ClusterSLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation

Jiahui Huang[1] Sheng Yang[1,2] Zishuo Zhao[1] Yu-Kun Lai[3] Shi-Min Hu[1]*
[1]BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing
[2]Alibaba A.I. Labs, China [3]Cardiff University, UK
huang-jh18@mails.tsinghua.edu.cn, shengyang93fs@gmail.com, wingedkuriboh@126.com
LaiY4@cardiff.ac.uk, shimin@tsinghua.edu.cn

Abstract

We present a practical backend for stereo visual SLAM which can simultaneously discover individual rigid bodies and compute their motions in dynamic environments. While recent factor graph based state optimization algorithms have shown their ability to robustly solve SLAM problems by treating dynamic objects as outliers, the dynamic motions are rarely considered. In this paper, we exploit the consensus of 3D motions among the landmarks extracted from the same rigid body for clustering and estimating static and dynamic objects in a unified manner. Specifically, our algorithm builds a noise-aware motion affinity matrix upon landmarks, and uses agglomerative clustering for distinguishing those rigid bodies. Accompanied by a decoupled factor graph optimization for revising their shape and trajectory, we obtain an iterative scheme to update both cluster assignments and motion estimation reciprocally. Evaluations on both synthetic scenes and KITTI demonstrate the capability of our approach, and further experiments considering online efficiency also show the effectiveness of our method for simultaneous tracking of ego-motion and multiple objects.

1. Introduction

Perceiving and modeling surrounding environments are the foundation of navigating modern Autonomous Things (AuT), which is achieved via Simultaneous Localization and Mapping (SLAM) formulated with onboard sensors. With the booming demand of service robots and self-driving cars, SLAM technology is now facing more challenging scenarios, *e.g*., low-cost sensors which introduce considerable noise when running in complicated *dynamic* scenes.

Recent advanced visual SLAM approaches applicable for dynamic scenes can be divided into two categories considering their treatment of dynamic components: exclusion [1, 7, 24, 5] or segmentation [36, 37, 4, 44]. While the first category chooses to exclude these components to ensure robust camera ego-motion tracking, the latter category inclines to further segment these components into multiple instances (*i.e*. rigid bodies). Although it is tolerable to discard minor movements in an almost static environment, most scenarios including autonomous driving and multi-robot collaborating [34] require explicit motion information of the surroundings to help with decision making and scene understanding. In these cases, segmentation approaches are preferred over exclusion solutions.

Existing segmentation based dynamic SLAM systems detect and model dynamics through either semantics from deep learning [37, 4] or motion consistency [22, 36]. Deep neural networks have shown their effectiveness for object detection and semantic segmentation [19, 8] in the past few years. But the problems applying them to SLAM systems are two-fold: First, they can only detect *movable* a-priori dynamic categories (*e.g*. cars or people) but cannot recognize arbitrary *moving* instances. Second, the performance of such models heavily depends on the amount of available computing resources, which brings deployment issues in restricted platforms (*e.g*. embedded computing devices).

From another perspective, methods exploiting motion consistency for segmentation [27, 22, 36, 44] achieve acceptable performance without such problems. These solutions aim to find inconsistencies in landmark observations between adjacent frames. Current methods discover these inconsistencies by identifying outliers apparently violating predefined motion models and causing a high error residual. However, these works have not sufficiently utilized the information calculated during the SLAM process, especially tracked long-term 3D motions, which are effective for obtaining better segmentation considering motion consistency.

In this paper, we take a different approach to discovering motion inconsistencies with a key observation that motions

*corresponding author.

of the dynamic components in a scene are the essence of landmark drifting, and thus propose to cluster their motions according to the rigidity throughout the time for deducing underlying rigid bodies simultaneously with the maximum-a-posteriori (MAP) estimation in the *backend*. Compared to the frontend, SLAM backend provides the convenience of globally discovering and processing long-term scene characteristics, facilitating the fusion of history information and hence having the potential of providing more accurate motion segmentation results.

In summary, we revisit the potential of a SLAM backend, and propose an approach which can distinguish individual rigid bodies through their locally consistent and globally inconsistent 3D motions. The proposed backend for stereo cameras, namely ClusterSLAM (Alg. 1), iteratively consists of two sub-modules for simultaneously handling cluster assignments and motion property estimation. The main advantages of our proposed algorithm are:

(1) In contrast to recent SLAM backends, our algorithm *clusters* rather than excludes dynamic landmarks in the scene, and further estimates their motions.

(2) Measurement uncertainties of keypoints are taken into account in both clustering and estimation to improve the accuracy of cluster assignments and motion property estimation.

(3) We use chunks of input frames for consensus clustering and a decoupled factor graph optimization procedure to maintain the overall system efficiency.

2. Related Work

Visual SLAM in dynamic environment. As introduced in the previous section, exclusion and segmentation approaches are two main techniques for visual SLAM. Many exclusion solutions [2, 24] utilize externally computed information such as optical flow to prune outlier observations in order to achieve more accurate ego-motion estimation, while others [1, 7, 30] instead choose to add robust M-estimator into the MAP optimization framework to automatically down-weight noisy observations. Contrarily, segmentation methods like [26, 23, 12] use tracked sparse features to perform motion consistency analysis and motion segmentation; dense approaches taking RGBD input [36, 37, 4, 44] combine the registration residual of dense model alignment and the geometric features for enhanced segmentation and tracking. More techniques for dynamic SLAM are summarized in [38].

Multibody motion segmentation. Previous methods for motion segmentation are mainly based on subspace factorization techniques [9, 28], statistical modeling and sampling [15, 3, 45], epipolar/trilinear constraints [42, 41], object/scene flow [27, 2, 20, 43], energy minimization [47, 23, 21], and deep learning based instance-level detection [19, 12, 4, 14] (*i.e.* tracking-by-detection). Our strategy for segmenting multiple instances is different from previous approaches: Instead, we found that after the noise-aware refinement of the 3D trajectories of landmarks, the consistency of motion can be extracted and grouped in an *unsupervised* way to present landmark-wise associations, thus deducing their underlying rigid bodies. Furthermore, such detection may reciprocally contribute to a finer estimation of the landmark trajectories. This strategy is not sufficiently exploited in recent segmentation modules.

Clustering approaches. We refer readers to a recent review [46] on clustering approaches, which divides cluster algorithms into several categories. Since it is difficult to find an effective way of representing a *single* dynamic landmark by a feature vector, most approaches except those hierarchy based are not directly applicable to the motion clustering problem. Based on the property of relative stationarity between landmarks, our clustering approach calculates *pairwise* motion inconsistency to form a motion distance matrix, and utilizes a bottom-up hierarchy-based algorithm [17, 39] to achieve the clustering in $O(n^2 \log n)$ time. We will also show in Sec. 4.4-B that the chosen clustering method is superior to alternative methods.

3. ClusterSLAM

As a SLAM backend illustrated in Alg. 1, the goal of our approach is to obtain the position and the cluster assignment of these landmarks, as well as the motion of each cluster. We use two major modules (clustering and SLAM) in an iterative scheme to solve for and refine these variables simultaneously.

In the clustering module (Sec. 3.1), we establish a motion distance matrix (Sec. 3.1.1) to describe the inconsistency of motions of these landmarks pairwisely, and choose a hierarchical agglomerative clustering approach (Sec. 3.1.2) to merge them into clusters. Considering the computational complexity of such a matrix in long sequences, we partition input frames into short-term chunks and use a consensus clustering (Sec. 3.1.3) to conclude the long-term assignment of a landmark.

In the SLAM module (Sec. 3.2), we aim at solving the position of these landmarks simultaneously with the movement of these clusters. We first use a noise-aware point cloud registration and integration approach to initiate its shape (Sec. 3.2.1) for optimization, and then refine these positions and motions through a decoupled factor graph optimization method (Sec. 3.2.2), so that the previous motion distance matrix can be updated to continue the iteration.

As the basis of our algorithm, the stereo keypoint corresponding to the i-th landmark at frame t ($i, t \in \mathbb{N}^*$) is denoted as $\mathbf{x}_t^i = (u_L, v_L, u_R)$ where (u_L, v_L) are the coordinates on the left image and u_R is the horizontal coordinate in the right image. $\mathbf{X}_t^{\times,i} \in \mathbb{R}^3$ and $\Sigma_t^{\times,i} \in \mathbb{R}^{3\times3}$ respectively represent the local 3D coordinates and uncertainty of

the i-th landmark within the coordinate system $\times$ at frame t. The back-projection function $f : \mathbf{x}_t^i \to \mathbf{X}_t^{\mathbf{c},i}$ w.r.t. the stereo camera model projects the observation into the camera local coordinate system $\mathbf{c}$. In order to consider the pixel errors of these keypoint extraction methods [30, 11], we introduce the stereo noise model [18], where the extraction error of $\mathbf{x}_t^i$ reflected on $\mathbf{X}_t^{\mathbf{c},i}$ can be calculated as $\Sigma_t^{\mathbf{c},i} = \mathbf{J}_f \Sigma_t^i \mathbf{J}_f^\top$, where $\mathbf{J}_f$ is the Jacobian matrix of the back-projection function f, and Σ_t^i is the covariance of $\mathbf{x}_t^i$ assigned w.r.t. the keypoint extraction error in its image coordinates. For transformations and poses, we define $\mathbf{P}_t^{\mathbf{q}} \in \mathrm{SE}(3)$ for the pose of cluster $\mathbf{q}$ ($\mathbf{q} \in \mathbb{N}^*$) at frame t and $\mathbf{P}_t^{\mathbf{c}}$ for the pose of the stereo camera. We assign all static landmarks to a single static cluster with $\mathbf{q} = \mathbf{0}$, hence $\forall t, \mathbf{P}_t^{\mathbf{0}} \equiv \mathbf{I}$. For simplicity of subsequent equations, we denote relative transformation as $\mathbf{T}_t^{\mathbf{ab}} = (\mathbf{P}_t^{\mathbf{a}})' \cdot \mathbf{P}_t^{\mathbf{b}}$ ($\mathbf{P}'$ being the inverse of $\mathbf{P}$) for coordinate transformations and $\mathbf{R} \in \mathrm{SO}(3)$ the rotation part of $\mathbf{T}$. For details of how the frontend generates correspondences between landmarks (*i.e*. tracklets of feature points, the input of Alg. 1) on the input frames, we refer readers to Sec. 4.1 for our implementation details w.r.t. the evaluation datasets.

3.1. Clustering Landmarks

3.1.1 Motion Distance Matrix

Distance calculation. Our clustering approach for these extracted landmarks is based on the fact that any pair of landmarks located on the same rigid body, even with noisy measurements, should durably stay constant. Hence, we examine the property between landmarks by building a motion inconsistency matrix $\mathbf{D}$, with each element $d^{ij} \in \mathbf{D}$ depicting the inconsistency between the motions of two landmarks i and j, calculated with the following equation:

$$d^{ij} = \frac{1}{2} \operatorname*{avg}_t \left(\left\| l_t^{ij} - l_*^{ij} \right\|^2_{\sigma_t^{ij}} + \log \sigma_t^{ij} \right) + \alpha \max_t y_t^{ij},$$
$$l_*^{ij} \triangleq \sum_t \left(\frac{1}{\sigma_t^{ij}} \cdot l_t^{ij}\right) \Big/ \sum_t \frac{1}{\sigma_t^{ij}}, \quad y_t^{ij} = \left\| \mathbf{x}_t^i - \mathbf{x}_t^j \right\|^2_{\Sigma_t^{ij}} \tag{1}$$

where $\|\mathbf{x}\|^2_\Sigma \triangleq \mathbf{x}^\top \mathbf{\Sigma}^{-1} \mathbf{x}$ is the squared Mahalanobis distance with a covariance matrix $\mathbf{\Sigma}$. We use those frames t where both landmarks i and j are observed to calculate their distance. The first term is the 3D geometric distance term, which depicts the consistency of pairwise 3D spatial distance $l_t^{ij} \in \mathbb{R}$ from the frame t w.r.t. their maximum-likelihood l_*^{ij}. This term is obtained through the form of negative log-likelihood, and we kindly refer readers to our supplementary material for the derivation.

Generally, the vector form $\mathbf{l}_t^{\times,ij} = \mathbf{X}_t^{\times,i} - \mathbf{X}_t^{\times,j} \in \mathbb{R}^3$ instead of the scalar form l_t^{ij} would provide more accurate uncertainty estimation as $\Sigma_t^{\times,i} + \Sigma_t^{\times,j}$ for depicting motion consistency ($\times$ stands for any valid coordinate system), but its scalar form $l_t^{ij} = \|\mathbf{l}_t^{\times,ij}\|$ has the property of being invariant to local coordinate systems. Hence, we choose to use the scalar form l_t^{ij} and approximate its distribution using a 1-dimensional Gaussian as $l_t^{ij} \sim \mathcal{N}(l_*^{ij}, \sigma_t^{ij})$, with the variance σ_t^{ij} approximated as the error propagation from $\mathbf{l}_t^{\times,ij}$ to l_t^{ij}:

$$\sigma_t^{ij} \approx \frac{\|\mathbf{l}_t^{\times,ij}\|^2_{\Sigma_t^{\times,i}} + \|\mathbf{l}_t^{\times,ij}\|^2_{\Sigma_t^{\times,j}}}{\|\mathbf{l}_t^{\times,ij}\|^2}, \tag{2}$$

where each Mahalanobis term can be computed under any unified coordinate system $\times$.

The second term of Equ. 1 is a vision based prior term, based on the observation that we incline to group pixels which are close in the image space into one cluster and $\Sigma_t^{ij} = \Sigma_t^i + \Sigma_t^j$. Since the 2D distance of two landmarks in the image space is also dependent on the camera pose, we choose to pick the maximum rather than the average for calculating such a prior. The constant logarithm of covariance from this term is ignored since all extracted landmarks are treated with equal uncertainty in the image space. $\alpha = 4 \times 10^{-4}$ is a balance factor to control the importance of the prior. Combining both terms in such a noise-aware form enables us to take the uncertainty of measurements into account when clustering. An ablation study for ignoring these uncertainties during clustering is presented in Sec. 4.4-A.

Element rejection. If co-occurrences of a pair of landmarks are too rare (fewer than 4 times in our implementation), the maximum-likelihood estimation is no longer considered accurate due to the insufficiency of measurements. For such cases, we assign their corresponding d^{ij} as invalid. Hence, $\mathbf{D}$ may become a sparse matrix, but this

Algorithm 1 ClusterSLAM

Input: The observation of landmarks $\bigcup_{i,t} \mathbf{x}_t^i$ on different frames t.

Output: The cluster assignments $\theta : i \to \mathbf{q}$ ($\mathbf{q} = \mathbf{0}$ for the static cluster), the MAP relative position of 3D landmarks w.r.t. their cluster $\bigcup_i \hat{\mathbf{X}}^{\mathbf{q},i}$, the ego-motion of stereo camera $\bigcup_t \mathbf{P}_t^{\mathbf{c}}$, and the trajectory of each cluster $\bigcup_t \mathbf{P}_t^{\mathbf{q}}$.

$k \leftarrow 1$;
repeat
  /* Clustering module (Sec. 3.1). */
  for all partitioned chunks m **do**
    Build motion distance matrix $\mathbf{D}_m$ (Sec. 3.1.1);
    Cluster on $\mathbf{D}_m$ and get $\theta_m(i)$ (Sec. 3.1.2);
  Conclude $\theta(i)$ from $\bigcup_m \theta_m(i)$ (Sec. 3.1.3);
  /* SLAM module (Sec. 3.2). */
  for all clusters $\mathbf{q}$ **do**
    if $\mathbf{q} = \mathbf{0}$ **then** $\mathbf{b} \leftarrow \mathbf{c}$ **else** $\mathbf{b} \leftarrow \mathbf{q}$;
    Initialize $\bigcup_i \hat{\mathbf{X}}^{\mathbf{q},i}$ and $\bigcup_t \mathbf{P}_t^{\mathbf{b}}$ (Sec. 3.2.1);
    Optimize for $\bigcup_i \hat{\mathbf{X}}^{\mathbf{q},i}$ and $\bigcup_t \mathbf{P}_t^{\mathbf{b}}$ (Sec. 3.2.2);
  $k \leftarrow k + 1$;
until the clustering converges or k exceeds limit.

does not affect the performance of our hierarchical clustering (Sec. 3.1.2).

Iterative Scheme. In Equ. 1, l and σ are refined during multiple iterations, since the SLAM module may update the 3D position $\mathbf{X}_t^{\times,i}$ and $\mathbf{X}_t^{\times,j}$ of each landmark. We begin our iteration with these variables computed under the camera local coordinates $\mathbf{c}$, but instead transform them into a cluster-specific coordinates $\mathbf{q}$ in subsequent iterations once the shape of such a cluster is initialized.

3.1.2 Hierarchical Agglomerative Clustering

We use hierarchical agglomerative clustering (HAC) [39] which enables us to perform clustering on such a sparse distance matrix $\mathbf{D}$. At the beginning of clustering, we take each landmark i as a cluster, and then iteratively merge these clusters pairwisely until the distance between any pair of clusters (defined as the maximum distance between their landmarks) is larger than a given parameter ϵ (set to 60.0 in our implementation). This complete-linkage criterion [10] is chosen since the motion consistency between landmarks is not transitive, *i.e.*, the consistencies between landmarks i, j and j, z do not ensure the consistency between landmarks i and z. By implementing a heap structure over all the elements in $\mathbf{D}$ and keep track of their changes, the time complexity of HAC is $O(n^2 \log n)$. Several alternative choices for clustering are further compared in Sec. 4.4-B which shows the advantage of using HAC.

3.1.3 Consensus Clustering from Multiple Chunks

The size of $\mathbf{D}$ grows quadratically with the number of landmarks, which is positively related to the number of input frames. To speed up the clustering algorithm, we divide the input sequence into multiple chunks (100 frames each with 25 overlapped frames for cluster association) and perform the HAC clustering (Sec. 3.1.2) separately. The influences of the chunk size are further tested in Sec. 4.4-C.

Then, we perform consensus clustering based on all assignments computed from each individual chunk, by constructing a sparse vector for each landmark i as $y^i = \{\theta_m(i)\}_m$ to depict its per-chunk assignments, and perform the Iterative Voting Consensus algorithm [33]. We refer readers to the supplementary material for details.

3.2. SLAM for Clusters and Camera Ego-motion

3.2.1 Noise-aware Cluster Shape Initialization

The initialization process of a new cluster $\mathbf{q}$ aims at acquiring the estimation for the position of its landmarks $\hat{\mathbf{X}}_t^{\mathbf{q},i}$ (Note the difference between $\hat{\mathbf{X}}_t$ and the back-projected single-frame position $\mathbf{X}$, where the former represents estimated state concluding all historical frames up to the frame t). Similar to reconstruction pipelines [32], this process contains two operations, namely registration and integration, and we consider the uncertainty of frame observations in both of them. In our implementation, we maintain a Gaussian mixture $\mathcal{G}_t^{\mathbf{q},i}$ for each landmark i and regard $\hat{\mathbf{X}}_t^{\mathbf{q},i}$ as the mean for all components in the mixture.

Frame-to-Model Registration. When the first frame of a cluster $\mathbf{q}$ is encountered at frame t, we initialize the local coordinates of this cluster by assigning $\mathbf{T}_t^{\mathbf{qc}} = \mathbf{I}$ for integration. For subsequent frames, a frame-to-model registration is executed to obtain the transformation $\mathbf{T}_t^{\mathbf{qc}}$ which converts points from the current local coordinates of the frame $\mathbf{P}_t^{\mathbf{c}}$ to this constructed coordinate system, by optimizing the following equation:

$$\mathbf{T}_t^{\mathbf{qc}} = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i \min_g \left(\frac{1}{2} \left\| \mathbf{T}\mathbf{X}_t^{\mathbf{c},i} - \hat{\mathbf{X}}_{t-1}^{\mathbf{q},i} \right\|_{\Sigma_g^i}^2 - \mathbf{C}_g^i \right),$$
$$\mathbf{C}_g^i \triangleq \frac{1}{2} \log |\Sigma_g^i| + \log |\Sigma_g|, \tag{3}$$

where g traverses each component of the Gaussian mixture, and Σ_g is the g-th component of $\mathcal{G}_{t-1}^{\mathbf{q},i}$. $\Sigma_g^i = \mathbf{R}\Sigma_t^{\mathbf{c},i}\mathbf{R}^\top + \Sigma_g$. $\mathbf{C}_g^i$ is the constant factor introduced by the maximum-likelihood estimation [6]. This formulation can be viewed as a weighted ICP algorithm, with the weight being the uncertainty of both the frame and the model.

Frame Integration and Shape Refinement. After the transformation matrix of the latest frame t is robustly estimated, we update the Gaussian mixture $\mathcal{G}_{t-1}^{\mathbf{q},i}$ by inserting a new component with covariance $\Sigma_{g'} = \mathbf{R}_t^{\mathbf{qc}}\Sigma_t^{\mathbf{c},i}(\mathbf{R}_t^{\mathbf{qc}})^\top$ weighted by $1/|\Sigma_{g'}|$, and remove the component with the least weight from the mixture if the size of $\mathcal{G}_{t-1}^{\mathbf{q},i}$ exceeds 3. This strategy ensures the registration to be considered through one of the most reliable measurements. Then, we integrate the obervation into the landmark position $\hat{\mathbf{X}}_t^{\mathbf{q},i}$ as:

$$\hat{\mathbf{X}}_t^{\mathbf{q},i} = \underset{\mathbf{X}}{\operatorname{argmin}} \frac{\left\| \mathbf{X}_t^{\mathbf{c},i} - \mathbf{T}_t^{\mathbf{cq}}\mathbf{X} \right\|_{\Sigma_t^{\mathbf{c},i}}^2}{|\Sigma_{g'}|} + \sum_g \frac{\left\| \hat{\mathbf{X}}_{t-1}^{\mathbf{q},i} - \mathbf{X} \right\|_{\Sigma_g}^2}{|\Sigma_g|}, \tag{4}$$

where the first term is used for adding current observations and the rest terms for previous observations.

Equs. 3 and 4 can be solved efficiently with the Gauss-Newton method and QR decomposition, respectively. In practice, we choose to fix Σ_g^i in Equ. 3 during each iteration to make the registration easier to solve. We compare such a probabilistic form of registration and integration to the traditional point-to-point registration in Sec. 4.4-D. The final initialized cluster poses and landmark positions are assigned as $\mathbf{T}^{\mathbf{qc}} = \mathbf{T}_T^{\mathbf{qc}}$ and $\hat{\mathbf{X}}^{\mathbf{s},i} = \hat{\mathbf{X}}_T^{\mathbf{s},i}$, respectively, where T is the index of the last frame.

3.2.2 Decoupled Factor Graph Optimization

Traditional factor graph optimization only treats static landmarks $\hat{\mathbf{X}}^{\mathbf{0},i}$ and camera ego-motion $\bigcup_t \mathbf{P}_t^{\mathbf{c}}$ as objectives. Assuming that the dynamic scene comprises multiple rigid bodies, we additionally treat the motion of all clusters $\bigcup_{\mathbf{q},t} \mathbf{P}_t^{\mathbf{q}}$ and their landmarks $\bigcup_{\mathbf{q},i} \hat{\mathbf{X}}^{\mathbf{q},i}$ as objectives. Then the BA energy function for each individual cluster can be written as:

$$\mathbf{E}_{\mathbf{q}} \triangleq \sum_{i,t} \rho(\left\| \mathbf{x}_t^i - f'\left((\mathbf{P}_t^{\mathbf{c}})' \mathbf{P}_t^{\mathbf{q}} \hat{\mathbf{X}}^{\mathbf{q},i}\right)\right\|_{\Sigma_t^i}^2), \qquad (5)$$

where $\rho(\cdot)$ is an optional robust kernel [1] and f' is the inverse of f, *i.e.*, the stereo projection model.

We use the following three candidate optimization strategies to test their differences: First, *fully-coupled* optimization tries to solve $\mathbf{E} = \sum_{\mathbf{q}} \mathbf{E}_{\mathbf{q}}$ w.r.t. variables of all clusters jointly. Second, *decoupled* optimization is performed following three steps: (1) solve $\mathbf{E}_{\mathbf{q}}$ ($\mathbf{q} \neq \mathbf{0}$) for $\bigcup_{\mathbf{q}\neq\mathbf{0}}\{\bigcup_i \hat{\mathbf{X}}^{\mathbf{q},i}, \bigcup_t \mathbf{T}_t^{\mathbf{cq}}\}$ by regarding $(\mathbf{P}_t^{\mathbf{c}})'\mathbf{P}_t^{\mathbf{q}}$ as a single variable; (2) solve $\mathbf{E}_{\mathbf{0}}$ to obtain the camera ego-motion $\bigcup_t \mathbf{P}_t^{\mathbf{c}}$ and static landmark positions $\bigcup_i \hat{\mathbf{X}}^{\mathbf{0},i}$; (3) composite the ego-motion and these transformations to generate the motion of clusters as $\bigcup_{\mathbf{q},t} \mathbf{P}_t^{\mathbf{q}}$. Third, *semi-decoupled* optimization replaces the above step (2) by solving the whole objective function $\mathbf{E} = \sum_{\mathbf{q}} \mathbf{E}_{\mathbf{q}}$ rather than $\mathbf{E}_{\mathbf{0}}$.

Both Hessian matrix based theoretical analysis (in supplementary material) and our experiments (Sec. 4.4-E) demonstrate the suitability and time efficiency of the *decoupled* strategy and we adopt this method in our final algorithm.

4. Evaluation

4.1. Datasets and Parameter Setup

Our experiments are performed on two synthetic datasets (SUNCG [40], CARLA [13]) and one real-world dataset (KITTI [16]).

Synthetic datasets. The SUNCG dataset [40] provides 3D models for constructing indoor scenes, and we build 3 scenes with 2 sequences on each, respectively. In these sequences, 2-5 instances as well as the stereo camera are dynamic, with their motions generated manually in 6 Degrees of Freedom. The CARLA simulator [13] is used for generating outdoor car-driving scenes. We use its engine to simulate streets with multiple driving vehicles and generate 4 sequences for experiments.

Ground-truth landmarks are extracted through random sampling among the vertices of these models, and we add a maximum of 1.5 pixels noise to both u, v coordinates for simulating noisy landmark observations on these stereo frames. The synthetic stereo camera has a resolution of 1280×720 and a horizontal Field-of-View of $90°$, with its baseline set to 10cm for indoor SUNCG and 50cm for outdoor CARLA, respectively.

Real-world dataset. We use 3 KITTI raw sequences (0013, 0015, and 0017) and Superpoint [11] for feature point extraction. A similar step as in [23] is performed in the frontend to find associations and generate landmark tracklets. Please refer to our supplementary materials for detailed information about these synthetic and real-world scans.

Parameters and Hardware Setup. Since both the baseline and scales are different in indoor and outdoor scenes, we use two sets of parameters for these two scenarios, respectively. For indoor scenes, these parameters remain as presented in Sec. 3. For outdoor scenes, we adjust ϵ to 90.0 regarding the change in stereo baseline and bigger size of vehicles, and raise the size of each chunk to 200 to maintain the density of $\mathbf{D}$ (*i.e.*, sufficient pairwise distances for intra-chunk clustering). We utilize the g2o framework [25] for implementing some of those proposed least-squares optimizations. All of the experiments for the backend are executed on an Intel Core i7-8700K, 32GB RAM desktop computer with a GTX 1080 GPU for timing.

4.2. Full Backend Performance

Baselines. For comparing the full backend performance, three candidate baseline methods are built: (1) Full Bundle Adjustment, where BA is performed on all visible landmarks assuming they are static. (2) Progressive DCS (dynamic covariance scaling), which takes a step beyond the Full BA through the robust dynamic covariance scaling kernel [1] for determining those dynamic objects one by one during each iteration. Specifically, the landmarks with average error larger than $\chi_N^2 + 0.3(\chi_M^2 - \chi_N^2)$ will be marked as dynamic and introduced to the next iteration (χ_N^2 represents its smallest reprojection error and χ_M^2 the largest). The algorithm will continuously segment one consensus object after each iteration, until the number of outliers is smaller than 10. (3) Semantic Segmentation, where a Mask R-CNN model trained on the MS-COCO dataset [19] is employed to get the instance-level segmentation of each input frame. These predicted labels are used to vote for the final labeling of each landmark through recursive Bayesian. In conclusion, these three categories represent the classical, robust strategy, and sequential tracking-by-detection methods respectively as discussed in Sec. 2.

Evaluation Criteria. We then use the following metrics to quantitatively compare the performance: (1) $\log \chi^2$, as the logarithm of reprojection error in BA, reflecting the accordance of these optimization results w.r.t. the input constraints. (2) RMSE, as the pointwise Root Mean Square Error for the position of each tracked landmark w.r.t. its ground truth position, measuring the quality of mapping. (3-5) ATE and R./T.RPE, as the RMSE of Absolute Trajectory Error and the Rotational/Translational Relative Pose

Table 1. Quantitative comparison on synthetic sequences.

	Indoor Sequences								Outdoor Sequences							
	log χ^2	RMSE(m)	ATE	R.RPE	T.RPE	Acc.(%)	β_{VI}	Time(s)	log χ^2	RMSE(m)	ATE	R.RPE	T.RPE	Acc.(%)	β_{VI}	Time(s)
Full BA	9.61	2.10	0.53/-	0.48/-	0.87/-	52.73	1.19	5.45	10.92	14.39	12.94/-	0.73/-	42.55/-	81.39	0.84	6.00
P. DCS	12.80	1.53	0.63/1.61	0.49/1.82	0.99/2.60	56.05	1.18	3.83	13.85	11.22	9.36/-	0.73/-	37.61/-	80.22	0.82	**1.86**
Sem. SEG	9.31	0.84	0.31/0.51	0.12/0.87	0.49/0.95	69.60	1.19	**3.82***	8.55	2.69	1.65/3.09	0.18/0.32	2.34/**8.11**	**96.70**	**0.24**	5.28*
Ours w/o U	7.88	1.21	0.35/0.34	0.15/0.37	0.53/0.57	65.60	0.96	9.60	8.56	2.48	1.86/5.13	**0.02**/0.40	3.18/12.32	86.51	0.64	6.96
Ours w/o I	8.65	1.05	0.15/0.31	0.05/0.57	0.21/0.55	76.16	1.06	9.47	9.70	9.56	2.12/3.44	0.47/0.20	4.47/9.94	81.83	0.57	5.84
Ours	**7.15**	**0.44**	**0.01/0.12**	**0.01/0.29**	**0.02/0.22**	**91.54**	**0.40**	11.15	**6.52**	**0.63**	**0.53/3.37**	**0.02/0.18**	**1.10**/8.65	94.15	0.27	6.14

* Mask R-CNN [19] prediction time is excluded.

Error w.r.t. the ground truth motions, showing the quality of the motion estimation (with camera ego-motion and object motion separately recorded). ATE and R./T.RPE are measured in meters, radians and meters, respectively. (6-7) Clustering accuracy, taken as the best among all permutations of ground truth and prediction label correspondences and β_{VI}, as the variation of information distance [29]. These two metrics reflect the performance of segmentation. (8) Running Time of the whole backend when all frames of the sequence are considered as a batch.

Results and Analysis. Quantitative comparisons on all synthetic sequences are averaged and listed in Tab. 1, with variations of our methods presented together but further discussed in Sec. 4.4-A. Despite our proposed method requiring more running time for processing, it outperforms other methods on the performance of tracking and mapping. Although Mask R-CNN [19] for segmenting individual objects is better than ours on outdoor sequences, it requires a pre-trained model and extra time for prediction (it costs on average 40.5 and 67.0 additional seconds in the frontend for indoor and outdoor sequences, respectively), and does not work for object categories not present in the training data. Besides, the Progressive DCS scores best in running time due to a better Gauss-Newton quadratic approximation of its cost function [1].

In account of their quality, we found that the Progressive DCS only tends to reject dynamic outliers in its first pass. In its second pass few outliers can be detected even though the landmarks left may contain multiple rigid bodies. This is because the dramatically reduced sparse observation constraints cannot provide enough information to determine the accurate kinematics (low signal-to-noise ratio). Despite higher clustering accuracy of the Semantic Segmentation than ours, its estimation of motions and landmark positions is affected by their imprecise masks, since an inaccurate edge of the mask appearing on the border of an object may erroneously categorize its nearby landmarks, and eventually influence the backend performance. Furthermore, we show visual comparisons of different clustering results on sample frames (Fig. 1) and the motion trajectories (Fig. 2), and our method produces more accurate results.

Performance on KITTI. Our algorithm is also tested on KITTI. The estimated trajectories are further smoothed by applying Gaussian interpolation [35] to reduce jitter. We compare and list the results in Tab. 2 with a visualized sample shown in Fig. 3. In addition to a better camera ego-motion, our algorithm can further detect and track multiple moving cars.

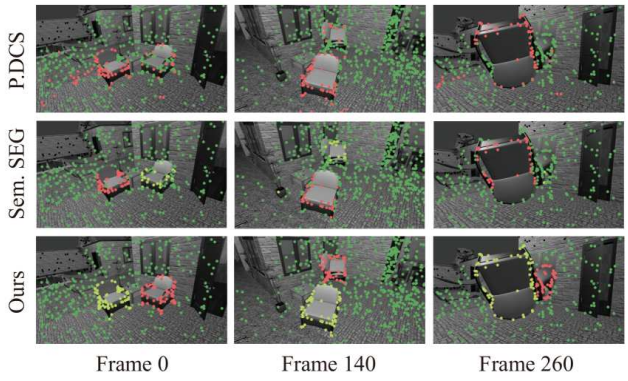

Figure 1. Visual comparisons on a SUNCG sequence. Landmarks are colorized by their index of cluster.

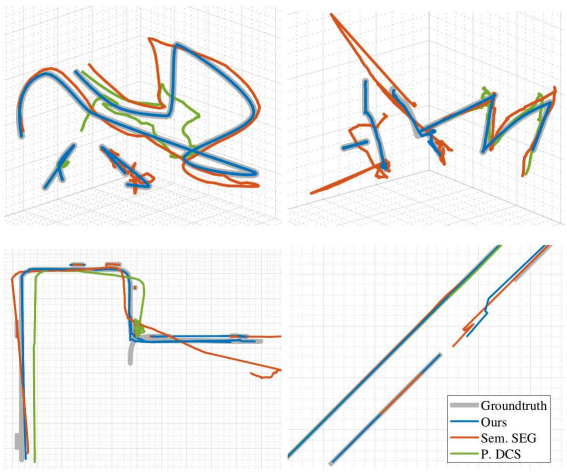

Figure 2. Recovered trajectories on synthetic sequences. Resolutions of the major grids (solid lines) are 1m, 1m, 20m and 5m, resp. Disconnected trajectories stand for multiple dynamic instances.

Table 2. Ego motion comparison on KITTI sequences.

	0013			0015			0017		
	ATE	R.RPE	T.RPE	ATE	R.RPE	T.RPE	ATE	R.RPE	T.RPE
Sem. SEG	2.65	**0.06**	**4.70**	2.64	0.07	8.35	0.77	0.09	1.11
Ours	**2.12**	0.07	5.50	**1.32**	**0.03**	**3.64**	**0.27**	**0.02**	**0.40**

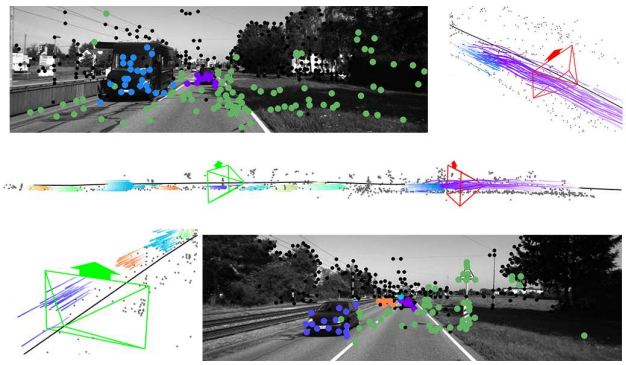

Figure 3. Our results on the KITTI dataset. Middle: Overview of the street with the camera trajectory (black) and multiple clusters colorized through index and time. Top and Bottom: Sample images and the visualized point clouds in the perspective view.

Table 3. Quantitative comparisons to existing systems.

	ATE	R.RPE	T.RPE	Acc.(%)	β_{VI}	Hz
Indoor Sequences						
ORB-SLAM2	0.03/-	**0.01**/-	0.02/-	(52.73)†	(1.19)†	**8.5**
Murali *et al.*	0.03/-	**0.01**/-	**0.01**/-	(52.73)†	(1.19)†	4.9
DynSLAM	0.54/0.19	1.10/0.73	2.60/0.40	61.12	1.21	2.0
Ours	**0.01/0.12**	**0.01/0.29**	0.02/0.22	**91.54**	**0.40**	*
Ours (RT)	0.03/**0.12**	**0.01**/0.30	0.05/**0.21**	85.27	0.60	2.2
Outdoor Sequences						
	ATE	R.RPE	T.RPE	Acc.(%)	β_{VI}	Hz
ORB-SLAM2	2.82/-	0.84/-	6.09/-	(81.39)†	(0.84)†	**9.0**
Murali *et al.*	1.19/-	0.53/-	3.45/-	(81.39)†	(0.84)†	5.0
DynSLAM	3.95/4.32	0.96/**0.09**	9.61/9.44	93.73	0.44	2.1
CarFusion	-/2.97	-/0.22	-/9.39	93.02	0.51	*
Ours	**0.53**/3.37	**0.02**/0.18	**1.10**/8.65	**94.15**	**0.27**	*
Ours (RT)	0.92/**1.53**	0.04/0.20	1.82/**3.35**	88.58	0.51	7.1

†These methods do not detect dynamics, so Acc. and β_{VI} are listed as the values when assigning all landmarks into one static cluster.
*Offline methods.

4.3. Real-time SLAM System Performance

The evaluation in Sec. 4.2 is performed by taking the entire video sequence as a single batch and running our algorithm over it. In real-time scenarios, it is intractable to run batch optimization on all acquired frames. We hereby present implementation strategies to build an online version of our system which can run at 7.1Hz for outdoor cases.

Implementation Details. Similar to [23], we restrict the size of input frames to our backend algorithm to a small neighborhood of 30 recent frames and optimize within the window periodically. Detected clusters between each different runs of the backend are associated by counting coexisting landmarks. Pairs of history and new clusters are associated if the landmark overlap ratio is over 70%, otherwise the clusters are either split or dropped. After cluster association, the involved landmarks are aligned to find the best transform between the history and new trajectories to connect them. Experiments show that this implementation takes 80.10ms/20.68ms on average for every iteration of optimization for indoor/outdoor sequences, respectively due to their different numbers of involved landmarks.

Comparisons and Analysis. We additionally compare our online version, denoted as 'Ours (RT)', to the batch version (denoted as 'Ours'), stereo ORB-SLAM2 [30], its variant proposed by Murali *et al*. [31], DynSLAM [4] and CarFusion [12] in terms of accuracy and speed. Tab. 3 comparatively shows our effectiveness of precisely acquiring both trajectory and clustering through the proposed algorithm. The criteria used are the same as Sec. 4.2.

ORB-SLAM2 [30] and Murali *et al*.'s system [31] are only designed for ego-motion tracking, which perform on par with ours in indoor cases. But for outdoor sequences where large part of the scene is dynamic as in road junctions, they fail to precisely track the ego-motion, causing large trajectory error or even tracking loss.

Both DynSLAM [4] and CarFusion [12] acquire segmentation through an external deep network instead of optimization. Besides, DynSLAM [4] does not contain a backend optimization and suffers from cumulative drift. CarFusion [12] addresses a different input, *i.e*., multiple video sequences captured alongside the road, so we feed the groundtruth ego-motion to avoid its failure on tracking and concentrate on assessing the motion of the dynamic objects. We find its performance dependent on the precision of car keypoint detection, where inaccurate detections may conflict with their intra-frame smoothing constraints and generate undesirable results.

4.4. Ablation Study

Ablation studies are performed and discussed for those modules presented in Sec. 3.

A. Formulation of Motion Distance. We evaluate the effectiveness of Equ. 1 by validating the necessity of its forms, specifically on: (1) The noise-aware formulation by switching the Mahalanobis form of the first term to Euclidean and replacing the weighted average l_*^{ij} into unweighted (denoted as **w/o U**); (2) The second vision based prior term through directly removing it (denoted as **w/o I**). Their results are listed in Tab. 1 with other parts of the algorithm unchanged. For outdoor sequences, **w/o U** requires more iterations to converge, and therefore is slower than our method. Since most metrics except running time shown in the table are worse than ours, we prove the necessity of both forms on improving the final quality of the backend.

B. Alternative Clustering Methods. We compare both the clustering accuracy and β_{VI} of our agglomerative clustering method with Spectral Clustering (SC) and Affinity Propagation (AP) methods. All these three methods do not rely on a feature vector of each element and therefore are suitable for our scenario with the dense pairwise matrix form. We eliminate all non co-visible landmarks and build dense motion distance matrix $\mathbf{D}$ for comparison. For this

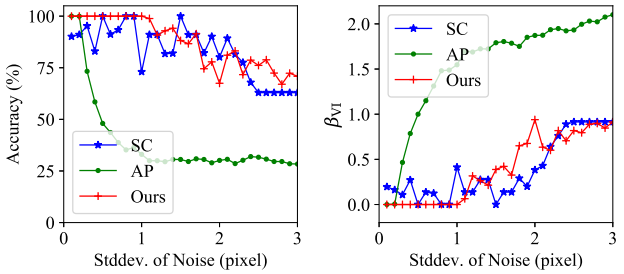

Figure 4. Accuracy and variation of information comparison of different clustering methods with respect to noise.

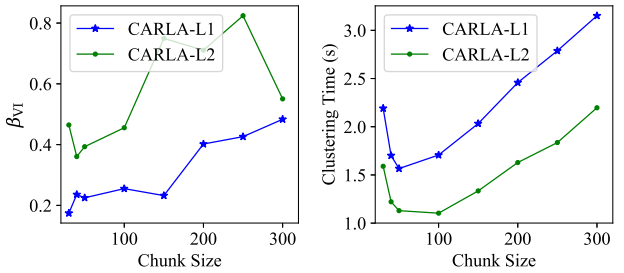

Figure 5. Chunk size vs. quality of clustering and computational time on two long CARLA sequences.

experiment, we generate landmark observations on these synthetic scans with the standard deviation of noise varied from 0.0 to 3.0 pixels to test the robustness.

Their performance w.r.t. the standard deviation of noise is shown in Fig. 4. We found that the clustering accuracy of AP drops quickly when noise becomes severe. Besides, in the case of small noise, SC is not as stable as ours (It is not fully accurate even if the input observations do not contain any noise). For the motion distance matrix $\mathbf{D}$ with size 260×260, the average running time of our approach is only 6.3ms while AP takes 13ms and SC takes 180ms on average.

C. The size of Chunks. We use the two long CARLA sequences (denoted as CARLA-L1 and CARLA-L2) to investigate how the length of chunks would affect β_{VI} and running time: We adjust the chunk size from 30 to 300 and plot the results in Fig 5. In general, smaller chunks present better clustering results but reduce the number of observations (*i.e.*, challenging the density of $\mathbf{D}$). It also increases the total running time since the *inter-chunk* merging requires more computations.

D. Noise-aware Cluster Initialization. Tab. 4 shows our results compared to a traditional point-to-point ICP approach, which neglects the uncertainty in both the registration and the integration phases. Our method works better especially on these outdoor sequences, since the uncertainty of the disparity becomes more important when the baseline is larger. It is also notably slower than the traditional ICP due to the requirement of computing all the covariances of observations and replacing the SVD by the Gauss Newton optimization. Nevertheless, the overall time to initialize all the poses is not a bottleneck in the backend.

Table 4. Comparison of the initialization methods.

	Indoor		Outdoor		
	ATE	RMSE(m)	ATE	RMSE(m)	Time(ms)
Point-to-Point ICP	0.07/**0.13**	0.22	7.83/1.37	8.54	**0.01**
Noise-aware ICP	**0.01**/0.15	**0.12**	**0.98/0.61**	**0.94**	0.12

Table 5. Comparison of different optimization schemes.

	Indoor		Outdoor	
	RMSE(m)	Time/Iter(s)	RMSE(m)	Time/Iter(s)
Fully-Coupled	**0.034**	0.83	0.73	0.57
Semi-Coupled	0.047	1.04	**0.12**	1.07
Decoupled	0.047	**0.57**	**0.12**	**0.53**

E. Decoupled optimization. The three alternative optimization strategies in Sec. 3.2.2 are tested with their RMSE and running time per iteration listed in Tab. 5. All optimizers run for 20 iterations. As a result, the *decoupled* strategy is the fastest and does not obviously show different quality in comparison to the *semi-decoupled* strategy. The *fully-coupled* strategy obtains better results indoor but the pose estimation on these dynamic clusters and the camera ego-motion may interfere with each other and present unexpected results especially on those noisy outdoor sequences.

5. Conclusion

Limitations. Despite the general applicability of our approach, there are several limitations worth noticing: (1) Our backend algorithm relies on the quality of landmark extraction and association from the frontend. Although false associations may be numerically filtered through the robust kernel, excessive errors can cause unexpected results from our backend. (2) Our algorithm may fail to detect dynamic objects with insufficient landmarks, since their recovered trajectory would be more severely affected by every single noisy landmark.

In this paper we presented ClusterSLAM, a general SLAM backend to simultaneously cluster rigid bodies and estimate their motions. In the future, our formulation of the multi-body factor graph optimization can be enhanced by external measurements to further develop its functionality, and it is also worth attempting to utilize the long-term consistency from the backend to reciprocally refine the data association in the frontend.

Acknowledgements. We thank anonymous reviewers for the valuable discussions. This work was supported by the National Key Technology R&D Program (Project Number 2017YFB1002604), the Joint NSFC-DFG Research Program (Project Number 61761136018) and the Natural Science Foundation of China (Project Number 61521002).

References

[1] Pratik Agarwal, Gian Diego Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard. Robust map optimization using dynamic covariance scaling. In *IEEE International Conference on Robotics and Automation*, pages 62–69, 2013.

[2] Pablo F. Alcantarilla, José J. Yebes, Javier Almazán, and Luis M. Bergasa. On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments. In *IEEE International Conference on Robotics and Automation*, pages 1290–1297, 2012.

[3] Haleh Azartash, Kyoung-Rok Lee, and Truong Q. Nguyen. Visual odometry for RGB-D cameras for dynamic scenes. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1280–1284, 2014.

[4] Ioan Andrei Bârsan, Peidong Liu, Marc Pollefeys, and Andreas Geiger. Robust dense mapping for large-scale dynamic environments. In *IEEE International Conference on Robotics and Automation*, pages 7510–7517, 2018.

[5] Berta Bescos, José M Fácil, Javier Civera, and José Neira. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, 2018.

[6] Yan-Pei Cao, Leif Kobbelt, and Shi-Min Hu. Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras. *ACM Transactions on Graphics*, 37(5):171:1–171:16, 2018.

[7] Luca Carlone, Andrea Censi, and Frank Dellaert. Selecting good measurements via L1 relaxation: A convex approach for robust estimation over graphs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2667–2674, 2014.

[8] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[9] João Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.

[10] Daniel Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

[11] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, pages 224–236, 2018.

[12] N. Dinesh Reddy, Minh Vo, and Srinivasa G. Narasimhan. CarFusion: Combining point tracking and part detection for dynamic 3D reconstruction of vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1906–1915, 2018.

[13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[14] Ruochen Fan, Fang-Lue Zhang, Min Zhang, and Ralph R. Martin. Robust tracking-by-detection using a selection and completion mechanism. *Computational Visual Media*, 3(3):285–294, 2017.

[15] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[16] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[17] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *ACM Sigmod Record*, volume 27, pages 73–84, 1998.

[18] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[20] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2462–2470, 2017.

[21] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *International Journal of Computer Vision*, 97(2):123–147, 2012.

[22] Mariano Jaimez, Christian Kerl, Javier Gonzalez-Jimenez, and Daniel Cremers. Fast odometry and scene flow from RGB-D cameras based on geometric clustering. In *IEEE International Conference on Robotics and Automation*, pages 3992–3999, 2017.

[23] Kevin M. Judd, Jonathan D. Gammell, and Paul Newman. Multimotion visual odometry (mvo): Simultaneous estimation of camera and third-party motions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3949–3956, 2018.

[24] Deok-Hwa Kim and Jong-Hwan Kim. Effective background model-based RGB-D dense visual odometry in a dynamic environment. *IEEE Transactions on Robotics*, 32(6):1565–1573, 2016.

[25] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.

[26] Abhijit Kundu, K. Madhava Krishna, and C.V. Jawahar. Realtime multibody visual SLAM with a smoothly moving monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2080–2087. IEEE, 2011.

[27] Philip Lenz, Julius Ziegler, Andreas Geiger, and Martin Roser. Sparse scene flow segmentation for moving object detection in urban environments. In *2011 IEEE Intelligent Vehicles Symposium*, pages 926–932, 2011.

[28] Ting Li, Vinutha Kallem, Dheeraj Singaraju, and René Vidal. Projective factorization of multiple rigid-body motions.

In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, 2007.

[29] Marina Meilă. Comparing clusterings by the variation of information. In *Learning theory and kernel machines*, pages 173–187. 2003.

[30] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[31] Varun Murali, Han-Pang Chiu, Supun Samarasekera, and Rakesh Teddy Kumar. Utilizing semantic visual landmarks for precise vehicle navigation. In *IEEE International Conference on Intelligent Transportation Systems*, pages 1–8, 2017.

[32] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[33] Nam Nguyen and Rich Caruana. Consensus clusterings. In *IEEE International Conference on Data Mining*, pages 607–612. IEEE, 2007.

[34] Liam Paull, Guoquan Huang, Mae Seto, and John J. Leonard. Communication-constrained multi-AUV cooperative SLAM. In *IEEE International Conference on Robotics and Automation*, pages 509–516, 2015.

[35] Abhijeet Ravankar, Ankit Ravankar, Yukinori Kobayashi, Yohei Hoshino, and Chao-Chung Peng. Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. *Sensors*, 18(9):3170, 2018.

[36] Martin Rünz and Lourdes Agapito. Co-Fusion: Real-time segmentation, tracking and fusion of multiple objects. In *IEEE International Conference on Robotics and Automation*, pages 4471–4478, 2017.

[37] Martin Runz, Maud Buffier, and Lourdes Agapito. MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 10–20, 2018.

[38] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. Visual SLAM and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, 51(2):37, 2018.

[39] Robert R Sokal. A statistical method for evaluating systematic relationship. *University of Kansas science bulletin*, 28:1409–1438, 1958.

[40] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1754, 2017.

[41] Rene Vidal and Richard Hartley. Three-view multibody structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):214–227, 2008.

[42] René Vidal, Yi Ma, Stefano Soatto, and Shankar Sastry. Two-view multibody structure from motion. *International Journal of Computer Vision*, 68(1):7–25, 2006.

[43] Zhi-Feng Xie, Yu-Chen Guo, Shu-Han Zhang, Wen-Jun Zhang, and Li-Zhuang Ma. Multi-exposure motion estimation based on deep convolutional networks. *Journal of Computer Science and Technology*, 33(3):487–501, 2018.

[44] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Midfusion: Octree-based object-level multi-instance dynamic SLAM. In *IEEE International Conference on Robotics and Automation*, 2019.

[45] Xun Xu, Loong Fah Cheong, and Zhuwen Li. Motion segmentation by exploiting complementary geometric models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2859–2867, 2018.

[46] Guan Yuan, Penghui Sun, Jie Zhao, Daxing Li, and Canwei Wang. A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, 47(1):123–144, 2017.

[47] Cui-Cui Zhang and Zhi-Lei Liu. Prior-free dependent motion segmentation using Helmholtz-Hodge decomposition based object-motion oriented map. *Journal of Computer Science and Technology*, 32(3):520–535, 2017.