Direct Sparse Odometry (DSO) and Delayed Marginalization Visual-Inertial Odometry (DM-VIO)

Final Submission - Mobile Robotics

Aniruth Suresh
2022102055
IIIT , Hyderabad, India
aniruth.suresh@students.iiit.ac.in

Aryan Garg
2022102074
IIIT , Hyderabad, India
aryan.garg@students.iiit.ac.in

Pratyush Jena
2022111016
IIIT , Hyderabad, India
pratyush.jena@research.iiit.ac.in

***Abstract*—This report presents the final project findings on Direct Sparse Odometry (DSO), a state-of-the-art visual odometry method designed to estimate camera motion in real-time using sparse keypoints. Additionally, we explore an enhanced variant of DSO that integrates Inertial Measurement Unit (IMU) data and employs Delayed Marginalization (DM-VIO) to improve performance.**

DSO combines a fully direct probabilistic model that minimizes photometric error with consistent, joint optimization of all parameters, including geometry—represented as inverse depth in a reference frame—and camera motion. Real-time performance is achieved by omitting the smoothness prior used in other direct methods, instead sampling pixels uniformly across the images.

The goal of this project is to implement and analyze both DSO and DM-VIO, evaluate the impact of incorporating IMU data and loop closure, and benchmark their performance against the base DSO implementation.

I. Problem Statement

The primary challenge in **Visual Odometry (VO)** is accurately estimating the motion of a camera in real-time using visual data, while maintaining high robustness, efficiency, and precision. Traditional odometry methods, such as *Parallel Tracking and Mapping (PTAM)* and *Visual Odometry (VO)* techniques like *Feature-based VO*, often face difficulties in large-scale environments, dynamic scenes, and **low-texture areas**. Additionally, many of these methods rely on computationally expensive feature matching techniques, such as ***SIFT*, *SURF*, or *ORB***, to extract keypoints, which significantly impact runtime and hinder real-time applicability.

Direct Sparse Odometry (DSO) seeks to overcome these challenges by utilizing a sparse set of keypoints for camera pose estimation, **without relying on feature matching**. The combination of **Sparse and Direct** methods provides several advantages: the sparse formulation **avoids the use of a smoothness prior**, which simplifies the optimization and improves efficiency, while the direct approach **skips the feature extraction step** and minimizes the photometric error directly on the image data. This results in a more robust and computationally efficient system suitable for real-time applications.

Developing robust visual-inertial odometry (VIO) systems presents several challenges, including the **irreversible nature of marginalization**, which fixes the linearization points of connected variables and prevents updates to marginalized states. **Accurately initializing and utilizing IMU data is also difficult,** particularly under photometric uncertainty, limiting the reliability of scale and gravity estimation.To overcome these challenges, **DM-VIO** introduces two key innovations: ***delayed marginalization*,** which allows IMU data to be injected into already marginalized states, and ***pose graph bundle adjustment***, enabling robust IMU initialization and improved performance across diverse scenarios.

We begin by discussing and explaining **Direct Sparse Odometry (DSO)**, presenting its methodology and results. Following this, we delve into the details of **DM-VIO**, highlighting its novel features and improvements. Finally, we compare the results of DSO and DM-VIO to **analyze the impact of incorporating IMU data and delayed marginalization**.

II. Model Pipeline and Architecture - DSO

In this section, we will go through each block of the overall pipeline, which is depicted in 1 , to provide a brief overview and understanding of the complete architecture.

A. Calibration

Direct Sparse Odometry (DSO) uses both **geometric** and **photometric** camera calibration to model the image formation process comprehensively.

1) Geometric Camera Calibration: For geometric calibration, DSO employs the well-known ***pinhole camera model***. Radial distortion is removed in a preprocessing step to simplify the model. The forward projection of a 3D point $\mathbf{X} = (X, Y, Z)^T$ onto the 2D image plane $\mathbf{x} = (x, y)^T$ is given by:

$$\mathbf{x} = \Pi_c(\mathbf{X}) = \left(\frac{fX}{Z} + c_x, \frac{fY}{Z} + c_y \right) \tag{1}$$

where f is the focal length, (c_x, c_y) are the coordinates of the principal point, and $\mathbf{X}$ is the 3D point in world coordinates. The 2D point $\mathbf{x}$ is in the image plane.

The reverse (or back-projection) formula is used to convert a 2D pixel $\mathbf{x} = (x, y)$ back to a 3D ray in the camera coordinate system:

$$\Pi_c^{-1}(\mathbf{x}) = (X, Y, Z) = \left(\frac{(x - c_x)Z}{f}, \frac{(y - c_y)Z}{f}, Z \right) \quad (2)$$

In practice, the back-projection relies on an initial estimate of Z, which is typically determined via optimization techniques.

2) Photometric Camera Calibration: For photometric calibration, DSO uses an image formation model that accounts for **non-linear response and lens attenuation**. The observed pixel intensity $I_i(x)$ in frame i is related to the irradiance $B_i(x)$ by the following equation:

$$I_i(x) = G(t_i V(x) B_i(x)) \quad (3)$$

where G is the non-linear response function, accounting for the exposure time t_i, and $V(x)$ represents lens attenuation (vignetting) at pixel x. The image is photometrically corrected by:

$$\boxed{I_i'(x) := t_i B_i(x) = \frac{G^{-1}(I_i(x))}{V(x)}} \quad (4)$$

where $I_i'(x)$ is the corrected image. In the remainder of this report, I_i will always refer to the photometrically corrected image I_i', unless stated otherwise.

B. Model Formation

In this section, we will discuss the key components that form the loss function used in Direct Sparse Odometry (DSO). The two primary terms that construct the loss function are the **photometric cost** and the **marginalization cost**

1) Photometric Error / Energy - E_{photo}: The photometric error of a point $p \in \Omega_i$ in reference frame I_i, observed in a target frame I_j, is defined as the **weighted Sum of Squared Differences (SSD) over a small neighborhood of pixels**. We use a special 8 pixels pattern arranged in a slightly spread pattern (2) provides a good trade-off between computational efficiency and sufficient information.

The photometric error is mathematically expressed as:

$$\boxed{E_{p_j} = \sum_{p \in N_p} w_p || \left(I_j[p'] - b_j \right) - \frac{t_j e^{a_j}}{t_i e^{a_i}} \left(I_i[p] - b_i \right) ||_\gamma} \quad (5)$$

where:

- N_p is the set of pixels included in the SSD,
- t_i, t_j are the exposure times for images I_i and I_j,
- $\| \cdot \|_\gamma$ denotes the Huber norm,
- p_0 is the projected point position of p with inverse depth d_p, given by

$$p_0 = \Pi_c \left(R \Pi_c^{-1}(p, d_p) + t \right) \quad (6)$$

where:

$$R, t := T_j T_i^{-1}$$

and T_j and T_i are the transformation matrices of the camera frames at time j and i, respectively.

The error function also includes an **affine brightness transfer function**:

$$e^{-a_i}(I_i - b_i)$$

This formulation ensures that the scalar factor e^{-a_i} remains positive and avoids numerical issues arising from exponential drift. Unlike most prior work, the brightness transfer factor is parameterized logarithmically to prevent negative values.

In addition to using robust Huber penalties, a **gradient-dependent weighting** w_p is applied:

$$w_p = \frac{c^2}{c^2 + ||\nabla I_i(p)||_2^2}$$

This weighting function **down-weights pixels with high gradients**, which can be interpreted as adding small independent geometric noise to the projected point position p_0 and marginalizing it to approximate small geometric error.

The photometric error E_{photo} over all frames and points is given by:

$$\boxed{E_{\text{photo}} = \sum_{i \in \mathcal{F}} \sum_{p \in P_i} \sum_{j \in \text{obs}(p)} E_{p_j}} \quad (7)$$

where i runs over all frames F, p runs over all points P_i in frame i, and j runs over all frames in which point p is visible.

2) Marginalization Error / Energy - $E_{marginalization}$: This cost will be discussed in the context of windowed optimization, but the final result is a combination of the photometric cost and the marginalization cost:

$$\boxed{E_{\text{total}} = E_{\text{photo}} + E_{\text{marginalization}}} \quad (8)$$

C. Windowed Optimization

We optimize the total error (Equation 8) using a **Sliding Window approach with the Gauss-Newton algorithm**, which provides a good trade-off between speed and flexibility.

In this report, we use the $\oplus$ notation to express linearized pose increments as elements of the Lie algebra $\mathfrak{se}(3)$, specifically for transformations in $\mathrm{SE}(3)$. Given a pose increment $x_i \in \mathfrak{se}(3)$, we can represent it directly as a 6-dimensional vector $x_i \in \mathbb{R}^6$. The $\oplus$ operator is defined as:

$$\oplus : \mathfrak{se}(3) \times \mathrm{SE}(3) \to \mathrm{SE}(3),$$

using a left-multiplicative formulation, where

[1]* All figures and plots in this report are created independently and are not taken from the main reference papers unless explicitly stated otherwise.

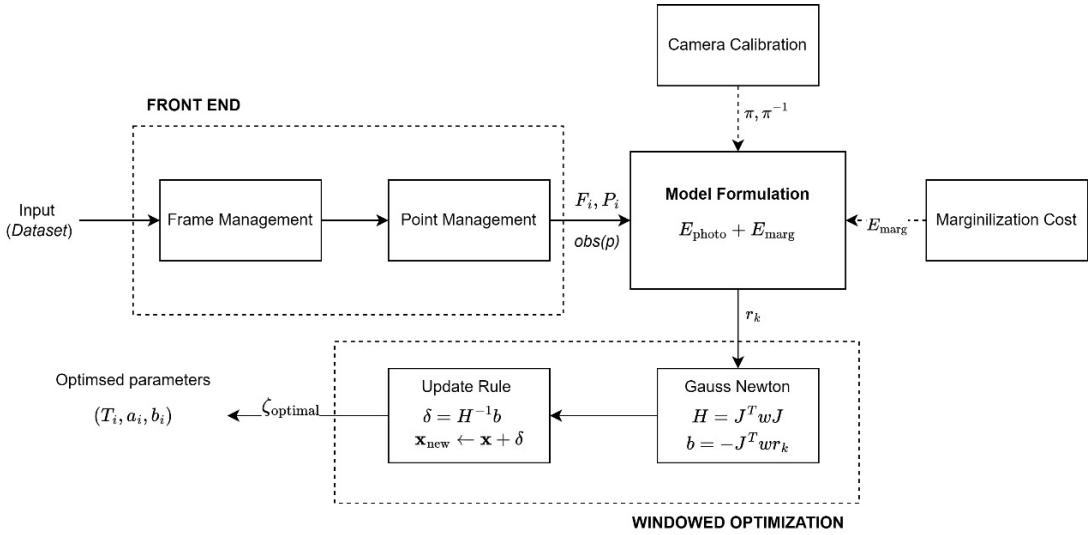

Fig. 1. The above flowchart completely describes the entire architecture, from the front end to the final update step. The process begins with the **Front-End Architecture**, which takes in the frames of the dataset and **outputs the feature points F, the 3D points P_i, and the set of observed points obs(P).** These represent the points and frames after marginalization, which includes **active keypoints and frames**. Next, the **Model Architecture block** takes the results from the **camera calibration model** (both forward and reverse formulas) along with the **marginalization cost**, and returns the residual vector. Finally, the **Windowed Optimization** process uses the Gauss-Newton algorithm to optimize the parameters and outputs the final optimized values. This entire process represents the complete flow of DSO.

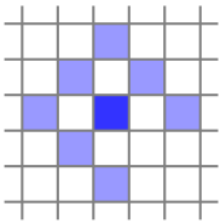

Fig. 2. Residual pattern used in energy computation. The light blue squares represent neighboring pixels considered when calculating the energy for the central dark blue pixel. (DSO)

$$x_i \oplus \mathbf{T}_i := e^{\hat{x}_i} \cdot \mathbf{T}_i. \tag{9}$$

We will use $\zeta \in \mathrm{SE}(3)^n \times \mathbb{R}^m$ to denote **all optimized variables**, including camera poses, affine brightness parameters, inverse depth values, and camera intrinsics.

The current state estimate is hence given by $\zeta = x \oplus \zeta_0$, where x represents the incremental updates and ζ_0 is the evaluation point.

We compute the Gauss-Newton system as:

$$H = J^T W J \tag{10}$$

and

$$b = -J^T W r \tag{11}$$

where $W \in \mathbb{R}^{n \times n}$ is the diagonal matrix containing the weights, $r \in \mathbb{R}^n$ is the stacked residual vector, and $J \in \mathbb{R}^{n \times d}$ is the Jacobian of r.

During optimization – as well as when marginalizing – residuals are always evaluated at the current state estimate, i.e.,

$$r_k = r_k(x \oplus \zeta_0) \; = (I_j[p'(T_i, T_j, d, c)] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[p] - b_i) \tag{12}$$

where $(T_i, T_j, d, c, a_i, a_j, b_i, b_j) := x \oplus \zeta_0$ are the current state variables the residual depends on. The Jacobian J_k is evaluated with respect to an additive increment to x, i.e.,

$$J_k = \frac{\partial r_k((\delta + x) \oplus \zeta_0)}{\partial \delta} \tag{13}$$

It can be decomposed as:

$$\boxed{J_k = \begin{pmatrix} \frac{\partial I_j}{\partial p'} & \frac{\partial p'((\delta+x)\oplus\zeta_0)}{\partial \delta_{\text{geo}}} & \frac{\partial r_k(\delta+x)\oplus\zeta_0}{\partial \delta_{\text{photo}}} \end{pmatrix}} \tag{14}$$

where δ_{geo} denotes the "**geometric**" parameters (T_i, T_j, d, c), and δ_{photo} denotes the "**photometric**" parameters (a_i, a_j, b_i, b_j).

From the resulting linear system, an increment is computed as $\delta = H^{-1}b$ and added to the current state:

$$x_{\text{new}} \leftarrow \delta + x.$$

After each update step, we update ζ_0 for all variables that are not part of the marginalization term, using

$$\boxed{\zeta_0^{\text{new}} \leftarrow x \oplus \zeta_0 \quad \text{and} \quad x \leftarrow x_0}$$

This update step **results in the optimized parameter ζ,** which incorporates the changes from the optimization process.

D. Marginalization and Marginalization Cost - E_{marg}

When the **active set of variables becomes too large**, old variables are removed by marginalization using the Schur complement. When marginalizing frame i, we first marginalize all points in P_i, as well as points that have not been observed in the last two keyframes. Remaining observations of active points in frame i are dropped from the system.

Marginalization proceeds as follows: Let $E^{'}$ denote the part of the energy containing all residuals that depend on state variables to be marginalized. We first compute a Gauss-Newton approximation of $E^{'}$ around the current state estimate $\zeta = x \oplus \zeta_0$. This gives

$$\begin{aligned} E^{'}(x \oplus \zeta_0) &\approx 2(x - x_0)^T b + (x - x_0)^T H (x - x_0) + c \\ &\approx 2x^T (b - H x_0) + x^T H x + (c + x_0^T H x_0 - x_0^T b). \end{aligned} \tag{15}$$

where x_0 denotes the current value (evaluation point for r) of x. The constants c and c_0 can be dropped .

This is a **quadratic function on** x**,** and we can apply the **Schur complement** to marginalize a subset of variables.

Written as a linear system, it becomes:

$$\begin{pmatrix} H_{\alpha\alpha} & H_{\alpha\beta} \\ H_{\beta\alpha} & H_{\beta\beta} \end{pmatrix} \begin{pmatrix} x_\alpha \\ x_\beta \end{pmatrix} = \begin{pmatrix} \beta_\alpha^{'} \\ \beta_\beta^{'} \end{pmatrix},$$

where β denotes the block of variables we would like to marginalize, and α the block of variables we would like to keep. Applying the Schur complement yields:

$$\hat{H}_{\alpha\alpha} x_\alpha = \hat{b}_\alpha^{'}, \quad \text{with}$$

$$\hat{H}_{\alpha\alpha} = H_{\alpha\alpha} - H_{\alpha\beta} H_{\beta\beta}^{-1} H_{\beta\alpha}, \quad \hat{b}_\alpha^{'} = b_\alpha^{'} - H_{\alpha\beta} H_{\beta\beta}^{-1} b_\beta^{'}$$

The residual energy on x_α can hence be written as:

$$\boxed{E^{'}(x_\alpha \oplus (\zeta_0)_\alpha) = 2 x_\alpha^T \hat{b}_\alpha^{'} + x_\alpha^T \hat{H}_{\alpha\alpha} x_\alpha.} \tag{16}$$

This is the marginalization cost E_{marg} (equation 8).

E. Visual Odometry Front-End

1) Frame Management: The front-end of the Direct Sparse Odometry (DSO) algorithm begins by tracking new frames with respect to a set of active reference frames. The set of reference frames, typically limited to N_f (we use $N_f = 7$), is used to track points, and new frames are either discarded or added as keyframes. After a new keyframe is created, the photometric error is optimized, and one or more frames are marginalized to ensure efficient computational resources.

For simplicity, in this report, frame management is discussed briefly, and further details of the process will be elaborated in the final report.

2) Point Management: In this work, point management differs from traditional methods by focusing on a subset of image data, carefully selecting and tracking points based on their gradient magnitude and spatial distribution.

Step 1: Candidate Point Selection. The image is divided into 32×32 blocks, and a region-adaptive gradient threshold is computed for each block:

$$\text{Threshold} = \hat{g} + g_{\text{th}},$$

where $\hat{g}$ is the **median absolute gradient within the block** and g_{th} is a **global constant (set to 7)**. For each block, we select the pixel with the highest gradient if it surpasses the threshold.

Step 2: Candidate Point Tracking. After selecting points, they are tracked in subsequent frames along the **epipolar line by minimizing the photometric error**. From the best match, the **depth and variance** are computed, constraining the search for the next frame. The computed depth serves as an initialization for further tracking.

Step 3: Candidate Point Activation. When old points are marginalized, new point candidates are activated to maintain uniform spatial distribution. All active points are projected onto the most recent keyframe, and new candidates are selected based on maximizing the distance to existing points. Candidates created in the second or third block-run require a larger distance for activation to avoid clustering.

Point management, as described, is briefly outlined here. More details will be discussed in the final report.

III. Datasets

For testing the functionality of the implemented DSO, we utilized two publicly available datasets:

1) **TUM Mono Dataset:** This dataset is available at TUM Mono Dataset.

2) **EUROC MAV Dataset:** The EUROC dataset can be accessed at EUROC MAV Dataset.

We load the images from these datasets along with the corresponding calibration files. Additionally, we have incorporated the **gamma and vignette effects to account for the nonlinearities of the camera**, which helps improve the performance of the system.

IV. Achievements - Low Texture Regions

This section highlights the major advantage of using Direct Sparse Odometry (DSO) over traditional methods, specifically in low-texture regions.

Although traditional SLAM methods rely on **keypoint detection**, which is computationally expensive, they often **fail to identify keypoints in low-texture regions**. However, DSO overcomes this limitation and can successfully track and reconstruct the scene, even in regions with weak intensity gradients, such as mostly white walls.

However, DSO overcomes this limitation by **using intensity variations instead of relying solely on keypoints**. It works by directly using **pixel intensities** from the camera images

to estimate the camera pose and scene structure. This allows DSO to successfully track and reconstruct the scene, even in challenging conditions like mostly white walls with minimal texture.

Refer to Figure 3 and observe the region of interest. Although this region lacks strong gradients, DSO is still able to detect keypoints and process the scene successfully.

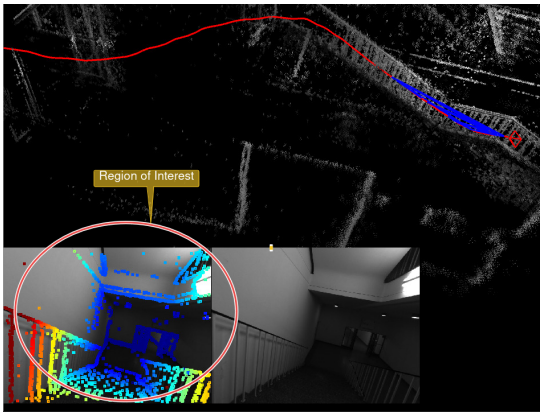

Fig. 3. Example of a low-texture region where DSO successfully detects keypoints and reconstructs the scene.Carefully observe the region of interest .

V. Results and Performance

In this section, we present the results of experiments on various datasets . The complete 3D reconstruction (Mapping) and the camera position and orientation estimates (Localization) are shown from 4 to 6

VI. Issues with DSO and Motivation for DM-VIO

Traditional methods such as Direct Sparse Odometry (DSO) have demonstrated impressive results, but they come with certain limitations that hinder their overall performance, especially when it comes to incorporating inertial data and resolving scale ambiguity in monocular systems.

- **Marginalization:** In traditional methods, once marginalization is performed, it is **irreversible**. Marginalization depends on linearization points, which are approximations of the state variables at the time of processing. However, as new information is incorporated, these linearization points may become outdated, leading to inconsistent priors.
- **Inability to Incorporate IMU Data Directly:** In DSO and similar methods, marginalized states cannot be updated with new data, such as IMU measurements, making it difficult to refine the solution and leverage inertial information effectively.
- **Loss of Scale Information:** A significant limitation of monocular visual odometry systems is that they **cannot inherently estimate the absolute scale of the environment.**

DM-VIO addresses these limitations through several innovations that make it more effective in dynamic environments. The primary motivation for developing DM-VIO is to **incorporate inertial measurement unit (IMU) data** into the visual odometry framework, improving both accuracy and robustness.

- **Delayed Marginalization:** DM-VIO introduces the concept of delayed marginalization, where the marginalization process is deferred using a **second factor graph**. This allows the system to delay marginalization and update the factor graph with new information, keeping the linearization points consistent and improving system accuracy over time.
- **Scale Observability:** Unlike monocular systems, which struggle with scale ambiguity, IMU data provides **acceleration measurements** that enable the system to directly observe scale. This crucial information helps resolve scale ambiguity, leading to more accurate trajectory estimation.
- **Gravity Alignment:** IMU data also enables precise estimation of the gravity direction, which is typically challenging to estimate from purely visual data. This allows us to estimate the orientation better .

These advancements make DM-VIO a powerful solution to the challenges faced by traditional monocular visual odometry methods.

VII. Initialization in DM-VIO and Proposed Strategy

One of the key challenges in integrating IMU data into visual odometry systems is the inability of the IMU to provide acceleration information during **constant velocity motion**. This limitation makes it difficult to accurately estimate scale, often resulting in **degenerate cases** where the system fails to infer the correct scale.

Requires Varied Motion: Accurate initialization of the IMU requires varied motion, including accelerations and rotations, to ensure that scale becomes observable.

If the motion trajectory lacks sufficient variation, **initialization can take an arbitrarily long time**. Moreover, prematurely initializing the IMU without adequate motion data can lead to incorrect scale estimation or introduce biases into the system. Therefore, **both extremes must be avoided** to ensure a stable and reliable initialization process.

Proposed Strategy in DM-VIO:

- **Step 1:** Start with a visual-only system and run an IMU initializer in parallel.
- **Step 2:** Continue optimizing scale and gravity direction in the main system after IMU initialization.
- **Step 3:** Use delayed marginalization to address the challenges of initialization and scale changes.

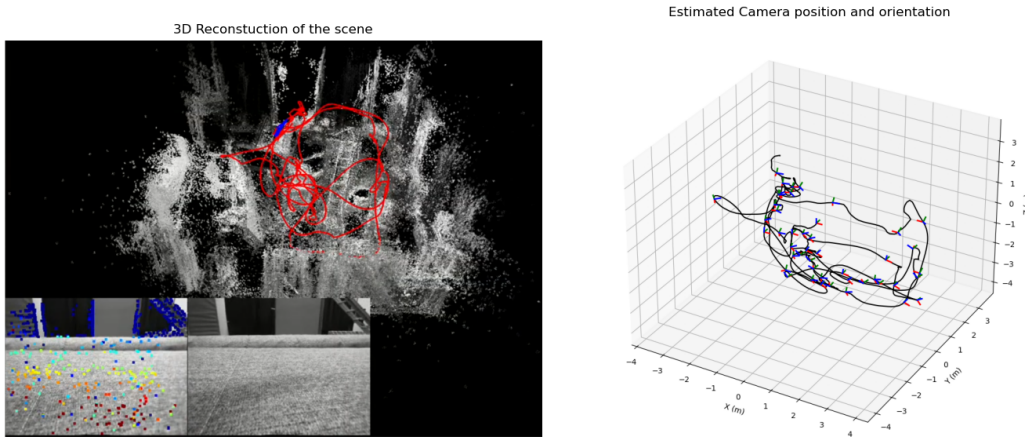

Fig. 4. In this visualization, we used the **EUROC V2-02-medium** dataset. Observe the complete 3D reconstruction of the scene and the estimated camera trajectory and position. The estimation is not perfectly accurate as we are plotting **every 1 in 10 frames** for computational efficiency. However, this serves as a good visualization to understand the estimation. The complete run of the scene is available at EUROC Dataset and the estimated camera positions are available at Estimated Camera Positions.

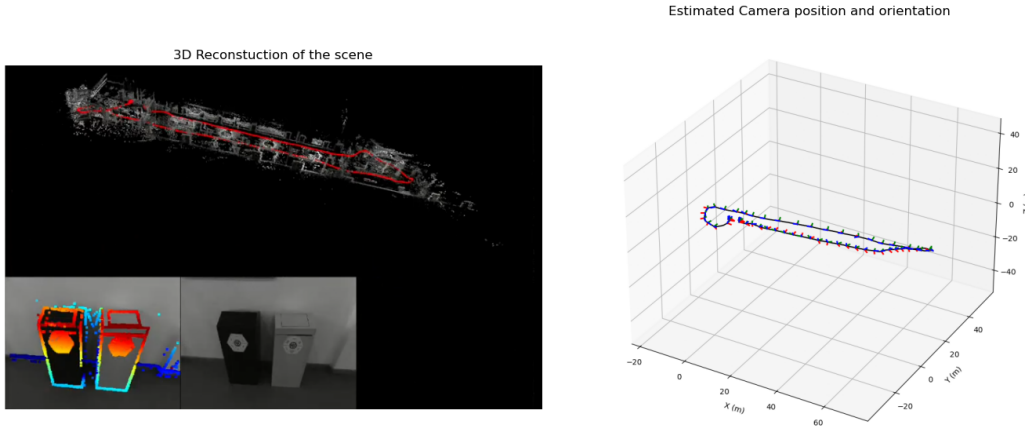

Fig. 5. **Sequence 40 from the TUM dataset**. The complete run is available at TUM Sequence 40 and the estimated camera positions are available at Estimated Camera Positions.

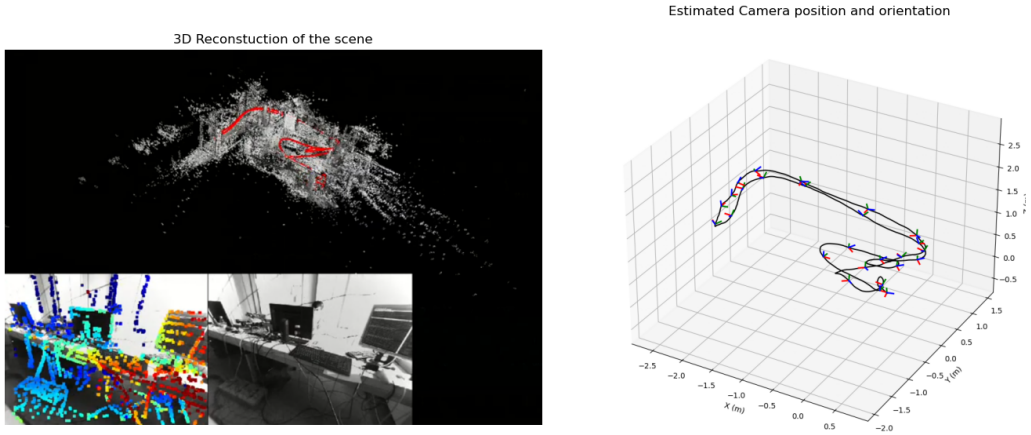

Fig. 6. **Sequence 14 from the TUM dataset**. The complete run is available at TUM Sequence 14 and the estimated camera positions are available at Estimated Camera Positions.

VIII. MODEL ARCHITECTURE: DM-VIO

The core of DM-VIO lies in the **visual-inertial bundle adjustment** performed for all keyframes. Similar to traditional approaches, we jointly **optimize both visual and IMU variables** within a combined energy function. This formulation is very similar to the cost function used in DSO, with the key difference being the inclusion of IMU information, which allows for better integration of motion data and refinement of the scale and trajectory.

The energy function in DM-VIO consists of photometric cost, the IMU cost, and the prior cost, as shown by:

$$\boxed{E_s = W(e_{photo}) \cdot E_{photo} + E_{imu} + E_{prior}} \tag{17}$$

A. Optimization Variables

We optimize both the scale and gravity direction as explicit variables.

The Visual Frame (V): The **visual frame (V)** is the coordinate system used by the camera. It has an ***arbitrary scale*** **and *rotation***. In the visual frame, motion is tracked based on image features, and this frame is used to estimate the *relative motion* between keyframes.

The IMU Frame (I): The **IMU frame (I)** is the coordinate system used by the *Inertial Measurement Unit (IMU)*. The IMU frame has a ***metric scale*** **(i.e., it has a well-defined, consistent scale), and its *z-axis*** is aligned with the direction of gravity.

To accommodate both sources of information in a meaningful manner, we need to ensure that **both coordinate systems are aligned.** Therefore, we need to find the rotational matrix between them.We are trying to optimize the scale s and the rotational matrix R_{V-I}

$$\boxed{\begin{aligned} P_i^I &= T^I_{w \leftarrow \text{imu}_i} = \Omega(P_i^V, S, R_{V-I}) \\ &= (R_{V-I})^{-1} S_{I \to V} (P_i^V)^{-1} (S_{I \to V})^{-1} T_{cam-imu} \end{aligned}} \tag{18}$$

- R_{V-I} represents the rotational matrix between the visual and IMU frames. It is the **primary variable being optimized** in the process.
- $S_{I \to V}$ is constructed using an identity rotation matrix and scaled (by s) translation. This formulation helps to **optimize the scale parameter**, which is crucial for aligning the two coordinate systems.
- $P_i^V = T_{\text{cam}_i \to \text{W}}$ denotes the **transformation from the camera frame** i **to the world frame** in the visual coordinate system. This transformation is used to relate the camera pose to the world frame.

The full state optimization problem is formulated as:

$$\boxed{s = \{s, R_{V-I}\} \cup \{s_i\}_{i \in F}} \tag{19}$$

where s_i represents the states for all active keyframes, defined as:

$$s_i = \{P_i^V, v_i, b_i, a_i, b_i, d_i^0, d_i^1, \ldots, d_i^m\}$$

In this equation:

- v_i denotes the velocity of the keyframe,
- b_i represents the bias associated with the keyframe,
- a_i and b_i are affine brightness parameters,
- d_i^j corresponds to the inverse depths of active points associated with the keyframe. The optimization process jointly adjusts all these parameters.

In this case, we optimize not only the parameters between frames but also the **scale parameter and the rotational relationship between the visual and inertial units**.

B. Photometric Cost: E_{photo}

E_{photo} is exactly the same as the one derived in Equation 7.

However, when the image quality is poor (e.g., due to motion blur, poor lighting, or low texture), the photometric residuals become large. Traditional photometric cost functions may inadvertently give more weight to visual data in such cases, even though it is less reliable.

The dynamic photometric weight is computed using the root mean squared photometric error:

$$e_{photo} = \sqrt{\frac{E_{photo}}{n_{residuals}}}$$

The weight $W(e_{photo})$ is defined as:

$$\boxed{W(e_{photo}) = \begin{cases} \lambda \cdot \left(\frac{\theta}{e_{photo}}\right)^2, & \text{if} \quad e_{photo} \geq \theta \\ \lambda, & \text{otherwise} \end{cases}}$$

Where E_{photo} is the total photometric error, and $n_{residuals}$ is the number of residuals, which **should be high** as it indicates that the system has detected and matched a large number of features across frames.

When $e_{photo} < \theta$, the weight remains constant at $W = \lambda$, allowing photometric data to contribute fully. On the other hand, when $e_{photo} > \theta$, the weight decreases, effectively **reducing the influence of photometric residuals** and increasing the reliance on IMU data.

In our case, we have set $\theta = 8$.

C. IMU Cost: E_{imu}

The IMU state s_i^I is defined as $s_i^I = \{P_i^I, v_i, b_i\}$, where: - P_i^I is the pose in the IMU frame, - v_i is the velocity, - b_i is the bias.

1) Delayed Marginalization: **Pre-Integration** refers to the process where IMU measurements between two keyframes are integrated to **predict the relative motion of the system**. By using the IMU data, we can predict the relative motion (position and velocity) between consecutive keyframes, even in the absence of direct measurements.

Using the preintegration technique, we can **predict the next state** $\hat{s}_j^I$ **based on the previous state** s_i^I. The predicted state $\hat{s}_j^I$ is computed using the preintegration data, which includes both position and velocity updates. Additionally, the uncertainty associated with the predicted state is represented

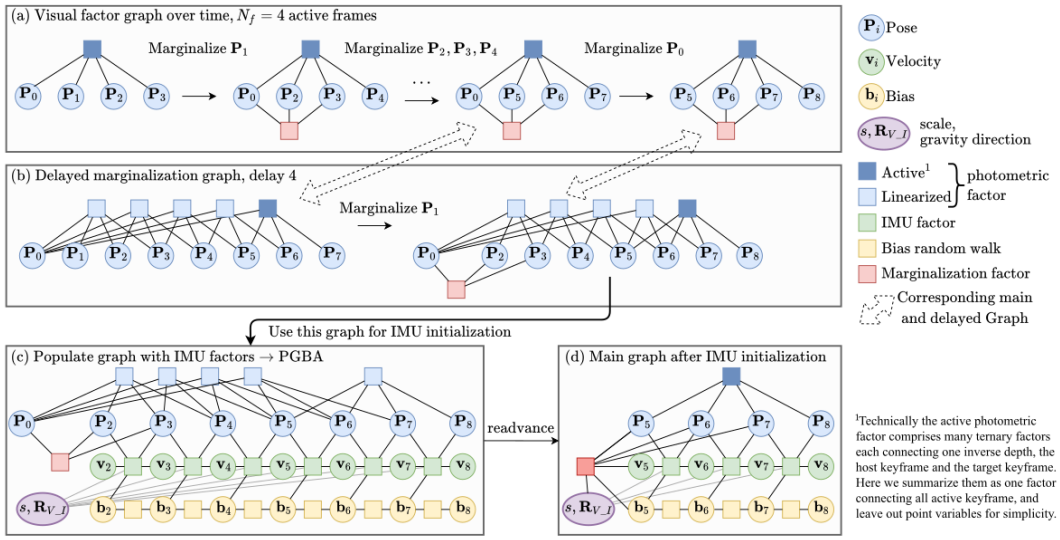

Fig. 7. a) Normal marginalization in the visual graph. Note that not always the oldest pose is marginalized. b) Delayed marginalization: We marginalize all variables in the same order as the main graph, but with a delay d (in practice d = 100). Marginalization in this graph is equally fast as marginalizing in the main graph c) For the pose graph bundle adjustment (PGBA) we populate the delayed graph with IMU factors. This optimization leverages the full photometric uncertainty. d) We readvance the marginalization in the graph used for PGBA to obtain an updated marginalization prior for the main system. This transfers inertial information from the initializer to the main system. DM - VIO

by the covariance matrix $\hat{\Sigma}_j$, which quantifies the confidence in the prediction.

The inertial error function penalizes deviations of the current state estimate from the predicted state. It is defined as follows:

$$E_{\text{imu}}(s_i^I, s_j^I) := (\hat{s}_j^I - s_j^I)^T \hat{\Sigma}_j^{-1} (\hat{s}_j^I - s_j^I) \tag{20}$$

In this context, $s_i^I - s_j^I$ represents the Lie subtraction, which is computed as $\log(R_i R_j^{-1})$, where:

- $(\hat{s}_j^I - s_j^I)$ represents the difference between the predicted state $\hat{s}_j^I$ and the current state s_i^I, applying the Lie subtraction operator.
- $\hat{\Sigma}_j^{-1}$ denotes the inverse of the covariance matrix associated with the predicted state $\hat{s}_j^I$. This matrix encapsulates the uncertainty in the prediction. **A larger covariance (indicating higher uncertainty) results in a smaller weight for the error term, reducing its influence on the optimization.** This reflects a lower confidence in the prediction and adjusts its contribution to the overall optimization accordingly.

D. Prior Cost: E_{prior}

Using the Schur complement as in the case of DSO, we **do not always marginalize the oldest pose**. Instead, a combination of newer and older poses is retained, provided they remain within the field of view as discussed in section II-E

However, **reverting marginalization** of a set of variables is inherently infeasible without redoing the entire marginalization process.

1) Delayed Marginalization: Marginalization, once performed, is irreversible. However, **delayed marginalization** offers a mechanism to postpone this process, allowing for greater flexibility in optimization. An example of delayed marginalization is illustrated in Figures 7a and 7b, demonstrating its application and benefits.

- In addition to maintaining the *normal marginalization prior*, a *secondary, delayed marginalization prior* and its corresponding factor graph are also preserved.
- In the delayed graph, marginalization of frames is deferred by a **delay parameter**, d.
- During this delay, marginalization is performed exclusively on the main graph and not on the delayed graph.
- Once the delay is reached, the first pose is marginalized in the delayed graph. Marginalization then proceeds at the **same pace and in the same order** as in the main graph.

Delayed marginalization helps preserve temporal information by deferring the removal of older poses, allowing them to **contribute longer to the optimization process and resulting in more accurate pose estimations**. It minimizes error accumulation by avoiding premature marginalization, thus reducing the loss of information and preventing the propagation of errors into subsequent frames. This approach enhances trajectory optimization by retaining **correlations between poses for a longer duration**, which is crucial for our specific optimization tasks.

In our case , we used a **delay of 100** to balance both computational complexity and the accuracy of performance.

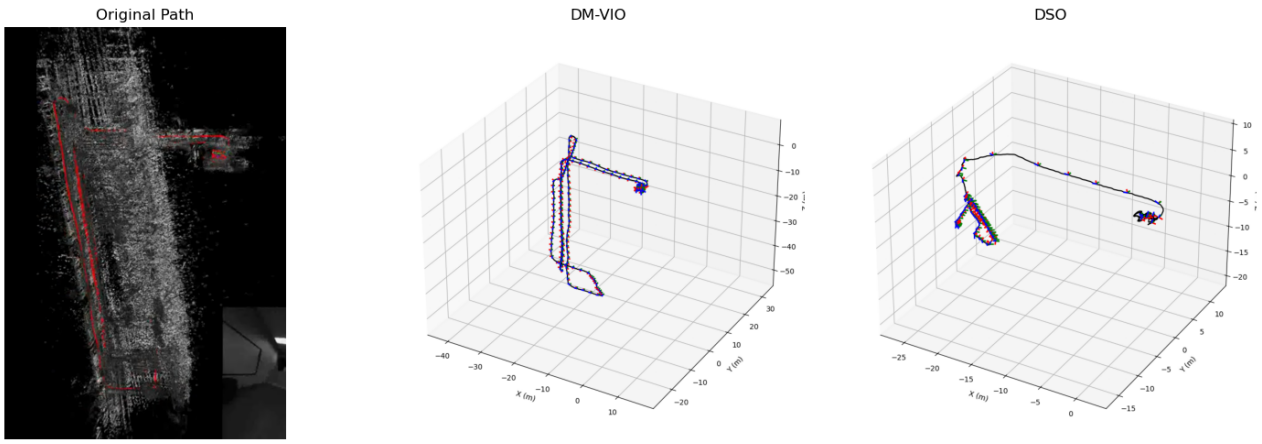

Fig. 8. This figure clearly highlights the advantages of using DM-VIO over DSO. The **loop closure in DM-VIO effectively corrects the trajectory**, whereas in the case of DSO, the estimate deviates significantly from the original path. This demonstrates the superiority of DM-VIO in maintaining accurate trajectory estimation.

2) Pose Graph Bundle Adjustment: The process of Pose Graph Bundle Adjustment (PGBA), depicted in Figures 7c and 7d, **incorporates IMU information into the optimization framework**. By populating the graph with IMU factors, we optimize the system to account for both visual and inertial uncertainties.

A significant advantage of utilizing delayed marginalization is the **ability to optimize over the full uncertainty space**, resulting in improved accuracy. Furthermore, delayed marginalization enables an additional critical operation: the graph can be **re-advanced.**The primary purpose of re-advancing is to generate a **marginalization prior**, such as $\mathbf{E_{photo}}$ for the main system. This prior encapsulates all the visual and inertial information accumulated up to a specific point in time.

A marginalization prior is a compact representation of the contributions made by variables that have been marginalized (removed) from the active optimization problem. Rather than **retaining all the variables in the optimization graph**, the **influence of the marginalized variables is preserved** in the form of a prior, which serves as a constraint on the remaining active variables.The marginalization prior **ensures that past observations and their contributions to the optimization process are not lost**, even after the variables have been removed.

IX. Datasets: DM-VIO

To evaluate the functionality of the implemented DM-VIO system, we utilized the following dataset:

- **EUROC TUMVI Dataset:** The EUROC dataset, which provides synchronized visual-inertial data, was used for our experiments. The dataset can be accessed at EUROC TUMVI Dataset.

We conducted evaluations on multiple sequences from this dataset. Many commonly used datasets lack this critical IMU information, making them unsuitable for testing the DM-VIO system.

X. Achievements - Loop Closure and Scale Ambuiguity

This section emphasizes the primary advantage of using DM-VIO over traditional methods, particularly in the context of loop closure regions.

As demonstrated in Figure 8, **loop closure** occurs effectively in DM-VIO, which leads to significant corrections in the estimated trajectory.

Furthermore, we observe that the **scale ambiguity** is resolved in the case of DM-VIO, as it accounts for the scale parameter directly in the main system calculations. This is clearly demonstrated in the figure below:

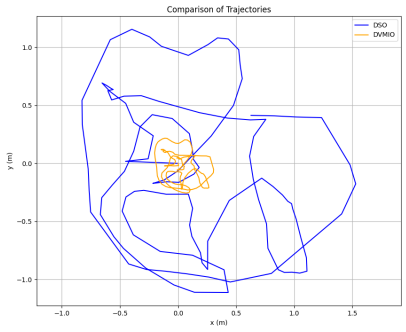

Fig. 9. Estimated trajectories for DSO (blue) and DM-VIO (yellow). The figure clearly shows that the scale ambiguity is effectively resolved in DM-VIO.

XI. Results and Performance : DM - VIO

In this section, we present the results of experiments on the EUROC TUMVI dataset as shown below :

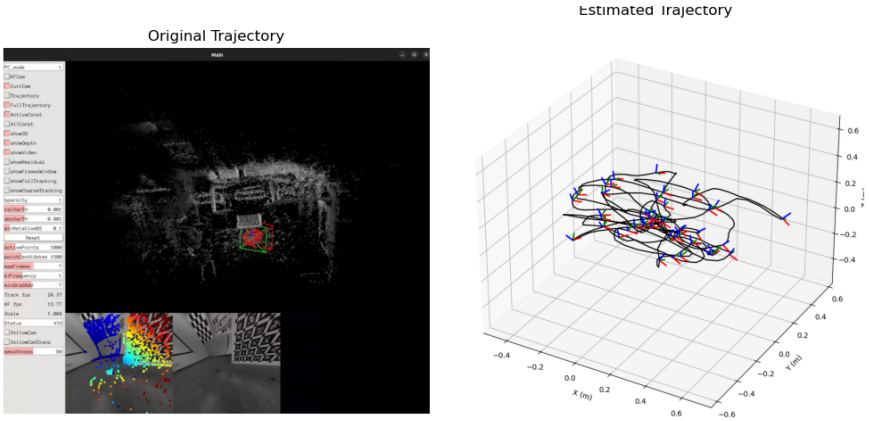

Fig. 10. In this visualization, we used the **Calib-cam-1** scene from the EUROC TUMVI Dataset.Observe the complete 3D reconstruction of the scene and the estimated camera trajectory and position. The estimation is not perfectly accurate as we are plotting **every 1 in 10 frames** for computational efficiency. However, this serves as a good visualization to understand the estimation. The complete run of the scene is available at Calib - cam - 1 and the estimated camera positions are available at Estimated Camera Positions.

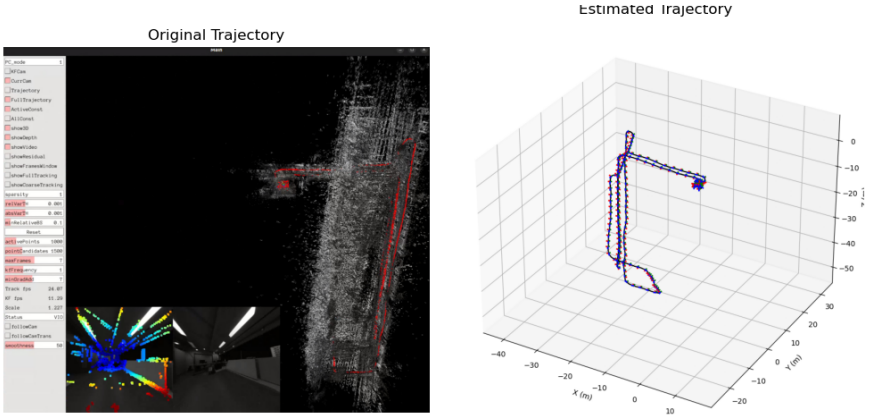

Fig. 11. **Magistrale5 -512 -16 scene from the EUROC TUMVI dataset**. The complete run is available at magistrale5 -512 -16 and the estimated camera positions are available at Estimated Camera Positions.

. The estimated trajectory is just flipped vertically for better visualisation .

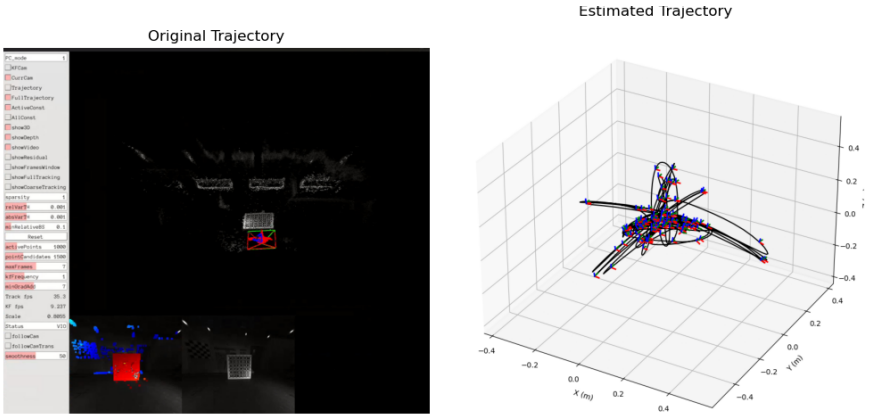

Fig. 12. **Calib-imu1-512-16 from the EUROC TUMVI dataset**. The complete run is available at Calib-imu1-512-16 and the estimated camera positions are available at Estimated Camera Positions.

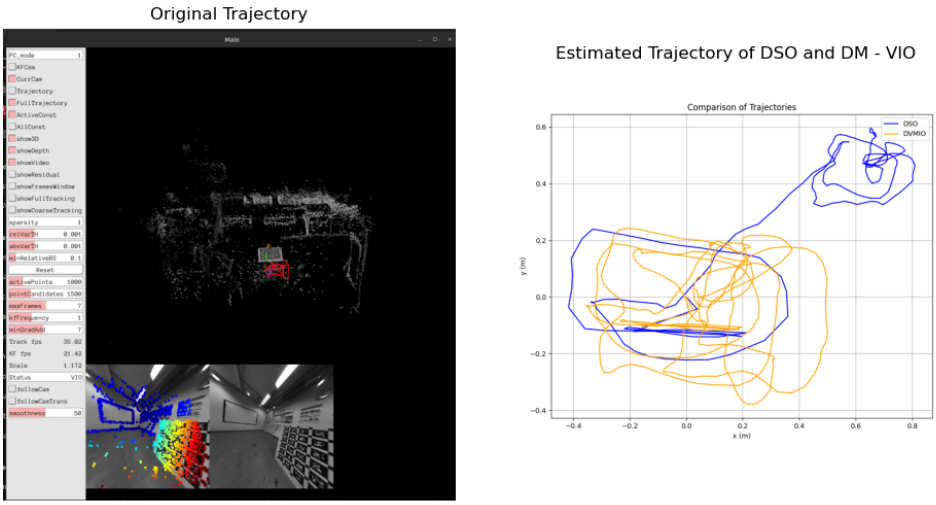

Fig. 13. In this visualization, we used the **Calib-cam-5** scene from the EUROC TUMVI Dataset. The figure shows a complete 3D reconstruction of the scene, along with the estimated camera trajectories for both DSO (blue) and DM-VIO (yellow). It is evident that there is **no straight motion in the ground truth**, but DSO estimates this incorrectly, while DM-VIO optimizes the trajectory using **loop closure and bundle adjustment**.

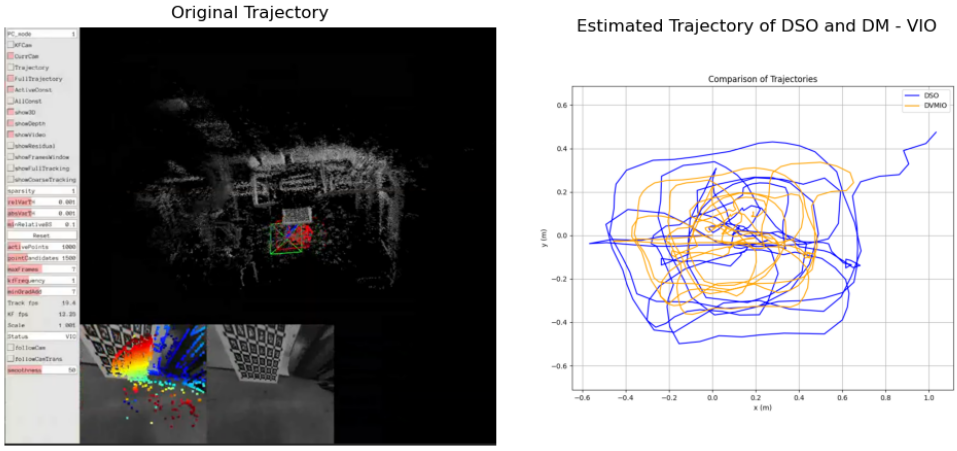

Fig. 14. **Cam-1 from the EUROC TUMVI dataset**. In this case, observe how DM-VIO **handles scale ambiguity** effectively, which results in a more accurate trajectory compared to DSO.

XII. Comparing and Benchmarking DSO and DM-VIO

In this section, we compare the results of DSO and DM-VIO on the same datasets. We overlay the original trajectory of the fastest path and the estimated trajectories for DSO (in blue) and DM-VIO (in yellow).

Clearly, as shown in the visualizations from 13 and 14 , DM-VIO performs better than DSO in all cases, achieving more accurate trajectory estimates by compensating for scale ambiguity and applying optimizations like loop closure and bundle adjustment. The performance of DM-VIO is a significant improvement over DSO, which is evident from the comparison of the estimated trajectories.

To objectively compare the performance of DSO and DM-VIO, we plotted the **Root Mean Squared Error (RMSE)** of both methods against the ground truth for three different scenes from the dataset. The results are as shown in Figure 15

XIII. Contributions

1. **Aniruth Suresh**: Contributed to the setup and implementation of Direct Sparse Odometry (DSO) and DM -VIO and played a key role in preparing the report .

2. **Aryan Garg**: Explored and benchmarked various versions of DSO to identify performance improvements and introduced the concept of DM-VIO for future exploration.

3. **Pratyush Jena**: Led the setup of DSO and DM - VIO and made significant contributions to both the implementation and the report (Results section) .

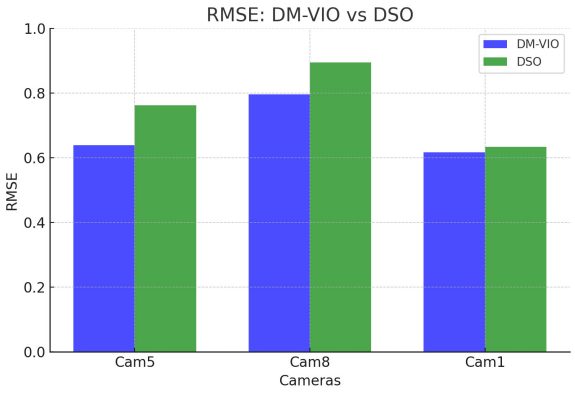

Fig. 15. Bar graph depicting the RMSE error for both DSO and DM-VIO across three scenes from the dataset.

XIV. Troubleshooting

During the implementation and setup of Direct Sparse Odometry (DSO), we encountered several challenges and bugs. Below are some of the major issues we faced:

1. **Pangolin Setup**: The Pangolin setup was incomplete, and the wrapper was not fully functional. As a result, we had to modify certain functions and configure it specifically for our use case.

2. **Compilation Issues**: While building the main codebase, we encountered issues such as missing OpenGL dependencies, frame buffer errors, and segmentation faults. To resolve these, we linked the correct version of OpenGL libraries, updated the CMakeLists file, and added constraints to ensure a minimum number of active points to prevent segmentation faults.

3. **Visualization**: To produce effective visualizations for mapping and localization, we developed several helper functions to visualize the camera trajectory and reconstructed map.

During the implementation and setup of DM - VIO, we encountered several challenges and bugs. Below are some of the major issues we faced:

1. **Data Association Issues**: One of the key challenges was the **occurrence of false data associations** in certain datasets. This was caused by errors in the processing pipeline, leading to incorrect matching of nodes. In such cases, the loop closure process would attempt to correct the trajectory drift, which in turn caused significant changes to the entire graph structure. To mitigate this, we implemented a **thresholding mechanism** to ensure that only reliable data associations were used in the optimization process, minimizing drift and improving the accuracy of the graph optimization.

2. **Computational Complexity**: Due to the integration of both visual and inertial data, the optimization process in DM-VIO tends to be computationally intensive. This led to performance bottlenecks, particularly when processing large datasets with many keyframes. To overcome this, we optimized the graph structure by pruning unnecessary nodes and using more efficient numerical solvers for the optimization process. Delay was just set to 100 instead of a very high value .

XV. Conclusion

In this work, we have **successfully implemented Direct Sparse Odometry (DSO)** and demonstrated its unique advantages over traditional methods, particularly in terms of efficiency and robustness in low-texture environments. The system exhibited strong performance in real-time visual odometry and 3D reconstruction tasks, highlighting its effectiveness in dynamic settings with limited feature points.

Furthermore, we explored and benchmarked the performance of **DM-VIO** against the standard DSO framework. Our analysis showed that DM-VIO provides significant improvements, particularly by **incorporating loop closure and bundle adjustment techniques**, which effectively correct the trajectory drift and enhance the overall pose estimation. The ability to handle **scale ambiguity and leverage both visual and inertial data o**ffers a substantial advantage over DSO in certain conditions. The comparative results confirm that DM-VIO is a promising alternative, especially in challenging environments where traditional methods like DSO may struggle.

In summary, both DSO and DM-VIO have demonstrated their respective strengths in visual odometry tasks, and the integration of inertial data through DM-VIO offers notable improvements in robustness, scale estimation, and trajectory correction.

References

[1] Direct Sparse Odometry, J. Engel, V. Koltun, D. Cremers 1
[2] A Photometrically Calibrated Benchmark For Monocular Visual Odometry, J. Engel, V. Usenko, D. Cremers 2
[3] DM-VIO: Delayed Marginalization Visual-Inertial Odometry Lukas von Stumberg , Daniel Cremers 3
[4] KITTI Dataset KITTI
[5] ASL Datasets EUROC
[6] DM-VIO: Delayed Marginalization Visual-Inertial Odometry :Lukas von Stumberg, Daniel Cremers DM - VIO