



Mapping

Graduate Course INTR-6000P

Week 7 - Lecture 13

Changhao Chen

Assistant Professor

HKUST (GZ)

Project Release

https://peaklab-hkust-gz.github.io/INTR_6000P/VNIV-fall-2025/project/

Overview

INTR 6000P - Introduction to Visual Navigation Final Project: Building and improving ORB-SLAM2 for Trajectory Construction

Welcome to the final project of INTR 6000P Introduction to Visual Navigation course! This project challenge you to develop your own SLAM system that can compute the camera trajectory in different situations.

 ****How to build ORB-SLAM 2 ****: https://github.com/raulmur/ORB_SLAM2?tab=readme-ov-file

This project asks you to build and improve ORB-SLAM2 to reconstruct trajectories from 7 provided datasets using a monocular camera, and to evaluate results against provided ground truth using EVO. You will submit fully reproducible code, evaluation outputs, and a technical report. This is an solo-only project.

Project Focus: A systematic enhancement of the ORB-SLAM2 framework for superior accuracy and robustness.

Base Algorithm: ORB-SLAM2 (mandatory codebase)

Key Areas: Feature matching, pose estimation, loop closing, and semantic integration.

Deliverables: Enhanced codebase, trajectory outputs, and a comprehensive technical report.

Project Release

https://peaklab-hkust-gz.github.io/INTR_6000P/VNIV-fall-2025/project/

Improvement Suggestions (not limited to):

1.Feature Detection & Matching

1. *Challenge:* ORB features are sensitive to illumination and viewpoint changes.
2. *Solution:* Integrate learned models like SuperPoint and LoFTR for superior matching reliability.

2.Pose Estimation

1. *Challenge:* Monocular scale ambiguity and tracking drift.
2. *Solution:* Fuse learned monocular depth and pose priors to stabilize tracking and recover metric scale.

3.Loop Closing & Back-End

1. *Challenge:* Robustness to long-term appearance changes and optimization efficiency.
2. *Solution:* Enhance place recognition and refine Pose Graph Optimization (PGO) and Bundle Adjustment (BA).

4.Semantic SLAM

1. *Challenge:* Dynamic objects corrupt the map and trajectory.
2. *Solution:* Leverage semantic information to identify and filter dynamic objects, building a persistent, static map.

Project Release

Evaluation Protocol

•Trajectory Difficulty & Completion:

- Easy Sequences: 3
- Medium Sequences: 2
- Hard Sequences: 2

Primary Metrics: Pose Accuracy (ATE, RPE) and overall robustness (tracking longevity).

Submission Deliverables

- 1.Code:** A well-documented, enhanced ORB-SLAM2 codebase.
- 2.Trajectory Outputs:** Estimated camera paths for all evaluation sequences.
- 3.Technical Report:** A detailed analysis of the implementation, results, and insights gained.

Grading Criteria

- Performance (50%):** Trajectory completion rate and pose accuracy.
- Technical Depth (20%):** Quality of the code and sophistication of the enhancements.
- Report & Understanding (30%):** Clarity and depth of the technical report, demonstrating a mastery of the underlying principles.

Deadline: **Submit all project materials by December 1, 2025.**

Presentation: **Present your findings in a 8-minute session on December 2, 2025.**

Recap: Kalman Filtering

2.3 Kalman Filter Equation

Prerequisites for optimal estimation: 1) Linear systems; 2) System noise and observation noise follow a normal distribution.

Prediction Process:

State-by-State Prediction:

$$\hat{X}_{k|k-1} = \Phi_{k,k-1} \hat{X}_{k-1}$$

One-step Variance Prediction:

$$P_{k|k-1} = \Phi_{k,k-1} P_{k-1} \Phi_{k,k-1}^T + \Gamma_{k-1} Q_{k-1} \Gamma_{k-1}^T$$

Mean Squared Error Matrix of the Previous Moment Optimal Estimate \hat{X}_{k-1}

Update Process:

Filter gain coefficient: $K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}$

Optimum estimate: $\hat{X}_k = \hat{X}_{k|k-1} + K_k (Z_k - H_k \hat{X}_{k|k-1})$

One-step prediction

New Information (One-Step Forecast Error of Observations)

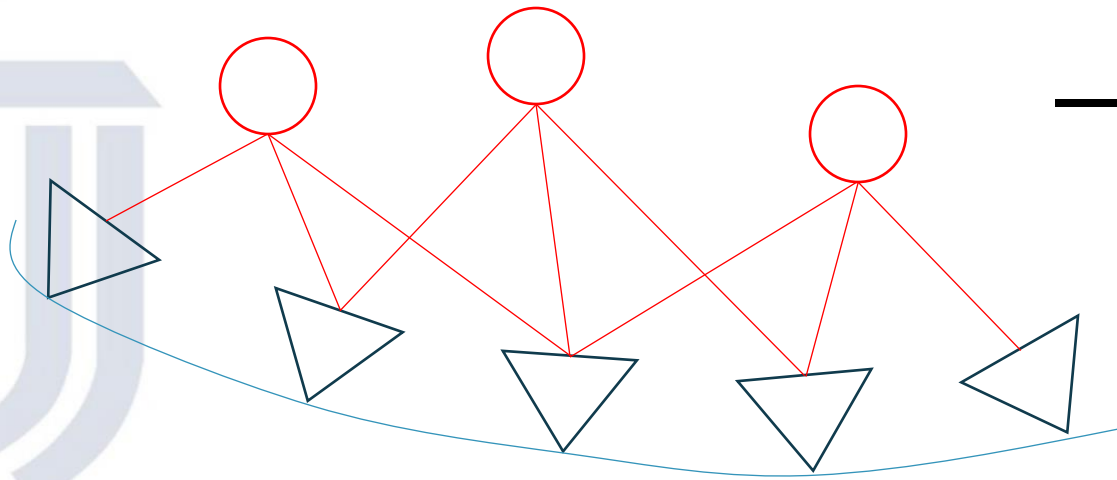
Optimal Estimator Variance Matrix:

$$P_k = (I - K_k H_k) P_{k|k-1} (I - K_k H_k)^T + K_k R_k K_k^T$$

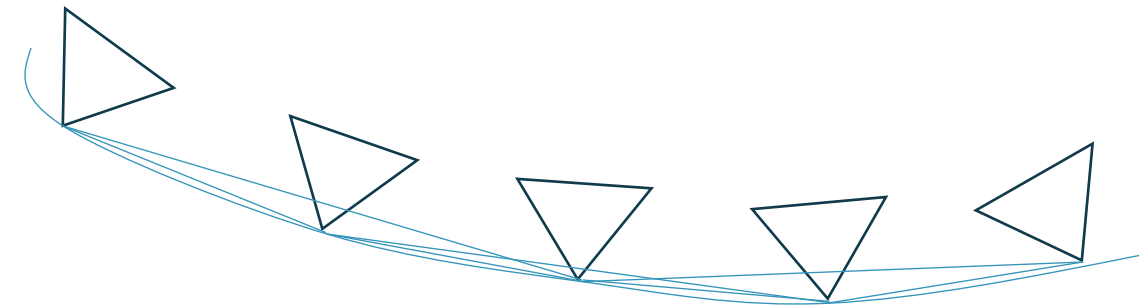
Recap: Problem with Full Bundle Adjustment

Bundle Adjustment (BA)

- Optimizes both camera poses and 3D feature point locations for high precision.
- However, in large-scale scenes, the huge number of feature points makes computation extremely heavy and inefficient.
- To address this, we introduce a simplified variant — **Pose Graph Optimization (PGO)**.

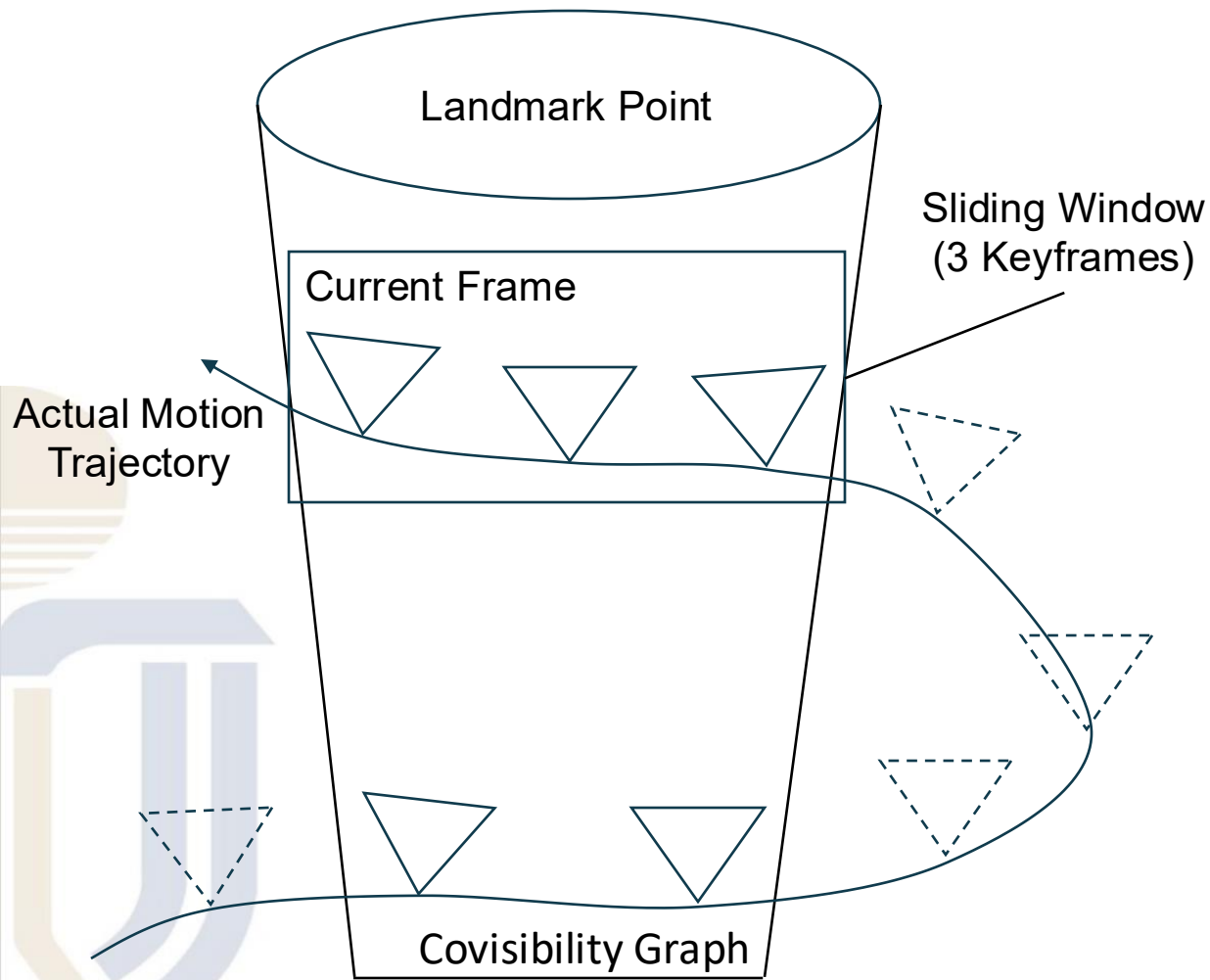


Bundle Adjustment



Pose Graph

Recap: Idea of Sliding Window Optimization



Schematic Diagram of Sliding Windows and Covisibility Graph

Sliding Window - A method that optimizes only recent keyframes within a fixed-size window to maintain accuracy while keeping computation efficient.

- Maintain only a **fixed-size local window** of recent keyframes and landmarks.
- Perform optimization only within this window.
- When a new keyframe arrives, **add the new keyframe** and **remove the oldest one**.
- This strategy balances accuracy and efficiency.

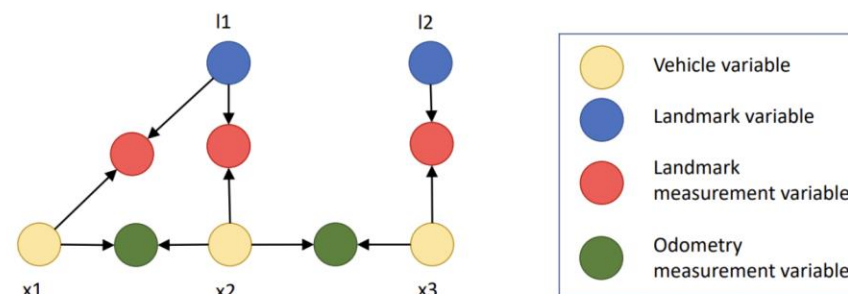
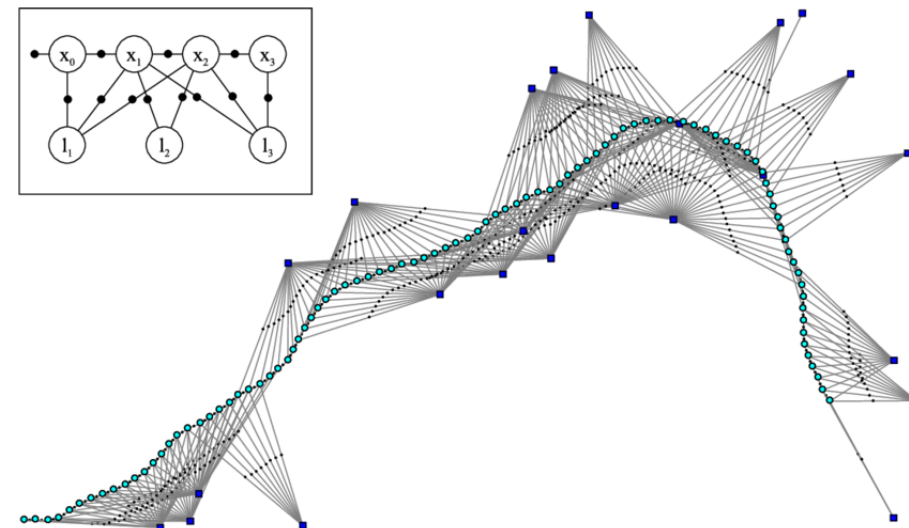
Recap: Concept of Pose Graph Optimization

Pose Graph – Simplifying the Optimization Problem

- In full Bundle Adjustment (BA), both camera poses and feature points are optimized.
→ Most computational cost comes from feature points.
- After multiple observations, feature points become stable; we only need accurate **camera poses**.
- **Idea:**
 - Represent each keyframe as a **node** (pose).
 - Represent spatial constraints (e.g., relative motion, loop closures) as **edges**.
 - Optimize only these poses instead of 3D landmarks
→ **Pose Graph Optimization (PGO)**.

Advantages:

- Reduces computation by eliminating landmarks.
- Suitable for large-scale SLAM or loop closure refinement.



Recap: Comparison of Optimization Libraries

Library	Core Concept	Strengths	Limitations	Used In
g2o	Sparse graph optimization (Gauss-Newton, LM, Dogleg)	Efficient, extensible, widely used	Slower on large 3D graphs	ORB-SLAM, SVO
Ceres Solver	Nonlinear least squares framework	Flexible, user-friendly, supports custom residuals	Needs good initialization, may get stuck in local minima	OKVIS, VINS-Mono
GTSAM	Factor graph + iSAM incremental smoothing	Incremental updates, supports multiple sensors	Higher memory use, slower in large static problems	SVO-GTSAM, academic & industry
SE-Sync	Convex semidefinite relaxation (SE(n))	Globally optimal, fast in high noise	Requires specific problem form, complex implementation	Research / high-accuracy robotics

The Mapping Problem

Input: sensor data (images, LiDAR, IMU)

Output: spatial representation of the environment

Challenges: dynamic objects, large-scale drift, scale ambiguity, real-time constraints

$$M=f(Z,X)$$

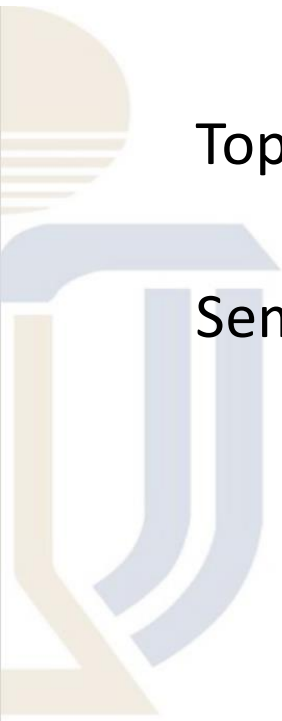
where Z are observations and X are poses.

2D vs. 3D Mapping

2D maps: for ground robots, indoor navigation

3D maps: essential for autonomous driving and drones

Types of Maps



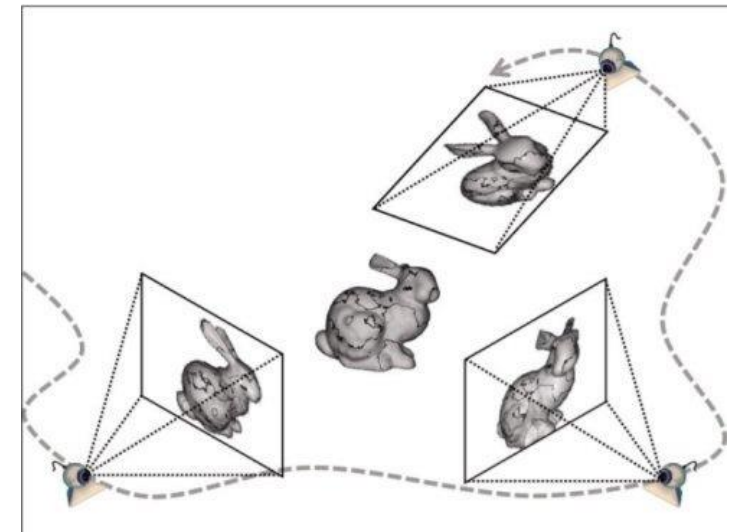
Map Type	Description	Examples
Metric	Precise geometric representation	Point cloud, voxel grid
Topological	Connectivity between places	Graph-based maps
Semantic	Adds meaning to structures	Road, building, pedestrian

Structure-from-Motion

Structure from Motion (SfM) reconstructs the 3D structure of a scene from a sequence of 2D images captured over time or from different viewpoints, without requiring prior camera calibration.

By establishing geometric correspondences between image points, SfM jointly estimates camera parameters and 3D scene geometry.

Corresponding points can be detected using feature descriptors such as SIFT or SURF, and tracked across frames using optical flow methods such as the Lucas–Kanade algorithm.

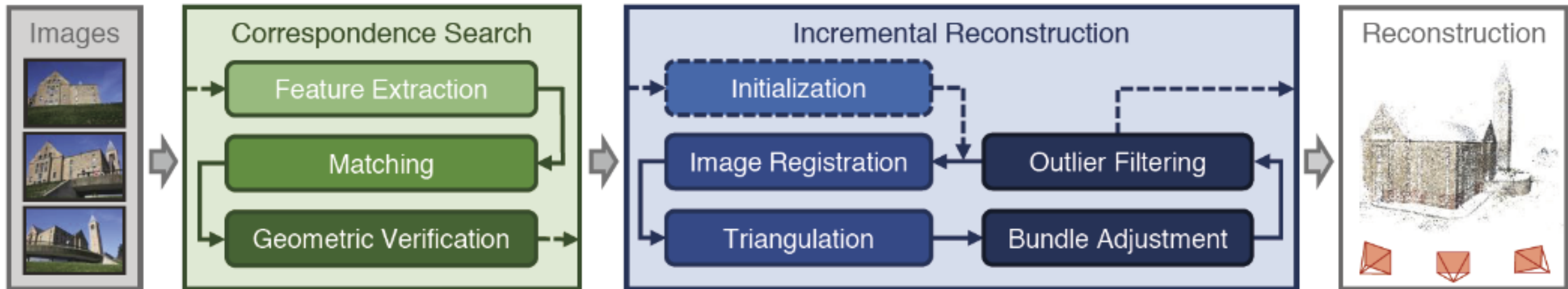


Structure-from-Motion

Colmap: Structure-from-Motion Revisited (2016)

SfM reconstructs 3D scene structure and camera poses from a set of 2D images.
The process can be divided into three main stages:

1. Correspondence Search
2. Feature Extraction & Matching
3. Incremental Reconstruction



Structure-from-Motion

Stage 1: Correspondence Search

- Identify overlapping regions between input images.
- Detect projections of the same 3D points in overlapping images.
- Output: a set of geometrically verified image pairs and their corresponding point projections.

Stage 2: Feature Extraction and Matching

1.Feature Extraction: Detect distinctive points in each image (e.g., SIFT, SURF).

2.Matching: Match features between image pairs.

3.Geometric Verification:

1. If a valid transformation maps enough feature points between two images, the image pair is considered geometrically verified.

• **Output:** A scene graph with images as nodes and verified image pairs as edges.

Structure-from-Motion

Stage 3: Incremental Reconstruction

Step 1 — Initialization:

- Select a pair or a small set of overlapping images with dense camera coverage.
- Ensures robust and accurate reconstruction.

Step 2 — Image Registration:

- New images are registered to the current 3D model using 2D–3D correspondences.
- Solve the PnP (Perspective-n-Point) problem to estimate camera pose.

Step 3 — Triangulation:

- New 3D points are triangulated from multiple registered views.
- Expands scene coverage incrementally.

Step 4 — Symbiotic Relationship:

- Image registration requires existing 3D structure.
- Triangulation requires registered images.



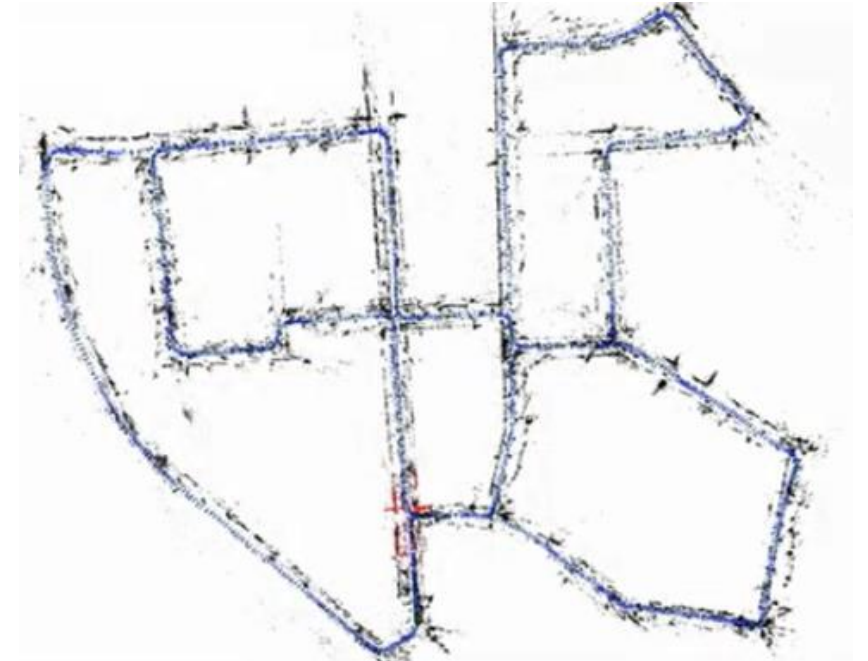
Mapping in Visual SLAM

Visual SLAM jointly estimates poses and maps

Bundle Adjustment (BA) refines 3D structure and camera trajectory

•Key components:

- Feature extraction
- Keyframe selection
- Map optimization (BA or Pose Graph Optimization)



Local vs. Global Mapping

Local mapping: short-term map for visual odometry

Global mapping: accumulated and optimized over time

Sparse vs. Dense Mapping

Sparse: Feature points or landmarks (e.g., ORB-SLAM)

Dense: Surface-level or volumetric reconstruction (e.g., DTAM, ElasticFusion)

Trade-offs: computation vs. detail, robustness vs. accuracy



Discussion:

How to enhance sparse mapping as dense mapping?

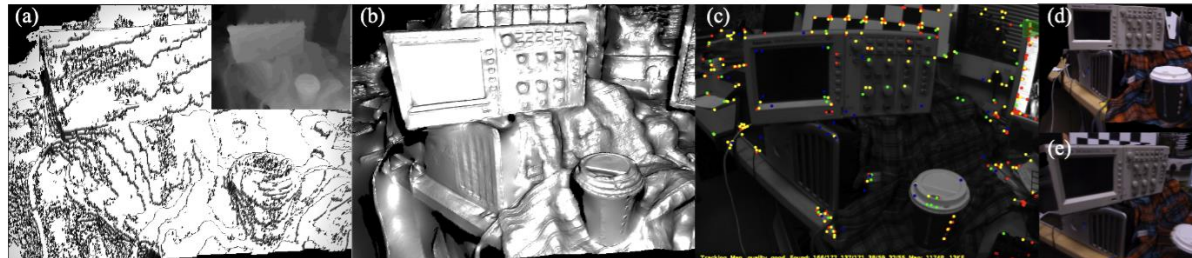
Dense Mapping

1) Direct Methods

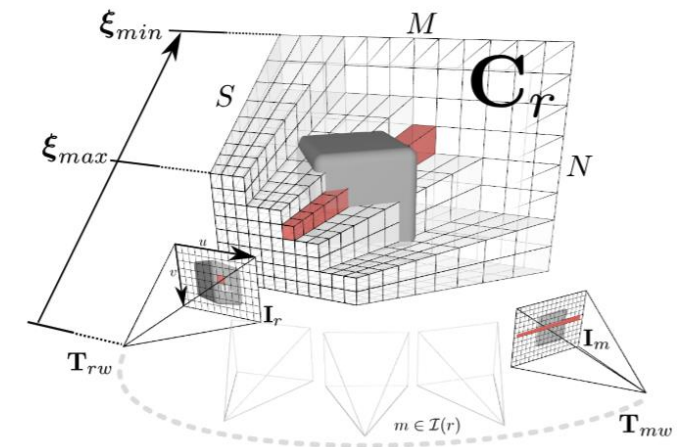
- **Direct methods** estimate camera motion and dense 3D structure directly from **pixel intensities**, without explicit feature extraction.
- Exploit **photometric consistency** between frames — assume brightness constancy across views.
- Produce **dense depth maps** by minimizing photometric error over all pixels.

DTAM (Dense Tracking and Mapping)

- Builds a dense depth map by optimizing photometric error across many keyframes.
- Simultaneously tracks camera pose and fuses dense geometry.
- Provides high-quality dense reconstructions but is computationally expensive.



Source: DTAM: Dense Tracking and Mapping in Real-Time

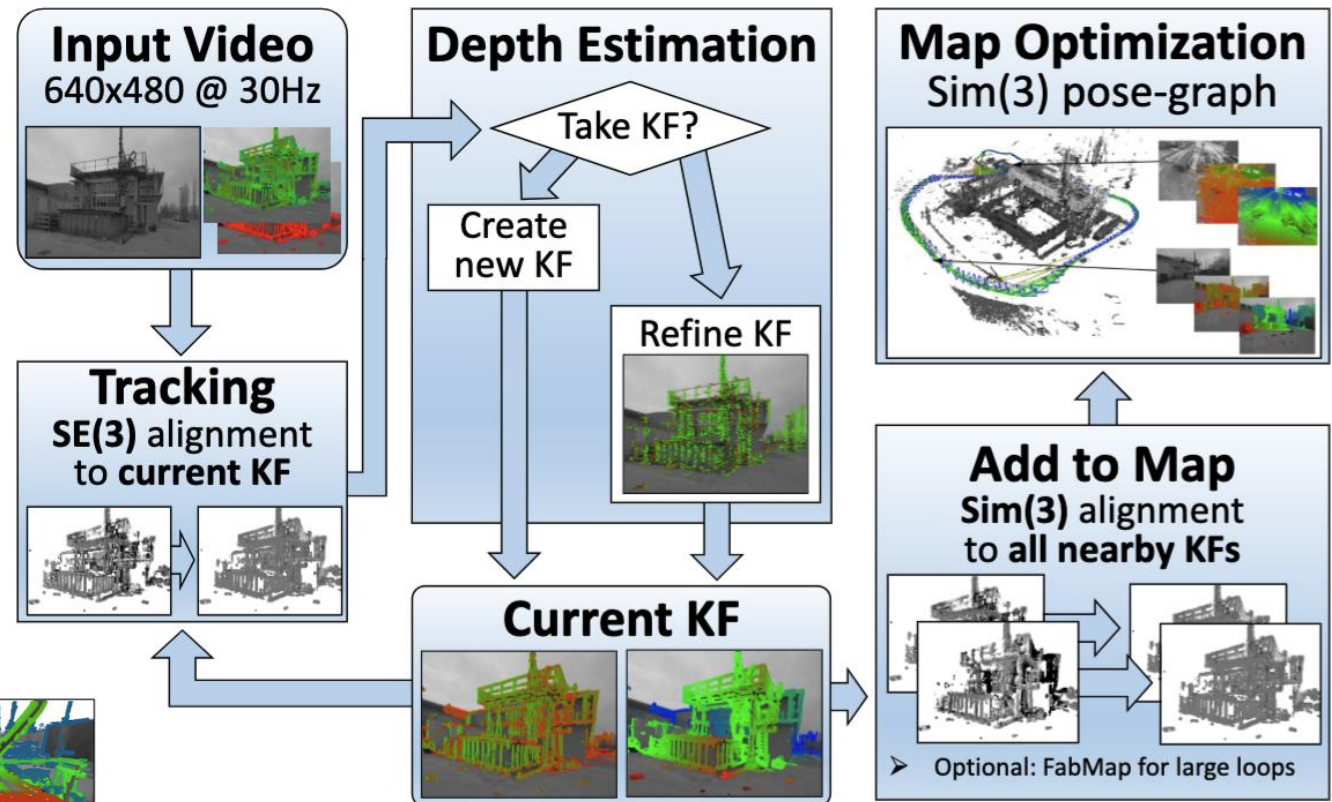
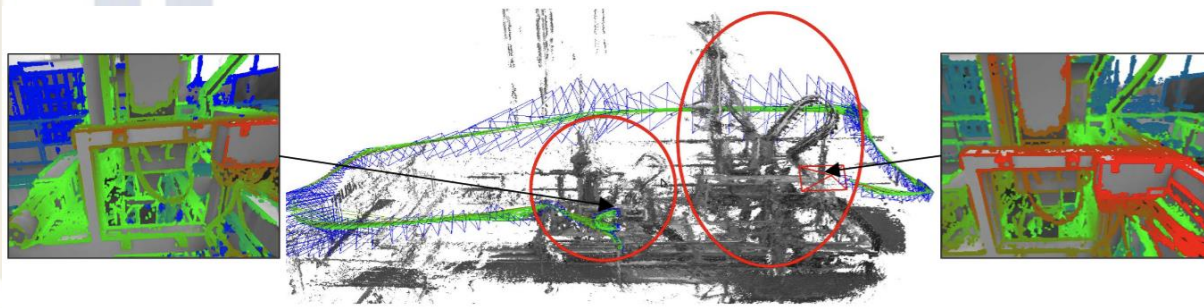


Dense Mapping

1) Direct Methods

LSD-SLAM (Large-Scale Direct SLAM)

- Semi-dense approach using only high-gradient pixels.
- Achieves real-time performance on CPU.
- Builds a pose-graph of keyframes and corresponding semi-dense depth maps.



Source: LSD-SLAM: Large-Scale Direct Monocular SLAM

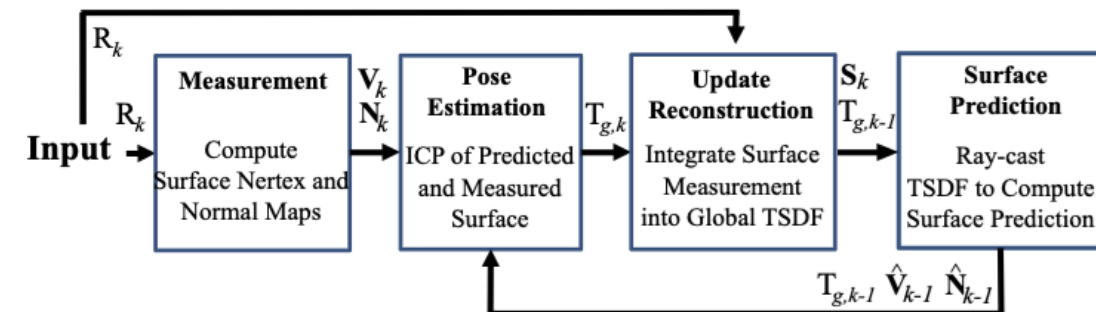
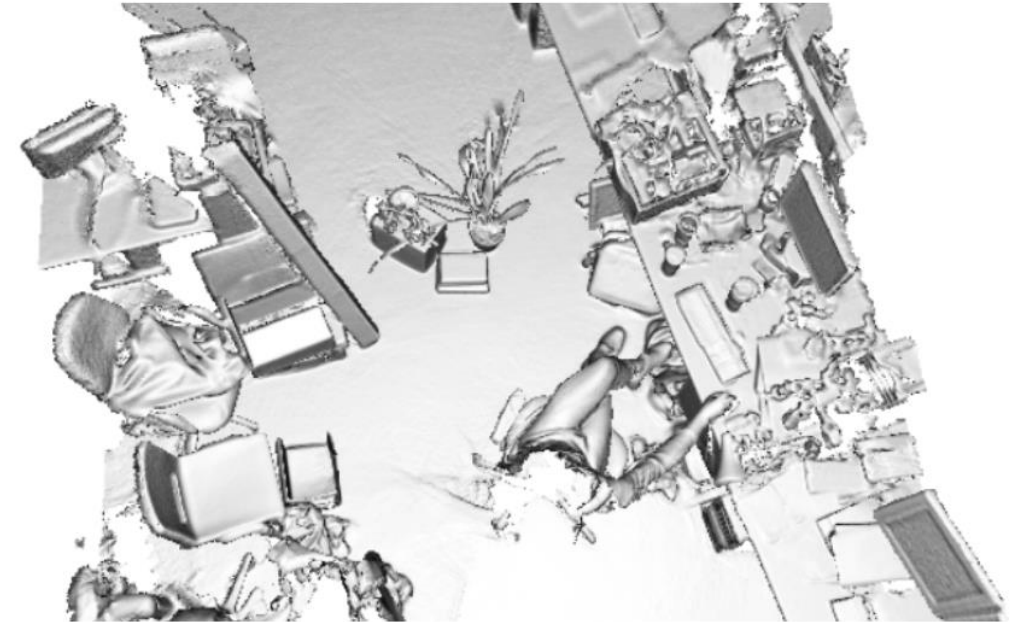
Dense Mapping

2) Volumetric Mapping

- Represent the environment as a **volumetric grid**, where each voxel stores a **Truncated Signed Distance Function (TSDF)** — the signed distance to the nearest surface.
- Surfaces are recovered as the zero-crossing of the TSDF field.
- Depth frames are **fused incrementally** into the volume for noise reduction and completeness.

- **KinectFusion (2011):**

- real-time dense reconstruction using depth sensors (RGB-D).
- Integrates each depth map into a global TSDF volume.
- Camera tracking via dense Iterative Closest Point (ICP).

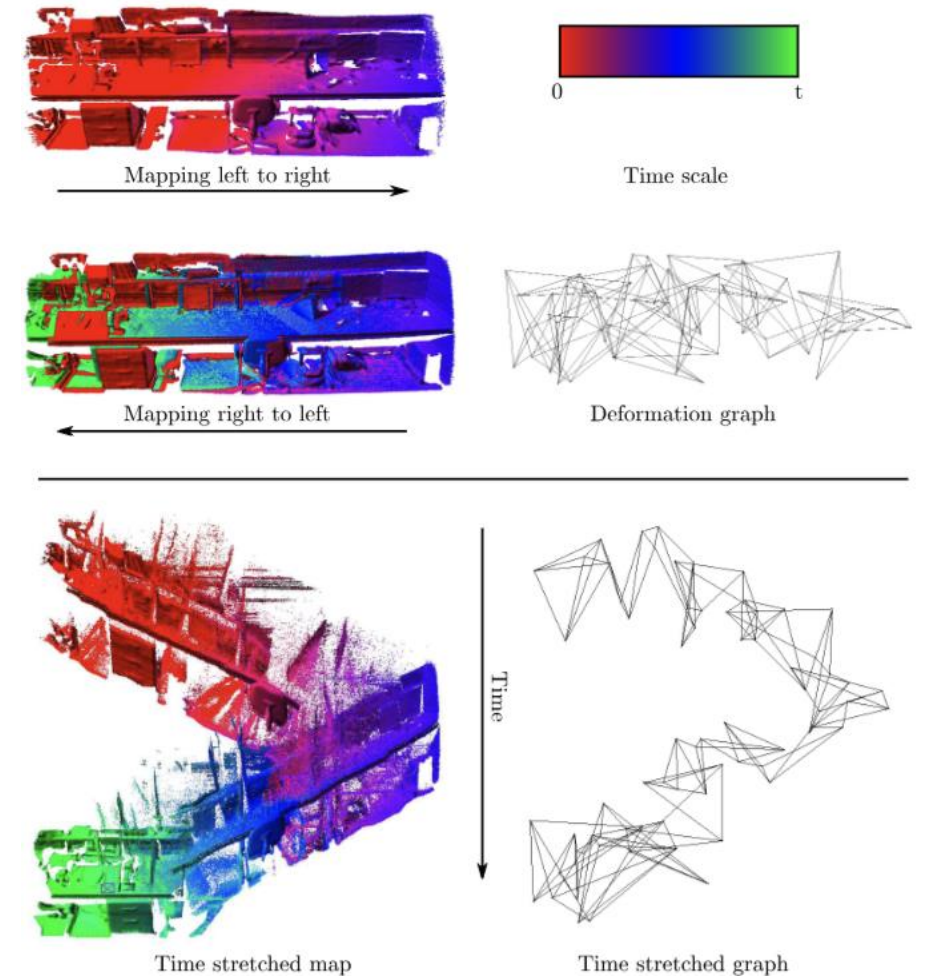
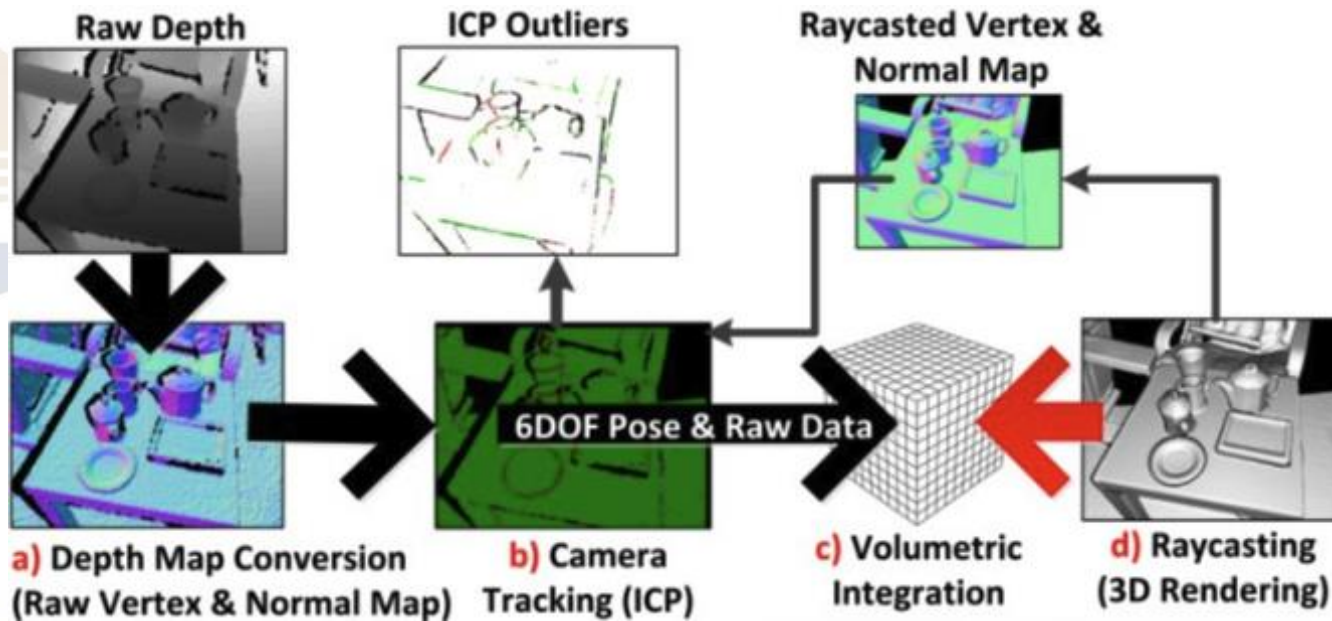


Dense Mapping

2) Volumetric Mapping

•ElasticFusion (2015):

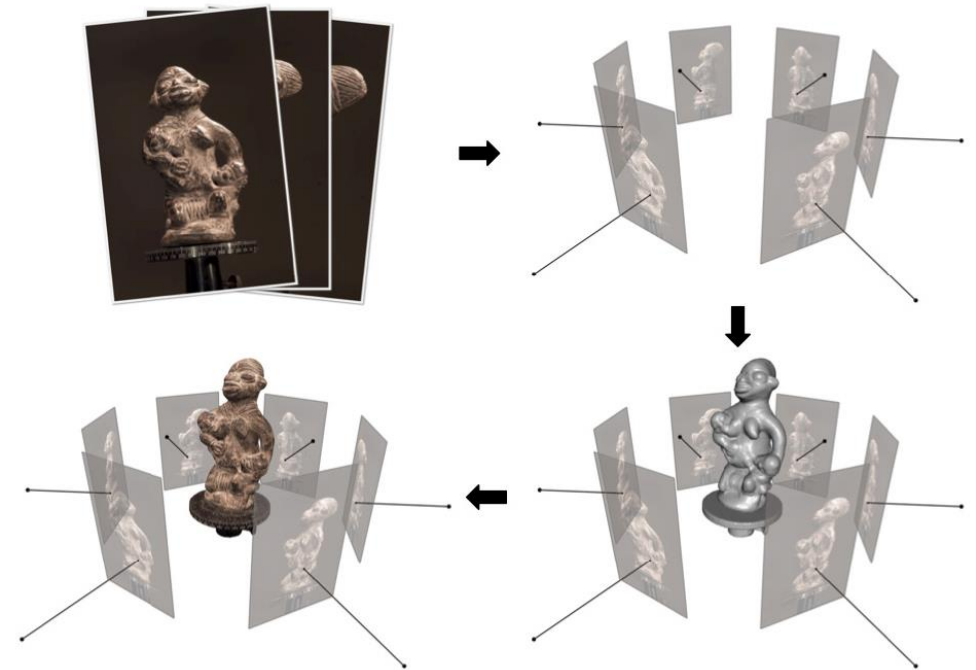
- Extends KinectFusion with **non-rigid deformation** to handle loop closures.



Dense Mapping

3) Depth Fusion

- **Depth fusion** combines multiple views or depth predictions to produce dense 3D reconstructions with improved accuracy and completeness.
- Can operate on **multi-view images** or **learned monocular depth maps**.
- **Multi-View Stereo (MVS):**
 - Reconstructs dense 3D geometry by matching pixels across multiple calibrated images.
 - Depth is estimated by minimizing reprojection error.
 - Variants: PatchMatch Stereo, COLMAP dense reconstruction.



Dense Mapping

3) Depth Fusion

• Learned Depth Fusion:

- Uses deep networks to predict or refine depth maps (e.g., DeepMVS, MVSNet).
- Fuses learned depth from multiple frames for robust reconstruction in challenging scenes.
- Enables dense mapping in monocular SLAM and self-supervised frameworks.

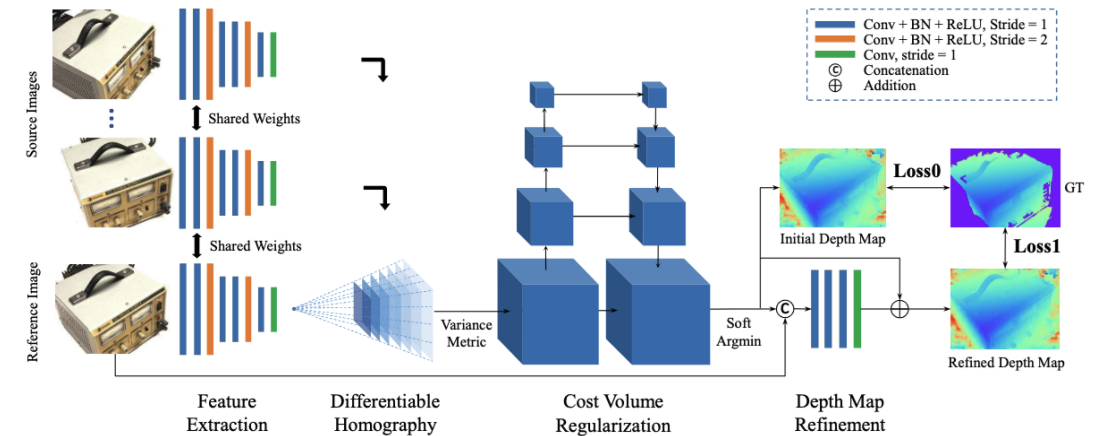
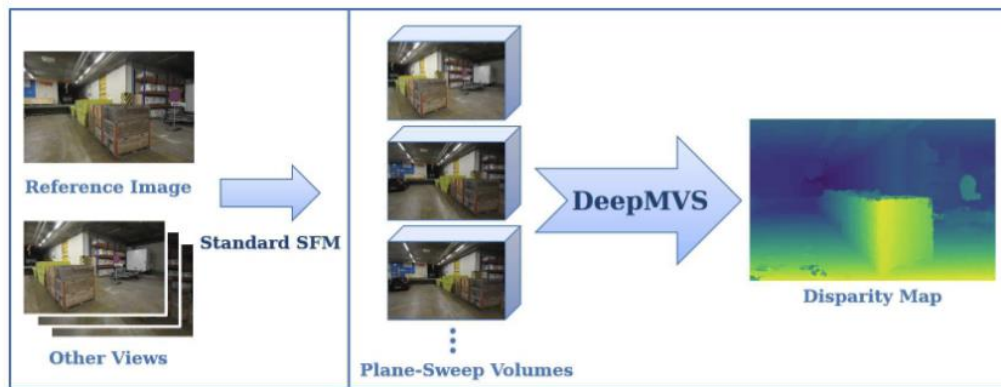
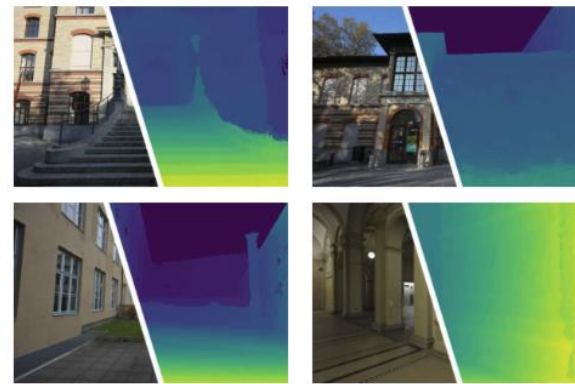


Fig. 1: The network design of MVSNet. Input images will go through the 2D feature extraction network and the differentiable homography warping to generate the cost volume. The final depth map output is regressed from the regularized probability volume and refined with the reference image



(a) Input

(b) Deep multi-view stereo



(c) Sample reconstruction results



Thanks for your attention!

Changhao Chen
HKUST (GZ)

changhaochen@hkust-gz.edu.cn

Homepage: [changhao-chen@github.io](https://github.com/changhao-chen)