Revisiting Fast and Accurate RGB-D Odometry for Real-World Use by Embracing Simplicity

Sumanth Nagulavancha Dhagash Desai Saurabh Gupta Luca Lobefaro
Cyrill Stachniss Ignacio Vizzo Tiziano Guadagnino

***Abstract*— Robust and accurate pose estimation of a robot is essential for many tasks. Over the past decades, researchers have explored odometry estimation using various sensors, including RGB cameras, LiDARs, and RGB-D cameras. RGB-D cameras, mainly, are attractive sensors as they provide color and depth. Despite the progress made in the RGB-D odometry approaches and systems sharing code, their practical use is often non-trivial due to technical debt in several open-source implementations. In some cases, approaches have been tuned to specific benchmarks, complicating the extension of implementations to different RGB-D sensors and environments. This paper introduces an easy-to-use and easy-to-understand yet robust RGB-D odometry pipeline adaptable across sensor platforms. We focus on simplicity and contribute a non-black-box RGB-D odometry approach and proper implementation to the community. Our open-source system utilizes ORB features for correspondence estimation and employs a frame-to-map registration strategy to estimate camera poses. Our system achieves performance on par with state-of-the-art methods while running faster than the sensor frame rate on a single-core CPU. Importantly, our approach does not require integrating IMU data or additional hardware like GPUs, making it easily deployable on various platforms.**

I. Introduction

Estimating the ego-motion of a robot, known as odometry, is essential for various robotic tasks. It is also a fundamental component in applications like simultaneous localization and mapping (SLAM) [14]. Recent advancements have introduced diverse odometry estimation techniques using different sensors, including RGB cameras [10], [11], [26], [34], LiDARs [5], [15], [50], [51], RADARs [20], and RGB-D sensors [24], [35], [36], [46]. RGB-D cameras offer the advantage of providing both RGB and depth measurements, effectively resolving scale ambiguities inherent in visual odometry estimation using an RGB camera. Despite being often less accurate, RGB-D cameras offer a cost-effective and energy-efficient alternative to LiDARs.

While the RGB-D odometry research field has made tremendous progress in the past decade [7], [9], [25], [35], available research implementations of RGB-D odometry systems can present quite significant technical debt. This results in software packages that are partially challenging to install and run due to the high number of software dependencies and pose a high barrier to entry for usage on real robots. Moreover, some implementations are tailored to specific benchmarks or datasets [45]. Additionally, some approaches [9], [25] are tightly coupled with particular software frameworks like the robot operating system (ROS) [41], limiting their usability to framework-specific data. Furthermore, a subset of methods require additional sensors such as an inertial measurement unit (IMU), demanding extra efforts for sensor synchronization or specialized hardware acceleration like GPUs to achieve real-time performance [8], [36], [48], thus making it harder to adapt them to other platforms. This highlights the necessity for a straightforward yet efficient RGB-D odometry system capable of standalone operation and easy generalization to other platforms without relying on specialized hardware or additional sensors.

The main contribution of this paper is an easy-to-understand yet efficient and effective RGB-D odometry estimation pipeline designed to compute the

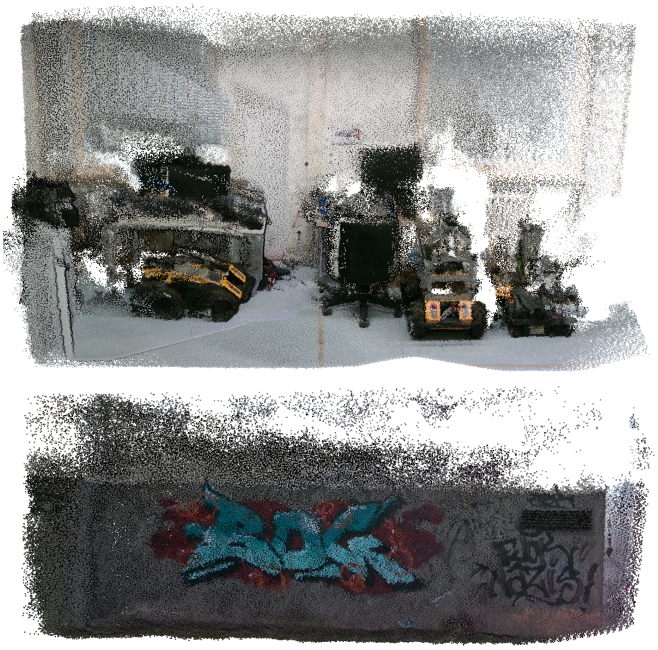

Fig. 1: Examples of reconstructions obtained from real-world data collected with an Intel RealSense D455. We showcase the geometric and photometric quality of the reconstruction obtained by exploiting the poses estimated with our approach.

The authors Sumanth Nagulavancha, Dhagash Desai, Saurabh Gupta, Luca Lobefaro, Tiziano Guadagnino and Cyrill Stachniss are with the University of Bonn, Center for Robotics. Sumanth Nagulavancha is additionally with Evitado Technologies, Germany. Cyrill Stachniss is additionally with the Lamarr Institute for Machine Learning and Artificial Intelligence. Ignacio Vizzo is with Dexory, UK

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 – 390732324 – PhenoRob, by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under STA 1051/5-1 within the FOR 5351 (AID4Crops), and by the German Federal Ministry of Education and Research (BMBF) in the project "Robotics Institute Germany", grant No. 16ME0999.

trajectory of a moving RGB-D camera at the sensor frame rate. We give an example application of our approach in Fig. 1. Our method does not require an IMU or specialized hardware such as GPUs, thereby realizing simplicity and generalizability to other platforms. We start from an initial pose estimate computed based on the previous motion of the camera and refine this initial pose by registering the input RGB-D frame with a local map representation. To this end, we extract image keypoints and descriptors from the RGB image and associate them with the depth information. We combine the image descriptors with the random sample consensus scheme (RANSAC) [12] for correspondence estimation between the frame and the local map representation. Furthermore, we implement a reliable registration scheme between the data associations using the iteratively reweighed least-squares [16] algorithm to estimate the pose of the sensor.

We make four key claims: our approach (i) performs on par with state-of-the-art methods in terms of odometry accuracy; (ii) computes the RGB-D odometry faster than the sensor frame rate on a single-core CPU; (iii) is significantly easier to understand and run compared to the existing approaches, with a smaller codebase and fewer software dependencies, and (iv) can operate in real-word scenarios and not just on benchmarks. These claims are backed up by the paper and our experimental evaluation. We provide an open-source implementation at: `https://github.com/sumanthrao1997/simple_rgbd_odometry.git`.

II. Related Work

Since the introduction of the first consumer-grade RGB-D camera, Microsoft Kinect, in 2011, numerous studies have explored RGB-D odometry and SLAM. In standard formulation, the RGB-D pose estimation problem is expressed using a probabilistic model on the unknown camera motion given noisy measurements as input. A maximum likelihood approach is employed to find the model parameters that maximize the probability of obtaining the measurements given the camera pose [10], [11], [25], [28], [35], [43]. RGB-D odometry estimation can be categorized into indirect, direct, and hybrid methods [52].

Indirect methods focus on optimizing the error between geometric correspondences in consecutive images. Various image descriptors, including SIFT [31], SURF [1], BRIEF [2], and ORB [42], can be used to estimate geometric correspondences [17], [18], [34], [43], [44]. RGBD-SLAMv2 [9] is the first open-source RGB-D pose estimation system designed to operate at sensor frame rate. It employs SIFT descriptors with RANSAC for robust geometric matching and least-squares optimization for camera pose estimation. ORB-SLAM3 [3] is a popular Visual SLAM pipeline supporting RGB-D cameras. It incorporates ORB features and operates in a frame-to-map alignment scheme for pose estimation with RGB-D cameras. Despite its popularity, ORB-SLAM3 lacks ongoing support and presents challenges in maintenance and comprehension due to its complex software interfaces. KISS-ICP [50] offers a simple open-source LiDAR odometry pipeline adaptable to RGB-D sensor data.

In contrast to feature-based approaches, direct methods operate on raw sensor measurements, assuming consistent pixel intensity across images to estimate camera motion by minimizing photometric error. KinectFusion [36] pioneered direct RGB-D odometry but is limited to small scenes, and operation at sensor frame rate requires a GPU. Kerl et al. [24] introduced a probabilistic approach for frame-to-frame motion estimation based on pixel intensity. Concha et al. [6] improved robustness with the integration of multi-view constraints. Fontan et al. [13] achieved comparable accuracy using only 24 points for tracking, eliminating the need for feature matching and enhancing robustness to blur and texture-less scenes. However, direct methods suffer from artifacts due to surface reflectance models and sensor artifacts arising from rolling shutter, auto-gain, and auto-exposure. Despite their advantages, none of these methods offer a usable open-source implementation.

Hybrid methods merge the advantages of direct and indirect approaches and are resilient to dynamic objects and low-texture environments. CPA-SLAM [32] integrates a global plane model with direct image alignment, while BundleFusion [7] combines sparse feature matching and dense error minimization. SDTAM [30] refines alignment with coarse-to-fine alignment strategy and ORB features. Color-ICP [40] optimizes jointly geometric and photometric errors in colored point clouds, addressing instability of iterative closest point alignment on smooth surfaces. Despite improved accuracy, hybrid methods pose challenges like complexity and computation and have limited open-source implementations. Color-ICP stands out for its simplicity and availability in the Open3D [53] library.

In summary, while many odometry approaches show promising results, their open-source versions often pose usability challenges [4], [21], [45]. These issues include limited documentation, hard-coded optimizations for specific RGB-D sensors, or the need for additional hardware like IMUs or GPUs. Approaches targeting simplicity exist. An example is KISS-ICP, which is designed for LiDARs. Our approach, inspired by the ORB-SLAM systems [3], [34], [35], leverages feature-based methods with ORB keypoints for efficiency and robustness. At the same time, we draw insights from KISS-ICP and design a straightforward RGB-D odometry system, achieving comparable accuracy to advanced methods while operating faster than the sensor frame rate and ensuring simplicity and generalizability across platforms.

III. Our Approach

Our method computes the trajectory of a calibrated mobile RGB-D camera by incrementally aligning the captured sensor frames. In short, we extract ORB [42] keypoints and descriptors from the RGB frame and associate the keypoints with corresponding depth values. We employ a voxel hash map known for its memory efficiency and quick correspondence search [33] as our local map. Finally, we implement a robust frame-to-map alignment strategy.

A. Extraction of ORB Features

Aligning a sequence of RGB-D frames involves extracting distinct features and matching them across consecutive frames. In line with the ORB-SLAM systems [3], [34], [35], we use the oriented FAST and rotated BRIEF (ORB) [42] feature descriptors for their efficient performance. These descriptors exhibit good invariance to changes in viewpoint and illumination and work consistently in many scenarios. Additionally, the binary nature of ORB features enhances memory efficiency, and the descriptor matching is fast by leveraging the Hamming distance metric.

For each input RGB-D frame we extract an intermediate representation called a Frame $\mathcal{F} = \{\boldsymbol{k}_{fi}, \boldsymbol{o}_{fi}, d_{fi}\}$, where $\boldsymbol{k}_{fi} \in \mathbb{R}^2$ denotes the i^{th} ORB keypoint extracted from the f^{th} RGB image, $\boldsymbol{o}_{fi} \in \{0,1\}^{256}$ is the corresponding binary descriptor and $d_{fi} \in \mathbb{R}^+$ is the corresponding depth value of the keypoint $\boldsymbol{k}_{fi}$ extracted from the depth image. Furthermore, we filter $\mathcal{F}$ based on a depth threshold r_{max} to extract $\mathcal{F}^*$ because the depth measurements from RGB-D cameras are typically reliable only until a certain distance.

B. Local Map

Similar to prior works [35], [37], [50], our RGB-D odometry pipeline employs a frame-to-map alignment approach. It aligns the current frame information with the information accumulated so far, i.e., a local map, to incrementally estimate the camera pose T_t by minimizing a geometrical error function. To achieve this seamlessly, we need a data structure that can store information from previously registered frames. Several RGB-D odometry systems [3], [13], [35] use a keyframe-based internal map representation. A recent study by Muglikar et al. [33] explores the voxel hash map data structure in visual odometry, offering a memory and computation-efficient solution while addressing the unbounded growth issue of keyframe-based representations.

In our implementation, we utilize a voxel grid as the local map representation, with each voxel having a size of $\nu \times \nu \times \nu$ and capacity for up to N_{max} points per voxel. To manage map growth, each voxel is designed as a circular buffer storing the most recent points. After registering a frame with 3D points in the local map, all keypoints are transformed to the camera pose T_t and stored within the local map along with their corresponding image descriptors. Additionally, we remove voxels beyond the sensor depth threshold r_{max} and behind the camera to prevent unbounded growth.

C. Frame to Map Alignment

To achieve reliable pose estimation T_t for a calibrated RGB-D camera, we start with the extraction of the features $\mathcal{F}^*$. We predict the camera's initial pose $\mathsf{T}_{\text{motion},t}$ with respect to its surroundings using the constant velocity model $\Delta\mathsf{T}_{\text{pred},t}$. To further refine our motion prior $\mathsf{T}_{\text{motion},t}$, we register the current frame $\mathcal{F}^*$ to the points in the local map $\mathcal{M}$. To this end, we first establish robust correspondences between $\mathcal{F}^*$ and the local map $\mathcal{M}$ using ORB descriptor matching in combination with RANSAC [12]. Finally, we formulate a geometric error on the frame-to-map correspondences and minimize the error with an iteratively reweighted least-squares solution [16] for robust pose estimation using the motion prediction $\mathsf{T}_{\text{ransac},t}$ obtained from the RANSAC step. Finally, we update our map and proceed with the next input RGB-D frame.

1) Motion Prediction: As highlighted by Vizzo et al. [50], odometry estimation in the context of mobile robots can be formulated as estimating the deviation between the robot's expected and actual motion. Various methods exist to predict the expected motion of a robot before proceeding with the frame-to-map alignment. These include the constant velocity model, spline fitting to previous camera poses, wheel odometry from encoders, and motion estimation with IMUs. The constant velocity model is the most widely applicable of these options as it is computed solely based on the relative motion between the previous camera poses. It also eliminates the need for extra sensors like IMU and wheel encoders or complex mathematical formulations like spline fitting. Moreover, RGB-D cameras record data at 30 - 60 frames per second, i.e., a new frame every 15 - 30 ms. As a result of the fairly high frame rate, the deviations from the constant velocity model in such short intervals are typically minimal.

Using the constant velocity model, we can predict the motion $\Delta\mathsf{T}_{\text{pred},t} \in \mathbb{SE}(3)$ at the current time step t by using the previous pose estimates $\mathsf{T}_{t-1} = [\boldsymbol{R}_{t-1}, \boldsymbol{t}_{t-1}]$ and $\mathsf{T}_{t-2} = [\boldsymbol{R}_{t-2}, \boldsymbol{t}_{t-2}]$, as:

$$\Delta\mathsf{T}_{\text{pred},t} = \begin{bmatrix} \boldsymbol{R}_{t-2}^\top \boldsymbol{R}_{t-1} & \boldsymbol{R}_{t-2}^\top (\boldsymbol{t}_{t-1} - \boldsymbol{t}_{t-2}) \\ \mathbf{0} & 1 \end{bmatrix}. \quad (1)$$

This motion prediction from the constant velocity model is then used to generate an initial guess for the remaining parts of the pipeline as $\mathsf{T}_{\text{motion},t} = \mathsf{T}_{t-1}\,\Delta\mathsf{T}_{\text{pred},t}$.

2) Descriptor Matching and RANSAC: For correspondence estimation between the keypoints in the current frame $\mathcal{F}^*$ and the 3D world points in the local map, we first query the local map. We specifically look for voxels inside a predefined search radius r of the current pose estimate $\mathsf{T}_{\text{motion},t}$. All the 3D points $\boldsymbol{x}_{mi}$, with their respective feature descriptors $\boldsymbol{o}_{mi}$, are extracted from the map. Then, the descriptors in $\mathcal{F}^*$ are matched against the descriptors $\boldsymbol{o}_{mi}$ in the map with a brute force search using the Hamming distance metric followed by the Lowe's ratio test [31] to establish bijective correspondences between $\mathcal{F}^*$ and $\mathcal{M}$.

To address outliers in descriptor matching arising from sensor noise, occlusion, and tracking errors, we filter the matches with RANSAC [12]. Various techniques such as Kabsch-Umeyama [23], [49] algorithm, P3P [27], EPNP [29], and DLT [19] can be employed for the geometric verification of descriptor matches within RANSAC. Our RANSAC scheme specifically incorporates the Kabsch-Umeyama algorithm due to its efficiency and minimal computation using only three pairs of corresponding points. Furthermore, the Kabsch-Umeyama algorithm also computes a motion refinement $\Delta\mathsf{T}_{\text{ransac},t} \in \mathbb{SE}(3)$, to increment our motion prediction $\mathsf{T}_{\text{motion},t}$ towards the actual pose T_t.

In our RANSAC scheme, given the correspondence set $\mathcal{C} = \{(\boldsymbol{k}_{fi}, d_{fi}, \boldsymbol{x}_{mi}) \mid \boldsymbol{k}_{fi} \in \mathbb{R}^2, d_{fi} \in \mathbb{R}, \boldsymbol{x}_{mi} \in \mathbb{R}^3\}$,

Algorithm 1: Robust non-linear least-squares

Input: Inlier correspondences $\mathcal{C}^*$ (2D-3D) from RANSAC, camera intrinsic matrix K, maximum iterations N, updated pose estimate after RANSAC step $\mathsf{T}_{\text{ransac},t}$, and convergence threshold ϵ

Output: Rigid transformation matrix $\mathsf{T}_t \in \mathbb{SE}(3)$

$n \leftarrow 0, \mathsf{T}_t \leftarrow \mathsf{T}_{\text{ransac},t}$
repeat
  $\boldsymbol{H} \leftarrow 0$ and $\boldsymbol{b} \leftarrow 0$
  foreach $\{\boldsymbol{k}_{fi}, \boldsymbol{X}_{mi}\} \in \mathcal{C}^*$ **do**
    $\boldsymbol{J}^{[i]} \leftarrow \boldsymbol{J}_{\pi}^{[i]} \mathsf{K} \boldsymbol{J}_{\mathsf{T}}^{[i]}$
    $\mathbf{e}^{[i]} \leftarrow \pi\left(\mathsf{K}\left(\mathsf{T}_t^{-1} \oplus \boldsymbol{x}_{mi}\right)\right) - \boldsymbol{k}_{fi}$
    $\omega^{[i]} \leftarrow$ kernel-weight $\left(\|\boldsymbol{e}^{[i]}\|_2\right)$
    $\boldsymbol{H} \leftarrow \boldsymbol{H} + \boldsymbol{J}^{[i]} \omega^{[i]} \boldsymbol{J}^{[i]}$
    $\boldsymbol{b} \leftarrow \boldsymbol{b} + \boldsymbol{J}^{[i]} \omega^{[i]} \boldsymbol{e}^{[i]}$
  end
  $\Delta \boldsymbol{x} \leftarrow \boldsymbol{H}^{-1} \boldsymbol{b}$
  $\mathsf{T}_t \leftarrow \exp\left(\Delta \boldsymbol{x}\right) \mathsf{T}_t$
  $n \leftarrow n + 1$
until $n \geq N$ *or* $\|\Delta \boldsymbol{x}\| < \epsilon$
Function `kernel-weight(`e`)`**:**
  $\omega \leftarrow \frac{1}{|e|}$
  return ω

we first unproject the frame keypoints $\boldsymbol{k}_{fi}$ using the provided camera calibration matrix K and the current motion prediction $\mathsf{T}_{\text{motion},t}$ using:

$$\boldsymbol{x}_{fi} = \mathsf{T}_{\text{motion},t} \oplus \left(d_{fi} \, \mathsf{K}^{-1} \begin{bmatrix} \boldsymbol{k}_{fi} \\ 1 \end{bmatrix} \right), \tag{2}$$

where, d_{fi} is the depth value, $\boldsymbol{x}_{fi}$ is the 3D point in world frame corresponding to keypoint $\boldsymbol{k}_{fi}$, and

$$\mathsf{T} \oplus \boldsymbol{p} = \mathsf{T} \begin{bmatrix} \boldsymbol{p} \\ 1 \end{bmatrix} \quad \text{with} \quad \mathsf{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{3}$$

In each iteration of the RANSAC scheme, we randomly sample 3 correspondences from $\mathcal{C}$ to compute $\Delta\mathsf{T}_{\text{ransac},t}$. We apply the transformation $\Delta\mathsf{T}_{\text{ransac},t}$ to the 3D frame points $\boldsymbol{x}_{fi}$ to verify inlier matches within RANSAC. Then, we compute the Euclidean distance to the corresponding map points $\boldsymbol{x}_{mi}$. Matches with distances larger than a predefined inlier correspondence threshold γ are considered outliers. The RANSAC scheme terminates after a fixed number of iterations or if a solution of high enough quality is found, i.e., we obtain more than 50% inliers. The RANSAC algorithm outputs the inlier correspondences $\mathcal{C}^*$ and the motion update $\Delta\mathsf{T}_{\text{ransac,t}}$. Finally, we update our motion prediction to $\mathsf{T}_{\text{ransac,t}} = \Delta\mathsf{T}_{\text{ransac,t}}\mathsf{T}_{\text{motion},t}$.

D. Robust Optimization

To achieve robust incremental global pose estimation T_t, we formulate a projective frame-to-map geometric error between the frame and map points within the inlier set $\mathcal{C}^*$. Then, we minimize this geometric error by employing an iteratively reweighted least-squares algorithm [16]. The choice of minimizing the projective error instead of the point-to-point distance arises due to inconsistencies in the depth measurements of RGB-D cameras. Given the uncertain depth measurements, we incorporate the L1-norm [22] robust kernel to reject outliers.

We formulate the projective error function in Eq. (4) and compute the refined pose estimate T_t by minimizing this error.

$$\sum_{(\boldsymbol{k}_{fi}, \boldsymbol{x}_{mi}) \in \mathcal{C}^*} \rho\left(\left\| \pi\left(\mathsf{K}\left(\mathsf{T}_t^{-1} \oplus \boldsymbol{x}_{mi} \right) \right) - \boldsymbol{k}_{fi} \right\|_2 \right). \tag{4}$$

Here, $\pi(\cdot)$ is a projection function as given by Eq. (5), and $\rho(\cdot)$ is the L1-norm robust kernel given by Eq. (6). We use the motion prediction $\mathsf{T}_{\text{ransac,t}}$ as the initial guess for T_t.

$$\pi(\boldsymbol{x}) = \begin{bmatrix} \frac{x_1}{x_3} & \frac{x_2}{x_3} \end{bmatrix}^\top \quad \text{where} \quad \boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^\top \tag{5}$$

$$\rho(e) = |e| \tag{6}$$

To solve the non-linear error function using the least-squares algorithm, we compute the Jacobian $\boldsymbol{J}$ for each correspondence set as given by Eq. (7), (8) and (9) as follows:

$$\boldsymbol{J}^{[i]} = \boldsymbol{J}_{\pi}^{[i]} \mathsf{K} \boldsymbol{J}_{\mathsf{T}}^{[i]}, \tag{7}$$

$$\boldsymbol{J}_{\mathsf{T}}^{[i]} = \begin{bmatrix} \boldsymbol{I}_{3\times 3} & \left[-\boldsymbol{x}_{mi} \right]_\times \end{bmatrix}, \tag{8}$$

$$\boldsymbol{J}_{\pi}^{[i]} = \begin{bmatrix} 1/x_3 & 0 & -x_1/x_3^2 \\ 0 & 1/x_3 & -x_2/x_3^2 \end{bmatrix}, \tag{9}$$

where $[\cdot]_\times$ transforms a vector to a skew symmetric matrix. Eq. (9) is computed as the Jacobian of Eq. (5).

Then, we update the pose as described in Algorithm 1. The termination criterion for the least-squares algorithm is based on a threshold ϵ over the magnitude of the correction applied to the pose in each iteration without explicitly setting a maximum number of iterations. As a result, we obtain the pose estimate T_t of each consecutive RGB-D sensor frame. Finally, the keypoints in the current frame $\mathcal{F}^*$ are transformed to the world frame using the estimated pose T_t and then are integrated into the local map using a first in, first out approach.

IV. Experimental Evaluation

The main focus of this work is to present a simple yet effective RGB-D odometry pipeline that we implement as described in this paper. We present our experiments that support our key claims: our approach (i) performs on par with state-of-the-art methods in terms of odometry accuracy; (ii) computes the RGB-D odometry faster than sensor frame rate on a single-core CPU, (iii) is significantly easier to understand and run than existing approaches, with a smaller codebase and less software dependencies, and (iv) can operate in real-word scenarios and not just on benchmarks.

		Translation (m)					Rotation (rad)				
Dataset	Sequence	Color-ICP	KISS-ICP	ORB-SLAM3	RTAB-Map	Ours	Color-ICP	KISS-ICP	ORB-SLAM3	RTAB-Map	Ours
TUM	fr1_xyz	0.025	0.011	**0.007**	**0.007**	**0.006**	0.037	0.012	**0.007**	**0.011**	0.013
	fr1_rpy	0.063	0.027	0.014	**0.012**	**0.011**	0.281	**0.029**	**0.014**	0.033	0.033
	fr1_desk	0.180	**0.017**	**0.012**	0.068	0.039	2.372	**0.018**	**0.012**	0.075	0.024
	fr1_desk2	0.180	0.022	**0.013**	**0.011**	0.041	2.335	0.036	**0.010**	**0.015**	0.057
	fr2_xyz	0.078	0.007	**0.002**	0.004	**0.003**	0.048	**0.006**	**0.004**	**0.006**	**0.004**
	fr2_rpy	0.025	0.008	**0.003**	**0.004**	0.008	0.117	0.006	**0.003**	0.005	**0.004**
	fr2_desk	0.213	0.033	**0.004**	0.006	**0.032**	0.113	0.020	**0.005**	**0.007**	0.020
	fr2_desk_with_person	0.242	0.014	**0.005**	**0.006**	0.010	0.105	0.010	**0.006**	**0.008**	0.009
	fr3_nostructure	0.061	0.018	**0.010**	**0.011**	0.013	0.026	**0.013**	**0.010**	0.014	0.014
	fr3_structure	0.050	**0.010**	**0.009**	**0.010**	0.011	0.020	**0.012**	**0.008**	**0.012**	0.014
	fr3_walking_static	0.091	**0.014**	0.022	**0.017**	**0.017**	0.032	**0.007**	**0.007**	**0.008**	**0.008**
Bonn	balloon	0.020	**0.018**	0.021	0.0431	**0.017**	0.019	**0.016**	0.020	0.0566	**0.018**
	crowd3	0.133	**0.021**	0.050	0.068	**0.024**	0.029	**0.013**	0.018	0.0719	**0.016**
	person_tracking	0.054	**0.027**	**0.021**	0.0809	0.033	**0.022**	**0.025**	0.027	0.1695	0.028
	removing_obstructing_box	0.045	**0.011**	0.046	0.0271	**0.010**	**0.026**	**0.011**	0.034	0.0591	**0.011**

TABLE I: Comparison of the relative pose error in translation and rotation for the TUM and Bonn RGB-D Dynamic datasets. The values highlighted with red in each row show the best-performing method on the sequence, and the blue color indicates the second-best method.

A. *Datasets and Metrics*

We evaluate our approach using data from the TUM RGB-D Benchmark [47] and the Bonn RGB-D Dynamic Dataset [39]. To evaluate the performance of odometry, we compute the root mean square of the relative pose error (RPE) in translation and rotation. The ground truth poses are provided with the datasets for evaluation.

We compare our approach with the well-known RGB-D SLAM approach, ORB-SLAM3 [3], by turning off the loop closure module in the pipeline. Similarly, we also compare against the popular open-source SLAM framework, RTAB-Map [28], using its RGB-D odometry component. Furthermore, we compare our approach with recently published approaches, KISS-ICP [50] and Color-ICP [40].

Throughout the experimental evaluation, our approach uses a consistent set of parameters. We start with the sensor depth set to $r_{\text{max}} = 5.0\,\text{m}$ and a camera calibration matrix K specific to the sensor in use. We set the maximum number of ORB keypoints detected in each frame to 1000. For the local map, we set the voxel size to $\nu = 0.5\,\text{m}$, and the maximum number of points in each voxel $N_{\text{max}} = 50$. For geometric verification of correspondences in RANSAC, the distance threshold is $\gamma = 0.2\,\text{m}$. In the optimization loop, the convergence criterion is $\epsilon = 0.001$.

B. *Odometry Performance*

The results of our first experiment support our first claim, namely that our proposed RGB-D odometry pipeline performs on par with existing state-of-the-art systems. To illustrate this, we evaluate the performance of our approach in terms of relative pose error (RPE) on the TUM-RGBD Benchmark (TUM) and Bonn RGB-D Dynamic Dataset (Bonn). Tab. I offer a comparative analysis in terms of relative pose error in translation and rotation, considering consecutive poses. It is important to note that we disable the loop closure module in ORB-SLAM3 and RTAB-Map for a fair evaluation of odometry systems.

In Tab. I, we observe that our method performs on par with state-of-the-art RGB-D odometry systems. Regarding RPE in translation, our approach performs better than all other baselines in 4 out of 15 sequences, and we are second-best performing in 4 out of 15 sequences. However, the accuracy between the two best-performing methods is usually close.

In both experiments, ORB-SLAM3 exhibits better results than our method due to its local bundle adjustment module. We omit a bundle adjustment step in our implementation for simplicity and yet perform comparably to ORB-SLAM3. Our approach closely parallels or surpasses the performance of ORB-SLAM3 and RTAB-Map in most sequences, showcasing its precision and reliability in minimizing drift. We explicitly highlight the performance of our approach in the presence of dynamic obstacles in the challenging Bonn RGB-D Dynamic dataset, where we are better than RTAB-Map and ORB-SLAM3. KISS-ICP shows competitive results in specific sequences in Tab. I because it uses the entire 3D pointcloud information to perform a dense alignment, although at the expense of processing time. Color-ICP's performance is inferior due to its frame-to-frame alignment.

C. *Computational Performance*

Our next experiment evaluates the processing speed of our pipeline, supporting our second claim that our approach can perform RGB-D odometry faster than the sensor frame rate. In the context of RGB-D odometry, a pipeline is considered to operate at the sensor frame rate when it can provide a pose estimate at 15 Hz or every 66 milliseconds. To evaluate the processing time, we run all the sequences 10 times each for every method, capturing the time taken to process each frame and averaging it over all the sequences for both the TUM and the Bonn RGB-D Dynamic datasets. We benchmarked the pipeline on an Intel NUC equipped with an Intel(R) Core(TM) i7-1270P CPU with 16 cores clocked at $4.8\,\text{GHz}$ and $16\,\text{GB}$ of RAM, running Ubuntu 22.04. Note that our approach uses only one core and a single thread.

Dataset	Approach	Processing Time Average (ms)	Standard Deviation (ms)
TUM	Color-ICP	322	62
	KISS-ICP	60	26
	ORB-SLAM3	**14**	**1**
	RTAB-Map	36	5
	Ours	25	8
Bonn	Color-ICP	322	62
	KISS-ICP	160	42
	ORB-SLAM3	**11**	**1**
	RTAB-Map	35	3
	Ours	36	12

TABLE II: Analysis of average processing time over 10 runs.

Codebase	Code-lines	Dependencies	Last commit
ORB-SLAM3	23041	8	3 years ago
Color-ICP	3095	**2**	2 years ago
KISS-ICP	759	3	2 weeks ago
RTAB-Map	6932	13	1 week ago
rgbdslam_v2	8399	7	6 years ago
ElasticFusion	5882	8	2 years ago
VINS-RGBD-FAST	16070	8	1 year ago
proSLAM	5509	10	6 years ago
dvo	3654	7	11 years ago
Ours	**475**	**2**	–

TABLE III: Comparison between the number of lines and mandatory software dependencies, evaluated in September 2024.

We report in Tab. II the mean and standard deviation of the processing time of each approach. Our method performs faster than the sensor frame rate on a single-core CPU, with an average processing time of 25 ms and 36 ms for the TUM and Bonn sequences, respectively. ORB-SLAM3 performs faster than our method, with a mean processing time of 14 and 11 milliseconds. However, ORB-SLAM3 and RTAB-Map use a multithreaded implementation instead of our single-threaded pipeline. Furthermore, our approach, RTAB-Map, and ORB-SLAM3, are the only methods that can process the frames faster than the sensor frame rate.

D. Software Comparison

In the third experiment, we aim to hint at the software implementation to support our claim that the proposed system is simple and easy to use from the software perspective compared to existing methods. To this end, we evaluate the code complexity of existing RGB-D odometry modules. A well-known evaluation criterion in this context is the number of code lines in a software package [38]. For the evaluation, we decided to include just the main library of each software package, as client code might differ in code lines depending on the specific task at hand. In Tab. III, we report the number of code lines for publicly available RGB-D odometry and SLAM systems. To make the evaluation fair, we exclude the loop closure and factor graph optimization modules from the line count when they are present. Furthermore, we report the number of mandatory software dependencies needed to compile the different pipelines as a proxy measurement of how the installation for each package is non-trivial. As we can see from Tab. III, our approach has the least software dependencies and the least code lines. Though we acknowledge that this evaluation is a proxy measurement of code complexity, assessing the complexity a user can face when trying to use a software package is still significant.

Fig. 2: 3D reconstructions of our custom dataset recorded in the robot lab of Institute of Photogrammetry, Bonn. The reconstruction was performed offline by using the poses estimated by our approach.

E. Offline 3D Reconstruction on Custom Dataset

We also evaluate our approach on a custom dataset, recorded using an Intel RealSense D455 camera on a Clearpath Husky robot. The dataset includes changing illumination, varying textures, and dynamic obstacles. This scenario helps demonstrate the capabilities of our approach to data collected by an actual robot rather than just on a benchmark. As we lack the ground truth poses, we perform a qualitative evaluation by reconstructing maps of the recorded environments. We use the poses obtained from our pipeline without any post-processing for noise or drift to generate a dense 3D Map of the environment. Despite the sensor noise in the RGB-D camera, it can be noted in Fig. 2 that the shape of the objects in the reconstructions is still consistent. This result demonstrates that the pose estimation was successful without needing extra parameter tuning in our approach.

V. Conclusion

This paper presents an easy-to-use and understandable yet effective and efficient approach for estimating RGB-D odometry. Our method relies on a few core components to realize the odometry pipeline, embracing simplicity and efficiency. Furthermore, it does not require specialized hardware such as GPUs, simplifying the deployment on real-world robotic platforms. We evaluate our approach on different datasets and provide comparisons to other existing techniques. The experiments suggest that our RGB-D odometry pipeline is on par with state-of-the-art odometry systems and performs well on benchmark datasets as well as real-world data with the same parameter set. Furthermore, our system operates faster

than the sensor frame rate in all presented datasets. Finally, our open-source code is less complex than other baselines, with fewer lines of code and software dependencies. We believe that the proposed system will constitute a new baseline for RGB-D odometry systems, not only in terms of performance but also in software design.

REFERENCES

[1] H. Bay, A. Ess, T. Tuytelaars, and L.V. Gool. Speeded-up robust features (SURF). *Journal of Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[2] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust IndependentElementary Features. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2010.

[3] C. Campos, R. Elvira, J.J.G. Rodríguez, J.M. Montiel, and J.D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Trans. on Robotics (TRO)*, 37(6):1874–1890, 2021.

[4] E. Cervera. Try to Start It! The Challenge of Reusing Code in Robotics Research. *IEEE Robotics and Automation Letters (RA-L)*, 4(1):49–56, 2019.

[5] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[6] A. Concha and J. Civera. RGBDTAM: A cost-effective and accurate RGB-D tracking and mapping system. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[7] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. BundleFusion: Real-time Globally Consistent 3D Reconstruction using Online Surface Re-integration. *ACM Trans. on Graphics (TOG)*, 36(3):1–18, 2017.

[8] K. Eckenhoff, Y. Yang, P. Geneva, and G. Huang. Tightly-Coupled Visual-Inertial Localization and 3D Rigid-Body Target Tracking. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1541–1548, 2019.

[9] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3D Mapping with an RGB-D Camera. *IEEE Trans. on Robotics (TRO)*, 30(1):177–187, 2014.

[10] J. Engel, V. Koltun, and D. Cremers. Direct Sparse Odometry. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(3):611–625, 2018.

[11] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2014.

[12] M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[13] A. Fontán, J. Civera, and R. Triebel. Information-Driven Direct RGB-D Odometry. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[14] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Trans. on Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[15] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Trans. on Robotics (TRO)*, 23(1):34–46, 2007.

[16] G. Grisetti, T. Guadagnino, I. Aloise, M. Colosi, B. Della Corte, and D. Schlegel. Least Squares Optimization: From Theory to Practice. *Robotics*, 9(3), 2020.

[17] T. Guadagnino, X. Chen, M. Sodano, J. Behley, G. Grisetti, and C. Stachniss. Fast Sparse LiDAR Odometry Using Self-Supervised Feature Selection on Intensity Images. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7597–7604, 2022.

[18] S. Gupta, T. Guadagnino, B. Mersch, I. Vizzo, and C. Stachniss. Effectively Detecting Loop Closures using Point Cloud Density Maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[19] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[20] D. Herraez, M. Zeller, L. Chang, I. Vizzo, M. Heidingsfeld, and C. Stachniss. Radar-Only Odometry and Mapping for Autonomous Vehicles. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[21] C. Hertzberg, R. Wagner, O. Birbach, T. Hammer, and U. Frese. Experiences in Building a Visual SLAM System from Open Source Components. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[22] P.J. Huber. *Robust Statistics*. Wiley, 1981.

[23] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[24] C. Kerl, J. Sturm, and D. Cremers. Robust Odometry Estimation for RGB-D Cameras. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[25] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[26] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proc. of the Intl. Symp. on Mixed and Augmented Reality (ISMAR)*, 2007.

[27] L. Kneip, D. Scaramuzza, and R. Siegwart. A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[28] M. Labbé and F. Michaud. RTAB-Map as an Open-Source Lidar and Visual Simultaneous Localization and Mapping Library for Large-Scale and Long-Term Online Operation. *Journal of Field Robotics (JFR)*, 36:416–446, 2018.

[29] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *Intl. Journal of Computer Vision (IJCV)*, 81:155–166, 2009.

[30] K. Liu, X. Gu, M. Yang, Y. Zhang, and S. Guan. Semi-Direct Tracking and Mapping with RGB-D Camera. In *Intelligent Robotics and Applications*, 2019.

[31] D. Lowe. Object recognition from local scale-invariant features. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 1999.

[32] L. Ma, C. Kerl, J. Stückler, and D. Cremers. CPA SLAM: Consistent Plane-Model Alignment for Direct RGB-D SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[33] M. Muglikar, A. Zhang, and D. Scaramuzza. Voxel Map for Visual SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[34] R. Mur-Artal, J. Montiel, and J.D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. on Robotics (TRO)*, 31(5):1147–1163, 2015.

[35] R. Mur-Artal and J. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. on Robotics (TRO)*, 33(5):1255–1262, 2017.

[36] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proc. of the Intl. Symp. on Mixed and Augmented Reality (ISMAR)*, 2011.

[37] R. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense tracking and mapping in real-time. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2011.

[38] A. Oram and G. Wilson. *Making Software: What Really Works, and Why We Believe It*. O'Reilly Media, Inc., 1st edition, 2010.

[39] E. Palazzolo, J. Behley, P. Lottes, P. Giguere, and C. Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[40] J. Park, Q. Zhou, and V. Koltun. Colored Point Cloud Registration Revisited. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[41] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. In *Proc. of the ICRA Workshop on Open Source Software*, 2009.

[42] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2011.

[43] D. Schlegel, M. Colosi, and G. Grisetti. ProSLAM: Graph SLAM from a Programmer's Perspective. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[44] D. Schlegel and G. Grisetti. Visual Localization and Loop Closing Using Decision Trees and Binary Features. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[45] D. Sharafutdinov, M. Griguletskii, P. Kopanev, M. Kurenkov, G. Ferrer, A. Burkov, A. Gonnochenko, and D. Tsetserukou. Comparison of Modern Open-source Visual SLAM Approaches. *Journal of Intelligent and Robotic Systems (JIRS)*, 107(3):43, 2023.

[46] F. Steinbrücker, J. Sturm, and D. Cremers. Real-Time Visual Odometry from Dense RGB-D Images. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2011.

[47] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[48] Z. Teed and J. Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. In *Proc. of the Conf. Neural Information Processing Systems (NIPS)*, 2021.

[49] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 13(4):376–380, 1991.

[50] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023.

[51] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. of Robotics: Science and Systems (RSS)*, 2014.

[52] S. Zhang, L. Zheng, and W. Tao. Survey and Evaluation of RGB-D SLAM. *IEEE Access*, 9:21367–21387, 2021.

[53] Q. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv preprint*, arXiv:1801.09847, 2018.