SRVIO: Super Robust Visual Inertial Odometry for Dynamic Environments and Challenging Loop-Closure Conditions

Ali Samadzadeh and Ahmad Nickabadi

Abstract—There has been extensive research on visual localization and odometry for autonomous robots and virtual reality during the past decades. Traditionally, this problem has been solved with the help of expensive sensors, such as light detection and ranging (LiDAR). Nowadays, the focus of the leading research in this field is on robust localization using more economic sensors, such as cameras and inertial measurement units. Consequently, geometric visual localization methods have become more accurate over time. However, these methods still suffer from significant loss and divergence in challenging environments, such as a room full of moving people. Scientists started using deep neural networks (DNNs) to mitigate this problem. The main idea behind using DNNs is to better understand challenging aspects of the data and overcome complex conditions such as the movement of a dynamic object in front of the camera that covers the full view of the camera, extreme lighting conditions, and the high speed of the camera. Prior end-to-end DNN methods did overcome some of these challenges. However, no general and robust framework is available to overcome all challenges together. In this article, we have combined geometric and DNN-based methods to have the generality and speed of geometric SLAM frameworks and overcome most of these challenging conditions with the help of DNNs and deliver the most robust framework so far. To do so, we have designed a framework based on VINS-Mono and shown that it can achieve state-of-the-art results on TUM-Dynamic, TUM-VI, ADVIO, and EuRoC datasets compared to geometric and end-to-end DNN-based simultaneous localization and mappings. Our proposed framework can also achieve outstanding results on extreme simulated cases resembling the aforementioned challenges.

Index Terms—Deep learning, inertial navigation, semantic segmentation, simultaneous localization and mapping, visual odometry.

I. INTRODUCTION

VISUAL inertial odometry (VIO) is the process of determining the location and orientation of an entity with the help of visual (camera) and inertial (inertial measurement unit (IMU)) sensors. A simultaneous localization and mapping (SLAM) framework is a system that has a global localization algorithm coupled with visual odometry [1]. Today, SLAM and VIO systems are being used in a wide variety of applications, including autonomous driving [2], [3], controlling and navigating aerial microautonomous vehicles and nanodrones [4], [5], [6], augmented reality, and enabling navigation on mobile devices [7]. In this article, it is assumed that the target entity only has a single RGB camera and a single IMU sensor.

The general pipeline of a modern SLAM framework consists of three major parts [8]: preprocessing, optimization, and pose graph optimization and loop closure. The preprocessing component typically consists of visual feature tracking and IMU preintegration between frames. There are direct [9], [10], keypoint-based [8], [11], and deep-learning-based [12], [13] visual feature tracking methods.

The optimization component typically starts with an initialization stage in which, algorithms like five-point [14] and triangulation are exploited to create an initial 3-D point cloud and reconstruct the initial poses. The process is also known as visual structure-from-motion (SfM) [15]. After initial SfM, the extracted poses and 3-D point cloud locations are fine tuned using bundle adjustment, and then, the visual data is matched with inertial data by estimating the scale and initial gravity vector. Finally, the optimization process starts based on visual and inertial residuals. These residuals are prominent in state-of-art VIO and SLAM frameworks [8], [10], [16].

Loop closure and pose optimization is the last standard component of SLAM frameworks. The goal of this component is to globally optimize the poses using the information provided by loop closure and map reuse mechanisms. The loop closure mechanism is a tool for reducing long-term error and improving the robustness of the models. It is claimed to be a crucial part of modern SLAM frameworks [11]. Fig. 1 shows an example of loop closure in a location visited both during the day and night. The three aforementioned components of a VIO system can be executed in parallel.

Despite the success of the current state-of-the-art three-component SLAM models, they fail dramatically in some challenging situations. For example, the presence of dynamic objects in the field of view of the camera violates the scene rigidity assumption of the base SLAM frameworks and degrades the visual features tracking quality. The presence of moving objects...
makes it difficult to build stable and accurate 3-D point cloud and poses. There have been many attempts, such as DynaSlam [17], to extend monocular and stereo SLAM to dynamic scenarios.

Also, SLAM methods often struggle to keep track of visual features when the tracked entity moves quickly, resulting in blurry or completely new frames. Another major challenge is the existence of external objects that cover most of the camera’s field of view and generate false visual cues of movement. In such cases, the visual odometry either provides wrong estimates or resorts to movements priors like assuming zero- or constant-speed movement.

Stable long-term localization is another challenge of SLAM as there are issues with inertial sensors and visual tracking. Regarding the inertial sensors, integration over time diverges due to the accumulation of errors in measurements. On the other hand, the loop closure, which plays a vital role in stabilizing SLAM systems in long time usages, fails to match photos of the same scene taken at different times (e.g., day and night) or viewing angles (e.g., entrance and exit directions of an area). These situations are almost impossible for the classical K-nearest neighbor (KNN) feature match and perspective-n-point (PnP) to handle.

In this article, we propose a novel SLAM framework to address the aforementioned challenges altogether. The main contributions of our proposed SLAM framework can be summarized as follows.

1) A hybrid visual preprocessing step is proposed to overcome the problem of dynamic objects at real-time speed with a higher recall rate compared to just using a semantic segmentation neural network (SSNN) for dynamic object filtering.

2) The optimization component is modified to be capable of switching between different choices of useful data sources (including visual, inertial, or both of them) to stay stable and provide accurate estimations in extremely dynamic scenes.

3) An inertial preprocessing component is introduced that solves the integration output divergence problem while it provides real-time output.

4) Finally, a novel transformer-based loop closure mechanism is employed for robust localization in day–night and enter–exit scenarios with a high recall rate and the same precision compared to competitors.

The rest of this article is organized as follows. Section II reviews the related works. Section III gives a full description of the proposed super robust visual inertial odometry (SRVIO) model. Section IV reports the results of the experiments. Finally, Section V concludes this article.

II. RELATED WORKS

In this section, we discuss the literature on SLAM and VIO frameworks and the methods for addressing the aforementioned challenges. First, in Section II-A, the most recent VIO and SLAM frameworks that utilize a single camera and a single IMU are reviewed. In Section II-B, methods that deal with moving objects and dynamic environments will be discussed. In Section II-C, the problem of inertial data integration and odometry divergence is explained. Also, the frameworks that have tackled this problem are described. Finally, the old problem of robust relocalization in challenging scenarios and the solutions provided for these situations are discussed in Section II-D.

A. Single-Camera Single-IMU SLAM or VIO Frameworks

There are many robust and successful frameworks that have been designed for single-camera single-IMU VIO, which can be divided into two categories: geometric and neural network-based methods. The former relies on traditional visual feature extraction methods, while the latter utilizes deep neural networks for feature extraction. One of the most famous geometric SLAM frameworks is VINS-Mono [8]. VINS-Mono’s preprocessing steps are well-known algorithms; keypoints are extracted using the good-features-to-track algorithm [18] and are tracked in video frames using the Lucas–Kanade [19] method. The optimization and initialization components of VINS-Mono are especially designed for this framework and play an essential role in its success. Despite the speed and accuracy of VINS-Mono in usual scenarios, it is not very stable in some challenging environments.

ORB-SLAM3 is another state-of-the-art geometric visual-inertial simultaneous localization and mapping (VI-SLAM) framework. ORB-SLAM3 is more robust than VINS-Mono because of addressing camera blur and visual feature loss, and IMU-camera translation matrices error. However, despite its robustness, it does not provide reliable outputs when dynamic objects appear in the scene. The loop closure of ORB-SLAM3 is similar to the classic PnP and the descriptor check method. This checking method has its downsides when the loop closure is required in day–night and enter–exit scenarios.

VI-DSO [9] is another well-known geometric VI odometry algorithm that is based on sparse and direct visual feature tracking between the frames and does not extract keypoints. This framework is also fast and accurate but has the same shortcomings as other geometric SLAM frameworks. Generally, geometric SLAM algorithms are fast, hand-engineered, and robust to standard conditions, but they fail to provide a robust localization in the extreme conditions stated in the previous section.

Recently, there has been an increasing interest in developing DNN-based visual odometry frameworks. UnDeepVO [20] was the first visual odometry framework based on DNNs. The first attempts did not have the same accuracy and robustness as the geometric SLAMs. However, more recent frameworks such as VINet [12] and D3VO [13] show promising results. VINet is
a VIO model that formulates motion estimation as a sequence-to-sequence problem and uses an end-to-end network to solve this problem. The visual preprocessing is done by a FlowNet convolutional neural network (CNN) in which the optical flow between frames is extracted, and the inertial preprocessing for integrating IMU data is accomplished via an LSTM neural network. ViNet has achieved state-of-the-art performance but still suffers from robustness issues in the challenging environments mentioned in Section I.

D3VO is a visual-only method that is more stable than ViNet. The stability of this approach comes from a neural network (NN) that estimates the depth of each frame and outputs the uncertainty coefficient of each pixel. This depth estimation NN uses the Bayesian filtering method introduced in [21] to solve the problem of image depth estimation in occluded and dynamic environments. The model shows outstanding performance and accuracy over various datasets. D3VO does not have an inertial data input or a loop closure mechanism (it is an odometry system, not a SLAM system). Therefore, it diverges in long-term scenarios and can fail in other challenging situations like when objects cover the camera view or extreme motion blur.

**B. Dynamic Robust SLAM Frameworks**

Today, almost all SLAM and VIO models are equipped with mechanisms to deal with nonrigid environments. One of the most straightforward methods uses random sample consensus (RANSAC) to detect outliers [16], [22]. Another reasonably simple method is to use a robust error function [23]. Unfortunately, these approaches fail when the majority of the keypoints belong to moving objects.

The more complex models tackle dynamic objects by identifying and dismissing dynamic keypoints or dynamic objects entirely. To identify moving objects, the common approach is to use SSNN to remove the areas belonging to dynamic object classes. Mask-SLAM [24] uses a mask produced by an SSNN to exclude undesirable feature points. DS-SLAM [25] is equipped with a real-time SNN running in an independent thread, which is coupled with moving consistency checking methods, enabling it to filter out the dynamic portion of the scene. Similarly, PSPNET-SLAM [26] uses an SSNN (PSPNET [27]) combined with ORB-SLAM2 to achieve the same goal. RGB-D SLAM [28] filters out dynamic keypoints instead of objects by only using depth edge points for visual odometry. These depth edges have static weights that indicate each point’s probability of belonging to a static object.

Dyna-SLAM [17] applies pixel-wise segmentation using a CNN to detect dynamic objects for the monocular and stereo cases. In the RGBD case, multiview geometry and deep learning are used to remove dynamic objects and in-paint the background occluded by these objects. Finally, Dynamic-SLAM [29] consists of a single shot detector (SSD) object detector for detecting dynamic objects, a missed detection compensation algorithm to improve the recall rate of the SSD component, and a feature-based visual SLAM that correctly handles the feature point of the dynamic objects.

Most of the methods mentioned previously achieve real-time dynamic object filtering. Nevertheless, when the dynamic object covers the whole field of view in extreme cases, the frameworks’ outputs diverge or show motion prior. This article shows that using another information source like the inertial sensor’s data can help to correct motion estimation errors in these extreme situations.

**C. Inertial Odometry (IO)**

Integrating inertial data (linear acceleration and rotation speed) provided by the IMU sensor is an excellent odometry method for short periods. However, in long term, the output diverges dramatically, as discussed and demonstrated in [30]. Some methods, such as [31], try to limit IMU integration error over time, but they are designed for particular applications such as pedestrian movements and are not very accurate as well. With the advances in deep learning methods in processing time-series data, new methods emerged to solve this problem known as IO.

One of the first successful algorithms, which used DNNs to cure the curse of drift in IO was IONet [32]. This method, with proper initialization, performs close to industrial SLAM frameworks such as Google Tango. IONet uses an LSTM-based architecture and polar coordinates vectors to estimate the pose at each step. It outperforms the previous classical attempts by a large margin.

The other idea was the use of denoising CNNs. For example, Bossard et al. [33] try to remove the noise that causes the integration to drift. Such methods require calibration data from a stationary entity with the IMU sensor. Almost all new emerging deep neural-network-based V1 SLAM/odometry frameworks have a performance or throughput gap compared to modern visual or visual-inertial geometric SLAM/odometry frameworks, but fusing them with geometric SLAM’s preprocessing units can show promising results over challenging scenarios. In this article, we have designed a denoising CNN to do the aforementioned task and have combined it with a geometric SLAM (details provided in Section III).

**D. Robust Relocalization**

Robust relocalization over extreme scenarios such as day–night, rainy or snowy weather, seasonal changes, extreme viewpoint changes, and similar scenarios is a big challenge in SLAM. So far, many methods have tried to overcome some of these challenges. The most interesting recent methods are X-view [34] and SuperGlue [35]. The X-view method uses SSNN over frames to build a graph of objects over frames; then, relocalization is done via subgraph matching. The method is interesting because it can solve almost all relocalization challenges, especially extreme viewpoint changes. The method can even relocalize the same place seen on the ground from aerial images. However, this method is not accurate and gives a rough estimation of the location and not a transformation matrix.

The SuperGlue provides an exact transformation between two frames of a video. This method is based on Superpoint [36] visual keypoint extractor and descriptor proposed by the same team. It uses graph transformers to find appropriate matches between the corresponding keypoints in frames and shows robustness over day–night conditions and rainy weather, but it does not show robustness over seasonal changes. Despite the success of
Superglue at the accurate estimation of transformation between images with robustness over extreme changes, it has never been used in a classical loop closure mechanism inside geometric SLAM. In this article, we use the Superglue and Superpoint inside a novel loop closure mechanism to improve the relocalization robustness and recall rate of our model while maintaining almost 100% precision.

III. SRVIO FRAMEWORK

As demonstrated in Fig. 2, the SRVIO consists of three main components: preprocessing, optimization, and pose optimization and loop closure. All of the components are reimplemented and redesigned and are different from the original VINS-Mono. However, this framework shares some identical module implementations with VINS-Mono including the pose graph optimization, the initialization module, the IMU preintegration, and the keyframe database. VINS-Mono is used as the base algorithm of SRVIO. To the best of our knowledge, VINS-Mono was the best-performing open-source algorithm with modular code design. The input to this framework is a single stream of frames and IMU data. The final output is the pose of the camera in each keyframe and the corresponding 3-D point cloud. In addition to the poses and locations, each frame’s keypoints are reported in the form of pixel locations and their descriptors. In the following subsections, each component of the model is discussed in detail.

A. Preprocessing

The preprocessing module of SRVIO is made up of two visual preprocessing and inertial preprocessing blocks, as seen in Fig. 2.

1) Visual Preprocessing: The visual preprocessing block of our SLAM system takes in a sequence of frames \(I_i\) and outputs corresponding keypoints matrices \(\mathcal{P}_i^{\text{vis}}\) between each pair of consecutive frames, as well as a weight for each frame, which represents the quality of the keypoints in that frame. To do this, it first employs a keypoint extractor, good-features-to-track [18], to track previous frame’s keypoints and extract new keypoints from the regions of frame \(i\) in which the old keypoints of frame \(i-1\) do not exist. The result is the set of initial keypoints of the current frame as \(\mathcal{P}_i^{\text{initial}} = \text{GoodFeaturesToTrack}(I_i, \mathcal{P}_i^{\text{vis}})\).

If keypoints extracted from the regions occupied by non-stationary objects are used, they will provide misleading clues for motion estimation. Hence, the initial keypoint set is then matched against the output of HRNet [37] as SSNN to decide whether they are possibly dynamic (like keypoints on humans, cars, and trains) or static (like keypoints on buildings or streets). This substep is known as the semantic filter in our model. The SSNN segments the input image into nonoverlapping regions labeled as different items of a predefined set of objects. The output of the SSNN is then converted to a binary mask indicating static and dynamic regions of the input frame. The dynamic region is then dilated to contain the borderline pixels, preventing the model from defining keypoints on these lines.

The SSNN can be used inside a SLAM framework by various approaches [17], [38]. The output of this SSNN is used for filtering out the undesired (dynamic) keypoints. However, it is also possible to use the SSNN output before the keypoint extraction step to restrict the search for keypoints to the stationary regions of the input frame. However, this may cause the model to be unstable as there might be many potentially dynamic objects (e.g., humans) that are already static in the scene, and rejecting all of the keypoints on these objects remains very few keypoints for proper tracking and optimization.

The SSNN is employed only when new keypoints are required (the new keypoints required when the objects get out of the scene or get covered). Hence, in most frames, keypoint tracker module just tracks the keypoints on static objects. The keypoint selection mechanism is also accurate since it allows the use of the keypoints that are wrongly labeled as dynamic, e.g., points on steady cars. To do this, SRVIO employs a three-step algorithm. First, RANSAC [39] is applied only to the keypoints labeled as static by the SSNN to find the initial
fundamental matrix transformation \( F_{i,i-1} \) between frame \( i \) and \( i-1 \). Next, a test is performed to check the consistency of all keypoints, including those labeled as dynamic objects by the segmentation network, with this initial fundamental matrix and only the truly moving keypoints are removed. Finally, the RANSAC is again applied to the new set of keypoints. These three steps can be formulated as follows:

\[
[\mathcal{P}^{\hat{C}_i}_{1,1}, F_{i,i-1}^{\hat{C}_i}] = \text{RANSAC} (\mathcal{P}^{C}_{initial}, \mathcal{P}^{C}_{i,i-1}) \quad (1)
\]

\[
\mathcal{P}^{C}_{i,2} = \left\{ \mathcal{P}^{C}_{i,initial} : |\mathcal{P}^{C}_{i,initial} F_{i,i-1}^{\hat{C}_i} P_{initial}^{-1} | < \epsilon \right\} \quad (2)
\]

\[
[\mathcal{P}^{C}_i, F_{i,i-1}^{\hat{C}_i}] = \text{RANSAC} (\mathcal{P}^{C}_{i,2}, \mathcal{P}^{C}_{i,i-1}) \quad (3)
\]

where \( \mathcal{P}^{C}_{initial} \) is the initial keypoint extracted from good-features-to-track algorithm, \( F_{i,i-1}^{\hat{C}_i} \) and \( F_{i,i-1}^{\hat{C}_i,1} \) are the fundamental matrix and the keypoint matrix corresponding to the initial static keypoints. Equation (2) checks the fundamental matrix test with the corresponding keypoints of \( i \)th frame and static keypoints of \( (i-1) \)th frame. The output \( (\mathcal{P}^{C}_{i,2}) \) is the new set of keypoints including the keypoints on static objects initially mislabeled as dynamic. Finally, in order to accurately obtain nondynamic keypoints and remove the outliers and dynamic keypoints, the RANSAC algorithm is performed again in (3). The \( \mathcal{P}^{C}_i \) is the final matrix of keypoints. Fig. 3 shows three examples of keypoint extraction and filtering in SRVIO along with the corresponding static/dynamic masks. The green, blue, and red dots on the frames are filtered, tracked, and not-tracked keypoints, respectively. The white (black) regions of the mask show static (dynamic) objects. As shown in this figure, the keypoints on the steady cars in the left picture are not filtered even though the semantic mask labels them as dynamic, while the keypoints on the train in the right picture are not used as expected.

After eliminating the moving keypoints, a weight parameter \( \Psi_{c_i} \) is computed for each frame that determines the quality of the keypoints of that frame. This frame weight is used in the optimization component and is defined as follows:

\[
\Psi_{c_i} = \frac{|\mathcal{P}^{C}_i|}{F_{max}} \quad (4)
\]

where \(| \cdot | \) denotes the size of the set of all remained keypoints inside the \( i \)th frame. The \( F_{max} \) is a constant indicating the maximum number of keypoints that can be present in a frame (experimentally set to 250 in this article). Accordingly, the frame weight is defined as the number of selected static points in proportion to the maximum points extracted by the keypoint extractor algorithm for a frame. This weight helps to decrease attention to the visual error where the visual clues are unreliable because the dynamic objects are covering the camera’s view field or the keypoint extractor is unable to extract a proper number of keypoints due to motion blur or similar challenges inside frames.

2) Inertial Preprocessing: In the inertial preprocessing component, a denoising convolutional neural network (DCNN) is used to help eliminate unwanted IMU noise and mitigate the divergence issue. As shown in Fig. 4, the proposed DCNN consists of two clone networks: one for gyro and the other for the accelerometer data denoising. The gyro DCNN gets \( N \) sequential IMU data from time steps \( i - N \) to \( i \) and outputs gyro correction \( \hat{\omega}_i \) for the \( i \)th IMU measurement. The accelerometer DCNN is the same as the gyro DCNN except that it is trained to estimate the accelerometer correction \( \hat{a}_i \) for the \( i \)th IMU data. The correction fixes the IMU bias and random walk noise by separating the noise part of the noisy IMU data. The correction is performed as follows:

\[
\hat{C}_{\omega} = \hat{S}_{\omega} M_{\omega} \cdot \hat{\omega}_i = \hat{C}_{\omega} \cdot \omega_{i_{\text{MU}}} + \hat{\omega}_i \quad (5)
\]

\[
\hat{a}_i = \hat{C}_{a} \cdot a_{i_{\text{MU}}} + \hat{a}_i \quad \hat{C}_{\omega} = \zeta_{\omega}^a + \zeta_{\omega}^a \quad (5)
\]

where \( M_{\omega} \) is axis misalignment matrices and \( \hat{S}_{\omega} \) are scale factors; therefore, \( \hat{C}_{\omega} \) is the correction matrix of the raw input data. These parameters are unique for each IMU and are learned for each dataset. The \( M_{\omega}, \hat{\omega}_i, \) and \( \hat{\omega}_i \) correspond to IMU raw measurements, denoising correction values of the corresponding DCNN, and the final corrected values, respectively. \( \zeta_{\omega}^a \) and \( \zeta_{\omega}^a \) are the accelerometer and gyro data quality scores for \( i \)th IMU.
data. This score improves the framework’s robustness over faulty IMU sensors and similar challenges.

The denoising networks of our model (see Fig. 4) are trained in the following manner. First, the convolution layers are trained using the ground truth (GT) of the locations of the IMU calibration sequence. The raw IMU data are filtered according to (5), and then, integrated and compared to the GT. Finally, the loss is backpropagated to train the convolution layers. Then, the convolutional layers are frozen and the linear layers are trained with the same IMU calibration sequence. However, this network is trained to predict the IMU data quality. This score improves the framework’s robustness over faulty IMU sensors and similar challenges.

Moreover, since ground truth (GT) is not available with high frequency, the corrected IMU (gyro and accelerometer) data are integrated through time to lower the frequency and reach the following manner. First, the convolution layers are trained after the gyro denoising network training in order to have an accurate rotation matrix per data (exp(ωi)). The ρ(.) is the Huber-norm defined in (12). The final loss of each network is defined as the sum of the losses for 16 and 32 timesteps.

Hereafter in SRVIO, corrected values of the accelerometer and gyro are regarded as the IMU measurements on which the preintegrations and integrations are performed with the assumption that bias and noise standard deviation is near zero. Using the IMU data validity score (ζk), defined in (5), the IMU attention weight is defined as follows:

\[
\Psi_{bi} = \frac{\sum_{k \in M_{ij}} \zeta_k}{m}
\]

where \(M_{ij}\) is the set of IMU data inputs between \(i\)th frame and \(j\)th frame, and \(m\) is the size of this set. This weight parameter may be set to 1 if the user is confident about the IMU data quality.

B. Optimization

The location and pose estimation of SRVIO starts with an estimator initialization step and continues with an optimization-based VI-SLAM procedure combined with loop closure detection. The initialization component of our model is similar to those of the previous frameworks like VINS-Mono whose details are given in [8]. Thus, in this section, we only focus on the details of our proposed optimization module. In the following, we first define the residual terms for visual and IMU measurements and then introduce the final error function.

The formulas of this section use the following notations: \((\cdot)^w\), \((\cdot)^b\), and \((\cdot)^c\) represent the world frame, the body frame (i.e. the IMU frame), and the camera frame, respectively. \(R\) and \(q\) are used as matrix and quaternion of rotation, respectively. The subscript for \(R\) or \(q\) indicates the original frame, while the superscript denotes the destination frame. \(p\) is used to show translation. \(b_k\) and \(c_k\) are used to show the body frame and camera frame while taking the \(k\)th image. \(\otimes\) is used to denote multiplication operations between two quaternions. \(g^v = [0, 0, g]^T\) is the gravity vector. Finally, \(\hat{\cdot}\) denotes a noisy measurement.

1) IMU Measurement Residuals: Let us denote accelerometer and gyroscope denoised values by \(\hat{a}\) and \(\hat{\omega}\). The IMU preintegration terms are then defined as

\[
\alpha_{bi}^{b_k} = \int_{t \in [t_k, t_{k+1}]} \frac{R_k^{b_k}(\hat{a}_t)}{dt^2}
\]
\[
\beta_{bi}^{b_k} = \int_{t \in [t_k, t_{k+1}]} \frac{R_k^{b_k}(\hat{\omega}_t)}{dt}
\]
\[
\gamma_{bi}^{b_k} = \int_{t \in [t_k, t_{k+1}]} \frac{1}{2} \Omega(\hat{\omega}) \gamma_k^{b_k} dt
\]

where \(\Omega(.)\) is

\[
\Omega(\omega) = \begin{bmatrix} -\omega_y & \omega_x & -\omega_z \\ -\omega_z & 0 & \omega_y \\ \omega_x & -\omega_y & 0 \end{bmatrix}
\]

The residual for two IMU measurements in consecutive frames at body (IMU) coordinates \(b_k\) and \(b_{k+1}\) in the sliding
window can be written as

\[
\begin{bmatrix}
R^b \left(p^w_{b_k+1} - p^w_{b_k} + \frac{1}{2}g^w \Delta t_k - v^w_{b_k} \Delta t_k \right) - \hat{\chi}^b_{b_k+1}
\end{bmatrix}
\]

(16)

where \(\Delta t\) is the time interval, \(v\) denotes velocity, \(\hat{\cdot}\) are residuals for visual and IMU measurements, respectively, and \(\delta \beta^b_{b_k+1}\) is the 3-D error state representation of the quaternion. When the DCNN is enabled, the IMU biases are not optimized and bias terms will be considered constant and zero.

2) Visual Measurement Residual: The camera model defines camera measurement residuals on a generalized unit sphere in contrast to the traditional pinhole camera model. Considering the \(l\)th feature, which appears at the \(j\)th residual, the residual for observing that feature at the \(j\)th image is written as

\[
r^c(\hat{z}^c_j^l, \chi) = |b_1 b_2|^T, \quad (\mathcal{P}^c)^j_i = \frac{|\mathcal{P}^c|^j_i}{|\mathcal{P}^c|}
\]

\[
\mathcal{P}^c_{l} = \pi^c_{l} \left( \begin{array}{c}
\frac{u^c_i}{v^c_i} \\
\frac{v^c_i}{u^c_i}
\end{array} \right)
\]

(17)

where \(\pi^c_{l}\) denotes the first observation of \(l\)th feature in the \(i\)th image, while \(\pi^c_{l}\) is the observation of that feature in the \(j\)th image. \(\pi^c_{l}\) is the back projection function, which given a pixel location yields the corresponding unit vector using camera intrinsic parameters. The residual vector is projected onto the tangent plane since the vision residual has two degrees of freedom. \(b_1\) and \(b_2\) are bases spanning the tangent plane \(\mathcal{P}^c_{l}^j\).

3) Full Error Function: The full state of the window denoted by \(\chi\) can be written as

\[
\chi = [x_0, x_1, \ldots, x_n, x^b_c, \lambda_1, \ldots, \lambda_m]
\]

(18)

\[
x^k = [p^w_{b_k}, v^w_{b_k}, q^w_{b_k}], \quad \in [0, n]
\]

(19)

\[
x^b = [p^b_{b_k}, q^b_{b_k}]
\]

(20)

where \(x^k\) is the IMU state when the \(k\)th image is captured, \(p^w_{b_k}, v^w_{b_k}, q^w_{b_k}\) denote position, velocity, and the orientation of the body in the world frame at the time \(k\), respectively. \(n\) indicates the number of keyframes in the sliding window, while \(m\) is the number of features in that window. Finally, \(\lambda_i\) is the inverse depth of the \(l\)th feature in the sliding window from its first observation.

The base error function used in VINS-Mono [8] can then be written as

\[
\mathcal{R} = \|r_p - H_p \chi\|^2 + \sum_{k \in B} \|r^b_k(\hat{z}^b_{b_k+1}, \chi)\|_{p^b_{b_k}}^2 + \sum_{l, j \in C} \rho \left( \|r^c(\hat{z}^c_j^l, \chi)\|_{f^c_l}^2 \right)
\]

(21)

where \(r^b_k(\hat{z}^b_{b_k+1}, \chi)\) and \(r^c(\hat{z}^c_j^l, \chi)\) are residuals for visual and IMU measurements, respectively, and \(C\) is the set of features that have been observed at least twice in the current sliding window. \(r_p\) and \(H_p\) are prior information from marginalization and \(\rho(.)\) is the Huber norm [defined in (12)].

The Huber function \(\rho\) penalizes residuals contributed by outliner features, but it assigns equal importance to both camera and IMU residuals. However, camera measurements usually yield a more satisfying result when sufficient keypoints are present in the scene. IMUs are usually vulnerable to shifts over long periods, but they exhibit precise performance during short periods. The goal of the optimization component of SRVIO is to utilize the potential of each of the measurements to compensate for the deficiencies of the other measurement source in specific scenarios. To achieve this goal, a novel weighted error function is proposed as follows:

\[
\mathcal{R}_2 = \|r_p - H_p \chi\|^2 + \sum_{k \in B} \Psi_{b_k} \|r^b_k(\hat{z}^b_{b_k+1}, \chi)\|_{p^b_{b_k}}^2 + \sum_{l, j \in C} \Psi_{b_k} \rho \left( \|r^c(\hat{z}^c_j^l, \chi)\|_{f^c_l}^2 \right)
\]

(22)

This error function assigns a weight to each of the visual and inertial terms according to the number of keypoints present in the current window [see (22)]. The camera residual weight increases in proportion to the number of visual keypoints in the frame window and causes the model to rely on the visual clues in windows with a large number of keypoints. On the other hand, the error function mainly depends on IMU residual in frame windows with few keypoints.

C. Pose Graph Optimization and Loop Closure

The pose graph optimization and loop closure component is essential for the long-term accuracy of trajectory estimation as it corrects the large drifts that may occur over time. The algorithm needs to memorize the previous scenes it has visited to detect the loops. However, raw frames are not saved; instead, a bag-of-words representation of keyframes is stored in a visual database to decrease memory usage. To do this, a set of keypoints are detected in each keyframe. Each keypoint is then described as a visual word using the Superpoints keypoint descriptor [36]. Once a loop, i.e., the correspondence between the input frame and a frame in the keyframe, is detected, the estimated path is optimized to match the detected loop.
As shown in Fig. 2, the pose graph optimization and loop closure component consists of two blocks: loop closure detection block and pose graph optimization block, which closes the loops and enables map reuse. The loop closure component performs a yaw and translation optimization defined as follows:

\[
\min_{p, \psi} \left\{ \sum_{(i,j) \in E} \| r_{i,j} \|^2 \right\}
\]

\[
P_{i,j} = \tilde{R}_{i}^{w^{-1}} (P_{j}^{w} - \tilde{P}_{i}^{w})
\]

\[
\tilde{\psi}_{i,j} = \tilde{\psi}_{j} - \tilde{\psi}_{i}
\]

\[
r_{i,j}(P_{i}^{w}, \psi_{i}, P_{j}^{w}, \psi_{j}) = \left[ \tilde{R}_{i}^{w^{-1}} (P_{j}^{w} - P_{i}^{w}) - \tilde{P}_{i,j} \right] \tilde{\psi}_{j} - \tilde{\psi}_{i} - \tilde{\psi}_{i,j}
\]

where \( E \) is the set of all pose graph edges and (\( \tilde{\cdot} \)) indicates a constant value. Additionally, \( \tilde{\psi}_{i,j} \) is the constant previous yaw difference between two graph nodes.

The VINS-Mono algorithm divides (23) into loop closed edges and sequential edges to reduce the risk of wrong loop closure. However, the Superglue method is very robust, and most of the correspondences result in correct relocalization, and there is no need for such division.

The second block in this component is the loop closure detection block. The loop closure detection block is crucial for maintaining long-term trajectory estimation accuracy. This block should be robust to challenges such as light source, day/night, and viewpoint changes. Previous methods (e.g., VINS-Mono [8]) use some simple keypoint extractors and KNN mapping methods for this purpose. These algorithms suffer in performance when facing the aforementioned challenges. SRVIO, on the contrary, introduces a modern hybrid and much more robust loop detection mechanism. In this method, the initial keypoints are first extracted using the Superpoint model [36]. The keypoints are then filtered using the SSNN of the visual preprocessing component. In this stage, there is no need for checking the fundamental matrix test to add semidynamic objects’ keypoints because they may not stay steady in the long term.

Subsequent to this stage, the descriptors of keypoints of each keyframe are stored in a DBoW2 [41] database to detect new loops in the future. Whenever a new keyframe comes in, its \( K \) closest keyframes from the DBoW2 database are sought using a weak KNN algorithm. If the number of matched descriptors (keypoints) between the input frame and any of these keyframes is more than a predefined threshold (\( T_{\text{KNN}} \)), the corresponding pair of keyframes are passed to the Superglue [35] matching algorithm. Finally, there is a PnP and pose matching stage to determine the exact transformation between two keyframes. This transformation and the corresponding keyframes’ indices are passed to pose graph optimization block to close the detected loop. The Superglance correspondence detection algorithm allows SRVIO to match significant viewpoint changes, occluded scenes, day/night scenarios, and many other challenging conditions as demonstrated in [35].

| Model         | Static | Dyn | DeepNN | Geometric |
|---------------|--------|-----|--------|-----------|
| VINS-Mono     | x      | -   | -      | x         |
| ORB-SLAM      | x      | -   | -      | x         |
| VI-DSO        | x      | -   | -      | x         |
| dynamic-SLAM  | -      | x   | -      | x         |
| DS-SLAM       | -      | x   | -      | x         |
| DynaSLAM      | -      | x   | -      | x         |
| SLAMANTIC     | -      | x   | -      | x         |
| UnDeepVO      | x      | -   | x      | -         |
| D3VO [13]     | x      | -   | x      | -         |

SRVIO

If the Dyn column is checked for a model, then the model is designed for a dynamic environment. DeepNN shows methods that use deep neural networks (other than just SSNNs) for odometry or SLAM task. SRVIO is a hybrid (DNNs and geometric) SLAM framework.

| Dataset         | Challenges                           | Dynamic |
|-----------------|--------------------------------------|---------|
| TUM-RGBD dyn [43] | Moving person                       | x       |
| ADVIO [42]      | Moving people and cars               | x       |
| WHUVID [40]     | Moving and parked cars              | x       |
| EuRoC [45]      | Blurry and fast movement            | x       |
| TUM-VI [44]     | Occasional pedestrian move          | x       |
| Sim ADVIO seq. 22 | Bus covering camera view            | x       |
| Sim TUM-VI outdoor7 | Person covering camera              | x       |

The keyframe database contains the following data:

\[
[i, \tilde{P}_{i}^{w}, \tilde{p}_{i}^{w}, \tilde{q}_{i}^{w}, j, \tilde{P}_{j}^{w}, \tilde{p}_{j}^{w}, \tilde{q}_{j}^{w}, \mathbf{D}(u, v, \text{des})]
\]

where \( \tilde{p}_{i}^{w} \) and \( \tilde{q}_{i}^{w} \) are estimated position and orientation of the \( i \)th keyframe, respectively. \( j \) is the possible loop closed keyframe index. \( \tilde{p}_{i,j} \) and \( \tilde{q}_{i,j} \) are the estimated relative position and yaw angle between these two frames (\( i \) and \( j \)), respectively. \( \mathbf{D}(u, v, \text{des}) \) is the keypoint matrix of the \( i \)th keyframe containing pixel locations \((u, v)\) and des descriptors. This data matrix may be considered as the output of the proposed SLAM framework.

IV. EXPERIMENTS AND RESULTS

In this section, multiple experiments are performed to showcase the superiority of SRVIO in the aforementioned challenging conditions over the current state-of-the-art SLAM and VIO models.

A. Datasets and Parameter Setting

We have selected several datasets to compare SRVIO with other SLAM algorithms. An overview of the models and datasets is provided in Tables I and II, respectively. The ADVIO [42] sequence 22 is edited to include highly dynamic objects in order to compare the outcomes of the models in such situations. This dataset is recorded originally with a handheld device and contains shaky walk sequences. TUM-RGBD dynamic [43] is a standard visual-only dataset. This dataset is also recorded by a handheld device. Here, TUM-RGBD is used with some estimators to have IMU data. This modified dataset is then used...
Fig. 5. Keypoint selection ablation study. This experiment is done on Seq23 of WHUVID dataset [40]. This particular sequence contains static and dynamic cars and dynamic humans in an open parking lot. The baseline (BL) method is VINS-Mono with the conventional fundamental matrix and RANSAC-based outlier rejection method for keypoint selection mechanism. The BL + SS (semantic segmentation) method uses a semantic mask to filter out all possible dynamic objects (green keypoints on all cars). (a) Top view plot of the paths estimated by each method. The images of eight important frames are shown below this plot. An example of filtered keypoints is also depicted in the right part. (b) 6-DoF plot of this experiment.

B. Keypoint Selection Algorithm Evaluation

The goal of our proposed method is to discard the keypoints that are on moving objects and to retain the static keypoints even if they fall on a possibly dynamic object (e.g., a parked car). To show the difference between the classic RANSAC-based filtering, SSNN only, and our proposed method, we apply all methods to sequence 23 of WHUVID dataset [40]. The results are depicted in Fig. 5. The results show that the baseline model (VINS-Mono) has errors and drifts when moving cars appear in scenes 1, 4, 6, and 7. The method that uses SSNN to filter out the dynamic objects (BL+SS) does not work on this sequence as well. Most of the initial keypoints fall on the moving or parked cars in this sequence. Hence, it is almost impossible to track the frames without including the keypoints on the static cars. As the results show, using only the SSNN to filter out the dynamic objects (BL+SS) does not work on this sequence as well. Most of the initial keypoints fall on the moving or parked cars in this sequence. Hence, it is almost impossible to track the frames without including the keypoints on the static cars. As the results show, using only the SSNN to filter out all keypoints on cars causes diversion and large errors in scale estimation. Our proposed method keeps the keypoints on the static cars and gives the best trajectory estimation.

C. Ablation Study

The goal of this experiment is to demonstrate the role of each block of SRVIO framework in facing extreme dynamic environments. To do so, we investigate the performance of several submodels of SRVIO. In the rest of this section, VIO stands...
Fig. 6. Simulated outdoor7 sequence of the TUM-VI dataset has a human blocking most of the camera’s view. The human appears at second 10 and leaves at second 28.

Fig. 7. Results of the proposed SRVIO model and its submodels over simulated TUM-VI outdoor7 sequence. In this experiment, the loop closure mechanism is turned off.

| Model       | Sim1 ATE | Sim1 RPE | Sim2 ATE | Sim2 RPE |
|-------------|----------|----------|----------|----------|
| VIO         | 22.691   | 0.207    | 5.150    | 2.578    |
| VO+SS      | 12.826   | 0.706    | 1.223    | 0.709    |
| IO         | 19.712   | 0.094    | 5.084    | 0.363    |
| VIO+SS        | 9.457    | 0.061    | 1.665    | 0.336    |
| VIO+SS+W     | 7.548    | 0.026    | 3.709    | 0.348    |
| SRVIO      | 2.201    | 0.024    | 0.959    | 0.180    |

The error metrics are the rms of ATE and RPE (m). The errors are not divided by a path length of 29.8 m to have a better distinction with limited floating point numbers.

TABLE III
RESULTS OF THE PROPOSED SRVIO MODEL AND ITS SUBMODELS OVER ADVIO22-SIM (SIM1) AND TUM-VI OUTDOOR7-SIM (SIM2)

for a vanilla VIO/SLAM framework (here, it is VINS-Mono SLAM framework) with no SSNN, adaptive weighting, IMU denoising networks, or loop closure. IO is the same VIO/SLAM framework with visual residual removed from the optimization. Similarly, VO is another version of VI odometry/SLAM with no inertial residual inside the optimization formula. SS stands for the SSNN module, and if used, it means that the framework uses the visual preprocessing of the SRVIO. W stands for dynamic weighting. The IMU data quality part of dynamic weights (the second part is the visual quality score) is fixed to unity in this experiment. Sim1 and sim2 refer to simulated ADVIO sequence 22 and TUM-VI sequence outdoor7, respectively.

For the experiments of this section, we use two sequences whose trajectories are plotted in Figs. 6 and 8. The first sequence (see Fig. 8) is designed to simulate driving behind a bus or walking behind a large object. The second sequence (see Fig. 6) is designed to evaluate the robustness of the models against complete coverage of the camera view. In this sequence, the camera view is blocked from second 10 to second 28 by a human.

The results of SRVIO and its submodels over the aforementioned two sequences are reported in Fig. 8 lower image, Fig. 7, and Table III. The results indicate that vanilla VIO could never solve the task of odometry on ADVIO22-sim or TUM-VI-sim. Furthermore, the IO configuration diverges after a few seconds at both ADVIO22-sim (Sim1) and TUM-VI outdoor7-sim (Sim2), and therefore, the loss is very high compared to other configurations (see Table III). However, the RPE loss is relatively smaller than expected as the IO problem is the build-up error of noise inside inertial integration, and everything seems to be ok in the short term. The VIO configuration also diverges because the SSNN is not used and visual residual becomes large and the inertial residual cannot help to rescue the framework from divergence. The visual-only model combined with the semantic segmentation network (VO+SS) cannot converge at the Sim1 sequence when the bus comes near and covers the image. The same thing happens in the Sim2 sequence when a human completely covers the camera. This is because the VO+SS framework does not have enough static keypoints when the bus is near, and the odometry trajectory estimation diverges.

The results of the VIO equipped with the semantic segmentation (VIO + SS) are slightly different. This configuration’s odometry estimation does not diverge when the bus covers the camera in Sim1 sequence because inertial data assist the optimization. However, it diverges after a while because it does not have enough keypoints to track and optimize correctly. In the Sim2 sequence, the loss of keypoints and small inertial residual cause the algorithm to maintain movement bias (fixed pose), and when the human goes away, the trajectory continues to be accurate.

The VO combined with SSNN and dynamic weighting inside the optimization block (VIO+SS+W) seems to perform great at the sim1 sequence. Nevertheless, at the sim2 sequence, it increases the weight of the inertial residual, and hence, the divergence of the inertial integration happens. The need for the denoising neural network becomes evident in this experiment. The (VIO+SS+W) configuration shows the importance of having dynamic weighting. It also explains the vulnerability of solely using the weighting mechanism and placing too much attention on inertial data in the optimization step. The complete SRVIO (without loop closure) is the most successful model in both sim1 and sim2 sequences. SRVIO has the IMU denoising component, while VIO+SS+W configuration does not have it. Also, it shows how IMU denoising can result in robustness over
harsh camera-blocking situations and help the VIO framework work more accurately when visual features are not good.

To summarize, it can be concluded that using only SSNN is not enough, and methods similar to SLAMANTIC and DynaSLAM2 would not output accurate trajectories in extreme dynamic conditions. Also, other state-of-the-art methods like ORB-SLAM3 and VINS-Mono will not be as accurate as SRVIO in full or partial visual feature loss situations.

D. Loop Closure Module Evaluation

A vital part of our algorithm is its loop closure mechanism. The performance of this module should be evaluated to ensure high precision (preferably 100%) and a high recall rate. We have evaluated the loop closure module on the Newer college dataset [46]. This dataset consists of three areas namely quad, midsection, and parkland. The dataset is recorded using a hand-held recording system. This dataset consists of three loops of the quad, two loops of the midsection and quad, and three loops of parkland. These loops are clockwise and counter-clockwise resulting in normal and reverse (180° angle) loop closures. To evaluate the loop closure modules, we use 870 keyframes. These keyframes are selected based on the provided GT and each keyframe has a distance of almost three meters from the next keyframe. We construct the GT based on the location of the GT of the dataset. The GT is shown in Fig. 10. The recall-precision results are shown in Fig. 9. This figure shows that our loop closure algorithm achieves significantly higher recall rates for different precision levels compared to other methods. Fig. 9(a) shows normal loop closures (clockwise loops) recognized with and counter clockwise are treated the same way. In the other experiment [see Fig. 9(b)], the recall-precision is reported for reverse loop closure for places that are revisited in a very different angle (more than 120°) like associating a clockwise loop to a counter-clockwise one.

To better show the performance of our method, we have depicted the raw probability of similarity before any geometric checks in Fig. 10. There are multiple colored lines on these similarity matrices that show the frames that are from the same locations. For example, keyframes (place) 6 and 190 are in the same locations from very different angles. The results show that the VINS-Mono loop closure mechanism without geometric checks is not confident and has very low precision. The performance of ORB-SLAM3 loop closure mechanism is better than VINS-Mono but still has lower recall compared to our method, especially when the target place is visited from two different angles. The geometric checks help all methods to reach much higher precision on this dataset but do not improve the recall on loop closures from different angles. Also, our loop closure mechanism does not show false positives and the precision is 1 since all possible dynamic objects are filtered and there is a PnP check on the detected loop closures (similar to the 3-D-2-D and 3-D-3-D check inside the VINS-Mono framework). Hence, there will be no false positives to cause the map failure.

The results show that none of the methods are able to recognize a location in the parkland (woods) from very different
angles. Also in these locations, the precision drops quickly when the recall is increased. Moreover, Fig. 1 shows an example on a recorded trial to be a proof of concept that this loop closure module can detect loops and places that are seen in the day and revisited in the night.

E. Comparison With the State-of-the-Art Models

In this section, SRVIO’s performance is compared with state-of-the-art VIO and SLAM methods in two main experiments. In the first experiment, the accuracy of the SRVIO on standard dynamic datasets is evaluated and compared with state-of-the-art methods. The IMU data of the vision-only dataset is simulated using the GT and the additional SLAMANTIC output. The results reported in Table IV, indicate that when dynamic objects and some motion blur are present in the input sequences, conventional classical methods, e.g. ORB-SLAM3 and VINS-Mono, have large errors in trajectory estimation. Also, the results suggest that our SRVIO method is superior to the SLAMANTIC method even when the IMU data are simulated.

In the TUM-RGBD dynamic dataset, the walking sequences contain humans walking in front of the camera, and the camera moves rapidly. Also, there are some camera blurs and extreme brightness changes. As the results show (see Table IV), the challenge is relatively high; therefore, the ORB-SLAM3 and VINS-Mono frameworks diverge and give unacceptable results. However, other dynamic methods use SSNN, and thus, have a much better performance. Although the other methods have problems with camera blur and extreme light changes, the SRVIO framework compensates for the lack of proper visual data using IMU data and the adaptive weighting mechanism. Therefore, the SRVIO is the best choice for indoor datasets with dynamic objects and challenging environments similar to this dataset. Despite the use of simulated IMU data, our method outperforms all previous dynamic methods on this standard dataset.

F. Throughput Experiment

The goal of this experiment is to show the advantage of a hybrid SLAM design compared to state-of-the-art deep learning-based odometry designs. The default resolution for all algorithms is 1280 × 720 and the parameters are set to default values.
TABLE VII
THROUGHPUT OF THE SRVIO COMPARED WITH OTHER ALGORITHMS ON SEQUENCE 22 OF THE ADVIO DATASET WITH HD IMAGES

| Model         | Throughput (FPS) | Frame Processing time (msec) |
|---------------|------------------|-------------------------------|
| VINS-Mono [8] | 25               | 39.7                          |
| ORB-SLAM3 [11]| 43               | 23.2                          |
| SLAMANTIC [38]| 8.9              | 112                           |
| UndeepVO [20] | 50               | 20                            |
| D3VO [13]     | 10               | 100*                          |
| SRVIO         | 20               | 48.4                          |

*The D3VO result is gathered from the KITTI benchmark [47].

TABLE VIII
THROUGHPUT OF THE MODULES OF THE SRVIO COMPARED WITH THE STATE-OF-THE-ART CLASSICAL SLAMs ON SEQUENCE 22 OF THE ADVIO DATASET BASED ON THE SETUP EXPLAINED IN SECTION IV-F

| Model         | Visual (ms) | IMU (ms) | Opt (ms) | Loop closure (ms) | Total (ms) |
|---------------|-------------|----------|----------|-------------------|------------|
| VINS-Mono [8] | 8           | 0.7      | 31       | 82.6              | 39.7       |
| ORB-SLAM3 [11]| 20.2        | 0.2      | 3        | 6.4               | 23.2       |
| SRVIO         | 15.4        | 2        | 31       | 29                | 48.8       |

The numbers are in ms. pp and Opt stands for preprocessing and optimization, respectively.

provided by the authors. The data used for benchmarking are sequence 22 of the ADVIO dataset. The throughput of each algorithm is recorded assuming all programs and NNs are loaded on RAM and GPU RAM (ready to run or receive the new frames). The same system described in Section IV-A is used. The overall benchmark results are summarized in Table VII. The throughput of the D3VO algorithm is based on KITTI benchmarks [47] since there is no open-source code for D3VO.

The results show better throughput of our hybrid SLAM (combination of DNNs and classical algorithms) compared to other nonclassical methods (UndeepVO, D3VO, SLAMANTIC). Also, the good result of UndeepVO is not surprising since it is a very simple odometry DNN according to the results shown in Table V.

Furthermore, another experiment is conducted to show the throughput differences of each module of SRVIO compared to state-of-the-art classical SLAM algorithms (VINS-Mono and ORB-SLAM3). The results are represented in Table VIII.

The results show that the throughput of SRVIO is lower than ORB-SLAM3. The main reason behind this lower throughput is not the NN used inside the framework, but the base algorithm of SRVIO, which is VINS-Mono. Comparing the throughput of SRVIO to VINS-mono (the baseline), we can see the delay caused by SSNN is not substantial since we have not used it all the time. Also, the IMU denoising NN is very fast since it is a small NN with few layers. The loop closure component of SRVIO has higher throughput than VINS-Mono since SRVIO does not extract a large number of low-confidence keypoints. Also, SRVIO’s high recall rate comes from using the Superpoint algorithm coupled with SuperGlue.

V. CONCLUSION

As mentioned earlier, there were the following three unsolved challenges in existing state-of-the-art VIO/VO-SLAM frameworks:
1) dynamic objects especially camera blocking objects;
2) the divergence of IMU data odometry;
3) robust long-term loop closure in challenging conditions such as day/night and multiview situations.

Our proposed SRVIO framework was demonstrated to solve all of the problems at once. Moreover, our proposed framework can be used on any dataset with frame rates higher than 10fps. However, there were limitations to our algorithm. These limitations include the following:
1) gathering IMU data before testing;
2) dependence on the semantic segmentation algorithm;
3) the need for a processing system with GPU;
4) the fact that we have not taken into account the error of IMU denoising DNN when the visuals were blocked.

Also, the loop closure module filters all dynamic objects. Consequently, the loop closure does not work in a scene inside a parking lot when most of the keypoints are on the parked cars. On the other hand, if we include the static cars in the loop closure detection, then there will be false positives that cause map failure. This was a limitation to consider.

Our experiments on simulated data also showed the need for a dynamic and challenging dataset. There was a simulated dataset for this purpose called VIODE [48], but this dataset was not real and may cause some biases in hybrid or deep-NN-based SLAM frameworks. There should be a dataset with temporally missing visual and inertial data, dynamic objects, challenging loop closure scenes, and scenarios in which the visual features are not trackable (e.g., extreme motion blur). Such datasets can facilitate the design of more robust SLAM frameworks.

REFERENCES

[1] K. Youssif, A. Bab-Hadiashar, and R. Hoseinnezhad, “An overview to visual odometry and visual SLAM: Applications to mobile robotics,” Intell. Ind. Syst., vol. 1, no. 4, pp. 289–311, 2015.
[2] L. An, X. Zhang, H. Gao, and Y. Liu, “Semantic segmentation-aided visual odometry for urban autonomous driving,” Int. J. Adv. Robot. Syst., vol. 14, no. 5, 2017, Art. no. 1729881417735667.
[3] M. Sabry, A. Al-Kaff, A. Hussein, and S. Abdennadher, “Ground vehicle monocular visual odometry,” in Proc. IEEE Intell. Transp. Syst. Conf., 2019, pp. 3587–3592.
[4] Y. Lin et al., “Autonomous aerial navigation using monocular visual-inertial fusion,” J. Field Robot., vol. 35, no. 1, pp. 23–51, 2018.
[5] J. Delmerico and D. Scaramuzza, “A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 2502–2509.
[6] T. Do, L. C. Carrillo-Arce, and S. I. Roumeliotis, “High-speed autonomous quadrotor navigation through visual and inertial paths,” Int. J. Robot. Res., vol. 38, no. 4, pp. 486–504, 2019.
[7] K. Wu, A. Ahmed, G. A. Georgiou, and S. I. Roumeliotis, “A square root inverse filter for efficient vision-aided inertial navigation on mobile devices,” in Proc. Robot., Sci. Syst., vol. 2, 2015, pp. 2–10.
[8] T. Qin, P. Li, and S. Shen, “VINS-Mono: A robust and versatile monocular visual-inertial state estimator,” IEEE Trans. Robot., vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
[9] L. Von Stumberg, V. Usenko, and D. Cremers, “Direct sparse visual-inertial odometry using dynamic marginalization,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 2510–2517.
X. Xiong, W. Chen, Z. Liu, and Q. Shen, “DS-VIO: Robust and efficient stereo visual inertial odometry based on dual stage EKF,” 2019, arXiv:1905.00684.

C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. Montiel, and J. D. Tardos, “ORB-SLAM2: An accurate open-source library for visual, visual- inertial, and multimodal SLAM,” 2015, pp. 1874–1890.

R. Clark, S. Wang, N. Markham, and N. Trigoni, “VINO: Visual-inertial odometry as a sequence-to-sequence learning problem,” in Proc. 31st AAAI Conf. Artif. Intell., 2017, pp. 3995–4001.

Y. Yang, L. V. Stumberg, R. Wang, and D. Cremers, “3DVO: Deep depth, deep pose, and deep uncertainty for monocular visual odometry,” in Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit., 2020, pp. 1278–1289.

D. Nistér, “An efficient solution to the five-point relative pose problem,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 6, pp. 756–770, Jun. 2004.

J. L. Schonberger and J.-M. Frahm, “Structure-from-motion revisited,” in Proc. IEEE Conf. Comput. Vision Pattern Recognit., 2016, pp. 4104–4113.

R. Mur-Artal and J. D. Tardos, “ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras,” IEEE Trans. Robot., vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

B. Bescos, J. M. Fácil, J. Civera, and J. Neira, “DynSLAM: Tracking, mapping, and inpainting in dynamic scenes,” IEEE Robot. Automat. Lett., vol. 3, no. 4, pp. 4076–4083, Oct. 2018.

J. Shi et al., “Good features to track,” in Proc. IEEE Conf. Comput. Vision Pattern Recognit., 1994, pp. 593–600.

J.-Y. Bouguet et al., “Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm,” Intel Corporation, vol. 5, no. 1–10, pp. 4–14, 2001.

R. Li, S. Wang, Z. Long, and D. Gu, “UnDeepVO: Monocular visual odometry through unsupervised deep learning,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 7286–7291.

C. Liu, J. Gu, K. Kim, S. G. Narasimhan, and J. Kautz, “Neural RGB (r) D sensing: Depth and uncertainty from a video camera,” in Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit., 2019, pp. 10978–10987.

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: A versatile and accurate monocular SLAM system,” IEEE Trans. Robot., vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” in Proc. IEEE/ACM 6th Int. Symp. Mixed Augmented Reality, 2007, pp. 226–234.

M. Kaneko, K. Iwami, T. Ogawa, T. Yamasaki, and K. Aizawa, “MaskSLAM: Robust feature-based monocular SLAM by using semantic segmentation,” in Proc. IEEE Int. Conf. Comput. Vision Pattern Recognit. Workshops, 2018, pp. 258–266.

C. Yu et al., “DS-SLAM: A semantic visual SLAM towards dynamic environments,” in Proc. IEEE/RSJ Int. Conf. Intel. Robots Syst., 2018, pp. 1168–1174.

S. Han and Z. Xi, “Dynamic scene semantics SLAM based on semantic segmentation,” IEEE Access, vol. 8, pp. 43536–43570, 2020.

H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in Proc. IEEE Conf. Comput. Vision Pattern Recognit., 2017, pp. 173–182.

S. Li and D. Lee, “RGB-D SLAM in dynamic environments using static point weighting,” IEEE Robot. Automat. Lett., vol. 2, no. 4, pp. 2263–2270, Oct. 2017.

L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, “Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment,” Robot. Auton. Syst., vol. 117, pp. 1–16, 2019.

O. J. Woodman, “An introduction to inertial navigation,” Computer Laboratory, University of Cambridge, Cambridge, U.K., Tech. Rep. UCAM-CL-TR-696, 2007.

P. G. Savage, “Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms,” J. Guid., Control, Dyn., vol. 21, no. 1, pp. 19–29, 1998.

C. Chen, X. Lu, A. Markham, and N. Trigoni, “IONet: Learning to cure the curse of drift in inertial odometry,” in Proc. AAAI Conf. Artif. Intell., 2018, pp. 6468–6476.

M. Brossard, S. Bonnabel, and A. Barrau, “Denoising IMU gyrosopes with deep learning for open-loop attitude estimation,” IEEE Robot. Automat. Lett., vol. 3, no. 3, pp. 4796–4803, Jul. 2020.

A. Gawel, C. Del Don, R. Siegwart, J. Nieß, and C. Cadena, “X-View: Graph-based semantic multi-view localization,” IEEE Robot. Automat. Lett., vol. 3, no. 3, pp. 1687–1694, Jul. 2018.

P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, “Super-Glue: Learning feature matching with graph neural networks,” in Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit., 2020, pp. 4937–4946.

D. DeTone, T. Malisiewicz, and A. Rabinovich, “SuperPoint: Self-supervised interest point detection and description,” in Proc. IEEE Conf. Comput. Vision Pattern Recognit. Workshops, 2018, pp. 224–236.

Y. Yuan, X. Chen, and J. Wang, “Object-contextual representations for semantic segmentation,” in Proc. 16th Eur. Conf. Comput. Vis. 2020, pp. 173–190.

M. Schorghuber, D. Steininger, Y. Cabon, M. Humenberger, and M. Glaesa, “SLAMANTIC-Leveraging semantics to improve VSLAM in dynamic environments,” in Proc. IEEE Int. Conf. Comput. Vis. Workshops, 2019, pp. 3759–3768.

M. A. Fischer and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” Commun. ACM, vol. 24, no. 6, pp. 381–395, 1981.

T. Chen, F. Pu, H. Chen, and Z. Liu, “WHUVID: A large-scale stereo-IMU dataset for visual-inertial odometry and autonomous driving in chinese urban scenarios,” Remote Sens., vol. 14, no. 9, 2022, Art. no. 2033.

D. Gálvez-López and J. D. Tardos, “Bags of binary words for fast place recognition in image sequences,” IEEE Trans. Robot., vol. 28, no. 5, pp. 1188–1197, Oct. 2012.

S. Cortés, A. Sofin, E. Rahtu, and J. Kannala, “ADVIO: An authentic dataset for visual-inertial odometry,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 419–434.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in Proc. Int. Conf. Intell. Robot. Syst., 2012, pp. 573–580.

D. Schubert, T. Goll, N. Denmel, V. Usenko, J. Stücker, and D. Cremers, “The TUM V1 benchmark for evaluating visual-inertial odometry,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2018, pp. 1680–1687.

M. Burri et al., “The EuRoC micro aerial vehicle datasets,” Int. J. Robot. Res., vol. 35, no. 10, pp. 1157–1163, 2016.

M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, “The newer college dataset: Handheld LiDAR, inertial vision and vision with ground truth,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2020, pp. 4353–4360.

A. Geiger, P.Lenz, and U. Ratsch, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in Proc. Conf. Comput. Vis. Pattern Recognit., 2012, pp. 3354–3361.

K. Minoda, F. Schilling, V. Wiest, D. Floreano, and T. Yairi, “VIODE: A simulated dataset to address the challenges of visual-inertial odometry in dynamic environments,” IEEE Robot. Automat. Lett., vol. 6, no. 2, pp. 1343–1350, Apr. 2021.

Ali Samadzadeh received the B.S. degree in electrical engineering and the M.S. degree in artificial intelligence from the Amirkabir University of Technology (AUT), Tehran, Iran, in 2018 and 2021, respectively. His research interests include deep learning in 3-D computer vision and localization, high-performance computing, and video programming in machine learning.

Ahmad Nickabadi received the B.S. degree in computer engineering and the M.S. and Ph.D. degrees in artificial intelligence from the Amirkabir University of Technology (AUT), Tehran, Iran, in 2004, 2006, and 2011, respectively. Since 2012, he has been an Assistant Professor with the Computer Engineering Department, AUT. His research interests include the analysis of image and video content, and data analysis using deep learning and probabilistic graphical models.