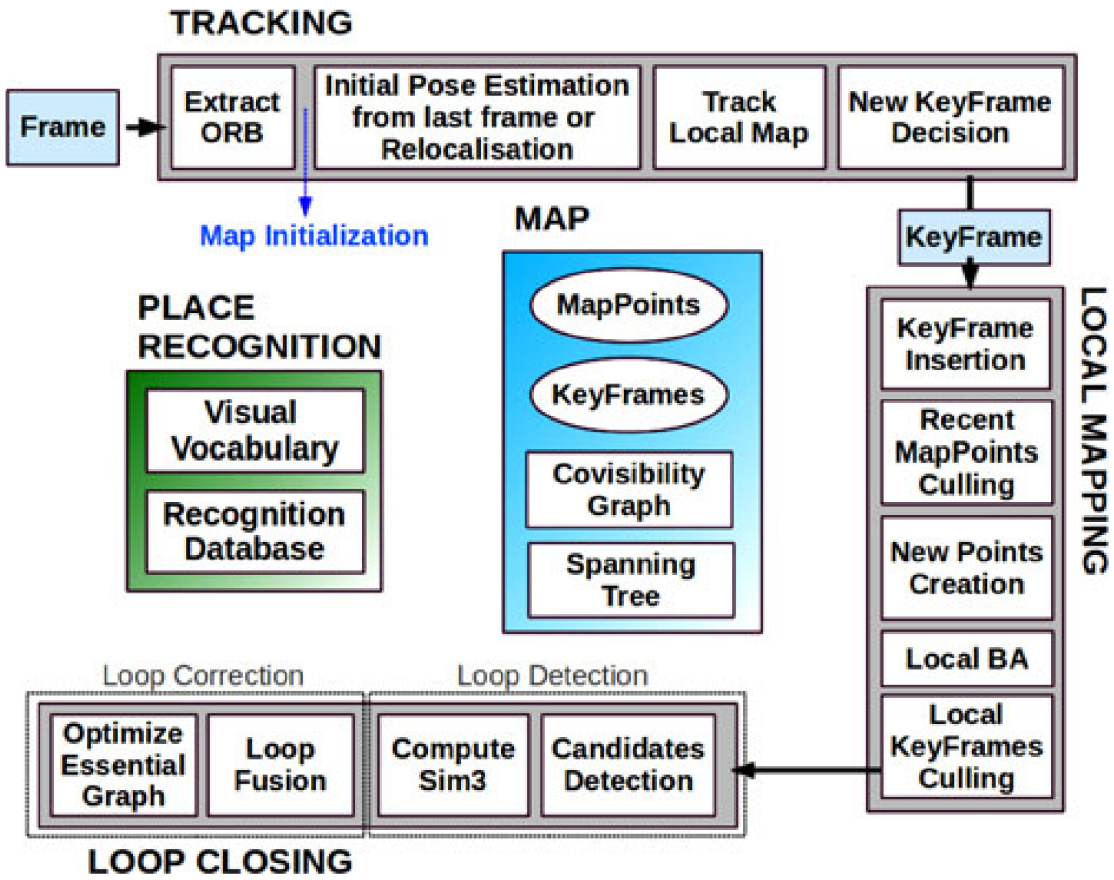
4. Solve PnP (using RANSAC + EPnP) to get a robust, initial pose estimate.

5. (Optional) Pose-only Bundle Adjustment: Refine the current camera pose by minimizing reprojection error over all inliers.

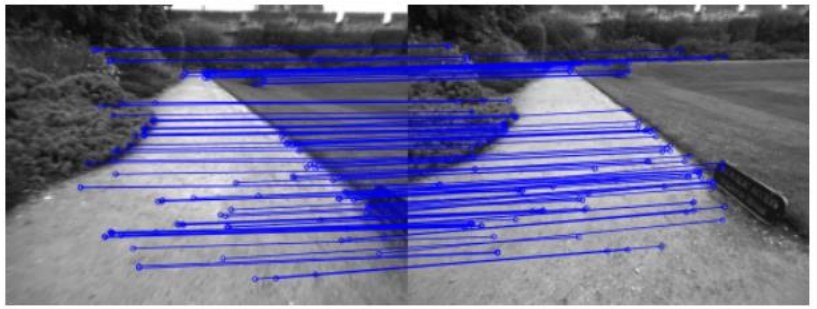
This provides the real-time, robust localization for the vehicle.

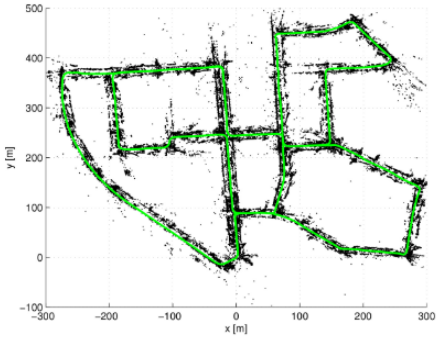
Feature-Based Visual SLAM

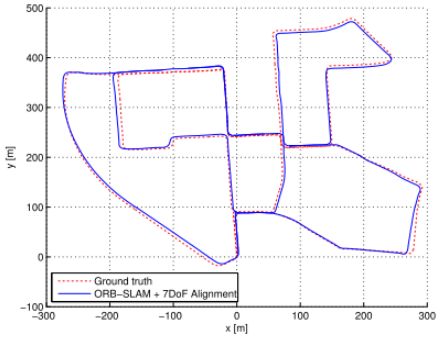
ORB-SLAM : A Versatile and Accurate Monocular SLAM System (2015)



1. Tracking: feature extraction, pose estimation from last frame or global relocalization
2. Local mapping: key frame insertion, update map, local bundle adjustment
3. Loop Closing: loop-closure detection, graph optimization



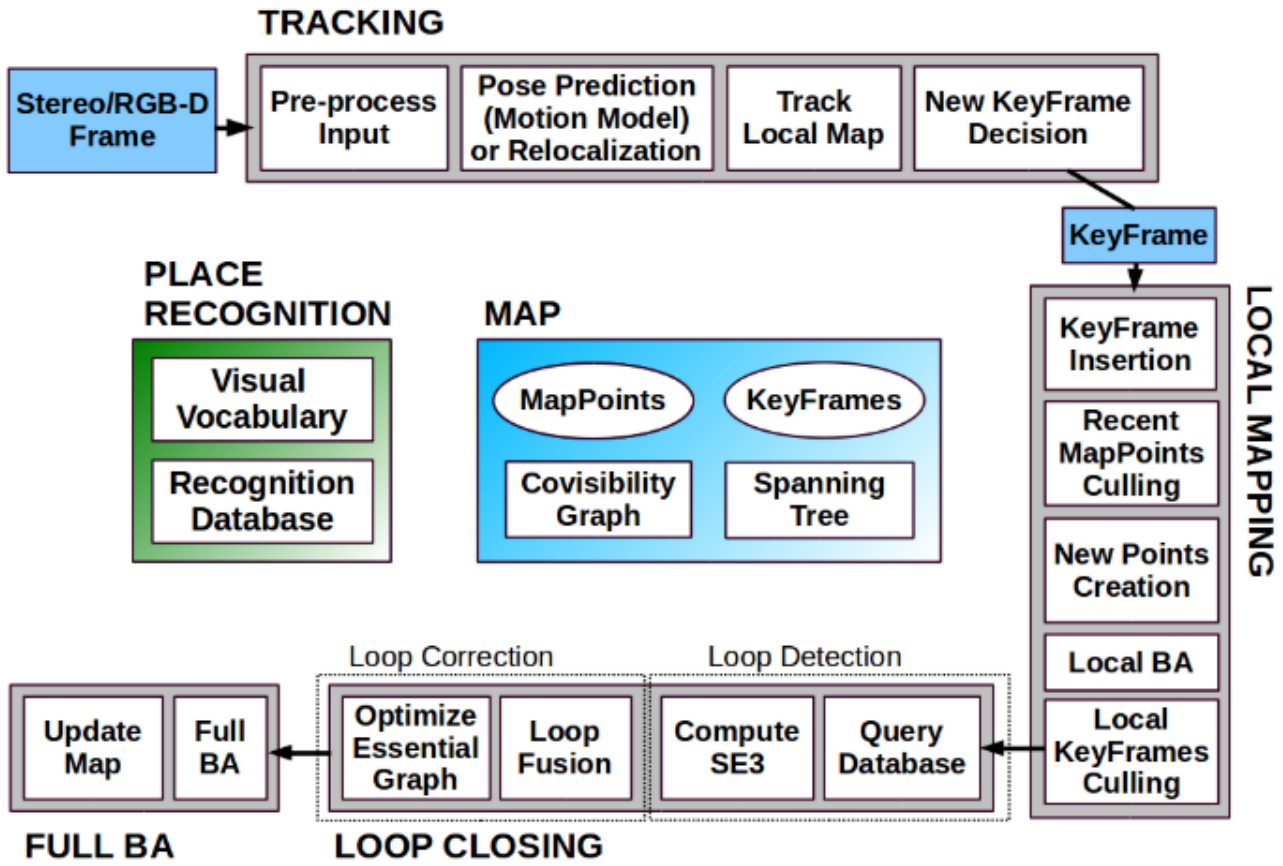


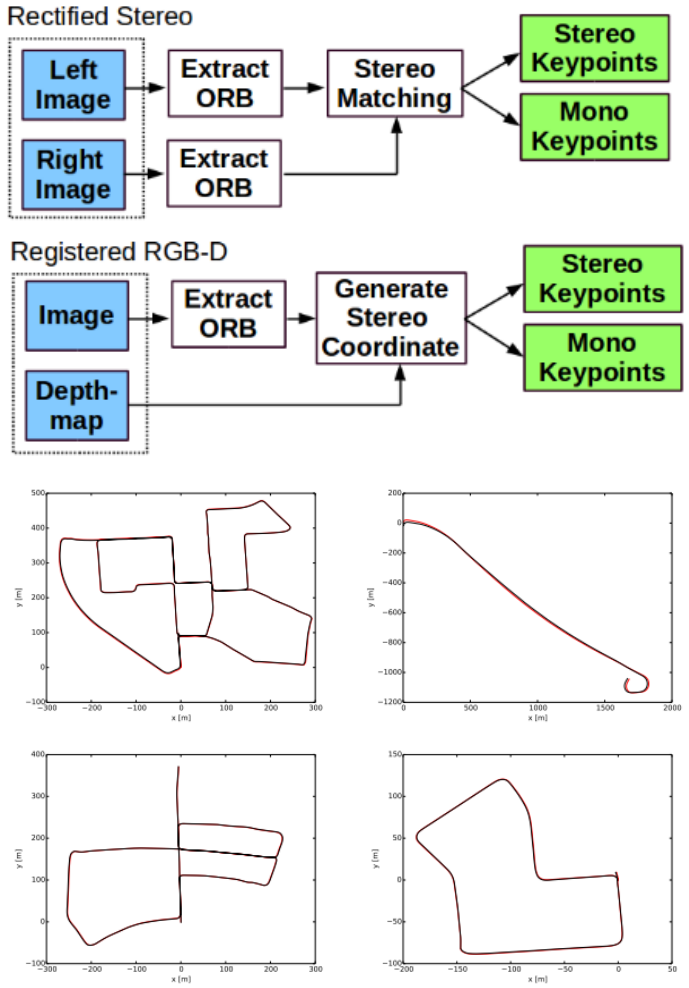


Feature-Based Visual SLAM

ORB-SLAM2 : an Open-Source SLAM System for Monocular , Stereo and RGB-D Cameras (2017)

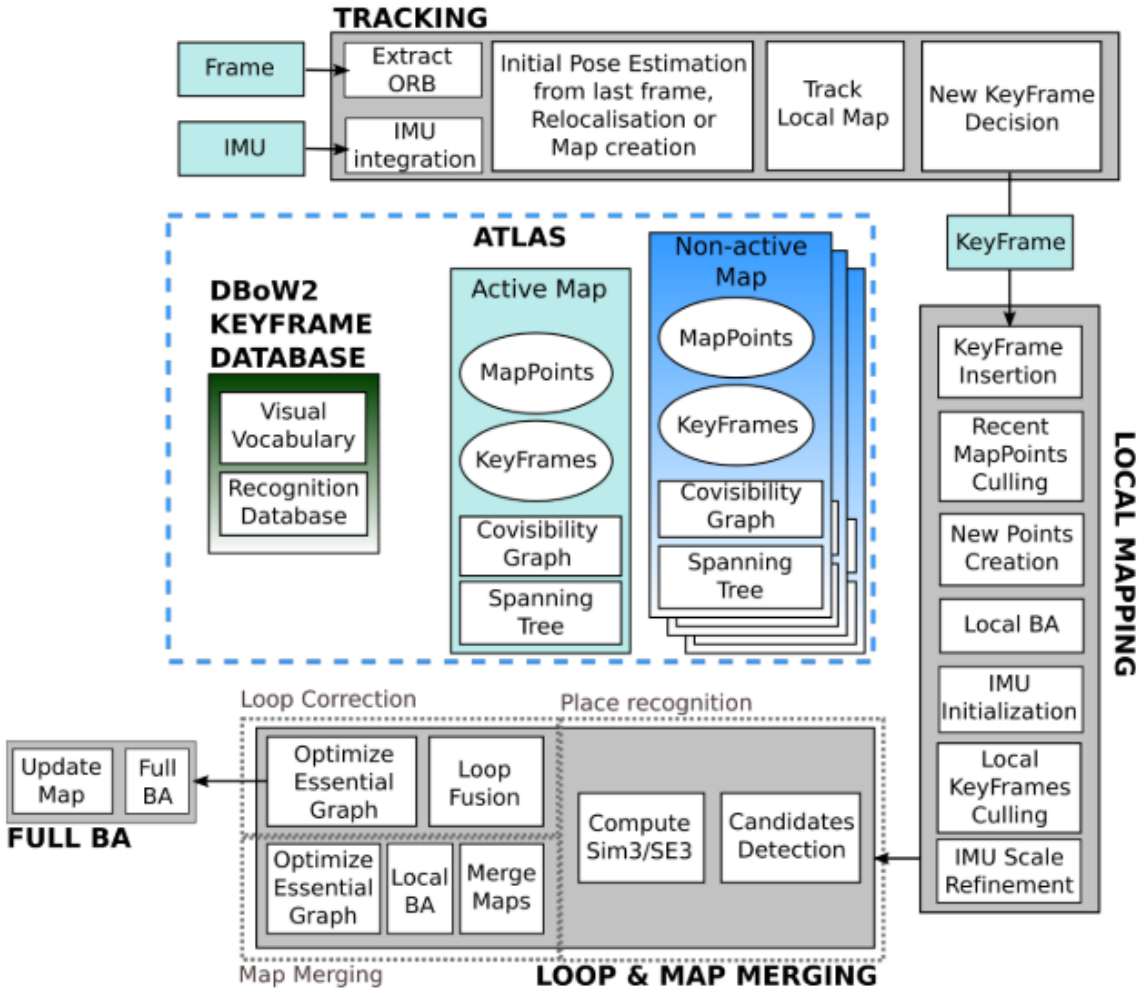
Support monocular, stereo and RGB-D cameras





Feature-Based Visual SLAM

ORB-SLAM3 : An Accurate Open-Source Library for Visual , Visual-Inertial and Multi-Map SLAM (2020)



1. **Short-term data association**: matching map elements obtained during the last few seconds, e.g. VO
2. **Mid-term data association**: matching map elements that are close to the camera whose accumulated drift is still small, e.g. local BA
3. **Long-term data association**: matching observations with elements in previously visited areas using a place recognition technique, e.g. pose-graph optimization

- Multi-map
- MAP Estimation

Code: https://github.com/UZ-SLAMLab/ORB_SLAM3

Feature-Based Visual SLAM

- Tracking (Localization): Estimate the camera pose for each new frame.
- Mapping (Mapping): Estimate the 3D structure of the environment (points, landmarks).
- Place Recognition (Loop Closing): Detect when the vehicle has returned to a previously visited area.
- Global Optimization (Graph Optimization): Correct the accumulated drift by adjusting the entire map and trajectory.

Component 1: Tracking - The Frontend

Goal: Estimate the camera pose for every new frame.
How it works:

1. Extract ORB features from the current frame.

2. Initialization: If no map exists, use 2D-2D epipolar geometry to bootstrap the first pose and map.

3. For subsequent frames:

- Motion Model: Predict the new pose from constant velocity assumption.

- Track Local Map: Search for matches between current frame and local map points.

- Pose Optimization: Use Motion-only Bundle Adjustment to refine the current camera pose by minimizing the reprojection error of the matched map points.

4. Keyframe Decision: Decide if the current frame is a keyframe (if tracking is weak, large change in viewpoint, etc.).

Component 1: Tracking - The Frontend

Problem: Processing every frame is computationally expensive and redundant.

Solution: Only process a selected subset of frames (Keyframes) in depth.

A Keyframe is chosen based on:

- Time passed since last keyframe.
- Large change in viewpoint (rotation/translation).
- Amount of new, unmatched features (tracking quality).

Keyframes store the full-resolution image, extracted features, and their pose.

Component 2: Local Mapping - The Middle-End

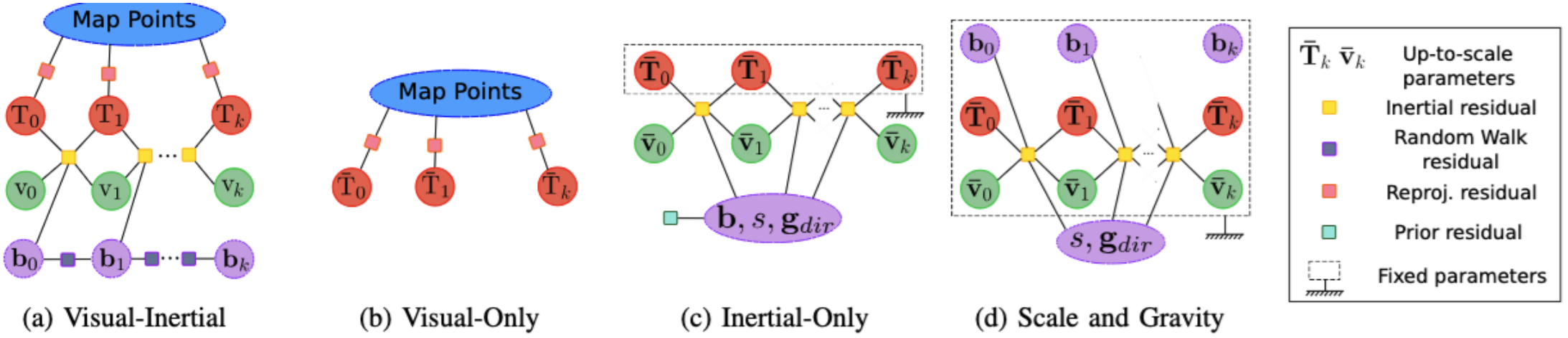
Goal: Refine the local map and create new 3D points.

Operates on new keyframes in a separate parallel thread.

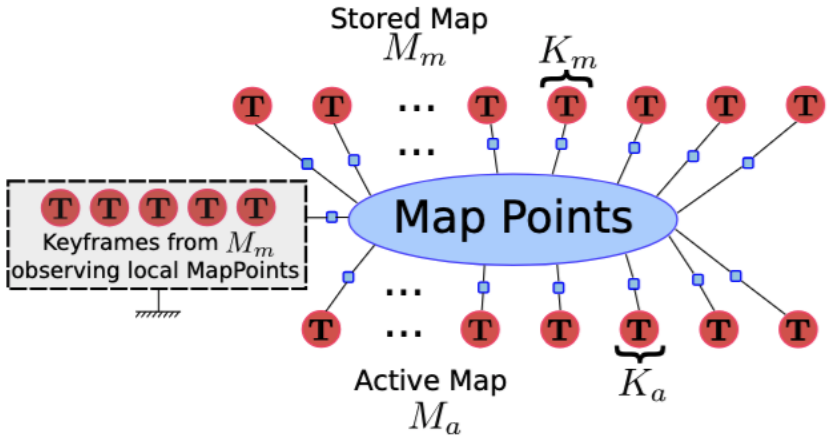
How it works:

1. Keyframe Insertion: Add the new keyframe to the map.
2. Recent Map Point Culling: Remove unstable map points that weren't found in many recent frames.
3. New Map Point Creation: Triangulate new 3D points from unmatched features between connected keyframes.
4. Local Bundle Adjustment: The core! Optimize the current keyframe, its covisibility keyframes, and all map points seen by them. This ensures local consistency.
5. Local Keyframe Culling: Remove redundant keyframes.

The Covisibility Graph



(a) Visual-Inertial (b) Visual-Only (c) Inertial-Only (d) Scale and Gravity



(a) Visual welding BA

A powerful data structure that defines the relationship between keyframes.

Two keyframes are connected if they share a sufficient number of common map points.

The local map for a keyframe is defined by its covisibility keyframes.

This makes optimization efficient and scalable.

Component 3: Loop Closing - The Backend

Goal: Detect large loops and correct the accumulated drift globally.

Runs in a separate parallel thread.

1. Loop Candidate Detection: Use a Bag-of-Words (BoW) place recognition module to find a previous keyframe that looks similar to the current one (but is not a neighbor).

2. Loop Validation: Compute a similarity transform between the current and loop keyframe using RANSAC and PnP to geometrically verify the match.

Loop Correction:

- Fuse Duplicate Map Points: Merge the two instances of the same physical landmark.

- Essential Graph Optimization: Perform a pose-graph optimization over a spanning tree of the covisibility graph to correct the drift. This efficiently aligns the two sides of the loop.

3. Full Bundle Adjustment (Optional): Run a full global BA in a separate thread to get the optimal solution (very slow, but can run in the background).

Component 4: Relocalization

What if tracking is lost? (e.g., due to a sudden motion, occlusion, or a bright light).

ORB-SLAM doesn't just give up.

It compares the current frame to all previous keyframes in the map using the Bag-of-Words model.

If a match is found, it computes the camera pose via PnP and continues tracking.

This makes the system incredibly robust.

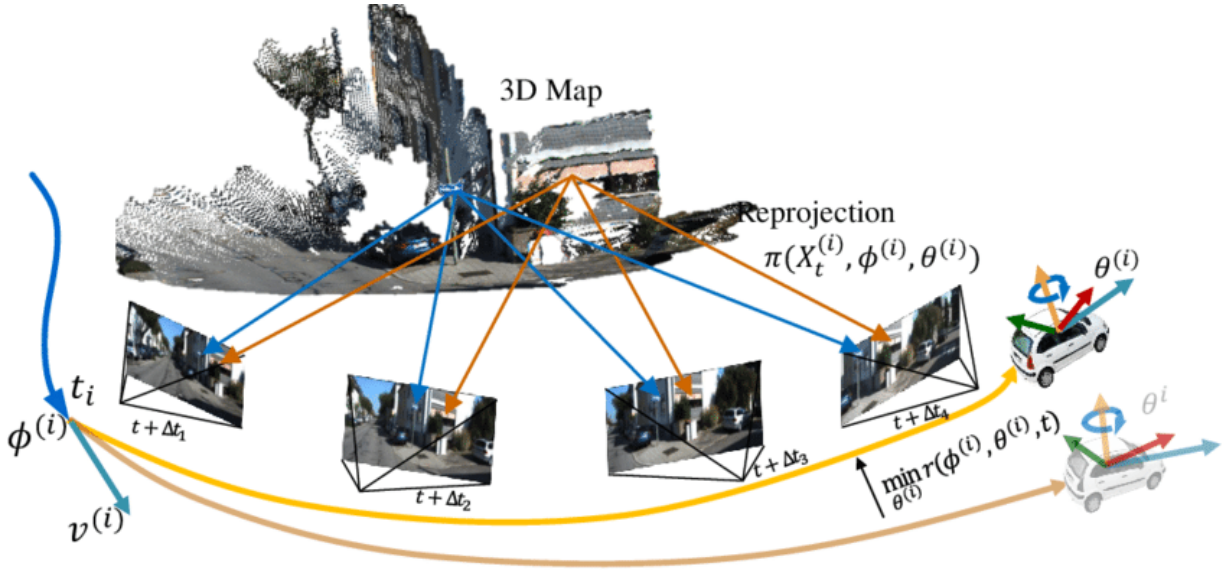
The Problem: Drift and Cumulative Error

Incremental Systems (like VO) have drift.

Small errors in each frame's pose estimation accumulate.

Errors in triangulating 3D points compound pose errors in subsequent frames.

PnP only optimizes one camera at a time. We need a global optimization to correct errors across all frames.



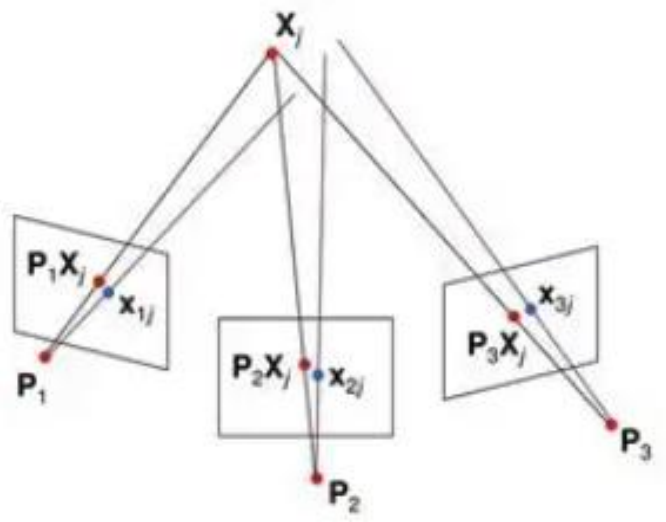
Bundle Adjustment

The Goal: Find the optimal configuration of all camera poses and all 3D points that is most consistent with all observed 2D image measurements.

"Bundle" refers to the "bundles" of light rays connecting camera centers to 3D points.

"Adjustment" refers to the iterative refinement of parameters.

It is a large-scale Non-Linear Least Squares optimization problem.



Bundle Adjustment

BA minimizes the total reprojection error across all cameras and all points.

$$\min_{\{\mathbf{R}_i,\mathbf{t}_i\},\{\mathbf{P}_j\}} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{v}_{ij} \, || \, \mathbf{p}_{ij} - \pi(\mathbf{K}[\mathbf{R}_i|\mathbf{t}_i]\mathbf{P}_j) \, ||_2^2$$

Where:

- m: number of cameras
- n: number of 3D points
- v_ij: Visibility function (1 if point j is seen in camera i, 0 otherwise)
- p_ij: Observed 2D image point of P_j in camera i
- π(...): Projection function

- **Parameters to Optimize:**
  - **All Camera Poses:** {R_i, t_i} for i = 1 ... m
  - **All 3D Points:** {P_j} for j = 1 ... n
- **Number of parameters can be huge:** e.g., 1000 frames & 10,000 points

→ ~33,000 parameters!

- **This is a massive, but very structured, optimization problem.**

Bundle Adjustment

We use iterative algorithms like Gauss-Newton or Levenberg-Marquardt.

1. Start with an initial guess (e.g., from PnP and triangulation).

2. Linearize the reprojection error function around the current guess (using Taylor expansion → requires the Jacobian matrix).

3. Solve a linear system ($H \delta x = -g$) for an update vector δx.

4. Update the parameters: $x_{new} = x_{old} + \delta x$.

5. Repeat until convergence.

Bundle Adjustment

The Key to Efficiency: Sparsity

The reprojection error for point P_j in camera i only depends on one point and one camera.

This means the giant Jacobian and Hessian (H) matrices are mostly zeros.

The Hessian has a very specific block-diagonal structure:

Camera-camera blocks are non-zero only if cameras see a common point.

Point-point blocks are diagonal.

Camera-point blocks create the structure.

Bundle Adjustment

Exploiting Sparsity: The Schur Complement

We can partition the linear system by grouping camera parameters (δc) and point parameters (δp):

$$\begin{bmatrix} \mathbf{H}_{cc} & \mathbf{H}_{cp} \\ \mathbf{H}_{pc} & \mathbf{H}_{pp} \end{bmatrix} \begin{bmatrix} \delta\mathbf{c} \\ \delta\mathbf{p} \end{bmatrix} = - \begin{bmatrix} \mathbf{g}_c \\ \mathbf{g}_p \end{bmatrix}$$

The Schur Complement Trick allows us to eliminate the point parameters δp and solve a much smaller system only for the cameras:

$$\left(\mathbf{H}_{cc} - \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{H}_{pc}\right)\delta\mathbf{c} = -\left(\mathbf{g}_c - \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{g}_p\right)$$

The matrix S = ... is called the reduced camera matrix.

It is much smaller and still sparse. Solving S δc = ... is efficient.

After solving for δc, we back-substitute to solve for δp.

Bundle Adjustment

The Role of BA in a VO or SLAM system

BA is computationally expensive. It is not run on every frame, but is used strategically:

Local BA: Optimizes the most recent k keyframes and all points they see. Runs frequently to correct short-term drift.

Global BA: Optimizes all keyframes and all points in the map. Runs after a loop closure to correct drift across the entire trajectory.

Pose-Graph Optimization: A related, lighter-weight method that can be used for very large-scale maps, optimizing only poses, not points.

Bundle Adjustment

BA Frameworks

Implementing efficient BA from scratch is complex. IV and robotics researchers use powerful libraries:

g2o (General Graph Optimization): A C++ framework for solving graph-based SLAM problems.

Ceres Solver (Google): A portable C++ library for modeling and solving large-scale optimization problems. Very popular for BA.

GTSAM (Georgia Tech Smoothing and Mapping): A BSD-licensed C++ library based on factor graphs.

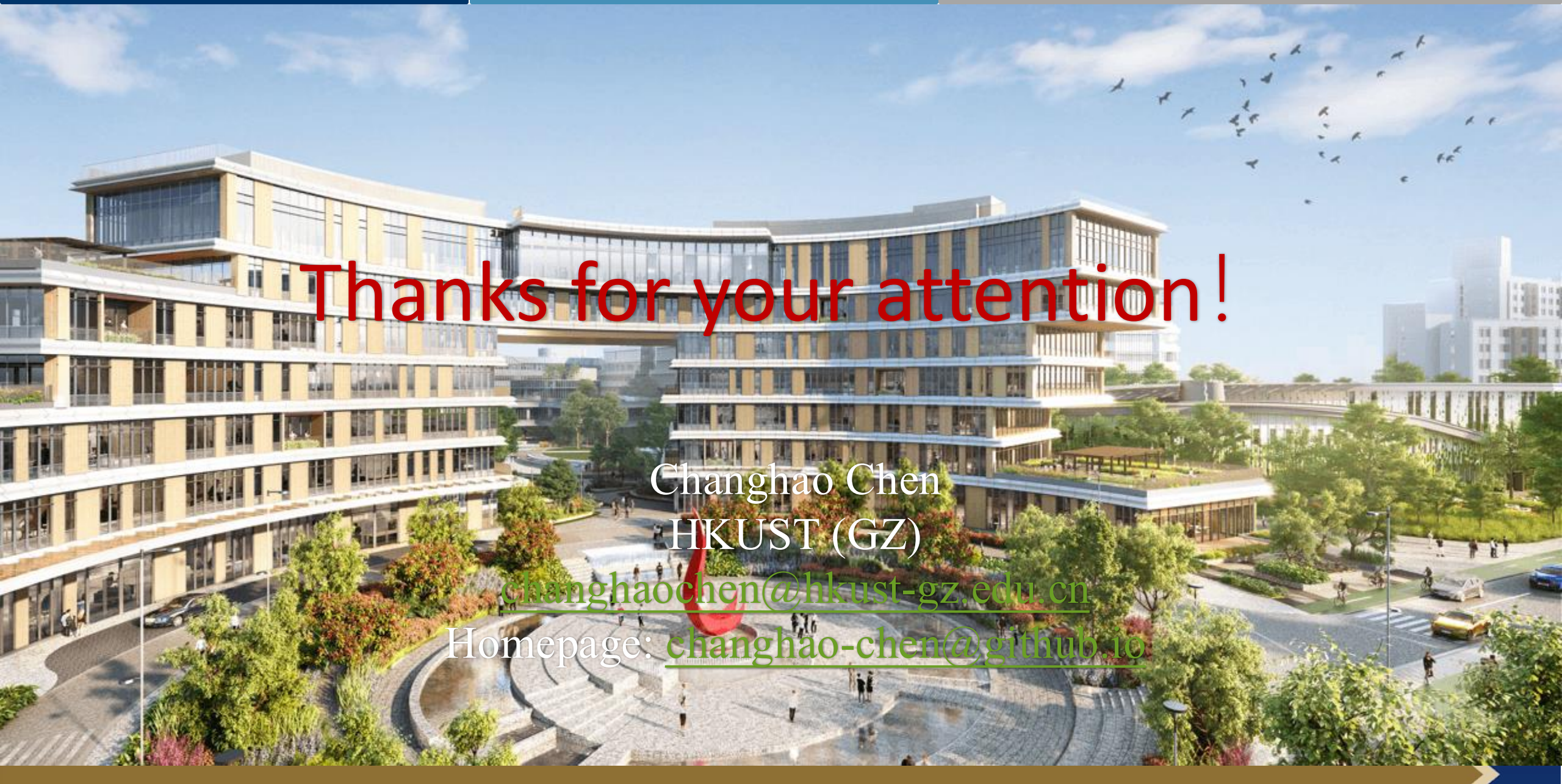
- Bundle Adjustment is the joint optimization of all camera poses and 3D points to minimize total reprojection error.
- It is the gold standard for achieving the most accurate and consistent map and trajectory, correcting cumulative drift.
- Solved via iterative non-linear least squares (Levenberg-Marquardt), made efficient by exploiting the sparse structure of the problem via the Schur Complement.
- Used as the backend in SLAM systems (Local BA and Global BA).

Summary

2D-2D (Epipolar): Bootstrapping motion and map from nothing.

3D-2D (PnP): Real-time camera tracking against a local map.

nD-nD (Bundle Adjustment): Global optimization for maximum accuracy and consistency.



Thanks for your attention!

Changhao Chen
HKUST (GZ)
changhaochen@hkust-gz.edu.cn
Homepage: changhao-chen@github.io