2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

# Tightly-Coupled Dynamic Object Tracking and RGB-D Inertial Odometry Estimation with Dual Quadrics

Toyozo Shimada[1], Kenji Koide[2], Aoki Takanose[2], Shuji Oishi[2], Masashi Yokozuka[2], and Jun Miura[1]

***Abstract*— This paper presents a robust RGB-D Inertial-Object odometry estimation framework for dynamic environments. The proposed method represents and tracks the surrounding objects as dual quadrics and estimates sensor pose and object parameters (pose, shape, and velocity) by jointly minimizing the depth point cloud registration factor, IMU preintegration factor, and visual object detection factor (bounding box factor). The use of the dual quadric representation enables handling detection of arbitrary objects (e.g., pedestrians, chairs and boxes) in a unified optimization framework. By continuously tracking objects across multiple frames, we identify points belonging to dynamic objects and remove them from the input point cloud. Although this point removal process inherently drops information in the input point cloud, the tight coupling of the object detection factor and the point cloud registration factor mitigates accuracy degradation. The experimental results showed that the proposed method enables robust and accurate odometry estimation in extremely dynamic situations, cases where a large part of the sensor view is occupied by walking pedestrians.**

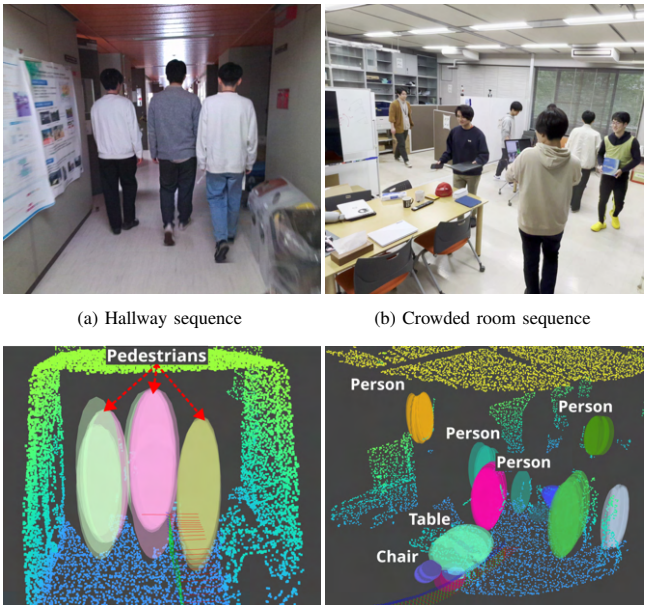
***Index Terms*— SLAM, ObjectSLAM, Odometry, Dynamic Environment**

## I. INTRODUCTION

Robust odometry estimation is one of the essential abilities for autonomous mobile robots. In recent years, the fields of Visual SLAM and Visual Odometry have made remarkable progress [1], [2]. In particular, methods utilizing RGB-D cameras have gained widespread adoption, enabling high-precision ego-motion estimation by providing dense geometric information at a low cost. However, the success of many of these techniques hinges on a foundational assumption: that the observed environment is static. In real-world scenarios, this assumption is frequently and easily violated by the presence of dynamic objects, such as pedestrians. In visual-based approaches [1], keypoints on moving objects do not follow the rigid motion of the sensor, causing their matches to become geometric outliers, which in turn degrades the accuracy and consistency of the final pose estimate. For registration-based methods [3][4], dynamic objects can cause significant geometric changes and reduce the overlap between point cloud pairs, making the registration process unstable and inaccurate.

The most common approach to deal with dynamic objects in RGB-D odometry is to naively detect and remove dynamic regions from the input, then estimate motion only from the remaining static information [5]. However, it inherently discards a large amount of surrounding information and suffers from degradation of accuracy and robustness in situations where many dynamic objects exist. In a highly dynamic environment where a lot of objects are moving and blocking a large part of the view of the sensor, only a small set of static measurements remains. Hence, accurate keypoint matching or depth point cloud registration become difficult due to degeneracy.

Object-based SLAM is an approach that treats surrounding objects as landmarks, and generates an object-oriented map. Among many object pose and shape representations, the dual quadric representation [6] has been gaining popularity recently. The dual quadric is an ellipsoid in 3D space that has nine degrees of freedom for representing the position, orientation and scale of an object. This representation has two major advantages. First, it enables handling arbitrary objects in a unified framework with a reasonable approximation of the object shape as an ellipsoid. Second, it can be projected




(a) Hallway sequence (b) Crowded room sequence

(c) Pedestrians in hallway (d) Objects in crowded room

Fig. 1: Object detection and tracking results obtained by the proposed method in dynamic indoor environments. Hallway sequence: (a) walking pedestrians occupy most of the sensor view; (c) dual quadric estimates. Crowded room sequence: (b) both people and a variety of other objects (e.g., tables and chairs) move; (d) results demonstrating our method's ability to track arbitrary objects.

This work was supported by the Japan Science and Technology Agency (JST) K Program, Grant Number [JPMJKP23G2].

[1]Toyozo Shimada and Jun Miura are with Department of Computer Science and Engineering, Toyohashi University of Technology, Toyohashi, Aichi, Japan, shimada.toyozo.lf@tut.jp

[2]Kenji Koide, Aoki Takanose, Shuji Oishi, and Masashi Yokozuka are with National Institute of Advanced Industrial Science and Technology, Tsukuba, Ibaraki, Japan

from the 3D space to the image space as a 2D bounding box in a differentiable way. This allows us to bridge 2D bounding box observations provided by contemporary visual object detectors (e.g., YOLOv11[7]) and factor graph-based sensor and object state estimation.

In this study, we developed a tightly coupled RGB-D, inertial and object odometry estimation method with dynamic object tracking. We adopted the dual quadric [6] to represent objects. Due to its flexibility and ease of 2D-3D data association, this representation allows us to keep tracking the pose, shape, and velocity of arbitrary surrounding objects over time. Based on the tracking result, we adaptively remove points belonging to dynamic objects with large velocities in the input depth point cloud. Then, we minimize the objective function composed of object observation, object movement, depth point cloud registration, and IMU constraints on a sliding window factor graph to simultaneously estimate the states of the sensor and surrounding objects. To ensure real-time processing, we employ a GPU-accelerated point cloud registration factor [8] and an incremental factor graph optimizer (iSAM2 [9]).

The main contribution of this work is three-fold:

- We propose an object-centric RGB-D odometry estimation method that represents objects as dual quadrics and handles both stationary and dynamic objects within a unified framework.
- By introducing constraints on object movement and estimating the velocity of objects, this system continuously tracks and identifies dynamic objects, and removes their corresponding area from the input depth point cloud.
- We show that the our factor graph structure based on the tight coupling of the point cloud registration factor, the IMU factor, and the visual object detection factor enables robust estimation under the existence of many dynamic objects.

## II. RELATED WORK

### A. Object-based SLAM

While conventional SLAM methods that build a map using geometric features of the environment such as visual feature points and LiDAR point clouds have been a common approach, object-based SLAM that use surrounding objects as landmarks has also been actively researched in recent years. In pioneering work, Salas-Moreno et al. proposed SLAM++ [10] to incorporate the tracking of predefined surrounding objects in the SLAM problem. While this system detects objects by matching RGB-D data with predefined CAD models, it can use only a limited variety of landmarks (5 classes) due to requirements associated with the CAD models. Thus, a more generic way to represent the pose and shape of objects was required to achieve a more generic SLAM system. Yang et al. [11] employed 3D cuboids as an object representation that did not require predefined 3D models. This approach constructs a 3D cuboid from 2D object bounding boxes using the vanishing point, and then jointly optimizes both the sensor pose and object pose using ORB feature points. QuadricSLAM [6] uses the dual quadric paradigm as a unified object representation. This approach allows us to directly optimize the sensor pose and object parameters by projecting the dual quadrics in the image space and generating bounding boxes fitted to them in a differentiable way. Wang et al. proposed VOOM [12], which uses dual quadrics and ORB features as landmarks leveraging the projected dual quadric regions to constrain and thereby reinforce ORB feature correspondences for more reliable pose refinement.

In the present work, we developed an odometry estimation method that jointly considers both object landmarks and geometric features. With the dual quadric representation, we can detect and track arbitrary objects through factor graph optimization and use the tracking results to filter outliers from the input point cloud.

### B. Tightly Coupled Sensor Fusion with IMU

Inertial information plays an important role in robust estimation of sensor motion. The tight coupling approach directly considers the consistency between raw-level observations of each sensor. In the context of visual-inertial SLAM, VINS-Mono [2] formulates a tightly coupled estimator that jointly minimizes IMU preintegration and feature reprojection residuals while estimating IMU biases. LiDAR-IMU tight coupling approach, which jointly minimizes IMU and point cloud registration errors on a unified objective function, has been widely studied due to its robustness and accuracy. Early work on tightly coupled LiDAR-IMU odometry based on an iterated Kalman filter, LINS was presented in [13]. More recent works employed sliding window-based optimization approaches [8] [14] that were more robust than those based on the iterated Kalman filter because of the repeated re-linearization of past sensor states [15].

Our method is built on recent sliding window-based methods by introducing surrounding object parameters as additional estimation subjects. In addition to depth point cloud matching and inertial information, visual object information is added, and all of them are jointly minimized on a single factor graph.

### C. Dynamic SLAM

Since many existing SLAM methods assume a static scene, estimation degrades when this assumption is violated. A common way to mitigate the effect of dynamics is to detect dynamic regions and remove their measurements before motion estimation. DS-SLAM [5], implemented on ORB-SLAM2 [16], combines semantic segmentation network (SegNet [17]) with motion consistency checking to reject features on moving objects and generate a dense semantic map. V3D-SLAM [18], also built on ORB-SLAM2, lifts segmented regions into 3D, uses geometric cues to classify them as static or dynamic, and masks the dynamic regions. Another line of work leverages tightly coupled visual-inertial estimation to suppress inconsistent measurements without explicit object models. DynaVINS [18] jointly optimizes

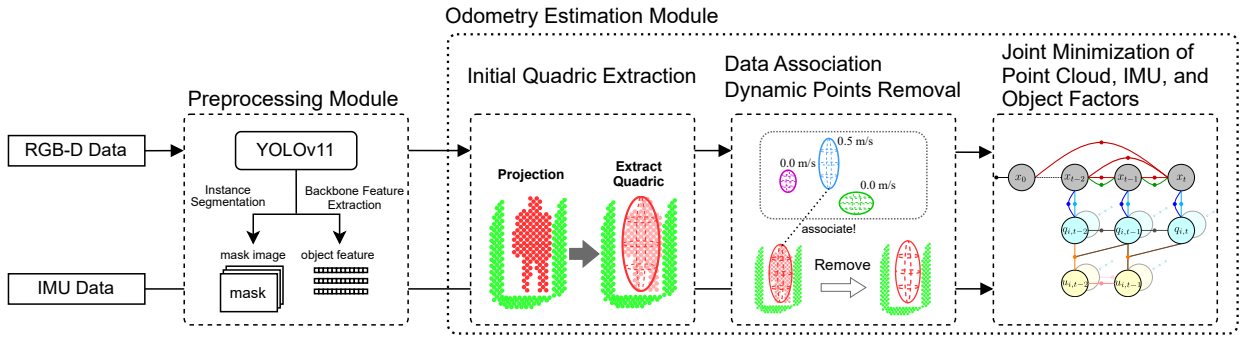



Fig. 2: Overview of proposed framework. For the given RGB image, object mask images and visual features are extracted as preprocessing. In odometry estimation, the point cloud and object mask images are fused to extract an initial dual quadric for each detected object. They are then associated with object landmarks (tracking instances) based on geometrical and appearance similarities, and the points belonging to dynamic objects are removed from the input point cloud. The states of the sensor and objects are then estimated through a sliding window factor graph optimization.

feature reprojection and IMU preintegration in a single window and down-weights features that are inconsistent with the inertial prediction, which improves robustness under heavy dynamics.

In the proposed method, we extract surrounding objects based on robust RGB image segmentation and depth-aided object instance tracking. Furthermore, we also use all surrounding objects as dynamic landmarks that aid robust odometry estimation in the presence of severe feature degeneration.

## III. PROPOSED METHOD

### A. System Overview

Fig. 2 shows an overview of the proposed system. This system takes as input RGB image $\mathcal{I}_t$, depth-derived point cloud $\mathcal{P}_t$, and IMU data and simultaneously estimates the states of the sensor and surrounding objects.The system first applies instance segmentation to the input RGB image using YOLOv11 [7], to obtain object masks $\mathcal{I}_{i,t}$ and bounding boxes $\boldsymbol{B}_{i,t}$, and we extract a backbone feature map $\boldsymbol{F}_t$. We compute a visual feature vector $\boldsymbol{f}_{i,t}$ by applying RoIAlign [19] to the backbone feature map $\boldsymbol{F}_t$ at the region specified by $\boldsymbol{B}_{i,t}$.The extracted object bounding box and visual feature vector paired with the point cloud are then fed to the odometry estimation module.

The odometry estimation module first roughly estimates quadric parameters $\boldsymbol{q}_{j,t}^{in}$ of each detected object by combining the 2D bounding box and the point cloud in the corresponding region. These quadrics are then associated with tracked object instances based on geometrical (motion prediction) and appearance (visual feature comparison) verifications. Points associated with an object with a large velocity are then removed from the input point cloud. Next, the states of the sensor and surrounding objects are estimated through a sliding window factor graph optimization with tightly coupled point cloud registration, IMU, and visual object bounding box factors. The system maintains a simplified object-oriented map for short-term object tracking. Note that we focused on robust and accurate local odometry estimation in this work, and thus the proposed system does not involve global trajectory optimization, such as loop closure or re-identification of objects, though they can be incorporated into the system in future work.

### B. Object Representation with Dual Quadrics

We employ dual quadrics [6] as an object representation. Let an object $\boldsymbol{q}_{i,t}$ be modeled with centroid translation $\boldsymbol{q}_{i,t}^{trans} = [t_1, t_2, t_3]$, rotation $\boldsymbol{q}_{i,t}^{rot} = [\theta_1, \theta_2, \theta_3]$, and axis-aligned bounding box size $\boldsymbol{q}_{i,t}^{shape} = [s_1, s_2, s_3]$. It can be represented as dual quadrics $\boldsymbol{Q}_{i,t}^* \in \mathbb{R}^{4\times4}$ as follows:

$$\boldsymbol{Q}_{i,t}^* = \boldsymbol{Z}\breve{\boldsymbol{Q}}_{i,t}\boldsymbol{Z}^T \tag{1}$$

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{R}(\boldsymbol{q}_{i,t}^{rot}) & (\boldsymbol{q}_{i,t}^{trans})^T \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{2}$$

$$\breve{\boldsymbol{Q}}_{i,t} = \begin{bmatrix} s_1^2 & 0 & 0 & 0 \\ 0 & s_2^2 & 0 & 0 \\ 0 & 0 & s_3^2 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{3}$$

where $\boldsymbol{Z}$ is a homogeneous transformation, and $\breve{\boldsymbol{Q}}_{i,t}$ is an ellipsoid of shape $\boldsymbol{q}_{i,t}^{shape}$ at the origin. The dual quadrics $\boldsymbol{Q}_{i,t}^*$ can be projected to the image plane as the following:

$$\boldsymbol{C}_{i,t}^* = \boldsymbol{P}_t\boldsymbol{Q}_{i,t}^*\boldsymbol{P}_t^T \tag{4}$$

where $\boldsymbol{P}_t = \boldsymbol{K}[\boldsymbol{R}_t|\boldsymbol{t}_t] \in \mathbb{R}^{3\times4}$ is the camera projection matrix composed of the intrinsic parameter $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ and the camera pose $\boldsymbol{R}_t \in \mathbb{R}^{3\times3}, \boldsymbol{t}_t \in \mathbb{R}^{3\times1}$. $\boldsymbol{C}_{i,t}^* \in \mathbb{R}^{3\times3}$ is a dual conic which denotes the ellipse on the image.

### C. Notation

We define the state $\boldsymbol{X}_t$ at time $t$ to be estimated as

$$\boldsymbol{X}_t = [\boldsymbol{x}_t, \{\boldsymbol{q}_{i,t}, \boldsymbol{u}_{i,t}\}_{i=1}^N] \tag{5}$$

$$\boldsymbol{x}_t = [\boldsymbol{T}_t, \boldsymbol{v}_t, \boldsymbol{b}_t] \tag{6}$$

where $\boldsymbol{x}_t$ is the sensor state, $\boldsymbol{T}_t = [\boldsymbol{R}_t|\boldsymbol{t}_t] \in SE(3)$ is the sensor pose, $\boldsymbol{v}_t \in \mathbb{R}^3$ is the velocity, $\boldsymbol{b}_t = [\boldsymbol{b}_t^a, \boldsymbol{b}_t^\omega] \in \mathbb{R}^6$ is the IMU linear acceleration and angular velocity bias, and $\boldsymbol{q}_{i,t} = [\boldsymbol{q}_{i,t}^{rot}, \boldsymbol{q}_{i,t}^{trans}, \boldsymbol{q}_{i,t}^{shape}] \in \mathbb{R}^9$ and $\boldsymbol{u}_{i,t} \in \mathbb{R}^3$ respectively denote the quadric parameters (rotation, translation, and shape) and the velocity of the $i$-th object being tracked at

time $t$.

The $j$-th visual object detection result $\boldsymbol{D}_{j,t}$ at time $t$ is defined as

$$\boldsymbol{D}_{j,t} = [\boldsymbol{B}_{j,t}, \boldsymbol{f}_{j,t}, \mathcal{I}_{j,t}] \tag{7}$$

where $\boldsymbol{B}_{j,t} = [x_{\min}, y_{\min}, x_{\max}, y_{\max}] \in \mathbb{R}^4$ is the detection bounding box, $\boldsymbol{f}_{j,t} \in \mathbb{R}^{768}$ is the visual feature, and $\mathcal{I}_{j,t}$ is the segmentation mask image.

### D. Initial Quadric Extraction

Given the point cloud $\mathcal{P}_t$ and segmented mask image $\mathcal{I}_{j,t}$, we obtain a rough estimate of the pose and shape of each surrounding object. First, the input point cloud is projected onto the image plane, and then the points $\mathcal{P}_j^o$ for each object are extracted by picking up points in the segmented region. Then, for each object point cloud $\mathcal{P}_j^o$, we calculate the centroid position $\boldsymbol{c} = [c_x, c_y, c_z]^T \in \mathbb{R}^3$. Additionally, we pick up the pairs of points in $\mathcal{P}_j^o$ that are most distant along the x, y, and z axes from the centroid to compute the size of the axis aligned bounding box $\boldsymbol{s} = [s_x, s_y, s_z]^T \in \mathbb{R}^3$ of each object. We then obtain observed estimates of the quadric parameters for surrounding objects as

$$\boldsymbol{q}_{j,t}^{in} = [\boldsymbol{0}, \boldsymbol{c}, \boldsymbol{s}] \tag{8}$$

### E. Data Association and Dynamic Points Removal

Given the observed estimates of quadric parameters, the detected objects are associated with tracked object instances (i.e., object landmarks). Each landmark maintains the latest timestamp $t$, the quadric parameter $\boldsymbol{q}$, the velocity $\boldsymbol{u}$, and a diagonal Gaussian appearance model parameterized by the per-dimension mean $\boldsymbol{\mu} \in \mathbb{R}^{768}$, variance $\boldsymbol{\sigma}^2 \in \mathbb{R}^{768}$. In summary, the $i$-th landmark $\boldsymbol{L}_i$ is defined as

$$\boldsymbol{L}_i = [\, t_i^l,\ \boldsymbol{q}_i^l,\ \boldsymbol{u}_i^l,\ \boldsymbol{\mu}_i,\ \boldsymbol{\sigma}_i^2\,]. \tag{9}$$

Once pairs of an object detection and its initial quadric estimate, $\{\boldsymbol{q}_{j,t}^{in}, \boldsymbol{D}_{j,t}\}_{j=1}^{N_{\text{observed}}}$, are obtained at time $t$, we first narrow the association candidates by geometric gating:

$$\boldsymbol{t}_i^{pred} = f^{\text{trans}}(\boldsymbol{q}_i^l) + \boldsymbol{u}_i^l\, \Delta t_i, \tag{10}$$

$$d_{ij} = \left\| \boldsymbol{t}_i^{pred} - f^{\text{trans}}(\boldsymbol{q}_{j,t}^{in}) \right\|, \qquad d_{ij} < d_{\text{th}}, \tag{11}$$

where $f^{\text{trans}}(\cdot)$ extracts the translation from a quadric and $\Delta t_i = t - t_i^l$. This yields the candidate set $\mathcal{C}_i = \{\boldsymbol{q}_{j,t}^{in}, \boldsymbol{D}_{j,t}\}^{N_{\text{candidate}}}$ for landmark $\boldsymbol{L}_i$.

For each candidate, we compute the Gaussian Naive Bayes log-likelihood under the landmark's appearance model:

$$\ell_{ij} = \log \mathcal{N}\big(\boldsymbol{f}_{j,t}\,;\, \boldsymbol{\mu}_i,\ \mathrm{diag}(\boldsymbol{\sigma}_i^2)\big) \tag{12}$$

$$= -\tfrac{1}{2} \sum_{k=1}^{768} \Big[ \log\big(2\pi\, \sigma_{i,k}^2\big) + \tfrac{(f_{j,t,k} - \mu_{i,k})^2}{\sigma_{i,k}^2} \Big]. \tag{13}$$

We additionally apply a Mahalanobis-distance gate to reject unlikely matches:

$$m_{ij}^2 = \sum_{k=1}^{768} \tfrac{(f_{j,t,k} - \mu_{i,k})^2}{\sigma_{i,k}^2}, \qquad m_{ij}^2 \le \tau_m. \tag{14}$$

Among the gated candidates, we select the association that maximizes the log-likelihood:

$$\{\hat{\boldsymbol{q}}_{j,t}, \hat{\boldsymbol{D}}_{j,t}\} = \underset{\mathcal{C}_i}{\text{argmax}}\ \ell_{ij}. \tag{15}$$

Since gating allows duplicates, we resolve conflicts in descending order of $\ell_{ij}$ and remove assigned detections from the remaining candidate lists (greedy nearest-neighbor assignment). Detections that fail the Mahalanobis gate for all landmarks are initialized as new landmarks. After association, we update the landmark's appearance model online using the Welford's algorithm [20].

Subsequently, we remove dynamic points from the input point cloud. Let the $i$-th input object $\boldsymbol{q}_{i,t}^{in}$ now be associated with the $j$-th landmark $\boldsymbol{L}_j$. If the estimated velocity of $\boldsymbol{L}_j$ is larger than a threshold ($\|\boldsymbol{u}_j^l\| > u_{th}$), then we consider the landmark dynamic and remove the point cloud $\mathcal{P}_i^o$ corresponding to $\boldsymbol{q}_{i,t}^{in}$ from the input point cloud $\mathcal{P}_t$.

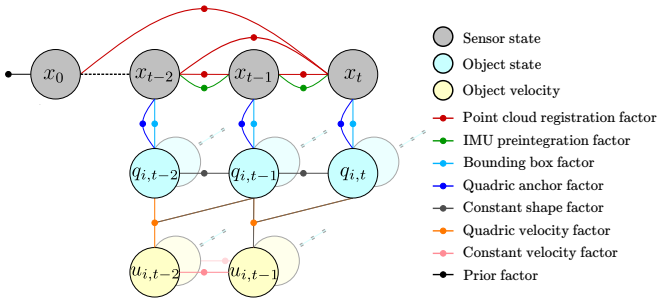



Fig. 3: Proposed factor graph structure. The proposed factor graph consists of a tightly coupled point cloud registration factor, IMU factor, visual object detection factor, and object motion factors. Because we use all detected surrounding objects, the graph involves multiple object states $q_{i,t}$.

### F. Factor Graph Formulation

In this system, we simultaneously estimate the states of the sensor and surrounding objects by optimizing the factor graph shown in Fig. 3. The proposed factor graph consists of six types of factors, *1)* Point cloud registration factor, *2)* IMU preintegration factor, *3)* Bounding box factor, *4)* Quadric anchor factor, *5)* Constant shape factor, and 6) Object motion factors. To limit the computation cost, we optimize the states in the most recent 5 seconds (i.e., sliding window optimization) using the iSAM2 [9] optimizer implemented in GTSAM [21].

*1)* ***Point Cloud Registration Factor****:* This factor constrains two sensor poses $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ such that the registration error between their corresponding depth point clouds $\mathcal{P}_i$ and $\mathcal{P}_j$ is minimized. To keep the consistency with past observations, we create this factor between the current frame and all past frames in the optimization window. As the registration error function, we employ a GPU-accelerated voxelized GICP error function (VGICP) [22]. VGICP approximates the local geometrical shape around each point $\boldsymbol{p}_k$ as a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{C}_k)$. For efficient nearest neighbor search, it discretizes the target point cloud $\mathcal{P}_j$ into voxels. The VGICP error function $e^M(\mathcal{P}_i, \mathcal{P}_j, \boldsymbol{T}_i, \boldsymbol{T}_j)$

is defined as follows:

$$e^{M}(\mathcal{P}_i, \mathcal{P}_j, \boldsymbol{T}_i, \boldsymbol{T}_j) = \sum_{\boldsymbol{p}_k \in \mathcal{P}_i} e^{\text{GICP}}(\boldsymbol{p}_k, \boldsymbol{T}_i^{-1}\boldsymbol{T}_j) \tag{16}$$

$$e^{\text{GICP}}(\boldsymbol{p}_k, \boldsymbol{T}_{ij}) = \boldsymbol{d}_k^T(\boldsymbol{C}'_k + \boldsymbol{T}_{ij}\boldsymbol{C}_k\boldsymbol{T}_{ij}^T)^{-1}\boldsymbol{d}_k \tag{17}$$

where $e^{\text{GICP}}(\boldsymbol{p}_k, \boldsymbol{T}_{ij})$ denotes the distribution-to-distribution distance between a source point $\boldsymbol{p}_k = (\boldsymbol{\mu}_k, \boldsymbol{C}_k) \in \mathcal{P}_i$ and its corresponding voxel $\boldsymbol{p}'_k = (\boldsymbol{\mu}'_k, \boldsymbol{C}'_k) \in \mathcal{P}_j$, and $\boldsymbol{d}_k = \boldsymbol{\mu}'_k - \boldsymbol{T}_{ij}\boldsymbol{\mu}_k$ is the residual between $\boldsymbol{\mu}_k$ and $\boldsymbol{\mu}'_k$.

*2) **IMU Preintegration Factor**:* To efficiently incorporate IMU data into the factor graph, we use the IMU preintegration factor [23]. Given an IMU measurement $\boldsymbol{a}_t, \boldsymbol{\omega}_t$, the sensor state after time interval $\Delta t$ can be modeled as follows:

$$\boldsymbol{R}_{t+\Delta t} = \boldsymbol{R}_t \exp((\boldsymbol{\omega}_t - \boldsymbol{b}_t^{\omega} - \boldsymbol{\eta}_t^{\omega})\Delta t) \tag{18}$$

$$\boldsymbol{v}_{t+\Delta t} = \boldsymbol{v}_t + \boldsymbol{g}\Delta t + \boldsymbol{R}_t(\boldsymbol{a}_t - \boldsymbol{b}_t^{a} - \boldsymbol{\eta}_t^{a})\Delta t \tag{19}$$

$$\boldsymbol{t}_{t+\Delta t} = \boldsymbol{t}_t + \boldsymbol{v}_t\Delta t + \frac{1}{2}\boldsymbol{g}\Delta t^2 + \frac{1}{2}\boldsymbol{R}_t(\boldsymbol{a}_t - \boldsymbol{b}_t^{a} - \boldsymbol{\eta}_t^{a})\Delta t^2 \tag{20}$$

where $\boldsymbol{g}$ represents the gravity vector, and $\boldsymbol{\eta}_t^{\omega}$ and $\boldsymbol{\eta}_t^{a}$ are the white noise in the IMU measurement. Then, by integrating them over time steps $i$ to $j$, we obtain the relative sensor motion $\Delta\boldsymbol{R}_{ij}$, $\Delta\boldsymbol{v}_{ij}$, and $\Delta\boldsymbol{t}_{ij}$ (see [23] for the detailed derivation). The error function $e^{\text{IMU}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is then defined as follows:

$$\begin{aligned} e^{\text{IMU}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = {} & \|\text{Log}(\Delta\boldsymbol{R}_{ij}^T\boldsymbol{R}_i^T\boldsymbol{R}_j)\|_{\Sigma}^2 \\ & + \|\Delta\boldsymbol{t}_{ij} - \boldsymbol{R}_i^T(\boldsymbol{t}_j - \boldsymbol{t}_i - \boldsymbol{v}\Delta t_{ij} - \frac{1}{2}\boldsymbol{g}\Delta t_{ij}^2)\|_{\Sigma}^2 \\ & + \|\Delta\boldsymbol{v}_{ij} - \boldsymbol{R}_i^T(\boldsymbol{v}_j - \boldsymbol{v}_i - \boldsymbol{g}\Delta t_{ij})\|_{\Sigma}^2 \end{aligned} \tag{21}$$

where $\|\cdot\|_{\Sigma}^2$ denotes the squared Mahalanobis distance with covariance $\Sigma$, and $\Sigma$ is propagated from the raw sensor noise covariance through the integration.

*3) **Bounding Box Factor**:* To simultaneously estimate the states of the sensor and surrounding objects by fusing geometrical and visual object detection information, we use the bounding box factor described in [6]. The error function is defined as follows:

$$e^{\text{bbox}}(\boldsymbol{T}_t, \boldsymbol{B}_{j,t}, \boldsymbol{q}_{i,t}) = \|\boldsymbol{B}_{j,t} - \beta(\boldsymbol{T}_t, \boldsymbol{q}_{i,t})\|_{\Sigma}^2 \tag{22}$$

where $\beta(\boldsymbol{T}, \boldsymbol{q}) = \hat{\boldsymbol{B}}$ is the quadric projection and bounding box fitting model formulated in [6]. This function first calculates a conic by projecting the quadric $\boldsymbol{q}$ to the image plane and obtains a bounding box $\boldsymbol{B}$ enclosing it. Then, the bounding box is refined until it encloses only the overlapping part of the conic and the image region, rather than simply circumscribing the entire conic, so that we can obtain accurate predictions for partial object observations. This factor imposes a constraint to minimize the error between the object bounding box $\boldsymbol{B}_{j,t}$ observed from the sensor pose $\boldsymbol{T}_t$ and the bounding box fitted to the quadric $\boldsymbol{q}_{i,t}$ projected on the image.

*4) **Quadric Anchor Factor**:* The bounding box factor has an ambiguity in the depth direction due to the camera projection. Although this ambiguity can be resolved with multiple observations from different viewpoints as in QuadricSLAM [6], in our online estimation setting, there is no guarantee that sufficient viewpoints are given for an object. We thus use the quadric anchor factor that constrains the relative position of the object to the sensor based on the 3D position of the object when it is observed, which is estimated from the point cloud. We define this factor as follows:

$$e^{\text{anchor}}(\boldsymbol{T}_t, \boldsymbol{q}_{i,t}) = \|f^{\text{trans}}(\widetilde{\boldsymbol{q}}_{i,t}) - \boldsymbol{T}_t^{-1} f^{\text{trans}}(\boldsymbol{q}_{i,t})\|_{\Sigma}^2 \tag{23}$$

where $\widetilde{\boldsymbol{q}}_{i,t}$ denotes the relative measurement of $\boldsymbol{q}_{i,t}$ from the sensor pose $\boldsymbol{T}_t$.

*5) **Constant Shape Factor**:* The objects detected by the preprocessing module are often only partially observed, thus the estimated shape parameters can fluctuate over time. To mitigate this, we assume that the shape of each object is constant over a short time interval and introduce a constant shape factor to suppresses temporal changes in the quadric shape. We define this factor as follows:

$$e^{\text{shape}}(\boldsymbol{q}_{i,t-1}, \boldsymbol{q}_{i,t}) = \|f^{\text{shape}}(\boldsymbol{q}_{i,t}) - f^{\text{shape}}(\boldsymbol{q}_{i,t-1})\|_{\Sigma}^2 \tag{24}$$

where $f^{\text{shape}}(\cdot)$ denotes a function that retrieves the shape component from the quadric parameter.

*6) **Object Motion Factors**:* To estimate an object velocity, we introduce object motion factors. The quadric velocity factor simply constrains the positions of an object in consecutive frames $\boldsymbol{q}_{i,t-1}$ and $\boldsymbol{q}_{i,t}$ to be matched with its velocity $\boldsymbol{u}_{i,t-1}$, as expressed in the following error function:

$$\begin{aligned} & e^{\text{vel}}(\boldsymbol{q}_{i,t-1}, \boldsymbol{q}_{i,t}, \boldsymbol{u}_{i,t-1}) = \\ & \qquad \|f^{\text{trans}}(\boldsymbol{q}_{i,t-1}) + \boldsymbol{u}_{i,t-1}\Delta t - f^{\text{trans}}(\boldsymbol{q}_{i,t})\|_{\Sigma}^2 \end{aligned} \tag{24}$$

where $\Delta t$ is the time difference of object observation.

Assuming that the velocity of an object is constant over a short time interval, we apply the constraints velocity factor between consecutive velocity variables for smoothing the object velocity. The error function is defined as follows:

$$e^{\text{cvel}}(\boldsymbol{u}_{i,t-1}, \boldsymbol{u}_{i,t}) = \|\boldsymbol{u}_{i,t} - \boldsymbol{u}_{i,t-1}\|_{\Sigma}^2 \tag{25}$$

This factor forces the velocity of an object to change smoothly and helps in mitigating noise in object observations.

In summary, the objective function of the proposed framework is defined as shown in Eq. 26. $\mathcal{X}$, $\mathcal{Q}$, and $\mathcal{U}$ denote sets of sensor states, object states, and object velocity states, respectively, in the optimization window $N^{\text{W}}$. In addition, $\mathcal{Q}_t$ represents the set of objects observed at time $t$. The error function $e^{\text{prior}}(\boldsymbol{x}_0)$ of the prior factor constrains the initial sensor state $\boldsymbol{x}_0$ to the fixed reference frame.

## IV. EVALUATION

To evaluate the robustness and accuracy of the proposed method, we recorded RGB-D data sequences with a Microsoft Azure Kinect. The dataset contains situations

$$e(\mathcal{X},\mathcal{Q},\mathcal{U}) = \sum_{\boldsymbol{x}_t\in\mathcal{X}}\sum_{s=t-N^{\mathrm{W}}}^{t-1}\underbrace{e^{\mathrm{M}}(\mathcal{P}_t,\mathcal{P}_s,\boldsymbol{T}_t,\boldsymbol{T}_s)}_{\text{Point cloud registration factor}} + \sum_{\boldsymbol{x}_t\in\mathcal{X}}\underbrace{e^{\mathrm{IMU}}(\boldsymbol{x}_{t-1},\boldsymbol{x}_t)}_{\text{IMU preintegration factor}} + \sum_{\boldsymbol{x}_t\in\mathcal{X}}\sum_{\boldsymbol{q}_{i,t}\in\mathcal{Q}_t}\underbrace{e^{\mathrm{bbox}}(\boldsymbol{T}_t,\boldsymbol{B}_{j,t},\boldsymbol{q}_{i,t})}_{\text{Bounding box factor}} + \sum_{\boldsymbol{x}_t\in\mathcal{X}}\sum_{\boldsymbol{q}_{i,t}\in\mathcal{Q}_t}\underbrace{e^{\mathrm{anchor}}(\boldsymbol{T}_t,\boldsymbol{q}_{i,t})}_{\text{Quadric anchor factor}}$$
$$+ \sum_{\boldsymbol{q}_{i,t}\in\mathcal{Q}}\underbrace{e^{\mathrm{shape}}(\boldsymbol{q}_{i,t-1},\boldsymbol{q}_{i,t})}_{\text{Constant shape factor}} + \sum_{\boldsymbol{q}_{i,t},\mathbf{u}_{i,t}\in\{\mathcal{Q},\mathcal{U}\}}\underbrace{e^{\mathrm{vel}}(\mathbf{q}_{i,t-1},\mathbf{q}_{i,t},\mathbf{u}_{i,t-1})}_{\text{Quadric velocity factor}} + \sum_{\boldsymbol{u}_{i,t}\in\mathcal{U}}\underbrace{e^{\mathrm{cvel}}(\boldsymbol{u}_{i,t-1},\boldsymbol{u}_{i,t})}_{\text{Constant velocity factor}} + \underbrace{e^{\mathrm{prior}}(\boldsymbol{x}_0)}_{\text{Prior factor}} \tag{26}$$

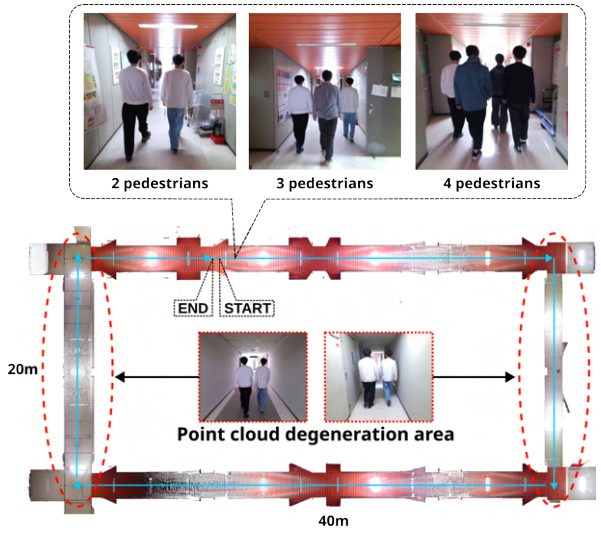



Fig. 4: Hallway experimental environment (hallway1). The Azure Kinect captures RGB-D and IMU data from just behind a predefined number of leading pedestrians while traversing a 120 m rectangular hallway in the clockwise direction (blue arrows). Due to the hallway environment, point cloud degeneracy becomes a serious problem even if the dynamic points are removed.

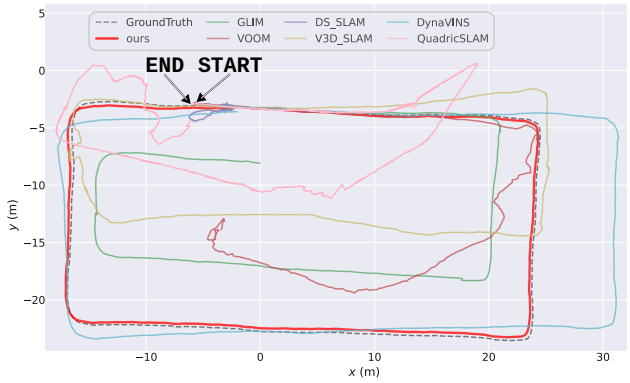



Fig. 5: Estimated trajectories for hw1_2 sequence.

challenging to odometry estimation because many dynamic objects (e.g., pedestrians) occupy the sensor view. Details on the conditions of these datasets are explained as follows:

*1)* ***Hallway****:* This is an environment in which two to four people walking abreast are continuously visible within the sensor's field of view while walking around a hallway loop as shown in Fig. 4. The dataset was collected using three hallways with different total lengths (120 m, 100 m, and 130 m). This poses a challenge for odometry estimation methods to deal with geometric degeneracy and the continuous presence of dynamic objects filling the field of view. We hereafter denote the hallway datasets using the notation "hw{hallway id}_{pedestrian number}".

*2)* ***Stairs****:* This is an environment in which two people walking abreast are continuously visible within the sensor's field of view while going up and down stairs from the first to the third floor (3×7×12 m). This environment is challenging due to its vertical structure, requiring the sensor to move along a non-planar trajectory while continuously observing dynamic objects.

*3)* ***Crowded room****:* This is an environment in which six people move around a room and rearrange the furniture. Because numerous objects of various categories (e.g., pedestrian, chair, table, and book) are moving, the scene is particularly challenging.

For each sequence, ground truth trajectories were obtained by aligning the sensor point clouds with a precise environment map captured using a survey-grade LiDAR (FARO Focus). We used an Intel Core i9-12900K (24 threads) with an NVIDIA GeForce RTX 4090 to run the proposed method.

**Comparison with Existing Methods:** We compared the proposed method with state-of-the-art 3D LiDAR SLAM method: GLIM [8]; object visual SLAM methods: QuadricSLAM [6] and VOOM [12]; and SLAM methods for dynamic environments: DS-SLAM [5], V3D-SLAM [24] and DynaVINS [18]. QuadricSLAM [6] was re-implemented by modifying our method: the object motion factor, the point cloud registration factor, and the IMU preintegration factor were removed from the factor graph, and the relative pose constraints between sensor states were pre-computed with GLIM [8] (i.e., loose coupling).

Table I shows the average RTEs [25] per 3 m of the proposed method and the existing methods, and Fig. 5 shows the estimated trajectories on the hw1_2 dataset. GLIM uses a tightly coupled LiDAR-IMU framework and its sliding window optimization successfully handled short-term point cloud degeneracy, providing a better result (0.07498-2.868 m). However, we can see from Fig. 5 that the sensor pose estimation became degraded after dynamic objects occluded the sensor in severely degenerated areas. DynaVINS likewise employs a tightly coupled visual-inertial framework, yet the sustained presence of dynamic objects makes reliable tracking of static features difficult, resulting in larger errors (0.33423-67.93296 m). In contrast, our proposed method achieved consistently accurate estimation even under such challenging conditions. This demonstrates that using dynamic objects not only for dynamic points removal but also as landmarks significantly contributes to maintaining

TABLE I: Comparison of relative trajectory error [m] with existing methods

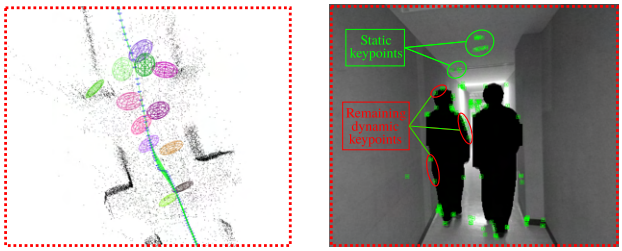
| Sequence | hw1_2 | hw1_3 | hw1_4 | hw2_2 | hw2_3 | hw2_4 | hw3_2 | hw3_3 | hw3_4 | stairs | crowded room |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Path length [m] | | 120 | | | 100 | | | 130 | | - | - |
| Front pedestrians | 2 | 3 | 4 | 2 | 3 | 4 | 2 | 3 | 4 | 2 | 6 |
| Ours | **0.08243** | **0.13059** | **0.29029** | **0.20697** | 0.88548 | 1.11728 | **0.03868** | **0.10551** | **0.15096** | **0.05684** | **0.11679** |
| GLIM [8] | 2.10579 | 1.35007 | 2.40985 | 2.37180 | 2.52949 | 2.86800 | 1.05559 | 1.69490 | 2.10418 | 0.07498 | 0.15189 |
| QuadricSLAM [6] | 1.77405 | 1.39869 | 2.15410 | 1.57964 | 2.05273 | 2.71470 | 1.22616 | 1.68382 | 1.83163 | 3.22397 | 2.29158 |
| VOOM [12] | 1.34585 | 1.50874 | 1.04852 | 0.44348 | **0.86033** | 0.73199 | 0.95759 | 1.22470 | 1.04409 | 1.74145 | 0.69484 |
| DS_SLAM [5] | 2.63422 | 4.33959 | 1.38081 | 1.10130 | 1.19675 | **0.70568** | 12.91585 | 8.12928 | 13.76414 | 0.29514 | 0.57612 |
| V3D-SLAM [24] | 0.99910 | 10.23673 | 12.00001 | 1.62478 | 15.11677 | 15.95537 | 2.48722 | 4.88607 | 6.81587 | 0.68631 | 0.66820 |
| DynaVINS [18] | 4.50960 | 4.70407 | 4.37366 | 4.04247 | 3.96510 | 4.35412 | 4.55843 | 16.61833 | 67.93296 | 3.63819 | 0.33423 |

**Bold** and underline respectively indicate the best and second best scores.

TABLE II: Relative trajectory error [m] in ablation study

| Sequence | * | ** | hw1_2 | hw1_3 | hw1_4 | hw2_2 | hw2_3 | hw2_4 | hw3_2 | hw3_3 | hw3_4 | stairs | crowded room |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Path length [m] | | | | 120 | | | 100 | | | 130 | | - | - |
| Front pedestrians | | | 2 | 3 | 4 | 2 | 3 | 4 | 2 | 3 | 4 | 2 | 6 |
| Ours | ✓ | ✓ | **0.08243** | **0.13059** | **0.29029** | **0.20697** | **0.88548** | **1.11728** | **0.03868** | 0.10551 | 0.15096 | **0.05684** | 0.11679 |
| ab_1 | ✓ | | 0.84986 | 1.67900 | 1.45821 | 1.32801 | 1.91947 | 2.29834 | 0.62792 | 1.09645 | 1.04214 | 0.07768 | 0.17086 |
| ab_2 | | | 0.93566 | 1.44821 | 1.53076 | 1.44313 | 1.91905 | 2.50757 | 0.62329 | 1.11598 | 1.41287 | 0.07647 | 0.15652 |
| ab_3 | | ✓ | 0.09014 | 0.14346 | 0.30154 | 0.66818 | 0.99667 | 1.16944 | 0.04001 | **0.10325** | **0.14962** | 0.05911 | **0.11517** |

*: object state optimization; **: dynamic points removal.
**Bold** and underline respectively indicate the best and second best scores.




(a) VOOM estimation result (b) V3D-SLAM detection result

Fig. 6: Object estimation results given input from hw1_2 sequence: (a) VOOM; (b) V3D-SLAM. In (b), moving objects are masked, yet keypoints remain along mask boundaries; in this frame they are present in a number comparable to the static keypoints.

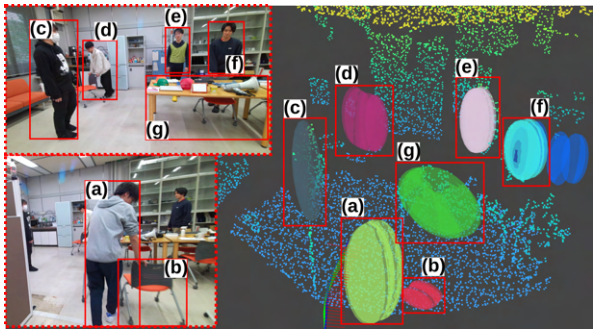



Fig. 7: Tracking results for arbitrary objects in crowded room sequence. Each colored ellipsoid in the 3D view indicates an estimated object, and multiple ellipsoids of the same color represent the same object tracked over time. Labels (a)-(g) correspond to the same object in both the input images and the 3D results.

accuracy in such a severe environment. VOOM achieved a relatively low RTE (0.44348-1.50874 m), but it lost tracking mid-sequence, failed to relocalize, and produced no further poses. QuadricSLAM and VOOM assume that all surrounding objects are static, so object estimates do not help odometry when objects move. As a result, VOOM treats each observation of a moving object as an independent static instance (Fig. 6a), which yields inconsistent object representations and degrades performance. DS-SLAM and V3D-SLAM showed lower accuracy overall (0.70568-12.91585 m and 0.99910-15.95537 m, respectively). V3D-SLAM masks moving objects, yet boundary keypoints remain. When static features are degenerate (Fig. 6b), RANSAC fails to reject all of them, outliers persist, and accuracy drops. In summary, across all sequences, our proposed method, which features dynamic points removal and continuous object tracking, achieved the highest accuracy (0.05684–1.11728 m) compared with existing methods.

**Ablation Study:** To reveal the effect of object optimization and dynamic points removal, we conducted an ablation study with the following configurations.

- ab_1: The dynamic points removal process is omitted from the system.
- ab_2: Objects are not treated as landmarks; only states of the sensor are optimized.
- ab_3: Objects are again not treated as landmarks, but dynamic points are still removed.

Table II reports the average RTEs per 3 m. Results of ablations ab_1 and ab_2 show that skipping dynamic point removal is the primary cause of large errors. Moreover ab_2 is slightly better than ab_1, indicating that adding dynamic landmarks without dynamic points removal can even degrade accuracy. Although ab_3, which removed only the dynamic points, achieved an accuracy comparable to that of our proposed method, our method still marginally outperformed it on many sequences. indicating that the continuous vi-

sual object detection constraints contribute to improving estimation accuracy. On the *crowded room* sequence, ab_3 slightly outperforms the full model, indicating that rich static geometry makes dynamic point removal sufficient for stable registration, whereas unstable object observations due to frequent occlusions conflict with the constant shape prior and degrade object estimation.

**Qualitative Analysis:** We further conducted a qualitative evaluation of the proposed method. Fig. 7 presents an example of object tracking results of the proposed method on the crowded room dataset. (a) and (b) respectively correspond to a chair and a person pulling it, while (d) shows a person carrying a chair. Due to the accurately estimated velocities with robust data association, the method successfully tracked such dynamic objects. Furthermore, (c), (e), and (f) represent stationary people, and (g) corresponds to a static table. All of these were also correctly estimated. These results show that the proposed method, using the dual quadric representation, can estimate and track arbitrary objects detected in images, regardless of class or motion.

**Processing Time:** We measured the processing times of the preprocessing module and the odometry estimation module on the hw1_3 sequence as a representative case: it has a medium path length and crowding level (three pedestrians). The average processing times per frame were approximately 19.67 ms for the preprocessing module and 46.33 ms for the odometry estimation module, respectively on an NVIDIA GeForce RTX 4090. The total processing time per frame was thus approximately 66 ms, which is well within the real-time requirements (100 ms per frame).

## V. CONCLUSIONS

In this paper, we presented a tightly coupled RGB-D inertial object odometry estimation method for dynamic environments. By integrating dual quadric-based object tracking into a RGB-D inertial odometry framework, our system enables continuous estimation of object pose and velocity, and removes dynamic points from the input point cloud. Experimental results across hallway, stairs, and crowded room scenarios show that our method achieves robust odometry even under severe point cloud degeneracy and dense dynamic interference, and the ability to track arbitrary objects.

## REFERENCES


[1] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "Orb-slam3: An accurate open-source library for visual, visual – inertial, and multimap slam," IEEE Transactions on Robotics, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[2] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[3] R. A. Newcombe, A. Fitzgibbon, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, and S. Hodges, "Kinectfusion: Real-time dense surface mapping and tracking," in 2011 10th IEEE International Symposium on Mixed and Augmented Reality. IEEE, Oct. 2011, pp. 127–136.

[4] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," The International Journal of Robotics Research, vol. 35, no. 14, pp. 1697–1716, Sept. 2016.

[5] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "Ds-slam: A semantic visual slam towards dynamic environments," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, Oct. 2018, pp. 1168–1174.

[6] L. Nicholson, M. Milford, and N. Sunderhauf, "Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam," IEEE Robotics and Automation Letters, vol. 4, no. 1, pp. 1–8, Jan. 2019.

[7] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[8] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Glim: 3d range-inertial localization and mapping with gpu-accelerated scan matching factors," Robotics and Autonomous Systems, vol. 179, p. 104750, Sept. 2024.

[9] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in 2011 IEEE International Conference on Robotics and Automation. IEEE, May 2011.

[10] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in 2013 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, June 2013.

[11] S. Yang and S. Scherer, "Cubeslam: Monocular 3-d object slam," IEEE Transactions on Robotics, vol. 35, no. 4, pp. 925–938, Aug. 2019.

[12] Y. Wang, C. Jiang, and X. Chen, "Voom: Robust visual object odometry and mapping using hierarchical landmarks," in 2024 IEEE International Conference on Robotics and Automation, 2024, pp. 10 298–10 304.

[13] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "Lins: A lidar-inertial state estimator for robust and efficient navigation," in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, May 2020, pp. 8899–8906.

[14] T.-M. Nguyen, D. Duberg, P. Jensfelt, S. Yuan, and L. Xie, "Slict: Multi-input multi-scale surfel-based lidar-inertial continuous-time odometry and mapping," IEEE Robotics and Automation Letters, vol. 8, no. 4, pp. 2102–2109, Apr. 2023.

[15] H. Strasdat, J. Montiel, and A. J. Davison, "Visual slam: Why filter?" Image and Vision Computing, vol. 30, no. 2, pp. 65–77, 2012.

[16] R. Mur-Artal and J. D. Tardos, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[18] S. Song, H. Lim, A. J. Lee, and H. Myung, "Dynavins: A visual-inertial slam for dynamic environments," IEEE Robotics and Automation Letters, vol. 7, no. 4, pp. 11 523–11 530, Oct. 2022.

[19] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in 2017 IEEE International Conference on Computer Vision (ICCV). IEEE, Oct. 2017.

[20] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," Technometrics, vol. 4, no. 3, pp. 419–420, Aug. 1962.

[21] F. Dellaert and G. Contributors, "borglab/gtsam," May 2022. [Online]. Available: https://github.com/borglab/gtsam

[22] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Globally consistent 3d lidar mapping with gpu-accelerated gicp matching cost factors," IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 8591–8598, Oct. 2021.

[23] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," IEEE Transactions on Robotics, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[24] T. Dang, K. Nguyen, and M. Huber, "V3d-slam: Robust rgb-d slam in dynamic environments with 3d semantic geometry voting," in 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, Oct. 2024, pp. 7847–7853.

[25] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, Oct. 2018, pp. 7244–7251.