Direct Visual Odometry

Graduate Course INTR-6000P

Week 5 - Lecture 9

Changhao Chen

Assistant Professor

HKUST (GZ)

Recap: Visual Pose Estimation

1. 2D-2D Pose Estimation (Monocular)

Input: Corresponding 2D points in two consecutive images.

Output: Relative camera rotation (R) and translation (t) up to scale.

Challenge: Scale ambiguity.

2. 3D-2D Pose Estimation (PnP)

Input: 2D image points and their corresponding known 3D points (from a map or previous reconstruction).

Output: Absolute 6-DOF camera pose (R, t).

Advantage: Provides metric scale.

Recap: Epipolar Geometry

We define the result of that multiplication as a single matrix:

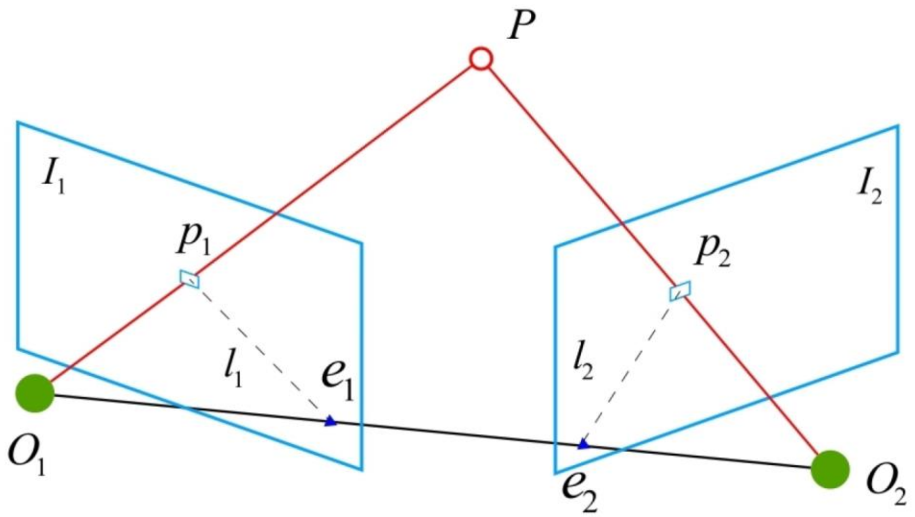
The Essential Matrix

$$\mathbf{E} = [\mathbf{t}]_{\times}\mathbf{R}^T$$

This allows us to write the coplanarity constraint in its most compact form:

$$\mathbf{P'}^T\mathbf{E}\mathbf{P'} = 0$$

This equation uses the 3D point P' as seen in the second camera's frame.



Recap: The 2D-2D Pose Estimation

1) Input: Image pair + 2D-2D correspondences (from SIFT, ORB, etc.).

2) Normalize Coordinates: (Optional but recommended) Transform points using K^{-1} to work with E for better numerical stability.

3) Robust Estimation: Use the 8-Point Algorithm inside a RANSAC loop to compute the Essential Matrix E.

4) Pose Extraction: Perform SVD on E to get the four possible (R, t) solutions.

5) Disambiguation: Perform a Cheirality Check (triangulation) to find the single physically possible pose.

6) Output: The relative camera pose $T = [R \mid t]$, an element of SE(3).

Recap: 3D-2D Pose Estimation (PnP)

The Reprojection Error
We can't solve the equations perfectly due to noise. Instead, we find the pose that minimizes the reprojection error.

Reprojection Error for a single point:

$$\text{error}_i = ||\, \mathbf{p}_i - \pi(\mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{P}_i)\, ||^2$$

where π([x, y, z]^T) = [x/z, y/z]^T is the perspective projection function.

Total Error we want to minimize:

$$(\mathbf{R}, \mathbf{t})^* = \arg\min_{\mathbf{R},\mathbf{t}} \sum_i ||\, \mathbf{p}_i - \pi(\mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{P}_i)\, ||^2$$

This is a non-linear least squares problem.

Recap: 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.

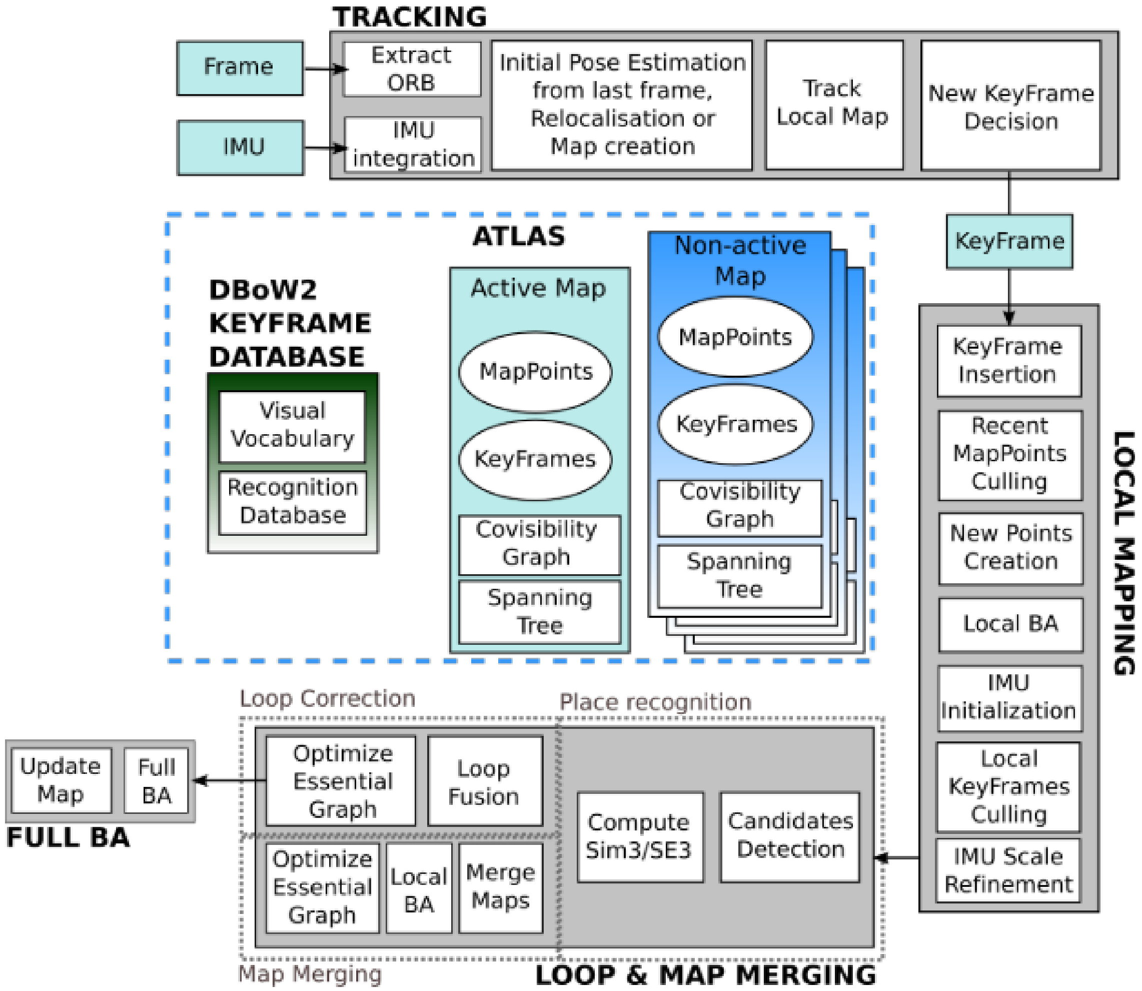
4. Solve PnP (using RANSAC + EPnP) to get a robust, initial pose estimate.

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.

Recap: 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

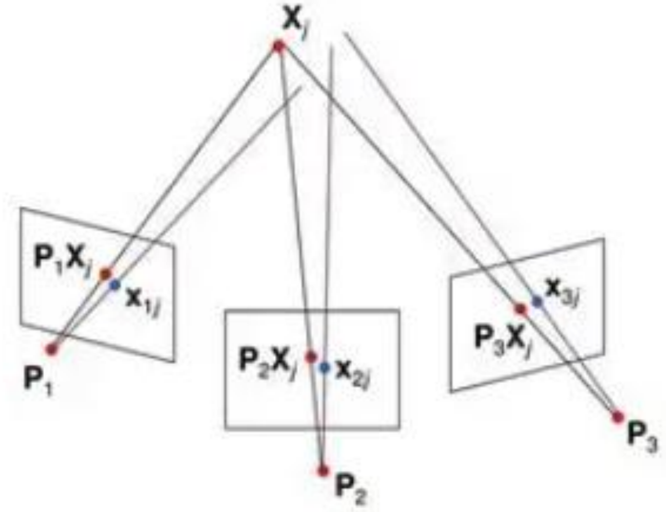
Recap: Bundle Adjustment

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.



Recap: 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.**

Feature-Based Visual Odometry: The Trade-Offs

Advantages

1) Robustness & Maturity

- Well-understood, battle-tested pipeline (e.g., ORB-SLAM).
- Highly resilient to outliers via **RANSAC** and robust descriptors.

2) Computational Efficiency

- Only processes a **sparse** set of features, not the whole image.
- Enables real-time performance on standard hardware.

3) Robust to Illumination

- Feature descriptors (e.g., ORB) are designed to be invariant to lighting changes, unlike photometric methods.

4) Strong Global Consistency

- **Bag-of-Words** models enable efficient **place recognition** and **loop closure**.
- Corrects drift effectively for large-scale mapping.

Limitations

1) Depends on Texture

- Fails in **textureless environments** (blank walls, smooth asphalt).
- Relies on finding a sufficient number of distinct keypoints.

2) Sparse Reconstruction

- Creates a sparse map of points.
- Lacks dense geometry, which is less useful for navigation and planning tasks.

3) Sensitive to Motion Blur

- Fast motion corrupts feature detection and description.
- Can lead to tracking loss in dynamic maneuvers.

4) Discrete Processing

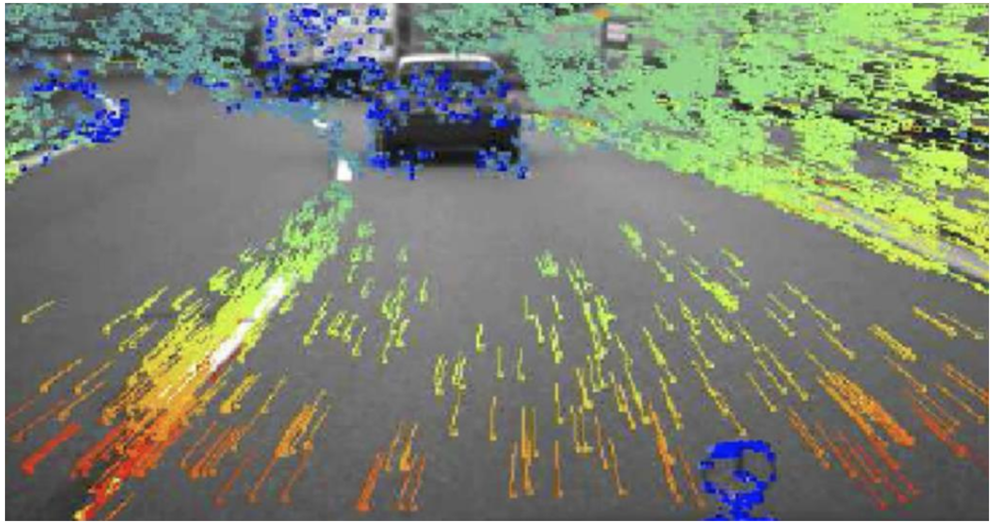
- Sensitive to incorrect **data association** (feature matching).
- A single bad match can corrupt motion estimation if not caught by RANSAC.

Feature-Based Methods vs. Direct Methods

Feature-Based Methods	**Direct Methods**
1. **Detect** distinctive features (e.g., corners).	1. **Use all or most pixel intensities directly.**
2. **Match** features across images.	2. **Assume brightness constancy** for a world point.
3. **Reconstruct** 3D points via triangulation.	3. **Minimize photometric error** to find camera motion.
4. **Motion from 3D-2D** correspondences (PnP).	4. **Geometry and motion are solved jointly.**
"Geometry-first"	**"Photometry-first"**

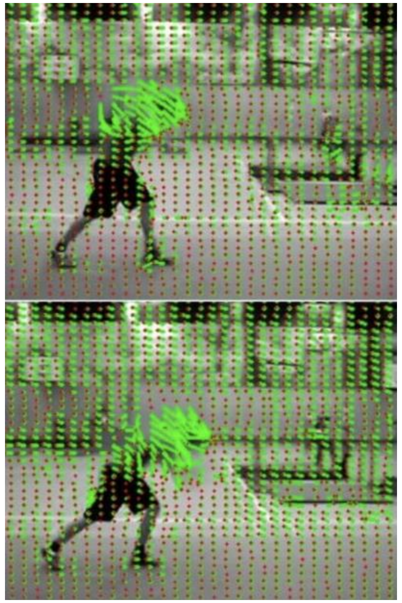
2D Optical Flow

Definition: The apparent motion of brightness patterns in the image plane between two consecutive frames.



•It is a 2D Vector Field: For each pixel (or region), optical flow is a vector (u, v) representing:

- u: horizontal displacement (pixels/frame)
- v: vertical displacement (pixels/frame)

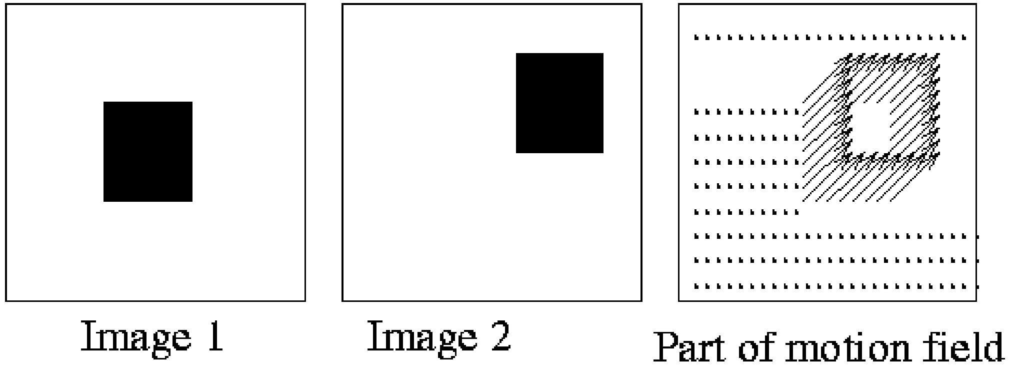


2D Optical Flow

The Apparent Motion Field

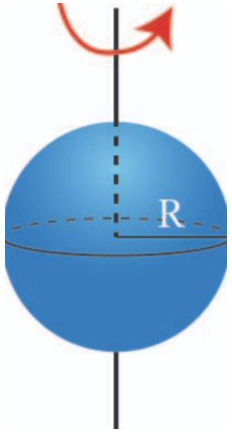
Crucial Distinction:

- **Optical Flow:** Apparent motion of brightness patterns.
- **Motion Field:** The real 2D projection of 3D motion vectors.



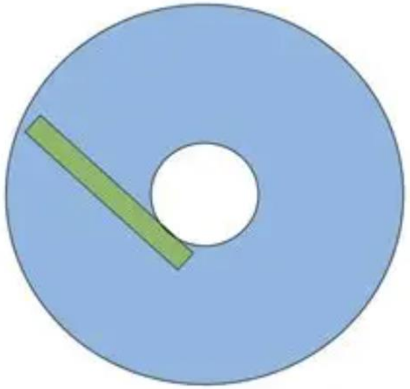
•**They are not always the same!**

•**Example 1:** A spinning barber pole (motion field is horizontal, but optical flow is vertical due to stripes).
•**Example 2:** A uniform, rotating sphere (real motion, but no optical flow due to lack of texture).



2D Optical Flow

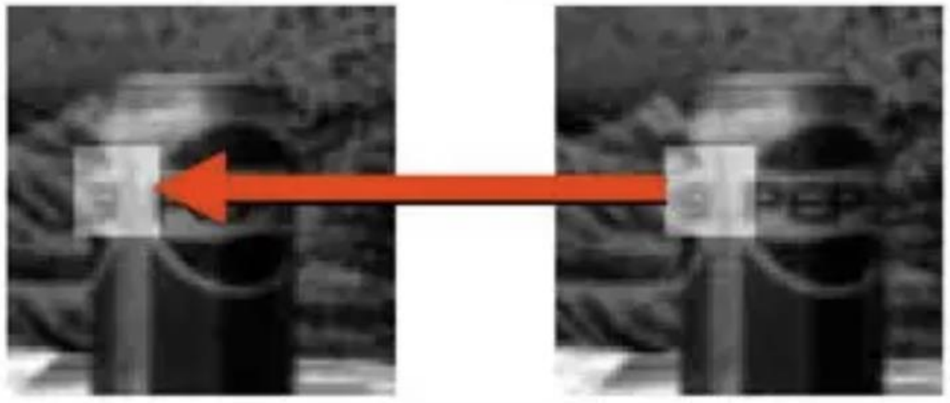
The Aperture Problem



Observation: Through the aperture, you can only perceive the motion *normal* (perpendicular) to the edge. The motion *along* the edge is invisible.

Implication: For a single pixel on a uniform edge, there are infinitely many (u, v) pairs that explain the observed change. The problem is **ill-posed**.

Assumption: Brightness Constancy



The Foundation: The brightness of a small pixel patch remains constant as it moves from one frame to the next.

Mathematically: $I(x, y, t) = I(x + u, y + v, t + dt)$

- I(...) is the image intensity.
- (x, y) is the pixel location at time t.
- (u, v) is the optical flow vector we want to find.
- dt is the time between frames.

2D Optical Flow

Optical Flow Constraint Equation

1.Start with Brightness Constancy: I(x, y, t) = I(x + u, y + v, t + dt)

2. Assume small motion (u, v) and apply a **1st-order Taylor Series Expansion** on the right-hand side:

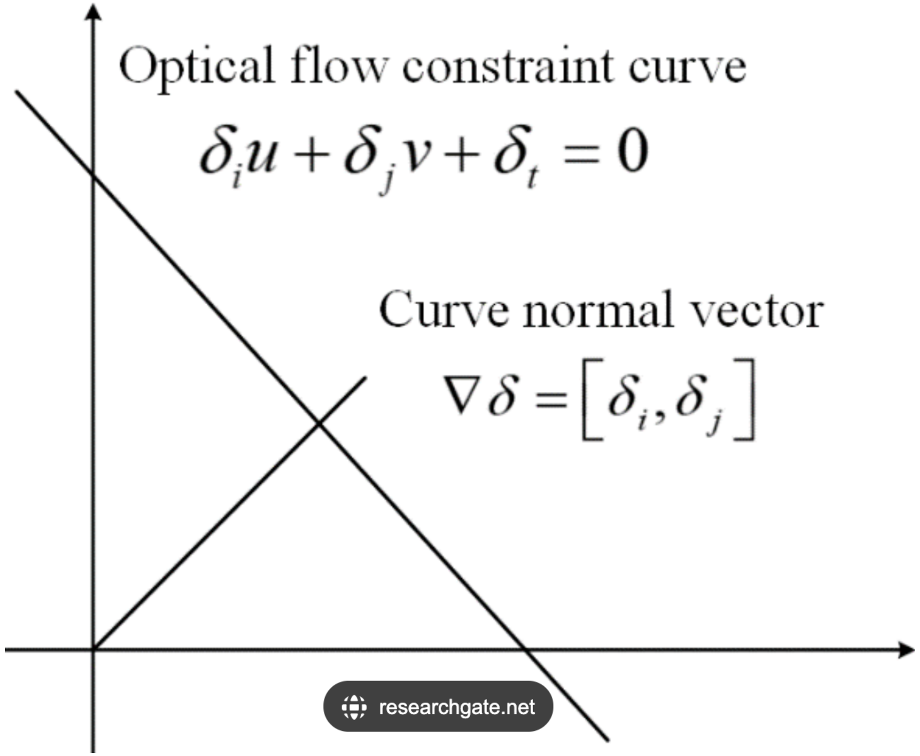
I(x+u, y+v, t+dt) ≈ I(x,y,t) + (∂I/∂x)u + (∂I/∂y)v + (∂I/∂t)dt

3. Plug back in and cancel I(x,y,t):
(∂I/∂x)u + (∂I/∂y)v + (∂I/∂t)dt = 0

4. Divide by dt and use shorthand

I_x = ∂I/∂x, I_y = ∂I/∂y, I_t = ∂I/∂t

I_x * u + I_y * v + I_t = 0


Optical flow constraint curve
$\delta_i u + \delta_j v + \delta_t = 0$
Curve normal vector
$\nabla \delta = [\delta_i, \delta_j]$
researchgate.net

One Equation, Two Unknowns

The Optical Flow Constraint Equation:
I_x * u + I_y * v = -I_t

This defines a line in (u, v) space. The true flow vector lies somewhere on this line.

2D Optical Flow

The Lucas-Kanade Method

The Solution: Impose a **local constraint**.

Lucas-Kanade (1981) Assumption: The flow vector (u, v) is **constant** within a small local neighborhood (e.g., a 5x5 or 7x7 window) around the pixel.

Now we have multiple equations for one (u, v)!

- Pixel 1 in window: I_x1 * u + I_y1 * v = -I_t1
- Pixel 2 in window: I_x2 * u + I_y2 * v = -I_t2
- ...
- Pixel n in window: I_xn * u + I_yn * v = -I_tn

When is this valid?

- Small inter-frame motion.
- The patch is on a single, roughly planar surface.

When does it fail?

- At motion boundaries (e.g., the edge of a car against the background).
- With large, non-rigid deformations.

2D Optical Flow

Solving the System of Equations

We have a system: A * [u; v] = b
Where:

- A = [[I_x1, I_y1]; [I_x2, I_y2]; ... [I_xn, I_yn]] (an n x 2 matrix)
- b = [-I_t1; -I_t2; ... -I_tn] (an n x 1 vector)

The Least-Squares Solution is:
[u; v] = $(A^TA)^{-1} A^Tb$

The Role of the Hessian Matrix

A^TA is a 2x2 matrix, often called the **Hessian** in this context.
A^TA = [ΣI_x², ΣI_xI_y; ΣI_xI_y, ΣI_y²]

Interpretation: This matrix captures the gradient structure of the patch.

For a solution to exist, $(A^TA)^{-1}$ must exist. This requires A^TA to be invertible, which means it must have full rank (rank=2).

This requires the patch to have gradients in at least two different directions (i.e., textured). A patch on a uniform edge (aperture problem) will have a singular Hessian.

2D Optical Flow

Limitations & Improvements of Basic LK

Limitations:

- **Small Motion Assumption:** Taylor series expansion fails for large motions.
- **Sparse Output:** Only computes flow where the Hessian is well-conditioned (e.g., at corners).
- **Fails at Motion Boundaries.**

Common Improvements:

- **Coarse-to-Fine (Pyramidal) LK:** Run LK on an image pyramid to handle large displacements.
- **Affine Motion Model:** Model the motion in the window as an affine transformation (6 parameters) instead of a simple translation (2 parameters).
- **Iterative Refinement:** Use the estimated flow to warp the image and re-estimate a residual flow.

Direct Visual Odometry

Photometric Consistency

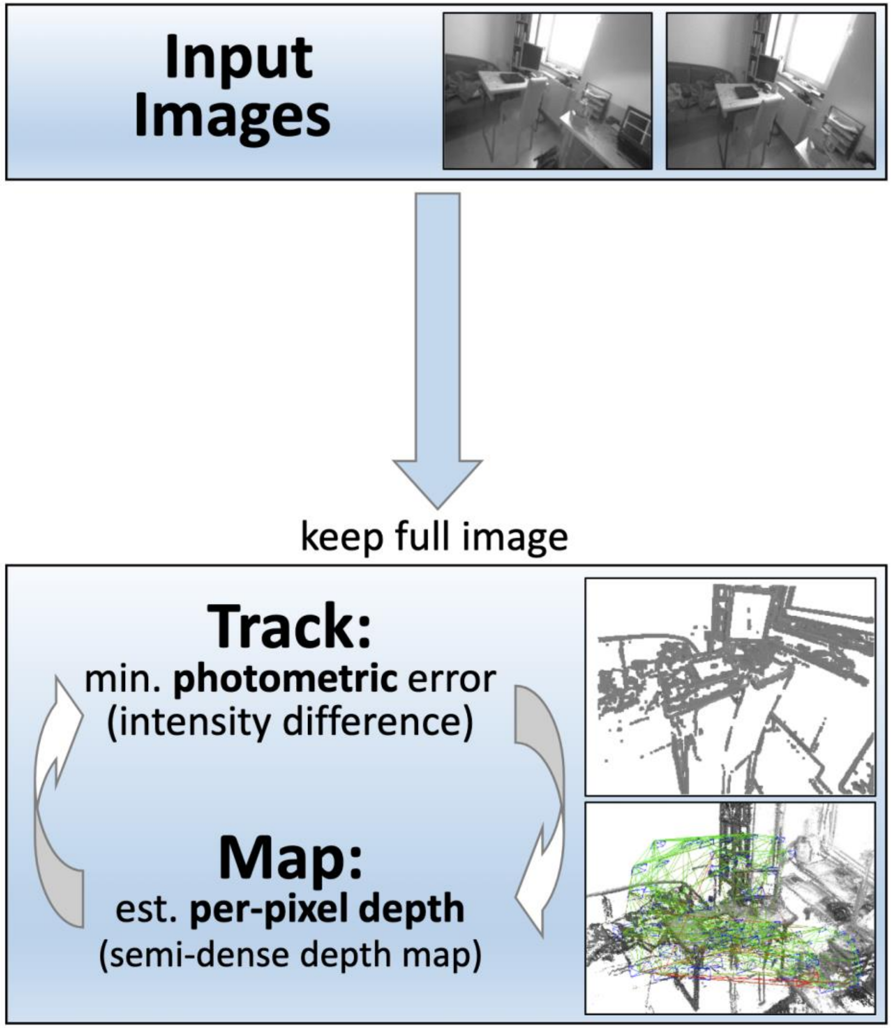
Core Idea: The brightness of a world point remains constant across successive frames.

Assumption: I(p, t) ≈ I(p', t+1)

- p is a pixel in the first image.
- p' is the *corresponding* pixel in the second image, projected from the same 3D point.

This is a **strong assumption!** It can be violated by:

- Non-Lambertian surfaces (specular reflections)
- Auto-exposure and auto-white balance
- Lighting changes



Direct Visual Odometry

The Mathematical Model: Photometric Error

Let P be a 3D point, visible in two images.

Its projection in the first image I_1 is p.
Its projection in the second image I_2 is p' = π(T * P).

Photometric Error: r_I = I_2(p') - I_1(p)

Goal: Find the camera pose T that **minimizes the sum of squared photometric errors** over all pixels.

T* = argmin_T Σ [I_2(π(T * P_i)) - I_1(p_i)]^2

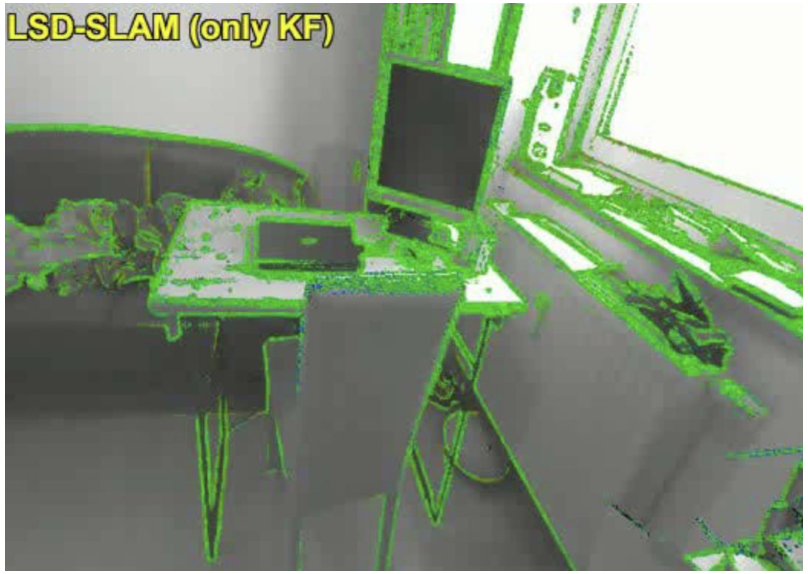
Direct Visual Odometry

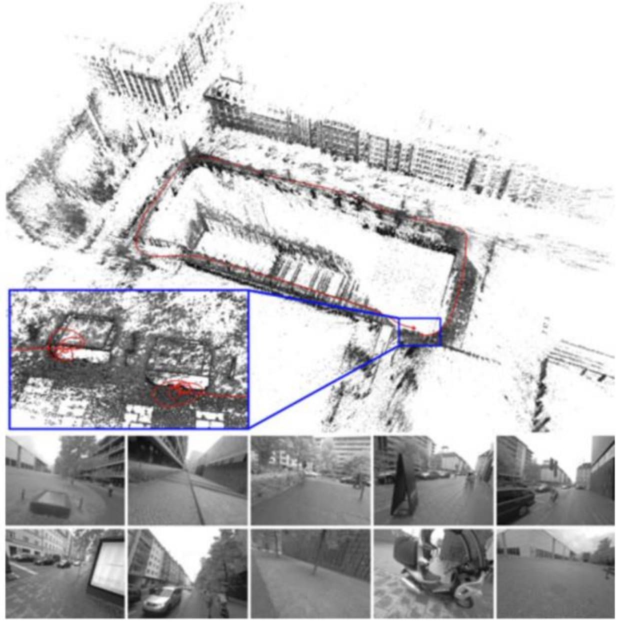
Dense vs. Semi-Dense vs. Sparse

Dense: Use **every single pixel** in the image. Very computationally expensive. Often used in RGB-D SLAM.

Semi-Dense: Use pixels with **sufficient image gradient** (i.e., not in completely flat regions). A good trade-off. (e.g., LSD-SLAM).

Sparse: Select a **sparse set of pixels** with high gradient. Much faster, more like feature-based in selection, but direct in alignment. (e.g., DSO).





Direct Visual Odometry

Formulating the Problem

We have a **cost function** to minimize: $E(T) = \Sigma r\_I^2$

This is a **non-linear least squares problem**.

How do we solve it? **Iterative Optimization** (Gauss-Newton / Levenberg-Marquardt).

We need to compute how the error changes as we tweak the pose T (i.e., the **Jacobian**).

Solution: Gauss-Newton Algorithm

- Start with initial guess T_0.
- Linearize the photometric error around T_0: r_I(T_0 + ΔT) ≈ r_I(T_0) + J * ΔT
- Solve for the pose update ΔT that minimizes the linearized error: $(J^T J)\, \Delta T = -J^T r\_I$
- Update the pose: T_1 = T_0 + ΔT
- Repeat until convergence.

•**The Jacobian J** contains the derivatives of the image I_2 with respect to the camera pose T.

Direct Visual Odometry

The Pipeline of a Direct VO System

Input Image -> Pre-processing -> Pixel Selection -> Motion Estimation (Tracking) -> Depth Estimation (Mapping) -> Map Management -> Output Pose

Step 1: Image Pre-processing

•**Why?** The photometric consistency assumption is fragile.

•**Key Calibrations:**

- **Vignette:** Compensate for darker corners of the image.
- **Gamma Correction:** Account for non-linear camera response.
- **Photometric Calibration:** Pre-compute a camera response function.

•**Rolling Shutter Compensation:** Model and correct for distortion from sequential row exposure.

Step 2: Selecting Good Pixels

•We want pixels where we can reliably estimate depth.

•**Ideal Pixels:** Have a high image gradient (e.g., along edges).

•**Bad Pixels:** No gradient (completely uniform regions).

•**Strategy:** Sample pixels from across the image, prioritizing those with high gradient magnitude.

Direct Visual Odometry

Step 3: Camera Motion Estimation (Tracking)

•**Goal:** Estimate the current camera pose T_k given the previous frame and the current frame.
•**How:** Use the Gauss-Newton optimization from Slide 11.
•**Input:** A set of points from the previous frame with their estimated depths.
•**Output:** The 6-DoF camera pose T_k that minimizes the photometric error to the current frame I_k.

Step 4: Depth Estimation (Mapping)

•**Goal:** Estimate the depth of selected pixels.
•**How:** For a *static* camera pose, the depth of a point is estimated by searching along the **epipolar line** in other frames for the best photometric match.
•**Filtering:** Depth is not calculated once. It is **refined over multiple observations** using a Kalman Filter or, more commonly, a running weighted mean and variance.

Direct Visual Odometry

Step 5: Map Management & Scale Handling

- **Map:** A collection of **reference keyframes** with their associated semi-dense depth maps.
- **Scale Ambiguity (Monocular):** Inherent to monocular vision. Direct methods suffer from this too.
- **Solutions:**
    - Use stereo or RGB-D camera (fixed scale).
    - **Fuse with IMU** (highly effective, standard in modern systems).
    - Use prior knowledge (e.g., known vehicle height).

Direct Visual Odometry

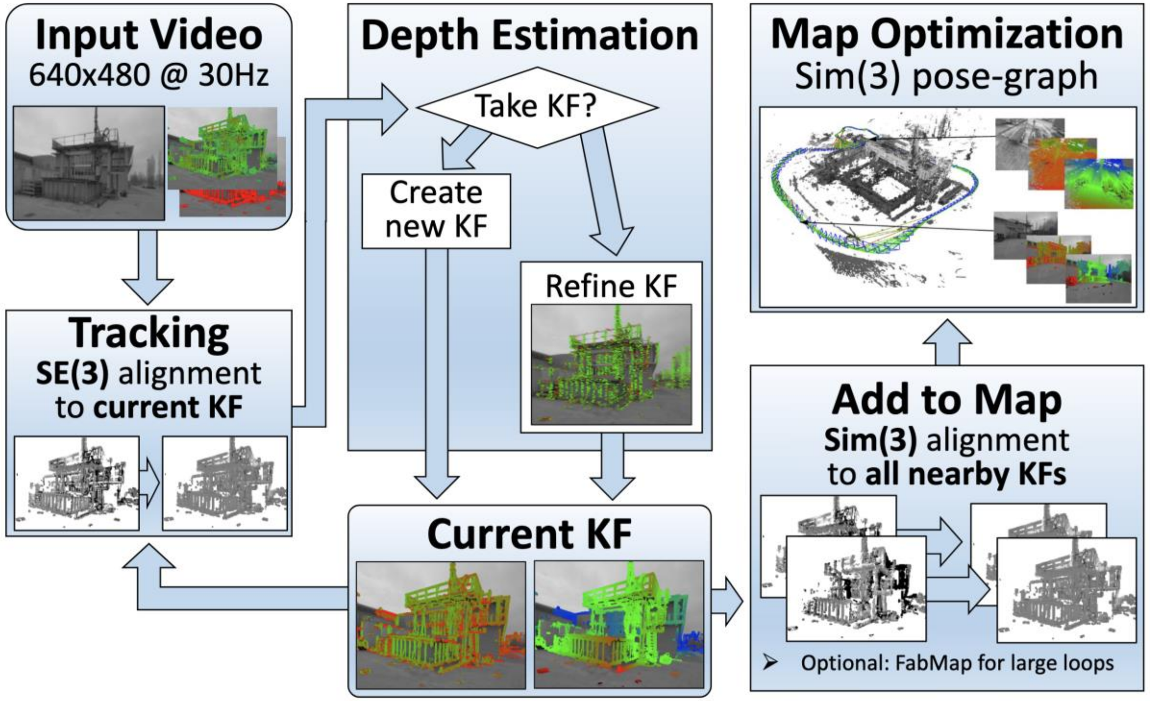
LSD-SLAM (Large-Scale Direct SLAM)

Semi-Dense, Monocular.

Key Idea: Builds a **pose graph** of keyframes, where the constraints between keyframes are estimated by direct image alignment.

Contribution: Showed that direct methods could be used for large-scale, consistent mapping, not just odometry.

Output: A globally consistent semi-dense 3D map of the environment.

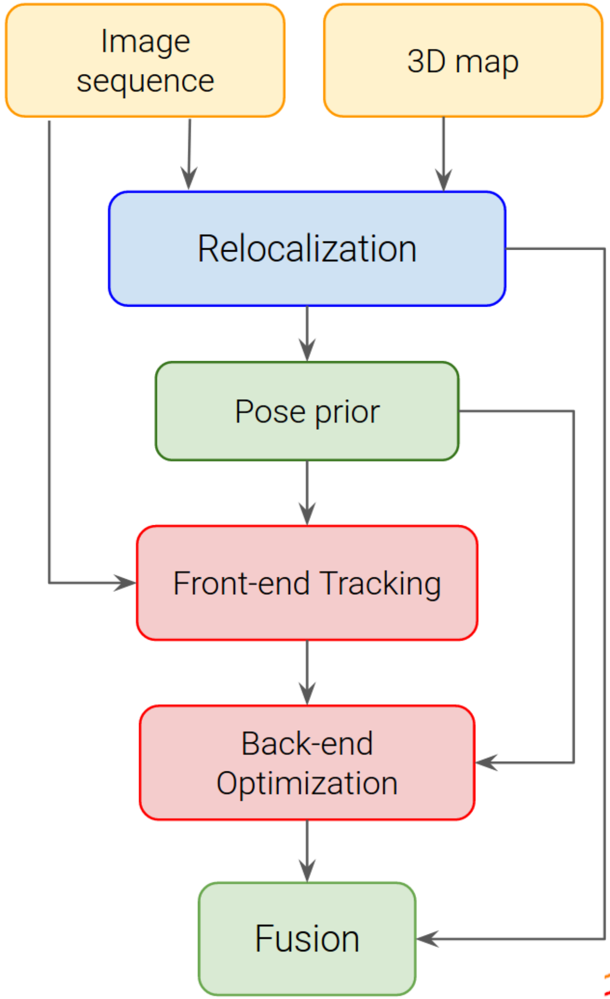


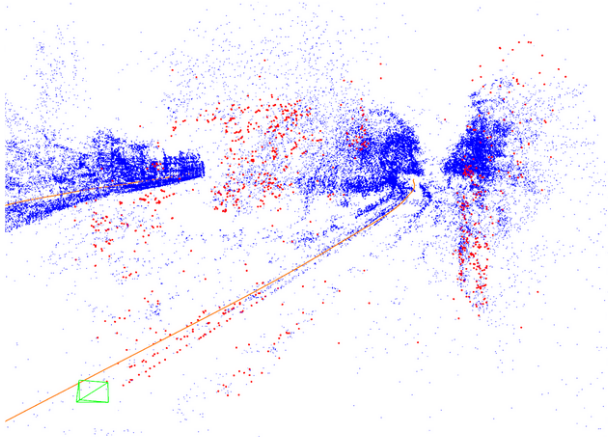
1) Engel, J., Sturm, J., & Cremers, D. (2013). Semi-dense Visual Odometry for a Monocular Camera. (LSD-SLAM)
2) Engel, J., Koltun, V., & Cremers, D. (2017). Direct Sparse Odometry. (DSO)
3) Forster, C., Zhang, Z., Gassner, M., Werlberger, M., & Scaramuzza, D. (2016). SVO: Semi-direct Visual Odometry for Monocular and Multi-Camera Systems.

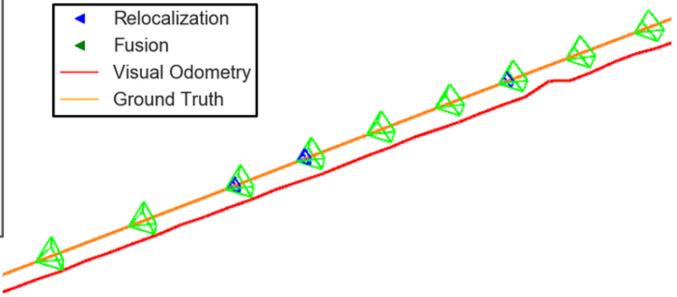
Direct Visual Odometry

DSO (Direct Sparse Odometry)

- Sparse direct method.
- Jointly optimizes over a window of frames for both camera poses and 3D point depths (windowed bundle adjustment on photometric error).







Summary

1. Direct Methods minimize photometric error instead of geometric reprojection error.

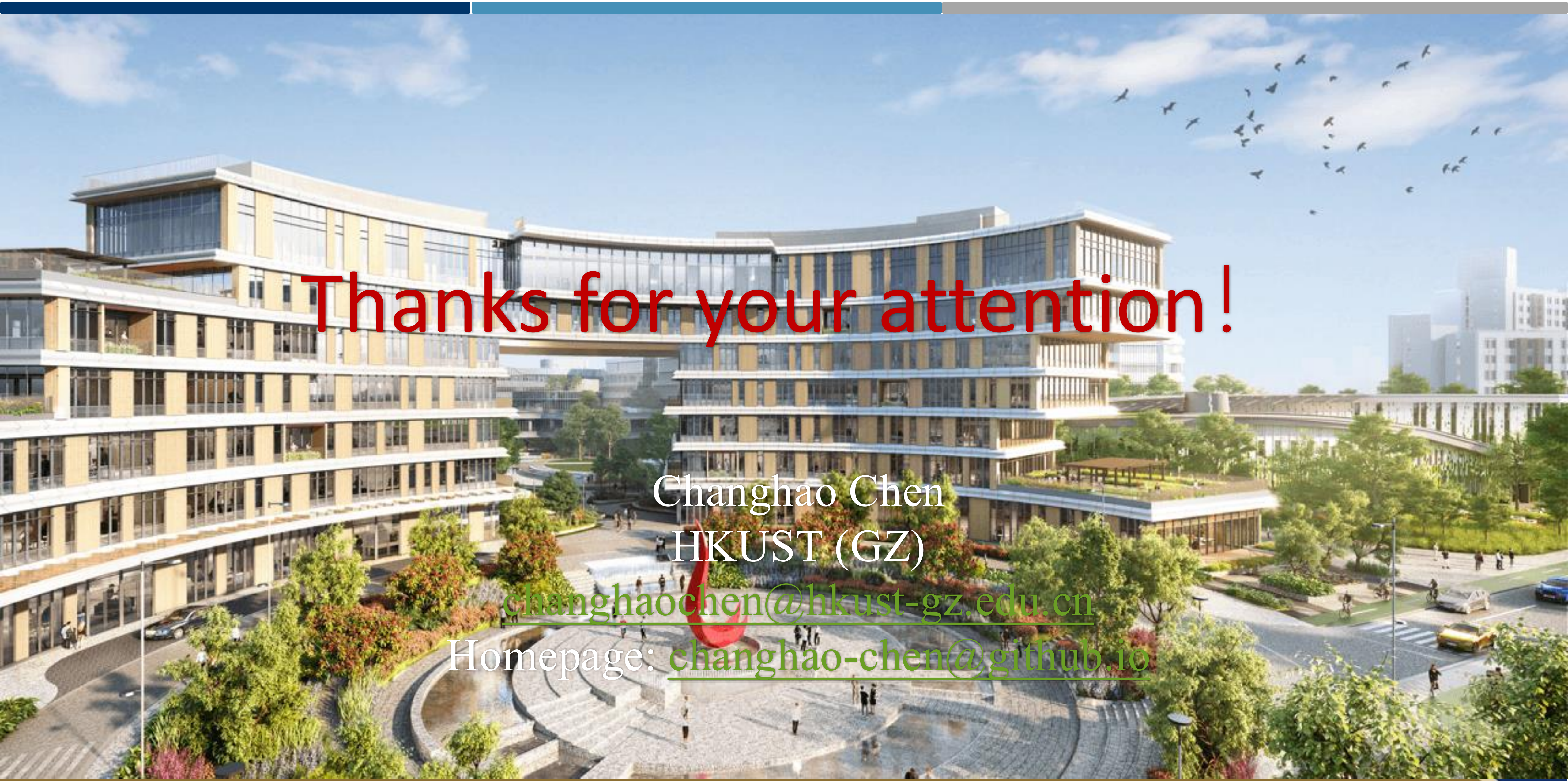
2. They rely on the photometric consistency assumption, requiring careful image pre-processing.

3. They excel in textureless environments and can produce semi-dense maps.

4. The core is a non-linear optimization problem solved iteratively.

5. They are sensitive to photometric changes and motion blur.

6. Direct Sparse Odometry (DSO) + IMU represents the cutting-edge for robust visual-inertial odometry in intelligent vehicles.



Thanks for your attention！

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