Fast Semantic-Aware Motion State Detection for Visual SLAM in Dynamic Environment

Gaurav Singh, *Student Member, IEEE*, Meiqing Wu, *Member, IEEE*, Minh V. Do, and Siew-Kei Lam, *Senior Member, IEEE*

***Abstract*— Existing visual SLAM (vSLAM) systems fail to perform well in dynamic environments as they cannot effectively ignore moving objects during pose estimation and mapping. We propose a lightweight approach to improve the robustness of existing feature based RGB-D and stereo vSLAM by accurately removing dynamic outliers in the scene that contribute to failures in pose estimation and mapping. First, a novel motion state detection algorithm using the depth and feature flow information is presented to identify regions in the scene with high moving probability. This information is then fused with semantic cues via a probability framework to enable accurate and robust moving object extraction to retain the useful features for pose estimation and mapping. To reduce the computational complexity of extracting semantic information in every frame, we propose to extract semantics only on keyframes with significant changes in image content. Semantic propagation is used to compensate for the changes in the intermediate frames (i.e., non-keyframes). This is achieved by computing the dense transformation map using the available feature flow vectors. The proposed techniques can be integrated into existing vSLAM systems to increase their robustness in dynamic environments without incurring much computation cost. Our work highlights the importance of distinguishing between motion states of potential moving objects for vSLAM in highly dynamic environments. We provide extensive experimental results on four well-known RGB-D and stereo datasets to show that the proposed technique outperforms existing vSLAM methods in indoor and outdoor environments under various dynamic scenarios including crowded scenes. We also perform our experiments on a low-cost embedded platform, i.e., Jetson TX1, to demonstrate the computational efficiency of our method.**

***Index Terms*— Visual SLAM (vSLAM), semantic segmentation, scene flow density, dynamic environment.**

I. Introduction

VISUAL Simultaneous Localization and Mapping (vSLAM) is a key component of modern autonomous robotic systems, augmented reality and visual positioning systems [1], [2]. vSLAM exploits the static correspondences in images to simultaneously estimate the pose of the ego-object and map of the environment as a joint problem. The outliers in the scene are the main cause of vSLAM estimation errors. Therefore, maximum consensus schemes (e.g. RANSAC [3], [4]) are generally adopted to remove these outliers from the static inliers, under the assumption that the majority of the scene contains static elements.

Recently, vSLAM achieved high pose estimation accuracy. ORB-SLAM2 [5], a state-of-the-art SLAM algorithm, is able to achieve pose estimation errors $\sim$1% of trajectory length i.e. average percentage pose error (APPE) on stereo KITTI odometry datasets [6], and about 0.015 meter absolute trajectory error (ATE) on RGB-D TUM-static datasets [7]. However, the accuracy of these vSLAMs [5] reduce significantly in dynamic environments, such as those in RGB-D TUM-dynamic dataset [7] and RGB-D BONN dataset [8]. This inconsistency in performance is due to the presence of large amount of moving objects (e.g. car, bicycle, pedestrian, etc.) in the scenes, which violates the assumption of maximum consensus schemes. These dynamic elements, which are prevalent in realistic environments, can cause failures in pose tracking and irreversible corruptions in the map.

To address this problem, recent works in dynamic SLAM [9], [10], [11], [12] exploited either geometric or semantic cues to identify and exclude the dynamic outliers prior to pose estimation. In geometric approaches, moving regions are extracted by utilizing dense optical flow [13], sparse optical flow [12], [14], [15], feature point trajectories [10], [16], depth information [17], map registration [8], etc. The work in [9] attempt to remove features associated with potential semantically moving objects (e.g. object-class car) so that they will not contribute to pose estimation and mapping. However, this direct feature removal step can result in losing useful information in scenes where potential moving objects are motionless, for example parked vehicles [6] or sitting person [7]. Our experiments show that in scenes containing large number of parked vehicles, the direct removal approach leads to reduction in features for pose estimation that causes drop in pose accuracy. In addition, the existing methods either have high computationally complexity [13], [16], or suffer from robustness issues [17], as discussed in Section II. The recent DynaSLAM II version [18] focuses on integrating multi-object tracking along with SLAM, but does not specifically address dynamic objects. It also requires high computational resources on GPU.

Our work increases the robustness of vSLAM in challenging scenarios for both indoor and outdoor dynamic scenes by removing the influence of moving objects from the vSLAM

Manuscript received 13 October 2020; revised 4 July 2021 and 26 June 2022; accepted 13 September 2022. Date of publication 25 October 2022; date of current version 5 December 2022. This work was supported in part by the RIE2020 Industry Alignment Fund—Industry Collaboration Projects (IAF-ICP) Funding Initiative; in part by the Singapore Telecommunications Ltd. (Singtel), through Singtel Cognitive and Artificial Intelligence Laboratory for Enterprises [SCALE@Nanyang Technological University (NTU)] (cash and in-kind contribution); and in part by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1, under Grant RG78/21. The Associate Editor for this article was Z. M. Kassas. *(Corresponding author: Gaurav Singh.)*

The authors are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: gaurav012@e.ntu.edu.sg).

Digital Object Identifier 10.1109/TITS.2022.3213694

pipeline in a computationally efficient manner. The proposed SGCSF-K (Spatial Graph Clustering and Semantic Fusion with KF Semantics Propagation) SLAM fuses semantic information with motion states of the scene objects in a probability framework to enable accurate and robust moving region extraction in order to weigh the useful features for pose estimation. First, sparse 3D scene flow is extracted by re-using the 2D feature correspondences and their depth information, that are already available from the initial step of vSLAM. A novel lightweight density based spatial graph clustering (SGC) algorithm is used to determine parts of the scene flow that correspond to moving regions in the scene. Second, semantic segmentation is utilized to extract the semantic knowledge in the scene. Unlike other vSLAM works [17], [19], [20], we only extract semantics in selected frames (keyframes). A lightweight mesh-flow based frame warping technique is proposed to propagate the semantic maps from the keyframes to the intermediate frames. This leads to significant reduction in computation time by avoiding the need to extract semantics in every frame. Finally, to overcome the uncertainty in the moving regions and semantics, a probability framework is proposed to fuse the motion states cue with the semantic cue to detect moving regions accurately. The features corresponding to the detected moving regions are then discarded, and remaining features are weighted in the final pose optimization based on the fusion probability. This increases the robustness of the random outlier removal scheme as the majority of features belong to static objects.

The proposed SGCSF-K SLAM is evaluated on standard datasets including indoor [7], [8] and outdoor [6] scenes, with both RGB-D and stereo inputs. Compared to the state-of-the-art dynamic vSLAM methods, SGCSF-K PSPNet achieves lowest pose errors on all three datasets, i.e., ATE 0.15m on TUM, ATE 0.63m on BONN, and 0.699% average pose errors (APE) on KITTI, whereas the faster version SGCSF-K EdgeNet achieves lowest error on BONN (ATE 0.71m) and comparable errors on KITTI (0.708% APE) and TUM (ATE 0.71m) datasets. Timing analysis on the NVIDIA Jetson TX1 reveals that SGCSF-K EdgeNet achieves the best tracking-time and accuracy trade-off among the dynamic SLAM methods.

The rest of the paper is organized as follows. Section 2 discusses the existing work in vSLAM particularly those that address the problems in dynamic environment. In Section 3, we describe the proposed framework to extract semantic-aware motion states for accurate moving region extraction in vSLAM. Ablation study, experiments and comparison with state-of-the-art vSLAMs are presented in Section 4. Finally, Section 5 concludes this paper.

II. Related Work

Modern vSLAMs estimate the pose either by exploiting the geometric relationship among the static feature correspondences observed in the images [5], [21], [22] or using end-to-end deep learning methods [23], [24]. The evaluation results in KITTI odometry evaluation platform [6] show that feature-based vSLAMs are the dominant methods in terms of accuracy and speed. The state-of-the-art ORB-SLAM2 [5] is one of the most versatile feature-based vSLAM algorithms that performs well for both indoor and outdoor scenarios. However, the pose estimation accuracy of ORB-SLAM2 degrades when a significant part of the scene is occupied by moving objects (e.g., vehicles) [19]. Similar behaviour is observed in other vSLAM [21] and visual odometry (VO) [22] systems. This is because the maximum consensus outlier removal schemes (e.g., RANSAC [5], [22]) can remove outlier features corresponding to moving objects only when its proportion in the scene is small compared to those associated with static objects. When significant number of features corresponding to moving object/s are present, they constitute to a motion field that impairs the effectiveness of the maximum consensus approach [9], [11], [19]. To address this problem, a number of works detect and remove features corresponding to the moving regions before pose and map estimation. These works can be categorized into three groups based on the cues they use to identify moving regions: geometric cue [10], [12], [13], [14], [15], [16], [25], semantic cue [9], [11], and fusion of geometric and semantic cue [17], [19], [20].

Geometric cue based methods exploit dense 2D optical flow [14], [26], sparse 2D optical flow [15], 3D flow [12], feature point trajectories [10], [13], [16], and dense model registration [27], [28]. Graph triangulation [15], feature clustering [10] or motion consistency [25] methods are utilized to identify the discontinuity in motion states. Graph triangulation techniques such as [15] construct Delaunay triangulation graph from optical flow (usually sparse), and then prune the graph by removing edges between flow velocity vectors with large velocity differences. While being computationally efficient, they can only partially capture the moving objects regions. Feature trajectory clustering techniques [10], [13] detect moving regions using features tracked over several frames. MVO [10] apply a multiple model fitting technique to estimate full motion of the camera and rigid objects. The parameterization of the motion transforms is done non-incrementally (i.e., relative to first frame) which may introduce severe linearization errors [29]. Sparse subspace techniques for trajectory clustering are still limited to noiseless data [30]. The dense approaches [27], [28], [31] performs the registration and alignment to the dense model and then exploits the registration residual with the geometric features for improving segmentation and tracking. Refusion [25] utilizes motion consistency based on truncated signed distance function (TSDF) to determine points on a moving object. It requires dense depth for generating detailed maps using TSDF based mapping, leading to higher computational requirements. More detailed information on these dynamic SLAM techniques can be found in [30].

MaskSLAM [9] and Detect-SLAM [11] try to remove features belonging to semantically labelled dynamic class of objects such as car, bicycle, pedestrian etc., irrespective of their actual motion state (moving or not). This results in loss of many static features (e.g., those associated with parked vehicles that occupy large parts of the scene) which could have contributed to effective pose estimation.

The third category of works, such as DS-SLAM [19], DynaSLAM [17], and DynaSLAM II [18], attempt to fuse both

geometric and semantic cues. DS-SLAM, which is built on top of ORB-SLAM2, detects and removes moving outliers based on semantic information extracted using SegNet [32], and geometric cue extracted by running moving consistency check on the features. It searches for outliers detected by moving consistency check that falls within the regions of semantically labelled dynamic objects. The moving consistency check is based on maximum consensus outlier detection, and due to the presence of random outlier distribution, it cannot guarantee true moving outliers. Moreover, the two cues are combined using simple AND operation without considering any uncertainties. DynaSLAM [17] combines ORB-SLAM2 with semantics and multi-view geometry cues. It detects moving features (called keypoints), and each keypoint is handled individually to determine if its depth change is larger than a threshold in order to declare it as moving. It then uses region growing over dense depth maps to obtain the full mask of the object. Finally, it declares the regions as moving if either the geometric region is detected as moving, or the semantic labels of that region belongs to the dynamic class. A main limitation of DynaSLAM is that it does not consider the coherence among the keypoints within one mask region, and therefore it over-removes static regions in some cases. DynaSLAM II mainly integrates the tracking of multiple objects with SLAM, rather than focusing on improving robustness of SLAM. Our previous work in [20] uses Delaunay triangulation for geometric graph clustering, and the resulting masks are fused with semantic information to obtain the moving regions. However, this method suffers from fragmented geometric clusters, which can only capture partial moving regions.

The works that exploit semantic cues in [9], [17], [19], and [20] perform semantic segmentation on every frame, which contributes to large computational bottleneck in the vSLAM systems. To address this, [33] extracts and uses semantics only from keyframes which exhibit large motion changes. For accurate moving region detection, semantics are required for every frame as the semantic labels corresponding to moving objects tend to change independently of the camera motion. If the keyframe semantics are to be utilized for intermediate frames, the semantic labels need to be propagated. Existing works on semantic label propagation [34], [35], [36] either use deep-learning based models [34], [35], [36] or dense optical flow [37], [38], which are computationally intensive.

Our work falls under the third category of dynamic SLAM methods that fuse geometric and semantic cues, and it differs from existing works as follows. The proposed SGCSF-K SLAM performs density-based SGC to extract geometric cues. We construct nearest neighbour spatial graph instead of Delaunay triangulation ([15], [39]), and perform density-based clustering over flow-pruned graph to obtain flow-coherent spatial clusters. Unlike DynaSLAM [17], the proposed SGCSF-K SLAM does not extract the geometric cue from individual features, but as a cluster of features that are spatially close with coherent motion flow. Moreover, instead of simple AND (DS-SLAM) or OR (DynaSLAM) operations, we generate distance based smooth fusion probabilities. Also, in contrast to DynaSLAM, the proposed method does not rely on accurate dense depth maps. Our work overcomes the computational and robustness challenges of extracting semantic cues in [9], [17], [19], [20], and [33] by utilizing a lightweight mesh-flow [40] concept based on sparse feature matches for semantic propagation.

The existing feature matches are utilized to generate uniform mesh-flow, which is then used to generate dense propagation maps for warping semantic labels across frames.

The following summarizes the main contributions of this paper:

- A novel lightweight density-based spatial graph clustering (SGC) algorithm to compute the geometric moving regions from the available sparse feature flow.
- Only keyframes are used to extract semantics to reduce the computation overhead. For the non-keyframes, the proposed semantic propagation method is used. Semantic propagation is achieved by computing mesh-flow from already available vSLAM feature matches.
- A new fusion approach that combines SGC based geometric cue and semantic cue in a probabilistic manner, which takes into account the uncertainties in both the cues. In addition, we weigh feature correspondences in the final pose estimation based on the fusion probability to reduce the effects of movable objects (e.g., ball, box, etc.) that are not classified as dynamic objects.

III. Proposed Method

Our work improves the accuracy of vSLAM in dynamic environments by removing major proportion of outlier features associated with moving objects, while keeping the computational cost low. The proposed SGCSF-K SLAM (see Fig. 1) detects moving object regions by exploiting the geometric motion (flow) patterns and semantics of the scene. First, a fast scene flow (sparse) based spatial graph clustering (SGC) algorithm is proposed to compute the locally coherent moving regions by geometric approach (Section III-A). Second, we perform semantic segmentation on keyframes and estimate the semantics for other intermediate frames. A lightweight mesh-flow based semantic propagation method which uses the already available feature correspondences of SLAM is presented in Section III-B. Third, a framework is designed that fuses the obtained geometric motion cues and pixel-wise semantics in a probabilistic manner (Section III-C). The proposed fusion framework takes both the uncertainties of the geometric motion cue and semantics into the probabilistic formulation, to accurately identify the moving object regions. The features corresponding to the highly probable moving regions are disregarded while the remaining features are weighted using the fusion probability and fed to the existing SLAM for final pose and map estimation. The proposed "moving feature removal" module (shown as yellow box in Fig. 1 in a general feature based vSLAM pipeline) only requires feature matches (with depth) and the image frame as inputs. The proposed module extracts semantic information for the keyframes, identifies moving features, and removes them. The remaining features are provided to the original vSLAM framework without any modification.

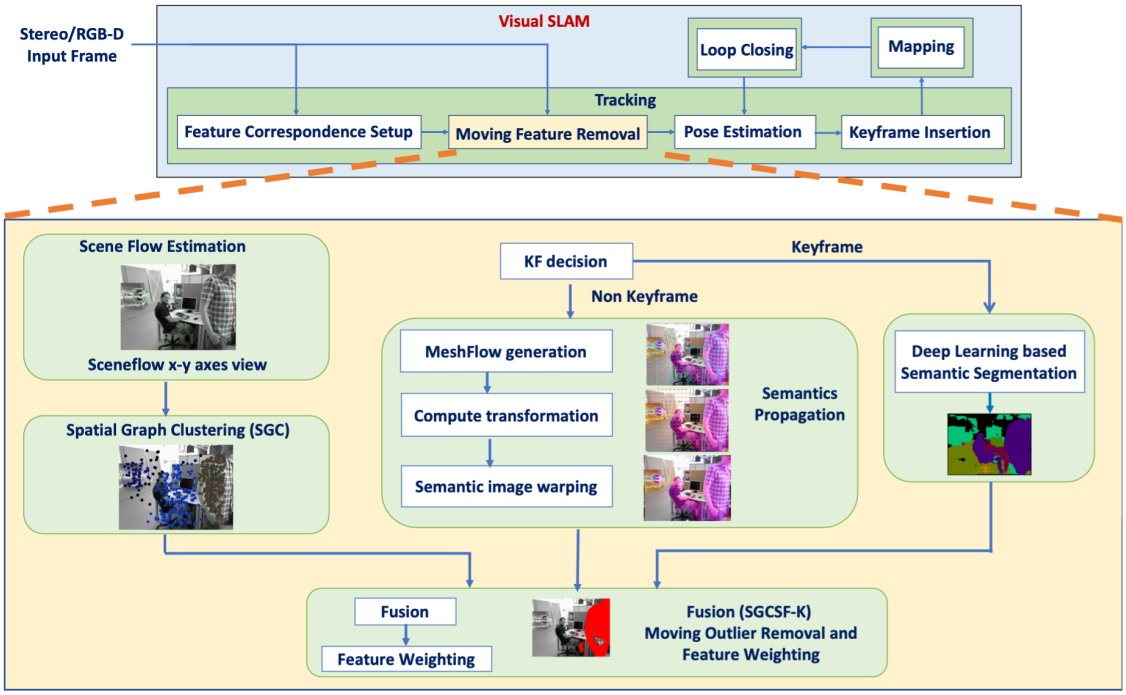


Fig. 1. Proposed SGCSF-K SLAM consists of SGC, keyframe only semantic propagation and fusion blocks. The intermediate steps are illustrated. The proposed moving feature removal (yellow) technique can be integrated into any existing vSLAM to improve its robustness in dynamic environments.

A. Scene Flow Density-Based Spatial Graph Clustering (SGC)

We utilize the features correspondences between sequential (previous and current) image frames for getting the sparse 2D optical flow, and the feature correspondences between left and right images is used for obtaining the disparity in the case of stereo camera setup. For RGB-D input stream, the depth values (i.e., disparity d) can be directly obtained [7]. Given the i^{th} feature point $p^i_{t-1}(u^i_{t-1}, v^i_{t-1}, d^i_{t-1})$ at time step $t-1$ and its correspondence $p^i_t(u^i_t, v^i_t, d^i_t)$ at time step t, where u, v represents the horizontal and vertical coordinates in the image and d refers to the disparity value, the corresponding 3D coordinates, $P^i_\tau = (X^i_\tau, Y^i_\tau, Z^i_\tau)^T$ can be reconstructed via camera triangulation process:

$$X^i_\tau = \frac{(u^i_\tau - c_u)\cdot b}{d^i_\tau}, \quad Y^i_\tau = \frac{(v^i_\tau - c_v)\cdot b}{d^i_\tau}, \quad Z^i_\tau = \frac{b\cdot f}{d^i_\tau}, \quad for\ \tau = t-1, t \tag{1}$$

where b is stereo baseline and (c_u, c_v) is principal point. The reconstructed world coordinates P^i_{t-1} and P^i_t using (1), takes the camera pose at time $t-1$ and time t as their coordinate origin, respectively.

Let $\tilde{T}_{t,t-1} = [\tilde{R}_{t,t-1}|\tilde{t}_{t,t-1}]$ be the approximate relative camera pose from $t-1$ to t. In order to estimate $\tilde{T}_{t,t-1}$, a straightforward way is to estimate the relative pose of current frame (F_t) with respect to the previous frame (F_{t-1}), which is also known as visual odometry (VO). But, as we do not know which feature points belong to moving objects, the VO estimated pose can be highly erroneous. Most recent vSLAM systems are keyframe-based [5], [41] and use keyframes for pose and map refinement by bundle adjustment optimization. The back-end map optimization improves keyframe's pose. In this work, we only include features associated with static category of objects in the keyframes by using extracted semantic information. Hence, the keyframes in the vSLAM pipeline contains relatively stable features than the non-keyframes (frame (F_{t-1})), where the keyframe selection policy remains the same as the base model (i.e., ORB-SLAM2 [5]). Hence, we make use of the keyframe to quickly estimate pose of the current frame, which is required for estimating the scene flow.

We propose a *fast-tracking* method, where we first estimate the pose $\tilde{T}_{t,k}$ of current frame F_t relative to last keyframe F_k and then obtain the relative pose $\hat{T}_{t,t-1}$ using already known previous frame pose $\hat{T}_{k,t-1} = \hat{T}^{-1}_{t-1,k}$ as shown in Fig. 2, and defined in (2) and (3). This approach of tracking with respect to keyframe provides more accurate estimate in dynamic scenes than VO (as VO tracks all features including moving outliers, from F_{t-1} to F_t). In *fast-tracking* method, the relative pose $\tilde{T}_{t,k} = [\mathbf{R}_{t,k}|\mathbf{t}_{t,k}]$ of current frame F_t with respect to the last keyframe F_k, is estimated by tracking corresponding matches $\psi_{t,k}$ between F_k and F_t i.e.

$$\{\mathbf{R}_{t,k}, \mathbf{t_{t,k}}\} = \underset{\mathbf{R}_{t,k},\mathbf{t}_{t,k}}{argmin} \sum_{i\epsilon\psi_{t,k}} \rho\left(\left\|x^i - \pi\left(\mathbf{R}_{t,k}\mathbf{X}^i_k + \mathbf{t}_{t,k}\right)\right\|^2_\Sigma\right) \tag{2}$$

where $x^i = [u^i, v^i]$, $\mathbf{X}^i_k$ is the corresponding matched 3D point in keyframe and operators π (projection), ρ (robustness), Σ (covariance) are same as used in standard motion-only bundle adjustment optimization [5]. Also, as $\hat{T}_{k,t-1} = (\hat{T}_{t-1,k})^{-1}$ is known from previous frame tracking, the relative pose $\tilde{T}_{t-1,t}$ can be written as

$$\tilde{T}_{t,t-1} = \tilde{T}_{t,k}\hat{T}_{k,t-1} \tag{3}$$

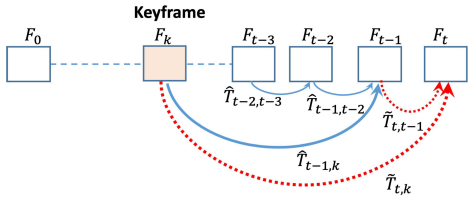


Fig. 2. Fast tracking concept: blue lines indicate known information from past SLAM estimations, red dotted pose $\tilde{T}_{t,k}$ is estimated with respect to last keyframe and $\tilde{T}_{t,t-1}$ is the relative transformation.

where, $\tilde{T}_{t,t-1} = [\tilde{\mathbf{R}}|\tilde{\mathbf{t}}]$ (subscripts are dropped for simplicity). Here, the estimated pose $\tilde{T}_{t,t-1}$ is not the final pose and we only use this approach to quickly estimate the required pose for the scene-flow and motion state decision (later in 7). We call this approach *fast-tracking* because this estimation can be done in less time than full-pose estimation. In fact, in full-pose estimation, the current pose is first estimated with respect to the last frame (i.e., VO) and then refined by tracking local map-points [5]. We use full-pose estimation for the final pose after detecting and removing the moving outliers, which will be discussed in later sections.

After compensating for the camera motion, coordinates $\tilde{P}_t^i = (\tilde{X}_t^i, \tilde{Y}_t^i, \tilde{Z}_t^i)^T$ are given by:

$$\tilde{P}_t^i = \tilde{\mathbf{R}} P_{t-1}^i + \tilde{\mathbf{t}} \tag{4}$$

Then, the position difference $V_t^i = (\Delta X^i, \Delta Y^i, \Delta Z^i)^T$ represents the scene flow (vector):

$$\Delta X^i = \tilde{X}_t^i - X_t^i, \quad \Delta Y^i = \tilde{Y}_t^i - Y_t^i, \ \Delta Z^i = \tilde{Z}_t^i - Z_t^i \tag{5}$$

In ideal case, if $\{\tilde{\mathbf{R}}, \tilde{\mathbf{t}}\} = \{\mathbf{R}, \mathbf{t}\}$ and the feature correspondences are free of errors (matching, depth, etc.), then the scene flow vectors for all static feature correspondences would be zero-magnitude. This would enable a simple thresholding operation to distinguish between static and moving feature correspondences. But in practice, this is not always feasible because of incorrect feature matches and use of motion approximation (4), some static points may even be associated with large magnitude scene flow. To overcome this problem, features points can be clustered on the basis of correlated scene flow vectors. However, apart from effect of the feature matching errors on scene flow, the scene flow uncertainty also increases with depth (z direction) from the camera [15]. Therefore, under these noisy conditions, global correlation (or affinity) of scene flow is feasible only when features are tracked for a longer duration as in [10], which in-turn suffers from problems such as loss of tracklets due to feature dropouts and higher complexity [10], [42]. In this work we utilize only feature matches between successive frames to perform lightweight clustering of features.

It was observed that features which are spatially close have more similar scene flow and are more likely to belong to same object. Hence, we utilize the two properties of spatial proximity and scene flow similarity together to construct a sparse connected graph, and perform clustering over it. We determine moving features as a group (clusters), instead of relying on highly uncertain single outlier based methods [17], [19]. We propose a spatial graph based scene clustering (SGC) technique to distinguish between static and moving points. The sparse graph based clustering also reduces the number of distance comparisons between feature points as only connected nodes are compared to each other for clustering.

We first construct a connected graph by using spatial proximity of features. This connected graph contains edges between neighbouring features (nodes) in space, but does not have any information regarding scene flow i.e. feature nodes with dissimilar flow are also connected. Hence, to account for this information in the graph, we perform pruning of graph edges to disconnect nodes that have distinctly dissimilar motion (scene flow) by using Mahalanobis distance (MD) between the scene flow vectors of the connected nodes. As compared to the absolute scene flow distance between nodes, the MD takes into consideration the uncertainty due to measurement noise in scene flow vectors by including covariance matrix. Next, a spatial graph with scene flow information is obtained. To obtain highly correlated clusters from this graph i.e., avoid the outliers within clusters we perform node density based clustering. The simple techniques such as connected component clustering has been used for graph clustering but they do not handle outlier within clusters. Similarly K-means clustering causes problems as anomalous points (outliers) will be assigned to the same cluster as normal data points. The anomalous points draw the cluster centroid towards them, making it harder to classify them as anomalous points. On the other hand, density-based clustering works by identifying dense clusters of points, allowing it to learn clusters of arbitrary shape and identify outliers in the data. We employed non-parametric hierarchical density based spatial clustering (HDBSCAN) as it does not require the number of clusters as a parameter. Also, basic DBSCAN can be seen as efficient variant of spectral clustering whereby, the connected components correspond to optimal spectral clusters, and DBSCAN finds connected components on the reachability graph [43]. Spectral clustering can be computationally intensive (up to $O(n^3)$) without approximation and other assumptions), and one has to choose the number of clusters k for both the number of eigen vectors to choose and the number of clusters to produce with k-means on the spectral embedding. Thus, for performance reasons we use the HDBSCAN algorithm. Finally, to classify the clusters into static and moving classes, we use average reprojection errors of the clusters that better represent the coherent features than individual reprojection errors. The uncertainty (noise) in reprojection errors is inherently handled by averaging them within clusters. This technique is shown in Algorithm 1 and discussed next.

In the first step, we construct a spatial graph $G_{sym}(n, k)$ using nearest neighbour algorithm KNN (see Fig. 3(b)). The symmetric spatial graph $G_{sym}(n, k)$ connects feature points $\mathbf{x}$ (Fig. 3(b)) with edges $\mathbf{e}$ based on their spatial closeness i.e. feature point euclidean distances $d(x_i, x_j) = \sqrt{(u^i - u^j)^2 + (v^i - v^j)^2}$, such that x_i and x_j are connected by $e_{i,j}$, if x_i is a member of k-nearest neighbours set $kNN(x_j)$ of x_j or vice-versa. Only a limited number of 'K' neighbours are connected. To construct this spatial graph we

Algorithm 1 Density Based Clustering on Flow Coherent Spatial-Graph

Step 1: Construct k-nearest-neighbor graph
$G_{sym}(n,k)$: x_i and x_j connected if $x_i \in \text{kNN}(x_j)$ or $x_j \in \text{kNN}(x_i)$
where, $\text{kNN}(x_i)$ denotes the set of the k nearest neighbors of x_i among $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n$
Step 2: Graph pruning (For each edge compute Mahalanobis distance)
for all edge, $e\{i,j\} \in G_{sym}(n,k)$ **do**
  $\Delta(\mathbf{V}_t^i, \mathbf{V}_t^j) = (\mathbf{V}_t^i - \mathbf{V}_t^j)^T \Sigma_{ij}^{-1} (\mathbf{V}_t^i - \mathbf{V}_t^j)$ ▷ Eq. 6
  if $\Delta(\mathbf{V}_t^i, \mathbf{V}_t^j) > \tau_{Mahala}$ **then**
    $\hat{G} \leftarrow$ remove edge $e\{i,j\}$ from G
  end if
end for
Step 3: Perform HDBSCAN clustering on $\hat{G}$
Clusters : $\{C_1, \ldots, C_n\} = f_{\text{HDBScan}}(\hat{G})$
Step 4: Motion state decision (For each cluster compute ARE)
for all $C_j \in \{C_1, \ldots, C_n\}$ **do** ▷ Eq. 7
  $ARE_j = \sum_{i \epsilon C_j} \left\| x^i - \pi \left(\mathbf{R}\mathbf{X}^i + \mathbf{t} \right) \right\|^2$
  if $ARE_j > \tau_{ARE}$ **then**
    $label_j \leftarrow$ moving
  else
    $label_j \leftarrow$ static
  end if
end for

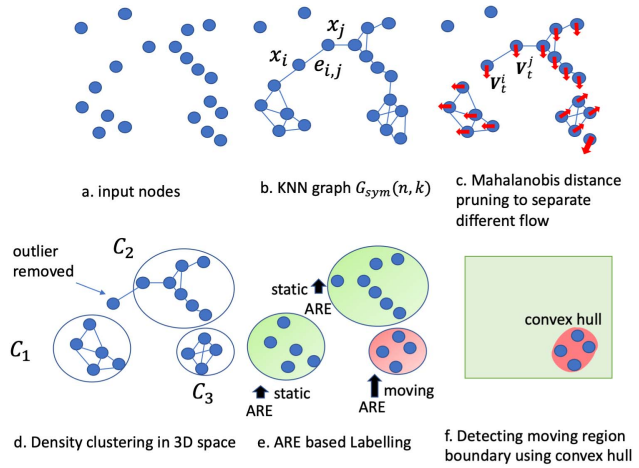


Fig. 3. SGC: Density based clustering on flow coherent spatial-graph.

use 2D position x_i of features instead of their 3D reconstructed position P_t^i to avoid the uncertainty introduced by triangulation [44].

After the graph has been constructed, some edges connect features with significantly different flow. To disconnect the nodes belonging to different motions patterns, a pruning step is performed on this graph. In the pruning step, we first compute Mahalanobis distance (MD) for each edge in the graph as follows. If each vertex represents a feature point by its flow vector V_t^i and its reconstructed 3D position P_t^i, then the edges of the graph are weighted based on the Mahalanobis distance $\Delta(\mathbf{V}_t^i, \mathbf{V}_t^j)$ of the associated scene flow vectors V_t^i, V_t^j as,

$$\Delta(\mathbf{V}_t^i, \mathbf{V}_t^j) = (\mathbf{V}_t^i - \mathbf{V}_t^j)^T \Sigma_{ij}^{-1} (\mathbf{V}_t^i - \mathbf{V}_t^j) \tag{6}$$

This takes into consideration the uncertainty incurred in the reconstructed position P_t^i, P_t^j due to measurement noise, where the modelled uncertainty is $\Sigma_{ij} = \Sigma_i + \Sigma_j$. Each Σ_i represents the covariance noise defined by $\Sigma_i = \mathbf{J_i S J_i}^T$. The measurement noise matrix $\mathbf{S}$=**diag**(0.5) pixel and $\mathbf{J}_i$ denotes Jacobian of the scene flow [45]. As high Mahalanobis distance represents heterogeneous pair of flow, we remove all edges with higher value than a predefined threshold τ_{Mahala}. By choosing suitable 'K' value for the KNN, the graph contains sufficiently connected components even after pruning phase. The empirically chosen 'K' value of 5 neighbors works well with all our experiments.

After the graph pruning step, few edges between nearby objects with nearly similar motion remain. If connected component analysis is used to retrieve clusters, it may lead to overly large segmented regions near to these moving objects, that sometimes overshadow static background regions (with low feature count). It is also observed that the feature nodes belonging to one moving object are densely distributed in 3D space compared to the sparsely distributed nodes in intermediate regions because pruned spatial (neighbourhood) graph $\hat{G}$ contains fewer inter-object edges than intra-object edges. And, $\hat{G}$ only contains connection between nodes with coherent flow. Moreover, the spatial 3D density of features (nodes) varies in different parts of the scene, for example, densely distributed in near distance regions and sparsely distributed in distant regions from camera due to the property of perspective projection. That is why, compared to the connected component retrieval, the density based clustering can easily bifurcate and extract such clusters based on the varying node distribution in the graph. Hence, we perform density based clustering using fast implementation of HDBSCAN clustering [46], [47], on the pruned graph $\hat{G}$. HDBSCAN is non-parametric method that finds the clusters of variable densities without having to choose any distance threshold. The input to the HDBSCAN is the pruned graph $\hat{G}$ with each node i represented by the 3D coordinates P_t^i. First, it constructs distance weighted graph $\hat{G}_d^W$ from $\hat{G}$. We experimented with euclidean, Mahalanobis, Manhattan and other distance metrics, but found euclidean as the better trade-off between clustering quality and computation cost. Then it build the minimum spanning tree of the distance weighted graph (efficiently via Prim's algorithm) and construct a cluster hierarchy of connected components. Finally, the cluster hierarchy is condensed based on minimum cluster size and extract the stable clusters from the condensed tree. The output of the clustering is a set of clusters such that $\{C_1, \ldots, C_n\} = f_{\text{HDBScan}}(\hat{G})$. Cluster C_j is a set of nodes id (features) within the j^{th} cluster and function $f_{\text{HDBScan}}(.)$ is the HDBSCAN clustering. This enables the detection of highly coherent clusters of feature points in terms of spatial geometry and motion (scene flow).

In the final motion state decision step, to label the clusters as moving or static, we calculate representative metric i.e.

average-reprojection-error (ARE) (see (7)) for each cluster C_j and compare it with empirically chosen threshold τ_{ARE}.

$$ARE_j = \sum_{i \epsilon C_j} \left\| x^i - \pi \left(\tilde{\mathbf{R}} \mathbf{X}^i + \tilde{\mathbf{t}} \right) \right\|^2 \tag{7}$$

$\{\tilde{\mathbf{R}}, \tilde{\mathbf{t}}\}$ is the transformation estimated using fast-tracking (3). All the clusters with lower ARE than the τ_{ARE} are labelled as static and other clusters as moving G_m (Fig. 5(a)). The main advantage of choosing ARE over individual reprojection error (RE) is that RE is highly variant due to random noise, false feature matches and depth. Thus, ARE represents a group of features in a cluster (in a local region), where the cluster represents coherent feature flow and spatial proximity. Hence, instead of classifying individual features, we classify clusters as moving or static. The binary masks for moving regions are identified using convex hull operation on the moving cluster members in the image domain. The usage of the sparse features features based geometric approach for motion state detection has a limitation that it cannot capture the complete mask of the moving objects, because the feature pruning step increases the sparsity. To remove majority of features lying on moving objects, we further exploit semantic information.

B. Keyframe Only Semantic Segmentation and Mesh Based Propagation

As an additional cue, we make use of the semantic information to assist final moving object decision making. The semantic information is extracted using existing deep learning based pixel-wise semantic segmentation network. Any state-of-the-art semantic segmentation like [48] and [49] can be utilized. In this work, we choose PSPNet [50], [51] which is ranked 1st place in ImageNet scene parsing challenge 2016, SUN Semantic segmentation challenge 2017 and WAD drivable area segmentation challenge 2018 [51]. Furthermore, the same PSPNet model can be used for both indoor and outdoor scenes. In spite using the pytorch optimized version [52], inference using PSPNet is computationally complex. If the inference is done for every frame, segmentation becomes the main bottleneck for achieving real-time performance on resource constrained devices. Therefore, we extract semantics only for selected frames i.e. keyframes rather than for all frames. However, the semantics are needed in every frame for moving objects removal, but the semantics gets outdated if used for the next frames in the sequence. To resolve this issue, we introduce semantics propagation using the available scene flow to compensate for the changes in the semantics from one frame to the next. This is called warping of semantics.

Warping semantics is in itself a computationally complex process, as it requires dense flow to perform warping between consecutive frames. Here, we only operate on uniformly spaced grid points, so that expensive dense optical flow can be replaced with already available feature correspondences. We adopt the idea of uniformly spaced flow from work on video stabilization [40], which is also known as mesh-flow. We aim to propagate the semantics using the flow due to the camera and moving objects. Thus, we create uniformly spaced sparse motion-field similar to mesh-flow while considering the true motion fields but without smoothing over past vertex profiles, and then using these motion-fields to warp the semantics.

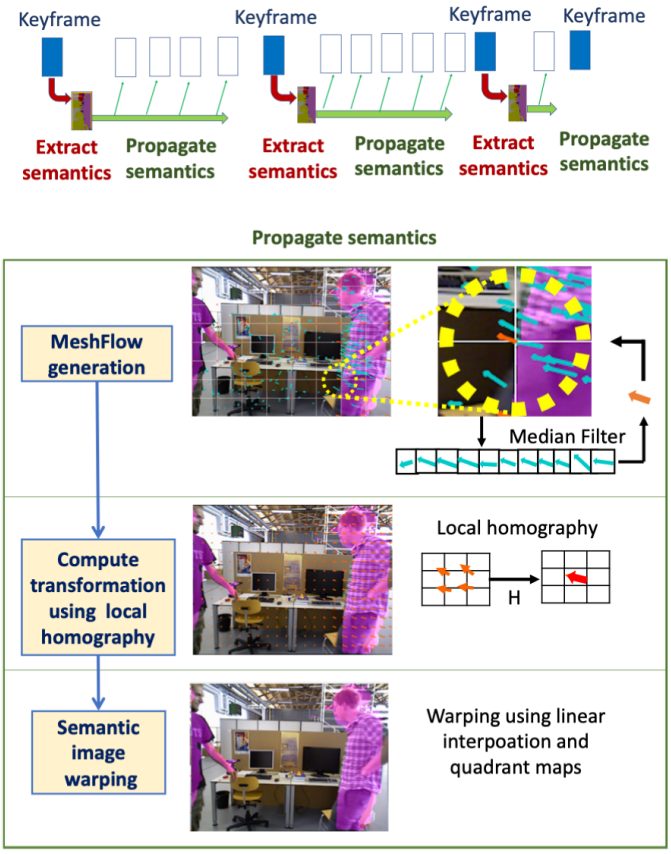


Fig. 4. **Mesh-flow generation**: Blue arrow at frame F_{t-1} shows the motion vector of the feature correspondence between frame F_t and F_{t-1}. This motion is propagated to the nearby grid vertexes within the eclipse using median filters to reduce effect of outliers. **Compute transformation**: The dense pixel-wise motion (orange arrows) is computed using quadrant homography. For display purpose uniform sparse motion is shown here. Larger arrow length indicates larger motion. **Semantic warping**: warped semantics overlapped with frame at t.

Specifically, we construct a uniform 2D grid (mesh) on the semantic image at time $t-1$. We then use the corresponding features between consecutive frames (F_{t-1} and F_t), to obtain motion vector at each feature location. For the i^{th} feature point $q^i_{t-1}(u^i_{t-1}, v^i_{t-1})$ at time step $t-1$ and its corresponding feature point $q^i_t(u^i_t, v^i_t)$ at time step t, the motion v^i at feature location q^i_t will be: $v^i = q^i_t - q^i_{t-1}$ (blue arrow in Fig. 4 (left most)). The neighbouring mesh vertices of the feature q^i_t should have a conforming motion as v^i [40]. So, each motion vector is transferred to its nearby mesh vertex, such that each mesh vertex receives motion vectors from its surrounding features within an eclipse. If the eclipse covering 3×3 cells is centered at vertex w_k (yellow circle in Fig. 4 (top)), any motion vector v^i is collected by the vertex w_k. As shown in Fig. 4, the vertex (orange) receives multiple motion vectors (blue arrows). To achieve spatial coherent motion at the vertices, a median filter f1 is used to filter the candidates as in mesh-flow [40], and the resulting filter response is then assigned back to w_k. This achieves sparse motion regularization and we obtain a uniformly spaced motion field (as illustrated in

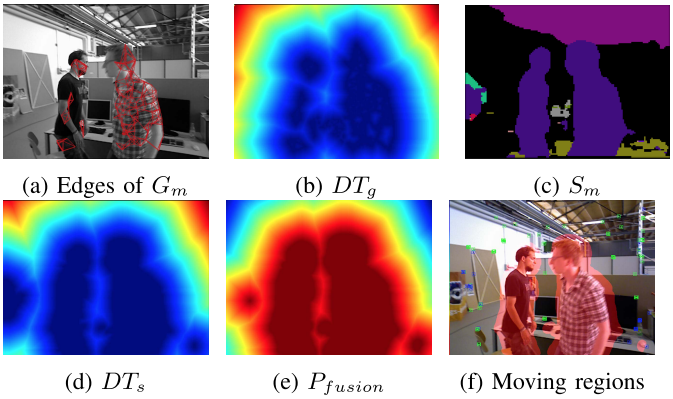
the middle of Fig. 4). Due to this sparse formulation, we obtain a lightweight spatially varying motion field. To get rid of outlier effects on motion field, another median filter f2 can be used for smoothing over nearby grid vertices. The first filter is used for removing small noises, while the second filter emphasizes on discarding large outliers. However, to avoid over-smoothing and removal of flow from moving objects, the f2 window size is kept small (2×2) depending on the chosen grid size. If grid size is kept large, it may lead to over-smoothing i.e. non-variant flow. On the other hand, if grid size is very small, it may receive few or no motion vector and is highly variant because of noise. Then, dense pixel-wise motion maps are computed using homography based on respective 4 quadrant vertices as follows. To compute the motion maps for a $quad(i, j)$, its corner pixels coordinates (u_i^{t-1}, v_j^{t-1}), $(u_i^{t-1}, v_{j+1}^{t-1})$, $(u_{i+1}^{t-1}, v_j^{t-1})$, $(u_{i+1}^{t-1}, v_{j+1}^{t-1})$ and their corresponding points (u_i^t, v_j^t),(u_i^t, v_{j+1}^t), (u_{i+1}^t, v_j^t), (u_{i+1}^t, v_{j+1}^t) are taken using the above mesh-flow and the homography matrix H is computed using 4-point method. Then, the dense maps i.e. correspondence pixel (u^{t+1}, v^{t+1}) for each pixel (u^t, v^t) inside the quadrant $quad(i, j)$ is found by transformation of (u^t, v^t) using H matrix. Finally, the semantic image from previous frame at time instant $t-1$ is remapped to the current frame at time instant $t-1$ using linear interpolation and dense maps according to mesh warp [53].

The moving cluster regions detected using SGC (last section) should be considered valid only when they are also semantically classified as movable class (e.g. person and car). However, the geometry motion cues and the semantic class cues contain significant uncertainties such that, a direct "AND" or "OR" operation of fusing both information will either lead to over-removal of a large number of static regions or failure in removing many moving outliers. Therefore, a fusion framework that takes into account the uncertainties in the two cues and robustly identifies the moving regions through the soft probabilistic fusion method is proposed as described in the following section.

C. Fusing Geometric and Semantic Cues

From SGC (Section III-A), the extracted G_m (Fig. 5(a)) provides an indication of the moving parts of the scene i.e. moving clusters (red edges). We generate a binary image $\mathcal{G}$ from the G_m such that, each pixel x_i in $\mathcal{G}$ is given value 1 if it lies inside moving cluster boundary in G_m (i.e estimated to be moving), otherwise 0. Another observation is obtained regarding dynamic classes (i.e. potential moving objects) in the form of semantic segmentation (Section III-B). It tells which regions of the images corresponds to dynamic classes. Similarly to the geometric method, a corresponding binary image $\mathcal{S}$ is obtained from the observed semantic image S_m to denote the potential moving objects. Thus, a pixel in $\mathcal{S}$ has a value 1 if corresponding pixel is labelled as one of the dynamic classes in S_m, otherwise value 0.

As discussed before, $\mathcal{G}$ and $\mathcal{S}$ contain uncertainties, most specially near the boundary regions of moving regions in $\mathcal{G}$ and dynamic classes in $\mathcal{S}$. Hence, the decision of final moving regions cannot be made directly using $\mathcal{G}$ and $\mathcal{S}$. Rather,



(a) Edges of G_m (b) DT_g (c) S_m (d) DT_s (e) P_{fusion} (f) Moving regions

Fig. 5. The process of fusion of SGC and semantic segmentation to obtain the moving region segmentation mask is depicted for sample frames from TUM dynamic sequence. The Edges (red) of geometrically detected moving clusters of G_m (a), geometric distance transform DT_g (b), semantics distance transform DT_s (d), probability after fusion P_{fusion} (e) and overlaid final moving regions (f) are shown.

we use a probabilistic model to compute the probability $p(g_i = mov|x_i)$ that a pixel x_i in original image corresponds to moving region based on $\mathcal{G}$ and the probability $p(s_i = dyn|x_i)$ that pixel x_i correspond to dynamic class based on $\mathcal{S}$. We utilize distance transform [54] to convert binary images $\mathcal{G}$ and $\mathcal{S}$ into the distance maps and then by applying Gaussian model we estimate the final fusion probability. The intuition is that the likelihood should decrease with the distance to the nearest region observed as moving region (value 1) in $\mathcal{G}$ or dynamic class (value 1) in $\mathcal{S}$. The probability model takes into consideration the uncertainty, and $\mathcal{G}$ and $\mathcal{S}$ can be fused in a more robust way.

Given the binary geometric observation map $\mathcal{G}$, a distance map DT_g is created using the distance transform technique, where Euclidean distance is adopted as illustrated in Fig. 5(b) and defined in (8) and (9).

$$D[i][j] = min\{Distance[(i, j), (x, y)] : B[x][y] = 1\} \quad (8)$$

$$Distance[(i, j), (x, y)] = \sqrt{(i-x)^2 + (j-y)^2} \quad (9)$$

Here, B refers to the binary map. Based on the distance map DT_g, we define the probability $p(g_i = mov|x_i)$ that pixel x_i correspond to moving region (mov) as

$$p(g_i = mov|x_i) = \frac{1}{\sqrt{2\pi}\sigma_g} e^{-\frac{1}{2\sigma_g^2} DT_g(x_i)^2} \quad (10)$$

where σ_g models the uncertainty in the SGC.

Similarly, given the binary semantic observation map $\mathcal{S}$, another distance map DT_s is created using (8) and (9). Based on DT_s (Fig. 5(d)), the probability $p(s_i = dyn|x_i)$ that pixel x_i correspond to dynamic class (dyn) is defined as

$$p(s_i = dyn|x_i) = \frac{1}{\sqrt{2\pi}\sigma_s} e^{-\frac{1}{2\sigma_s^2} DT_s(x_i)^2} \quad (11)$$

where σ_s models the uncertainty in the semantic image classification.

Since semantic segmentation $\mathcal{S}$ and scene flow based geometric segmentation $\mathcal{G}$ are independently estimated, we can reduce the individual uncertainty of detecting actual moving

regions by fusing both the estimates as follows to get true moving object likelihood $p\,(f_i = trueM|x_i)$.

$$p(f_i = trueM|x_i) = p(s_i = dyn|x_i) \cdot p(g_i = mov|x_i) = \frac{1}{2\pi\,\sigma_s\sigma_g} e^{-}\Big(\frac{DT_s(x_i)^2}{2\sigma_s^2} + \frac{DT_g(x_i)^2}{2\sigma_g^2}\Big) \tag{12}$$

The fused region can then be computed in terms of fusion probability as shown in Fig. 5(e), based on $\sigma_s = 40$ pixels and $\sigma_g = 80$ pixels. These values depend on the individual quality of segmentation. The final mask for definite moving region is evaluated by thresholding this estimated fusion probability as shown in Fig. 5(f). The threshold P_{high} is taken as 0.85 in the current implementation, which is selected based on the qualitative experiments.

As illustrated in Fig. 5, the geometric clusters (red clusters edges in Fig. 5(a) and its probability map in Fig. 5(b)) are not able to extract the complete moving persons due to the sparsity of features. But after applying the proposed fusion approach to get probability distribution, we can extract more accurate mask for the moving person as shown in Fig. 5(f). On the other hand, semantic segmentation is able to extract most regions of the person class (Fig. 5(c)), but it mis-classifies some of the regions (false purple blob that is semantically segmented as person in Fig. 5(c)). Our fusion approach removes such unwanted regions which were falsely segmented as dynamic class by semantic segmentation. As shown in Fig. 5(f), the false blob that was segmented as person is removed in the final mask. This is because, the binary mask obtained from geometric clustering detected this particular region as static.

Moreover, instead of directly removing all the features within the mask region using hard threshold, we also propose to use feature correspondences weighting in the pose estimation based on the estimated fusion probability. This helps in down-weighting the effect of features related to movable objects such as box or ball as seen in [8] datasets. As shown in Fig. 6 the feature correspondence that lie in the region of higher fusion probability p_f than P_{high} are directly removed i.e., given weighting factor $w_f = 0$, but feature correspondences lying in the mid range i.e., p_f between P_{high} and P_{low} are weighted in the pose estimation proportional to their fusion probability. The features lying below P_{low} regions are given highest weight $w_f = 1$, where $P_{high} = 0.85$ and $P_{low} = 0.5$. For simplicity, we used linear weighting directly injected as feature uncertainty factor in the pose optimization [5], the weighted information matrix is used inside robust kernel function. Other smoother weighting function can also be used (Fig. 6).

D. Moving Outlier Removal and Final Pose Estimation

The final mask obtained from the fusion approach is used to filter out the feature correspondences lying within the mask, leaving only static features for the pose and map estimation. After removing such features, in certain scenes where large proportion of the features belong to moving outliers, only few static features remain for good pose estimation. Thus, in such conditions we propose adaptive feature count for the SLAM as follows. In usual SLAM pipeline, the number of features extracted from each incoming image frame is fixed (say N) beforehand. Let number of feature matches (correspondences) obtained be n (variable). In frames containing large proportion of moving outliers, let the number of feature correspondences that remain after moving outlier removal be m. In the adaptive feature feature count, if m drops below M, we increase N step-wise by 200 features for the next frame, thus maintaining enough static features for final pose estimation. In the current work, N is set as 1000, and M as 30. The final pose is then estimated using remaining features, by full pose optimization using motion only bundle adjustment and pose refinement as in baseline ORB-SLAM2 [5].

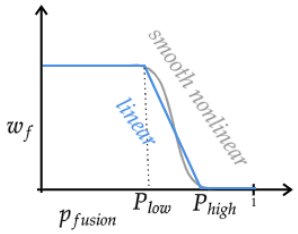


Fig. 6. Feature correspondences weighting based on fusion probability. Linear weighting in the mild dynamic region and directly removing features in highly dynamic regions.

IV. Experiments

A. Datasets

Most existing vSLAMs are evaluated on KITTI [6] benchmark and TUM-RGBD [7] datasets. KITTI was captured in low dynamic and static environments, and do not contain enough dynamic elements to test the fundamental assumptions of conventional vSLAMs. For instance, KITTI benchmark dataset only contains a few moving vehicles in certain video sequences (sequence 04) and does not have significant harmful effects on vSLAM estimates. On the other hand, TUM-RGBD dataset contains a separate set of seven dynamic sequences that we use for testing the robustness against dynamic elements in the scene. To further evaluate and analyze the proposed framework, we also include RGBD-BONN [8], which contains 24 highly dynamic challenging sequences. In ablation studies, we additionally run experiments on synthetic virtual KITTI (vKITTI) [55] dataset that provide two dynamic sequences (18 and 20). For RGB-D indoor datasets i.e. TUM and BONN, the absolute trajectory error (ATE) is a standard metric for evaluation, which gives average of absolute pose errors over a trajectory. The RMSE (root mean square error) of ATE is widely used to reduce randomness over multiple runs. Whereas, for outdoor KITTI datasets [6], average of percent pose errors (APPE) relative to different sub-sequences of length $(100, \ldots, 800)$ meters is used as standard metric. To account for the randomness, we take average of APPE over five experimental runs.

B. Implementation Details

We implemented our SGCSF-K SLAM on top of both ORB-SLAM2 and ORB-SLAM3 to compare with the state-of-the-art vSLAM systems. We conduct experiments on both

TABLE I
PARAMETER SETTINGS USED IN PROPOSED APPROACH

Parameters	Value
KNN graph factor K	5
Mahalanobis pruning threshold τ_{Mahala}	0.01*camera_fps (indoor) 0.15*camera_fps (outdoor)
ARE threshold τ_{ARE}	2 (indoor) 5 (outdoor)

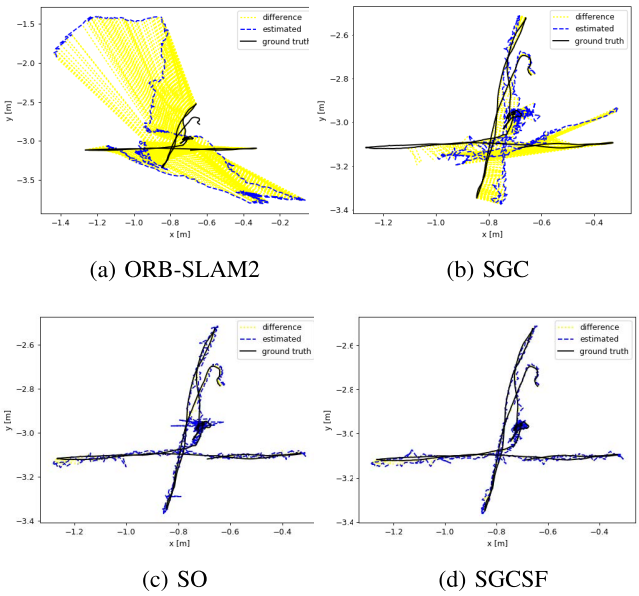
stereo and RGB-D setups, whereas some of the discussed vSLAMs were only implemented on RGB-D setup e.g., DS-SLAM, Refusion and SF. At the time of writing this paper, most existing methods still utilize ORB-SLAM2 i.e. DS-SLAM [19], Detect-SLAM [11], DynaSLAM [17] and BGS [20]. Therefore, for the ease of comparison, we also include the base ORB-SLAM2.

The pixelwise semantic labels are generated using pytorch implementation of PSPNet model [50] trained on Cityscape dataset for outdoor application and trained on ADEK dataset for the indoor application. For the outdoor datasets we select vehicle and pedestrian classes of objects as dynamic category. For the indoor datasets the people class is set to be in highly dynamic category, and movable objects i.e. box and ball as low dynamic category. The embedded platform used for the implementation of the experiments is NVIDIA Jetson TX1, which is a low-power embedded computing board with quad-Core ARM processor, NVIDIA Maxwell GPU with 256 CUDA Cores and 4 GB 64-bit LPDDR4 memory. For the basic SLAM pipeline i.e. feature extraction, tracking and mapping, we use CPU cores without any GPU acceleration. The Jetson GPU is used only for running semantic inference. Rest of the parameter settings are given in Table I.

C. Baseline vSLAMs

ORB-SLAM3 is considered as one of the state-of-art vSLAM, which performs well in various indoor and outdoor environments. Therefore, we include it as a baseline for the experiments. We also include other recent vSLAMs that target the dynamic situations i.e. DS-SLAM [19], Detect-SLAM [11], Refusion [25], Static-fusion i.e. SF [56], DynaSLAM [17] and BGS [20]. We run the open source code of ORB-SLAM3, DS-SLAM and BGS for the experiments by keeping their best parameter settings, and utilize the experimental values of Detect-SLAM, Refusion, DynaSLAM and SF reported in their paper.

1) Ablation Studies: We analyze the constituting components of the proposed SGCSF-K SLAM. Specifically, we study the effectiveness of the proposed feature density-based spatial clustering (SGC) for the purpose of moving object detection, and compare it with the base. Then we discuss the effectiveness of semantics only (SO) approach in tackling the dynamic elements. Further, we compare the SGCSF i.e., fusion of SGC and SO to remove moving objects with respect to the above geometric (SGC) and semantic (SO) approaches. Finally, the error and timing analysis of the complete SGCSF-K is performed to validate the effectiveness of semantic propagation. All these components have the same base as ORB-SLAM2 for ablation studies.



(a) ORB-SLAM2 (b) SGC

(c) SO (d) SGCSF

Fig. 7. TUM trajectory plots for ORB-SLAM2, SGC, SO and SGCSF with respect to the groundtruth of sequence fr3_wlk_xyz. Longer yellow lines indicate larger difference in pose from groundtruth.

a) Effectiveness of the spatial graph clustering (SGC): To evaluate the effectiveness of proposed method in removing moving objects from the scene, we conduct experiments on indoor TUM-RGBD dynamic scenes and outdoor vKITTI dynamic scene by utilising only the proposed SGC to detect and remove moving features. Results are shown in Table II and Table III. The comparison with the baseline, clearly shows that pose errors for SGC reduce significantly i.e. average error (drift) reduction of 68% on TUM and 53% on vKITTI. This validates the hypothesis that many outlier features of moving objects remain undetected through RANSAC outlier removal. These results also show that a pure geometric approach based on sparse features can reduce a significant number of the moving outliers.

We further analyze the proposed SGC approach by comparing it with another graph clustering approach i.e. delaunay triangulation based graph clustering (BG) reported in [20]. BG performs connected component clustering over the Delaunay triangulation graph [15]. As shown in Table II, SGC achieves lower pose errors than BG on sequences: fr3_wlk_xyz, fr3_wlk_rpy, fr3_wlk_half and fr3_sit_half. The overall reduction of average trajectory error (ATE) is from 0.124m (BG) to 0.114m (SGC). The overall reduction in pose error on vKITTI (Table III) is 1.67m (BG) to 1.20m (SGC). One primary reason for the improvement lies in the graph construction. The Delaunay triangulation graph (as in BG) contains large proportion of edges that connect the far features (implying features having largely different scene flow), as can be seen in Fig. 8 (left) before-pruning. After pruning removes these edges with different scene flow, the clusters obtained are highly fragmented, which only captures the moving regions partially, as shown in Fig. 8 (left). On the other hand, the proposed feature density based clustering performed on KNN-graph contains sufficient nearby edges. Hence, even after pruning and clustering, it captures significantly larger

TABLE II

POSE ERROR ON DYNAMIC TUM RGB-D DATASET, SHOWING RMSE OF ABSOLUTE TRAJECTORY ERRORS (ATE) IN METERS

Dynamicity level	*TUM Seq.*	*Baseline*	*BG (triang. graph clust.)*	*SGC (spatial graph clust.)*	*SO*	*SGCSF (fusion)*	*SGCSF-K (fusion with KF-only sem. prop.)*
High	*fr3_wlk_xyz*	0.752	0.149	0.122	0.025	**0.014**	0.015
High	*fr3_wlk_stat*	0.390	0.081	0.080	0.014	**0.007**	**0.007**
High	*fr3_wlk_rpy*	0.870	0.338	0.301	0.301	**0.029**	**0.029**
High	*fr3_wlk_half*	0.486	0.263	0.260	0.032	**0.025**	**0.025**
Low	*fr3_sit_stat*	0.009	0.008	0.007	0.007	**0.006**	**0.006**
Low	*fr3_sit_xyz*	**0.009**	**0.009**	**0.009**	0.014	0.010	0.010
Low	*fr3_sit_half*	0.026	0.026	0.021	0.019	**0.014**	**0.014**
	average	0.363	0.124	0.114	0.059	**0.015**	**0.015**

TABLE III

POSE ERROR ON VKITTI DATASET, SHOWING RELATIVE POSE ERRORS TO MEASURE THE DRIFT IN METERS

Dynamicity level	*vKITTI Seq*	*Baseline*	*BG (triang. graph clust.)*	*SGC (spatial graph clust.)*	*SO*	*SGCSF (fusion)*	*SGCSF-K (fusion with KF-only sem. prop.)*
Very low	*01*	**0.12**	**0.12**	**0.12**	0.20	**0.12**	**0.12**
Very low	*02*	**0.12**	**0.12**	**0.12**	**0.12**	**0.12**	**0.12**
Very low	*06*	0.45	0.41	0.41	0.36	**0.35**	0.43
Moderate	*18*	0.30	0.28	0.27	0.25	**0.20**	0.22
Moderate	*20*	11.89	7.42	5.08	6.00	3.54	**3.45**
	average	2.58	1.67	1.20	1.39	**0.87**	**0.87**

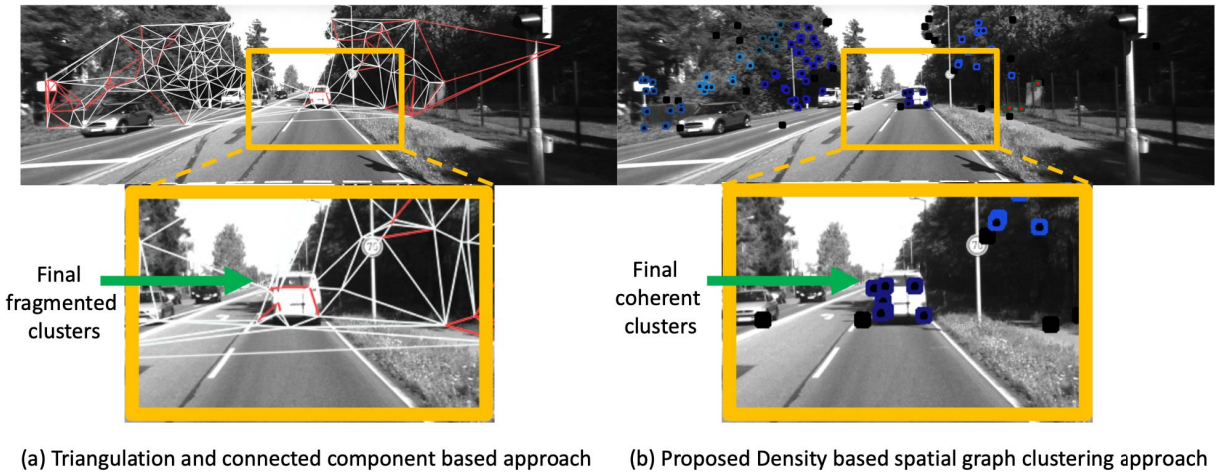


Fig. 8. Geometric clustering comparison: triangulation based graph clustering (left) used in BG versus proposed KNN graph and density based clustering i.e. SGC on flow coherent spatial-graph (right).

moving regions. This phenomenon can also be seen in Fig. 8 (right). It is noteworthy that the proposed SGC uses sparse feature matches which are already available in vSLAM to avoid additional computation overhead. The second reason for the improvement is the use of SGC instead of connected component retrieval (BG). This is because density-based clustering performs clustering by utilizing the spatial distribution of nodes over the connected graph. Thus, it is able to extract multiple clusters within a sub-graph if the distribution is multivariate and avoids the problem of over-segmentation. On the contrary, connected component retrieval can lead to over-segmentation that sometimes merges different objects into one cluster and such segments tend to overshadow (masks) the in-between static regions.

b) Effectiveness of semantics only information (SO): As semantics offer another cue for detecting potential moving objects, we inspect the effectiveness of semantics-only (SO) approach for feature removal. The semantics provide explicitly labelled region for the potential moving objects. In this ablation study, we choose PSPNet as the semantic segmentation model. The effectiveness of using SO can be easily verified from the improved pose estimates i.e. lower pose errors in Table II and Table III. On TUM (Table II), the proposed SO shows decrease in pose error over baseline, and significant improvement on vKITTI dynamic sequence 18 and 20 (Table III) due to better handling of dynamic elements. As expected, when the frames contain all dynamic objects (dynamic TUM-RGBD) that are actually in motion, using semantics based feature removal improves the pose estimates compared to the baseline. However, in frames where most of these potential moving objects are not in motion, the error increases. As can be seen in sequence 01 of vKITTI Table III, SO increases the pose errors. This is the major limitation of semantics-only approach and therefore our method fuses both geometric and semantic cues.

c) Effectiveness of probabilistic fusion (SGCSF): To test the effectiveness of the fusion approach, SGCSF (SGC+SO), we compare SGCSF with geometric-only approach SGC and semantic-only approach SO, as shown in Table II and III. The probabilistic fusion SGCSF approach achieves significantly

TABLE IV

POSE ERROR AND TIMING RESULTS ON DYNAMIC TUM RGB-D DATASET, SHOWING RMSE OF ABSOLUTE TRAJECTORY ERRORS (ATE) IN METERS AND AVERAGE TRACKING TIME PER FRAME IN SECONDS (IMPLEMENTED ON JETSON TX1)

TUM Seq.	*SGCSF PSPN*		*SGCSF-K PSPN*		*SGCSF EdgeN*		*SGCSF-K EdgeN*	
	error	time	error	time	error	time	error	time
fr3_wlk_xyz	**0.014**	2.065	0.015	0.804	0.016	0.346	0.017	**0.219**
fr3_wlk_stat	**0.007**	2.060	**0.007**	0.811	0.008	0.342	0.009	**0.217**
fr3_wlk_rpy	**0.029**	1.840	**0.029**	0.692	0.030	0.339	0.030	**0.204**
fr3_wlk_half	**0.025**	2.078	**0.025**	0.762	0.025	0.340	0.027	**0.209**
fr3_sit_stat	**0.006**	2.069	**0.006**	0.823	0.007	0.348	0.008	**0.218**
fr3_sit_xyz	**0.010**	2.160	**0.010**	0.826	0.012	0.347	0.012	**0.219**
fr3_sit_half	**0.014**	2.120	**0.014**	0.829	0.016	0.341	0.018	**0.219**
average	**0.015**	2.056	**0.015**	0.792	0.016	0.343	0.017	**0.215**

lower pose errors of 0.015m on TUM and 0.87m on vKITTI, in comparison to the SGC (with 0.114m on TUM and 1.20m on vKITTI) and SO (0.59m on TUM and 1.39m on vKITTI) (see Table II and Table III). This is due to the fact that SGCSF combines the individual cues together in probabilistic manner which overcomes the false static features removal problem of SO and incomplete masks problem of SGC, thus reducing the effect of individual uncertainties. The trajectory plots of SGC, SO, SGCSF are shown in Fig. 7 for the TUM sequence fr3_wlk_xyz.

d) Effectiveness of proposed SGCSF-K: The proposed SGCSF-K only performs semantic segmentation on keyframes, and the semantics are propagated in the intermediate frames. The performance of SGCSF-K with SGCSF (semantics extracted for every frame) is shown in Table IV. Furthermore, we also propose a much lighter version of SGCSF-K that utilizes EdgeNets, a fast implementation of ESPNetV2 [57]. EdgeNets has lower semantic segmentation quality than PSPNet but faster training and inference times. The trade-offs are discussed below.

First, we discuss the pose error performance using the PSPNet version. As can be seen (Table IV), the overall average pose error of the SGCSF-K PSPNet with (0.015m) is almost similar to SGCSF PSPNet (0.015m), except for a slight increase on sequence fr3_wlk_xyz. This is justified because SGCSF-K use the propagated semantics instead of the semantics generated using the PSPNet model for each frame. A similar performance is observed on vKITTI (Table III). The example absolute pose errors and xz-trajectory of the sequence 20 of vKITTI are shown in Fig. 9. The main advantage of semantic propagation is reduction in tracking time. The average tracking time reduces from 2.056 sec per frame (SGCSF PSPNet) to 0.792 sec per frame (SGCSF-K PSPNet) i.e. a 61% decrease in the average tracking time per frame. The average tracking time of SGCSF-K EdgeNets is 0.215 seconds, which is achieved at the cost of minor increase in pose error. It is noteworthy that even with sub-standard semantics from EdgeNets, SGCSF-K is still able to achieve low pose errors due to support from the proposed geometric SGC.

2) Comparison With the State-of-the-Art: In this section, we compare the two versions of the proposed dynamic vSLAM framework, SGCSF-K PSPNet and SGCSF-K EdgeNet, with DynaSLAM, DS-SLAM, and BGS, which are most relevant to our work, on the dynamic TUM-RGBD and RGBD-BONN. A few other related works that target dynamic environments are Refusion and SF (only for RGBD-BONN and TUM-RGBD). For completeness, we also discuss their results on KITTI benchmark dataset. Our implementations with ORB-SLAM3 are denoted as SGCSF-K PSPNet base ORB-SLAM3 and SGCSF-K EdgeNet base ORB-SLAM3 in Tables V, VI, VII, VIII. The implementations with ORB-SLAM2 are denoted as SGCSF-K PSPNet base ORB-SLAM2 and SGCSF-K EdgeNet base ORB-SLAM2.

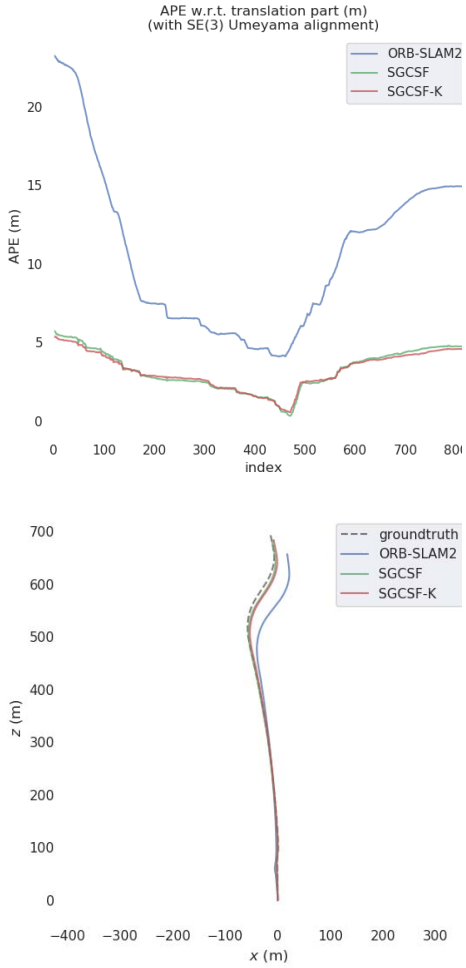


Fig. 9. APE (absolute pose error in metres) comparison [top] and trajectory comparison [bottom] for sequence 20 of vKITTI.

TABLE V

RESULTS ON TUM RGB-D DATASET, SHOWING RMSE OF ABSOLUTE TRAJECTORY ERRORS (ATE) IN METERS

TUM Seq.	*ORB-SLAM2*	*ORB-SLAM3*	*DS-SLAM*	*Detect-SLAM*	*Refusion*	*SF*	*DynaSLAM*	*BGS*	*SGCSF-K PSPN base ORB-SLAM2*	*SGCSF-K EdgeN base ORB-SLAM2*	*SGCSF-K PSPN base ORB-SLAM3*	*SGCSF-K EdgeN base ORB-SLAM3*
fr3_wlk_xyz	0.752	0.426	0.025	0.024	0.099	0.093	0.017	0.015	0.015	0.017	**0.014**	0.016
fr3_wlk_stat	0.390	0.780	0.008		0.017	0.015	**0.007**	**0.007**	**0.007**	0.009	0.008	0.009
fr3_wlk_rpy	0.870	0.437	0.444	0.296				0.029	0.029	0.030	**0.027**	0.028
fr3_wlk_half	0.486	0.326	0.030	0.051	0.104	0.681	0.026	**0.025**	**0.025**	0.027	**0.025**	0.026
fr3_sit_stat	0.009	0.010	**0.006**		0.009	0.014	0.007	**0.006**	**0.006**	0.008	**0.006**	0.008
fr3_sit_xyz	**0.009**	**0.009**	0.018	0.020	0.040	0.039	0.015	0.011	0.010	0.012	0.010	0.012
fr3_sit_half	0.026	0.025	0.026	0.023	0.110	0.041	0.028	0.019	**0.014**	0.018	**0.014**	0.018
average	0.363	0.287	0.079	0.082	0.063	0.147	0.017	0.016	**0.015**	0.017	**0.015**	0.017

TABLE VI

RESULTS ON BONN RGB-D DATASET, SHOWING RMSE OF ABSOLUTE TRAJECTORY ERRORS (ATE) IN METERS

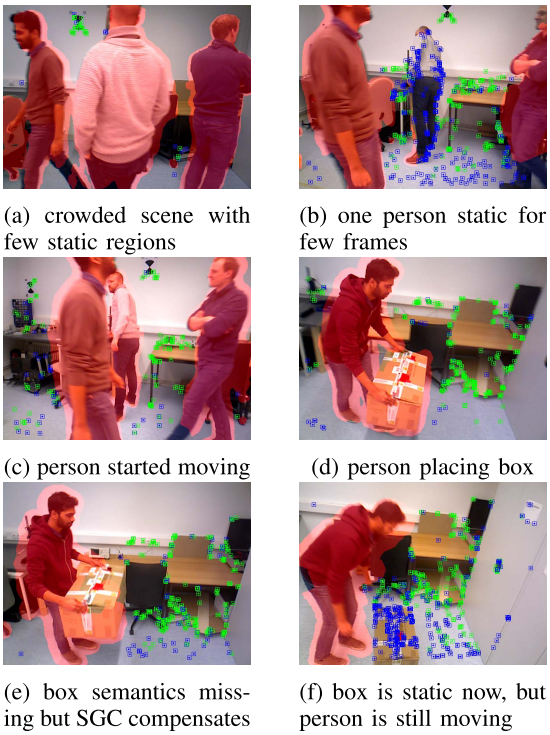
BONN Seq.	*ORB-SLAM2*	*ORB-SLAM3*	*DS-SLAM*	*Refusion*	*SF*	*DynaSLAM*	*BGS*	*SGCSF-K PSPN base ORB-SLAM2*	*SGCSF-K EdgeN base ORB-SLAM2*	*SGCSF-K PSPN base ORB-SLAM3*	*SGCSF-K EdgeN base ORB-SLAM3*
bal	0.431	0.529	0.041	0.175	0.233	**0.030**	0.038	0.035	0.036	0.036	0.036
bal2	0.228	0.216	0.031	0.254	0.290	0.029	0.033	0.028,	0.039	**0.027**	0.038
bal_trac	0.272	0.265	0.044	0.302	0.221	0.049	0.040	**0.034**	0.036	**0.034**	0.035
bal_trac2	0.054	0.061	**0.033**	0.322	0.366	0.035	0.051	0.045	0.046	0.045	0.046
crowd	0.796	0.613	0.073	0.204	3.586	**0.016**	0.035	0.032	0.053	0.032	0.049
crowd2	1.186	0.954	0.056	0.155	0.215	**0.031**	0.044	0.038	0.042	0.037	0.040
crowd3	0.658	0.572	0.083	0.137	0.168	**0.038**	0.063	0.051	0.064	0.049	0.060
kid_box	0.028	0.028	0.049	0.148	0.336	0.029	0.047	**0.025**	0.026	**0.025**	0.026
kid_box2	**0.024**	**0.024**	0.035	0.161	0.263	0.035	0.034	0.025	0.025	0.025	0.025
mov_nonObs_box	0.098	0.097	0.080	0.071	0.141	0.232	0.065	**0.023**	0.024	**0.023**	0.024
mov_nonObs_box2	0.040	0.042	0.046	0.179	0.364	0.039	0.044	textbf0.035	0.036	textbf0.035	0.036
mov_ob_box	0.649	0.637	0.105	0.343	0.331	0.044	0.307	**0.033**	0.040	**0.033**	0.039
mov_ob_box2	0.681	0.670	0.428	0.528	0.309	**0.263**	0.402	0.319	0.358	0.298	0.341
per_track	0.684	0.668	0.070	0.289	0.484	0.061	0.071	0.041	0.045	**0.040**	0.043
per_track2	0.949	0.865	0.067	0.463	0.626	0.078	0.068	0.036	0.036	**0.035**	0.036
pl_nonObs_box	0.829	0.801	0.120	0.106	0.125	0.575	0.086	**0.023**	0.027	**0.023**	0.027
pl_nonObs_box2	0.061	0.058	0.045	0.141	0.177	0.021	0.081	**0.021**	0.028	**0.021**	0.027
pl_nonObs_box3	0.338	0.325	0.060	0.174	0.256	0.058	0.052	0.033	0.035	**0.032**	0.035
pl_obs_box	0.352	0.356	0.375	0.571	0.330	**0.255**	0.373	**0.255**	0.276	0.256	0.278
re_nonObs_box	0.023	0.023	0.028	0.041	0.136	0.016	0.027	**0.015**	0.017	**0.015**	0.017
re_nonObs_box2	0.026	0.025	0.036	0.111	0.129	**0.021**	0.035	0.024	0.027	0.024	0.026
re_ob_box	0.362	0.357	0.362	**0.222**	0.334	0.291	0.341	0.317	0.348	0.311	0.340
synch	1.079	0.991	0.038	0.441	0.446	0.015	0.037	**0.014**	0.016	**0.014**	0.016
synch2	1.370	1.349	0.024	0.022	0.027	0.009	0.023	0.009	0.010	**0.008**	0.009
avg.	0.468	0.438	0.097	0.232	0.412	0.095	0.100	0.063	0.070	**0.062**	0.069

a) Performance on TUM-RGBD dynamic dataset: The comparison on TUM dynamic dataset in Table V shows that SGCSF-K PSPNet with ORB-SLAM2 and ORB-SLAM3 outperforms all other methods significantly as can be seen in overall average pose errors. The only exception is sequence fr3_sit_xyz, where ORB-SLAM2 and ORB-SLAM3 have a slightly lower pose errors. This is because fr3_sit_xyz sequence contains very slow moving person which is detected by the proposed SGCSF-K SLAM and causes removal of the whole person. This resulted in a minor increase in pose error on fr3_sit_xyz. The proposed SGCSF-K EdgeNet with ORB-SLAM2 and ORB-SLAM3 also performs on par with most of the other vSLAMs in terms of overall pose errors. Notably, ORB-SLAM3 has a pose error of 0.438m, which is lower than ORB-SLAM2 (0.468m). This is mainly due to the improved loop detection (place recognition module) of ORB-SLAM3. However, the proposed SGCSF-K method already removes most inliers to boost accuracy, therefore the final integrated "SGCSF-K PSPNet base ORB-SLAM3" has only marginally lower error than "SGCSF-K PSPNet base ORB-SLAM2" (which is not reflected in the table due to averaging).

b) Performance on bonn RGB-D dynamic dataset: The second set of experiments is performed on a recent dynamic Bonn RGB-D dataset, consisting of 24 highly dynamic scenes. In this dataset, people perform different tasks such as manipulating boxes or playing with balloons. These tasks often obstruct the camera and create challenging conditions for continuous camera tracking. Table VI shows the performance of different approaches on this dataset. As can be observed from the overall average error performance, SGCSF-K PSPNet base ORB-SLAM3 (0.062m) achieves the lowest errors among DS-SLAM (0.088m), Refusion (0.232m), DynaSLAM (0.095m) and BGS (0.100m). SGCSF-K PSPN base ORBSLAM3 has average pose errors of 0.062m, which is slightly lower than SGCSF-K PSPN base ORBSLAM2 (0.063m). Similarly, SGCSF-K EdgeN base ORBSLAM3 has pose errors of 0.069m as compared to 0.070m of SGCSF-K EdgeN base ORBSLAM2.

To analyze the consistency of the pose error performance, the deviation of error is important, hence we plot the errors for all the sequences using box plot as shown in Fig. 11. It is evident that the proposed SGCSF-K PSPN base ORB-SLAM2 and SGCSF-K PSPN base ORB-SLAM3 have the lowest maximum errors than the counterparts. Furthermore, the variance of the proposed SGCSF-K PSPN and SGCSF-K PSPN is also considerably lower than other methods, which implies that the proposed SGCSF-K performs consistently better than other methods on varying challenging dynamic conditions. The qualitative performance of SGCSF-K is also shown in Fig. 10.

c) Performance on KITTI dataset: This final set of experiments (Table VII) are performed to demonstrate that our framework does not overfit on a particular type of input data. We chose the well-known outdoor KITTI benchmark dataset for this experiment. The KITTI sequences contains very few scenes with moving objects, for example sequence 04. Sequence 04 contains a van moving in z-direction of the camera movement and hence is considered particularly challenging



(a) crowded scene with few static regions

(b) one person static for few frames

(c) person started moving

(d) person placing box

(e) box semantics missing but SGC compensates

(f) box is static now, but person is still moving

Fig. 10. The intermediate qualitative results of SGCSF-K in challenging BONN scenes: the red highlighted regions indicate high fusion probability, while remaining features are weighted in pose estimation. The blue points are the pure VO features and green are the already registered map-points. As can be seen in (b), the features on static person (highly dynamic potential candidate) are being used for VO only, but not included in the map. Proposed SGCSF-K is able to detect even small movements when person in (c) starts to move again. In (d) when box becomes static, corresponding features are used for VO.

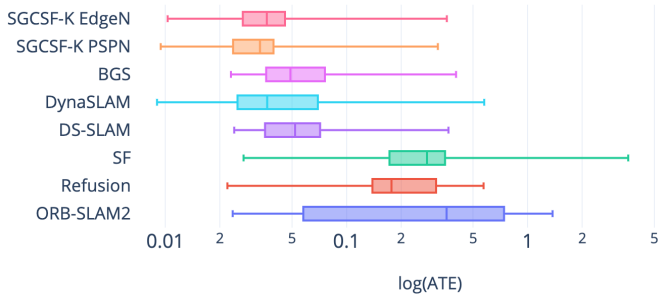


Fig. 11. Bonn error analysis: Box-plot showing pose error spread.

to detect using sparse feature flow with higher uncertainty in the depth information. Sequence 01 also contains a only few cars passing the ego vehicle. Other sequences are mostly static scenes. Hence the pose error reduction is significant only in sequence 04. The average percentage pose errors remain same from SGCSF-K EdgeN base ORBSLAM2 to SGCSF-K EdgeN base ORBSLAM3, and SGCSF-K PSPN base ORBSLAM2 to SGCSF-K PSPN base ORBSLAM3. In these environments, BGS has a slightly lower average error (0.699%) than the proposed SGCSF-K PSPN (0.699%) and SGCSF-K EdgeN (0.708%). This is because BGS uses semantics, which are directly obtained from the deep learning model at the expense of longer runtime. On the other hand, the

TABLE VII
KITTI DATASET: AVERAGE PERCENTAGE (% OF TRAJECTORY LENGTH) OF POSE ERROR (APPE)

KITTI Seq.	*ORB-SLAM2*	*ORB-SLAM3*	*DynaSLAM*	*BGS*	*SGCSF-K PSPN base ORBSLAM2*	*SGCSF-K EdgeN base ORBSLAM2*	*SGCSF-K PSPN base ORBSLAM3*	*SGCSF-K EdgeN base ORBSLAM3*
0	0.70	0.71	0.74	**0.68**	**0.68**	0.70	**0.68**	0.70
1	1.39	1.39	1.57	1.36	**1.35**	1.36	**1.35**	1.36
2	**0.76**	**0.76**	0.80	**0.76**	**0.76**,	**0.76**	**0.76**	**0.76**
3	**0.71**	**0.71**	0.69	0.73	**0.71**	**0.71**	**0.71**	**0.71**
4	0.48	0.48	0.45	0.39	**0.38**	**0.38**	**0.38**	**0.38**
5	0.40	0.40	0.40	**0.38**	**0.38**	0.39	**0.38**	**0.38**
6	0.51	0.51	0.50	**0.48**	0.49	0.49	0.49	0.49
7	0.50	0.49	0.52	**0.47**	0.50	0.50	0.50	0.50
8	1.05	1.05	1.05	**1.03**	1.04	1.04	1.04	1.04
9	0.87	0.87	0.93	**0.84**	0.86	0.86	0.86	0.86
10	0.60	0.60	0.67	**0.56**	0.60	0.60	0.60	0.60
average	0.724	0.724	0.756	**0.698**	0.699	0.708	0.699	0.708

TABLE VIII
COMPARISON OF THE STATE-OF-ART VSLAMS FOR RUNTIME AND POSE ERRORS OF THE MOTION ESTIMATION IN DYNAMIC SCENES. HERE $*$ DENOTES THE RUNTIME (PER FRAME) EVALUATED ON MORE POWERFUL HARDWARE. RED HIGHLIGHT INDICATES HIGH POSE ERRORS AND HIGH FRAME TIMES. BLUE TEXT INDICATES LOWEST POSE ERRORS AND LOWEST FRAME TIMES. GREEN BOX INDICATES BEST TRADE-OFF AMONG ALL METHODS

	Frame time Mean(s)	*Pose error TUM*	*Pose error BONN*	*Platform (system configuration)*
ORB-SLAM2	0.122	0.363	0.468	*Jetson TX1*
ORB-SLAM3	0.139	0.287	0.438	*Jetson TX1*
DS-SLAM	0.640	0.079	0.097	*Jetson TX1*
BGS SLAM	2.022	0.016	0.100	*Jetson TX1*
DynaSLAM	0.738*	0.017	0.095	*GPU*
Refusion	-	0.063	0.232	*GPU*
SF	0.030*	0.147	0.412	*GPU*
SGCSF-K (PSPNet) base ORBSLAM2	0.792	0.015	0.063	*Jetson TX1*
SGCSF-K (EdgeNet) base ORBSLAM2	0.215	0.017	0.070	*Jetson TX1*
SGCSF-K (PSPNet) base ORBSLAM3	0.808	0.015	0.062	*Jetson TX1*
SGCSF-K (EdgeNet) base ORBSLAM3	0.235	0.017	0.069	*Jetson TX1*

proposed keyframe based method SGCSF-K uses the semantic propagation method for non-keyframes to reduce the runtime. In some image frames of sequences 07 and 10, the features that fall on the car are too sparse, which degrades the propagation map and the subsequent semantics. This negatively impacts the effectiveness of the subsequent outlier removal step in our method. To deal with this problem, a feature distribution strategy is required to uniformly densify the features.

3) Timing Evaluation: One of the main advantage of the proposed SGCSF-K is in the keyframe semantics' propagation, which leads to computation cost savings from semantics extraction. The average tracking time per frame of different SLAMs are reported along with their mean average errors in Table VIII. ORB-SLAM2 has the lowest tracking time, but it does not handle dynamic scenes effectively. We aim to overcome the limitations of existing dynamic vSLAM systems by achieving closer-to-real-time performance on embedded platforms, while also improving the robustness in challenging environments. It can be observed that "SGCSF-K EdgeN base ORBSLAM2" has lower frame time complexity (i.e., 0.215s) than "SGCSF-K EdgeN base ORBSLAM3" (i.e., 0.235s), but only with a slightly lower pose error on BONN i.e., 0.069m compared to 0.070m. The increase in runtime from ORB-SLAM2 to ORB-SLAM3, is mainly because ORB-SLAM3 densifies data association to improve map accuracy and changes employed in the g2O optimizer, at the expense of a slightly higher computational cost [58].

Among the dynamic approaches (DS-SLAM, BGS, DynaSLAM, Refusion, SF and proposed), our SGCSF-K PSPNet base ORB-SLAM3 which has lowest overall errors, has a runtime of 0.808s that is lower than BGS (2.022s) and DynaSLAM (0.738s on powerful GPU). SF, Refusion and DS-SLAM have higher errors than the the SGCSF-K versions. Further, the fast version SGCSF-K EdgeNet ORB-SLAM2, with only 0.215s tracking time on Jetson, also achieves close to lowest overall errors after SGCSF-K PSPNet. Hence, there is a trade-off in time and accuracy. For more accurate requirements, SGCSF-K PSPNet can be used, but for much faster operational speed with near accurate performance, SGCSF-K EdgeNet is preferred. In summary, "SGCSF-K EdgeN base ORBSLAM2" and "SGCSF-K PSPN base ORBSLAM3" (highlighted in green in Table IV) achieves the best trade-off in time complexity and pose errors.

The typical timing cost SGCSF-K EdgeNet consist of $t_{light\_track}(0.0095s)$, $t_{identify_moving}(0.135s)$, $t_{remove\_outlier}(0.0038s)$, and the SLAM component timings $t_{preprocess}(0.0566s)$ and $t_{final\_track}(0.0306s)$. The $t_{identify\_moving}$, is dominated by semantics extraction, of which SGC (geometric clustering) and semantics extraction (for keyframes only) run in parallel. The average keyframe semantic extraction time is 1.971s and frame semantic propagation time is 0.0311s.

V. CONCLUSION

Our work addresses the dynamic outlier problem faced by existing vSLAM systems. While most existing systems utilize complex methods to avoid dynamic objects or use hard rules to remove features, we utilize already available information

in the vSLAM system i.e. feature flow between frames to detect moving regions and later fuse this information with semantic information. The proposed method only performs semantic segmentation on keyframes and propagate the semantic information to the intermediate frames. Extensive experiment results demonstrate that the proposed SGCSF-K achieves lowest pose errors at low computational cost compared to state-of-art dynamic SLAM systems.

REFERENCES

[1] D. Chekhlov, A. P. Gee, A. Calway, and W. Mayol-Cuevas, "Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual SLAM," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nov. 2007, pp. 1–4.

[2] K. Yousif, 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, Nov. 2015.

[3] D. Nistér, "Preemptive RANSAC for live structure and motion estimation," *Mach. Vis. Appl.*, vol. 16, no. 5, pp. 321–329, 2005, doi: 10.1007/s00138-005-0006-y.

[4] M. A. Fischler and R. 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, doi: 10.1145/358669.358692.

[5] R. Mur-Artal and J. D. Tardós, "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.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[7] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.

[8] 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. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 7855–7862.

[9] M. Kaneko, K. Iwami, T. Ogawa, T. Yamasaki, and K. Aizawa, "Mask-SLAM: Robust feature-based monocular SLAM by masking using semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 258–266.

[10] K. M. Judd, J. D. Gammell, and P. Newman, "Multimotion visual odometry (MVO): Simultaneous estimation of camera and third-party motions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 3949–3956.

[11] F. Zhong, S. Wang, Z. Zhang, C. Chen, and Y. Wang, "Detect-slam: Making object detection and slam mutually beneficial," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 1001–1010.

[12] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, "Three-dimensional scene flow," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, Sep. 1999, pp. 722–729.

[13] Y. Wang and S. Huang, "Towards dense moving object segmentation based robust dense RGB-D SLAM in dynamic scenarios," in *Proc. 13th Int. Conf. Control Autom. Robot. Vis. (ICARCV)*, Dec. 2014, pp. 1841–1846.

[14] Y. Fang and B. Dai, "An improved moving target detecting and tracking based on optical flow technique and Kalman filter," in *Proc. 4th Int. Conf. Comput. Sci. Educ.*, Jul. 2009, pp. 1197–1202.

[15] P. Lenz, J. Ziegler, A. Geiger, and M. Roser, "Sparse scene flow segmentation for moving object detection in urban environments," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 926–932.

[16] P. Ochs, J. Malik, and T. Brox, "Segmentation of moving objects by long term video analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1187–1200, Jun. 2013.

[17] B. Bescos, J. M. Fácil, J. Civera, and J. L. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, Oct. 2018.

[18] B. Bescos, C. Campos, J. D. Tardos, and J. Neira, "DynaSLAM II: Tightly-coupled multi-object tracking and SLAM," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5191–5198, Jul. 2021.

[19] C. Yu et al., "DS-SLAM: A semantic visual SLAM towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1168–1174.

[20] G. Singh, M. Wu, and S.-K. Lam, "Fusing semantics and motion state detection for robust visual SLAM," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 2764–2773.

[21] T. Pire, T. Fischer, G. Castro, P. D. Cristóforis, J. Civera, and J. J. Berlles, "S-PTAM: Stereo parallel tracking and mapping," *Robot. Auto. Syst.*, vol. 93, pp. 27–42, Jul. 2017.

[22] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D reconstruction in real-time," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2011, pp. 963–968.

[23] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 2043–2050.

[24] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 817–833.

[25] 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. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 7855–7862.

[26] B. K. Horn and B. G. Schunck, "Determining optical flow," *Proc. SPIE*, vol. 281, pp. 319–331, Nov. 1981.

[27] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "MID-fusion: Octree-based object-level multi-instance dynamic SLAM," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 5231–5237.

[28] M. Runz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 4471–4478.

[29] M. Henein, J. Zhang, R. Mahony, and V. Ila, "Dynamic SLAM: The need for speed," 2020, *arXiv:2002.08584*.

[30] M. R. U. Saputra, A. Markham, and N. Trigoni, "Visual SLAM and structure from motion in dynamic environments: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 1–36, Jun. 2018.

[31] M. Runz, M. Buffier, and L. Agapito, "MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Oct. 2018, pp. 10–20.

[32] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder–decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Jan. 2017.

[33] B. Yu et al., "30–4: Semantic simultaneous localization and mapping for augmented reality," in *SID Symp. Dig. Tech. Papers*, vol. 49, no. 1, May 2018, pp. 391–394.

[34] Y. Zhu et al., "Improving semantic segmentation via video propagation and label relaxation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8856–8865.

[35] X. Liang, E. Xing, and H. Zhou, "Dynamic-structured semantic propagation network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 752–761.

[36] D. Nilsson and C. Sminchisescu, "Semantic video segmentation by gated recurrent flow propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6819–6828.

[37] V. Badrinarayanan, F. Galasso, and R. Cipolla, "Label propagation in video sequences," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3265–3272.

[38] A. Y. C. Chen and J. J. Corso, "Propagating multi-class pixel labels throughout video frames," in *Proc. Western New York Image Process. Workshop*, Nov. 2010, pp. 14–17.

[39] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, Dec. 1996.

[40] S. Liu, P. Tan, L. Yuan, J. Sun, and B. Zeng, "MeshFlow: Minimum latency online video stabilization," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 800–815.

[41] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Visual SLAM: Why filter?" *Image Vis. Comput.*, vol. 30, no. 2, pp. 65–77, 2012.

[42] Y. Wu, Z. Zhang, T. S. Huang, and J. Y. Lin, "Multibody grouping via orthogonal subspace decomposition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2001, p. 2.

[43] E. Schubert, S. Hess, and K. Morik, "The relationship of dbscan to matrix factorization and spectral clustering," in *Proc. LWDA*, 2018, pp. 330–334.

[44] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011.
[45] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, vol. 31. Cham, Switzerland: Springer, 2013.
[46] L. McInnes, J. Healy, and S. Astels, "Hdbscan: Hierarchical density based clustering," *J. Open Source Softw.*, vol. 2, no. 11, p. 205, Mar. 2017.
[47] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 33–42.
[48] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 405–420.
[49] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 418–434.
[50] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2881–2890.
[51] H. Zhao. (2019). *Semseg*. [Online]. Available: https://github.com/hszhao/semseg
[52] M. P. Shah. (2017). *Semantic Segmentation Architectures Implemented in Pytorch*. [Online]. Available: https://github.com/meetshah1995/pytorch-semseg
[53] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1134–1141, Jul. 2005.
[54] K.-N. Lianos, J. L. Schonberger, M. Pollefeys, and T. Sattler, "VSO: Visual semantic odometry," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 234–250.
[55] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4340–4349.
[56] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "StaticFusion: Background reconstruction for dense RGB-D SLAM in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–9.
[57] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "ESPNetv2: A light-weight, power efficient, and general purpose convolutional neural network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9190–9200.
[58] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

Gaurav Singh (Student Member, IEEE) received the Ph.D. degree from the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He is currently working as a Scientist with the Agency for Science, Technology and Research (A*STAR), Singapore. His current research interests include visual odometry, SLAM, semantic 3-D mapping for navigation and perception applications in urban traffic scenes and indoors.

Meiqing Wu (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Engineering, Nanyang Technological University, Singapore, in 2016. She is currently working as a Research Fellow with the School of Computer Engineering, Nanyang Technological University. Her current research interests include stereo vision, motion analysis, object detection, and tracking for urban traffic scene understanding.

Minh V. Do received the Bachelor of Engineering degree in electronics and communication from the University of Da Nang, University of Science and Technology, in 2016. He is currently working as a Research Engineer with the School of Computer Science and Engineering, Nanyang Technological University. His current research interests include computer vision, machine learning, and embedded systems for urban traffic scene understanding.

Siew-Kei Lam (Senior Member, IEEE) received the B.A.Sc., M.Eng., and Ph.D. degrees from Nanyang Technological University (NTU), Singapore. He is currently an Associate Professor with the School of Computer Engineering (SCE), NTU. His research investigates custom computing techniques to meet the challenging demands for performance, energy-efficiency, cost, reliability, and security in edge intelligence. His research group develops novel design methodologies, domain-specific architectures, and architecture-aware algorithmic optimizations that will enable complex applications to run on edge devices to provide timely operations. He has published over 150 international refereed journals and conferences in these areas.