Computational Visual Media
https://doi.org/10.1007/s41095-020-0195-3 Vol. 7, No. 1, March 2021, 87–101
Research Article

ClusterSLAM: A SLAM backend for simultaneous rigid body clustering and motion estimation

Jiahui Huang[1], Sheng Yang[2], Zishuo Zhao[1], Yu-Kun Lai[3], and Shi-Min Hu[1] (✉)

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, their dynamic motions are rarely considered. In this paper, we exploit the consensus of 3D motions for landmarks extracted from the same rigid body for clustering, and to identify static and dynamic objects in a unified manner. Specifically, our algorithm builds a noise-aware motion affinity matrix from landmarks, and uses agglomerative clustering to distinguish rigid bodies. Using decoupled factor graph optimization to revise their shapes and trajectories, 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 simultaneously tracking ego-motion and multiple objects.

Keywords dynamic SLAM; motion segmentation; scene perception

1 Introduction

Perceiving and modeling surrounding environments are the foundation of navigating modern *autonomous things* (AuT), which is achieved by *simultaneous localization and mapping* (SLAM) using onboard sensors. With booming demand for service robots and self-driving cars, SLAM now faces more challenging scenarios, e.g., low-cost sensors which introduce considerable noise when used in complicated *dynamic* scenes.

Recent advanced visual SLAM approaches applicable to dynamic scenes can be divided into two categories according to their treatment of dynamic components: exclusion [1–4] or segmentation [5–8]. While the first category chooses to exclude these components to ensure robust camera ego-motion tracking, the latter category segments these components into multiple instances (i.e., rigid bodies). Although it is acceptable to discard minor moving components in an almost static environment, most scenarios including autonomous driving and multi-robot collaboration [9] require explicit motion information about the surroundings to help with decision making and scene understanding. In such cases, segmentation approaches are preferred over exclusion solutions.

Existing segmentation-based dynamic SLAM systems detect and model dynamics through either semantics from deep learning [6, 7, 10] or motion consistency [5, 11]. Deep neural networks have shown their effectiveness for object detection and semantic segmentation [12, 13] in the past few years. However, when applying them to SLAM systems, the problems are two-fold. Firstly, they can only detect *movable* a-priori dynamic categories (e.g., cars or people) but cannot recognize arbitrary *moving* objects. Secondly, the performance of such models heavily depends on the available computing resources, leading to deployment issues on restricted platforms (e.g., embedded computing devices).

1 BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: J. Huang, huang-jh18@mails.tsinghua.edu.cn; Z. Zhao, wingedkuriboh@126.com; S.-M. Hu, shimin@tsinghua.edu.cn (✉).
2 Alibaba A.I. Labs, Hangzhou 311121, China. E-mail: shengyang93fs@gmail.com.
3 School of Computer Science and Informatics, Cardiff University, Cardiff, CF24 3AA, UK. E-mail: LaiY4@cardiff.ac.uk.
Manuscript received: 2020-04-09; accepted: 2020-09-04

However, methods which exploit motion consistency for segmentation [5, 8, 11, 14] 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 which appear to violate predefined motion models, and have a high error residual. However, such work does not sufficiently utilize the information calculated during the SLAM process, especially tracked long-term 3D motions, which can be effectively used to obtain better segmentation by considering motion consistency.

In this paper, we take a different approach to discovering motion inconsistencies. Our key observation is that motions of the dynamic components in a scene are the basis of landmark drift, and so we cluster their motions according to rigidity over time. This allows us to determine underlying rigid bodies simultaneously with maximum-a-posteriori (MAP) estimation in the *backend*.

Compared to the frontend, using the SLAM backend provides the convenience of globally discovering and processing long-term scene characteristics, facilitating the fusion of historical information and hence having the potential to provide 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 but globally inconsistent 3D motions. The proposed backend for stereo cameras, which we call ClusterSLAM (Algorithm 1), iterates two sub-modules for cluster assignment 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 rigid motions.
2. Measurement uncertainties of keypoints are taken into account in both clustering assignments and motion property estimation, to improve the accuracy of both.
3. We use chunks of input frames for consensus clustering and a decoupled factor graph optimization procedure to maintain overall system efficiency.

A conference version of this paper exists in Ref. [15]. This journal version extends it by providing additional odometric evaluations on the KITTI [16] dataset, and qualitative results not in the original paper. We also have published our synthetic dataset to facilitate research in the community.

2 Related work

2.1 Visual SLAM in dynamic environments

As noted earlier, exclusion and segmentation are the two main techniques for visual SLAM in dynamic environments. Many exclusion solutions [3, 17] utilize externally computed information such as optical flow to prune outlier observations so as to achieve more accurate ego-motion estimation, while others [1, 2, 18] instead choose to add a robust M-estimator into the MAP optimization framework to automatically down-weight noisy observations. Alternatively, segmentation methods like Refs. [19–21] use tracked sparse features to perform motion consistency analysis and motion segmentation; dense approaches taking RGBD input [5–8, 22] combine the registration residual of dense model alignment and geometric features for enhanced segmentation and tracking. Further techniques for dynamic SLAM are summarized in Ref. [23].

2.2 Multibody motion segmentation

Previous methods for motion segmentation are mainly based on subspace factorization techniques [24, 25], statistical modeling, and sampling [26–28], epipolar or trilinear constraints [29, 30], object or scene flow [14, 17, 31, 32], energy minimization [20, 33, 34], and deep learning based instance-level detection [7, 10, 12, 21, 35] (i.e., tracking-by-detection). Our strategy for segmenting multiple instances differs from previous approaches. Instead, we find that after noise-aware refinement of the 3D trajectories of landmarks, consistent motions 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 landmark trajectories. This strategy is not sufficiently exploited in recent segmentation modules.

2.3 Clustering approaches

We refer readers to a recent review [36] which categorizes clustering algorithms. Since it is difficult

to find an effective way to represent a *single* dynamic landmark by a feature vector, most non-hierarchical approaches 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 hierarchical algorithm [37, 38] to perform clustering in $O(n^2 \log n)$ time. We show in Section 4.4 that the chosen clustering method is superior to alternative methods.

3 ClusterSLAM

3.1 Preliminaries

As a SLAM backend (illustrated in Fig. 1 and formally defined in Algorithm 1), the goal of our approach is to obtain the position and cluster assignment of each landmark, 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 in turn.

In the clustering module (Section 3.2), we establish a motion distance matrix (Section 3.2.1) to describe inconsistency of motions of pairs of landmarks, and use a hierarchical agglomerative clustering approach (Section 3.2.2) to merge them into clusters. Given the computational effort of using such a matrix over long sequences, we partition input frames into short-term chunks and use consensus clustering (Section 3.2.3) to determine the long-term assignment of a landmark.

In the SLAM module (Section 3.3), we aim to find the positions of these landmarks simultaneously with the movements of these clusters. We first use a noise-aware point cloud registration and integration approach to perform robust state initialization (Section 3.3.1), and then refine the positions and motions by a decoupled factor graph optimization method (Section 3.3.2), so that the previous motion distance matrix can be updated to continue the iteration.

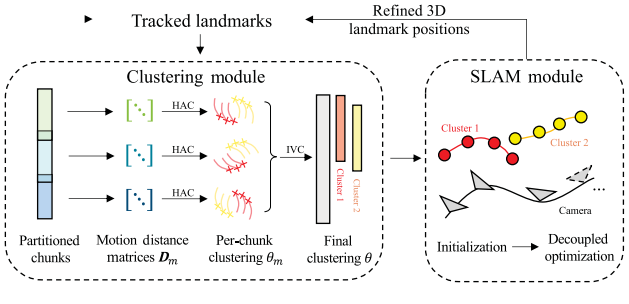


Fig. 1 ClusterSLAM pipeline (IVC means iterative voting consensus).

Algorithm 1 ClusterSLAM

Input: Observations of landmarks $\bigcup_{i,t} \boldsymbol{x}_t^i$ in frames at time t.
Output: Cluster assignments $\theta : i \to \boldsymbol{q}$ ($\boldsymbol{q} = \boldsymbol{0}$ for the static cluster), the MAP relative position of 3D landmarks w.r.t. their cluster $\bigcup_i \hat{\boldsymbol{X}}^{\boldsymbol{q},i}$, the ego-motion of stereo camera $\bigcup_t \boldsymbol{P}_t^{\boldsymbol{c}}$, and the trajectory of each cluster $\bigcup_t \boldsymbol{P}_t^{\boldsymbol{q}}$.
$k \leftarrow 1$;
repeat
  /* Clustering module (Section 3.2). */
  for all partitioned chunks m **do**
    Build motion distance matrix $\boldsymbol{D}_m$ (Section 3.2.1);
    Cluster on $\boldsymbol{D}_m$ to get $\theta_m(i)$ (Section 3.2.2);
  end for
  Determine $\theta(i)$ from $\bigcup_m \theta_m(i)$ (Section 3.2.3);
  /* SLAM module (Section 3.3). */
  for all clusters $\boldsymbol{q}$ **do**
    if $\boldsymbol{q} = \boldsymbol{0}$ **then** $\boldsymbol{b} \leftarrow \boldsymbol{c}$ **else** $\boldsymbol{b} \leftarrow \boldsymbol{q}$;
    Initialize $\bigcup_i \hat{\boldsymbol{X}}^{\boldsymbol{q},i}$ and $\bigcup_t \boldsymbol{P}_t^{\boldsymbol{b}}$ (Section 3.3.1);
    Optimize for $\bigcup_i \hat{\boldsymbol{X}}^{\boldsymbol{q},i}$ and $\bigcup_t \boldsymbol{P}_t^{\boldsymbol{b}}$ (Section 3.3.2);
  end for
  $k \leftarrow k + 1$;
until clustering converges or k exceeds a limit.

Stereo keypoints form the basis of our algorithm. The stereo keypoint corresponding to the i-th landmark at frame t ($i, t \in \mathbb{N}^*$) is denoted $\boldsymbol{x}_t^i = (u_\mathrm{L}, v_\mathrm{L}, u_\mathrm{R})$ where $(u_\mathrm{L}, v_\mathrm{L})$ are the coordinates in the left image and u_R is the horizontal coordinate in the right image. $\boldsymbol{X}_t^{*,i} \in \mathbb{R}^3$ and $\Sigma_t^{*,i} \in \mathbb{R}^{3\times3}$ respectively represent the local 3D coordinates and uncertainty of the i-th landmark in some coordinate system $*$ at frame t. The back-projection function $f : \boldsymbol{x}_t^i \to \boldsymbol{X}_t^{\boldsymbol{c},i}$ w.r.t. the stereo camera model projects the observation into the camera local coordinate system $\boldsymbol{c}$. In order to allow for the pixel errors of the keypoint extraction methods [18, 39], we introduce the stereo noise model [40], where the influence of the extraction error of $\boldsymbol{x}_t^i$ on $\boldsymbol{X}_t^{\boldsymbol{c},i}$ can be calculated as $\Sigma_t^{\boldsymbol{c},i} = \boldsymbol{J}_f \Sigma_t^i \boldsymbol{J}_f^\mathrm{T}$, $\boldsymbol{J}_f$ is the Jacobian matrix of the back-projection function f, and Σ_t^i is the covariance of $\boldsymbol{x}_t^i$ assigned w.r.t. the keypoint extraction error in its image coordinates. For transformations and poses, we define $\boldsymbol{P}_t^{\boldsymbol{q}} \in \mathrm{SE}(3)$ for the pose of cluster $\boldsymbol{q}$ ($\boldsymbol{q} \in \mathbb{N}^*$) at frame t and $\boldsymbol{P}_t^{\boldsymbol{c}}$ for the pose of the stereo camera. We assign all static landmarks to a single static cluster with $\boldsymbol{q} = \boldsymbol{0}$, hence $\forall t, \boldsymbol{P}_t^{\boldsymbol{0}} \equiv \boldsymbol{I}$. For simplicity of subsequent equations, we denote relative

transformations as $\boldsymbol{T}_t^{\boldsymbol{ab}} = (\boldsymbol{P}_t^{\boldsymbol{a}})' \cdot \boldsymbol{P}_t^{\boldsymbol{b}}$ ($\boldsymbol{P}'$ being the inverse of $\boldsymbol{P}$) for coordinate transformations, with $\boldsymbol{R} \in \mathrm{SO}(3)$ being the rotation part of $\boldsymbol{T}$. For details of how the frontend generates correspondences between landmarks (i.e., tracklets of feature points, the input of Algorithm 1) on the input frames, we refer readers to Section 4.1 for our implementation details w.r.t. the evaluation datasets.

3.2 Clustering landmarks

3.2.1 Motion distance matrix

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

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

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

Generally, using the vector form $\boldsymbol{l}_t^{*,ij} = \boldsymbol{X}_t^{*,i} - \boldsymbol{X}_t^{*,j} \in \mathbb{R}^3$ instead of the scalar form l_t^{ij} provides more accurate uncertainty estimation as $\Sigma_t^{*,i} + \Sigma_t^{*,j}$ for depicting motion consistency ($*$ stands for any valid coordinate system), but the scalar form $l_t^{ij} = \| \boldsymbol{l}_t^{*,ij} \|$ has the property of being invariant to local coordinate system changes. Hence, we 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 variance σ_t^{ij} approximated as the error propagation from $\boldsymbol{l}_t^{*,ij}$ to l_t^{ij}:

$$\sigma_t^{ij} \approx \frac{\left\| \boldsymbol{l}_t^{*,ij} \right\|^2_{\Sigma_t^{*,i}} + \left\| \boldsymbol{l}_t^{*,ij} \right\|^2_{\Sigma_t^{*,j}}}{\left\| \boldsymbol{l}_t^{*,ij} \right\|^2} \tag{2}$$

where each Mahalanobis term can be computed under any unified coordinate system $*$. The derivation is given in Appendix A.2.

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

Element rejection. If co-occurrences of a pair of landmarks are too rare (fewer than 4 in our implementation), the maximum-likelihood estimation is no longer considered accurate due to insuffient measurements. In such cases, we mark the corresponding d^{ij} as invalid. Hence, $\boldsymbol{D}$ becomes a sparse matrix for scenes with few co-occurrences. Our hierarchical clustering method (Section 3.2.2) can cope with such sparse input.

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

3.2.2 Hierarchical agglomerative clustering

We use hierarchical agglomerative clustering (HAC) [38] which enables us to perform clustering on the sparse distance matrix $\boldsymbol{D}$. At the beginning of clustering, we take each landmark i as a cluster, and then iteratively merge clusters pairwise 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 [41] is chosen since motion consistency between landmarks

is not transitive, i.e., consistency between landmarks i, j and j, k does not ensure consistency between landmarks i and k. By implementing a heap structure over all elements in $\boldsymbol{D}$ and keeping track of changes, the time complexity of HAC is $O(n^2 \log n)$. Several alternative choices for clustering are compared in Section 4.4, showing the advantage of using HAC.

3.2.3 Consensus clustering from multiple chunks

The size of $\boldsymbol{D}$ grows quadratically with the number of landmarks, which increases with the number of input frames. To speed up the clustering algorithm, we divide the input sequence into multiple chunks (100 frames with 25 frame overlap to the next chunk, for cluster association) and perform HAC clustering (Section 3.2.2) separately. The influence of chunk size is examined in Section 4.4.

Next, 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 [42]. Detailed in Algorithm 2, this algorithm resembles k-means in spirit but instead treats the assignments themselves as data points. The desired number of consensus clusters k is selected as the sum of numbers of clusters for all chunks for outdoor cases, and the maximum number of clusters for all chunks for indoor cases. The final cluster assignments θ are initialized randomly, with the probability of cluster $\boldsymbol{q} = \boldsymbol{0}$ (the cluster with static landmarks) set to 80% while other clusters are chosen uniformly. Clusters with too few landmarks ($\leqslant 2$) are pruned to make the total number of clusters controllable.

Algorithm 2 Iterative voting consensus

Input: a set of landmarks $\bigcup_i \boldsymbol{X}^i$ to be classified, a set of clusters $\bigcup_m \theta_m$ obtained from all chunks and the desired number of consensus clusters K.
Output: Consensus clustering θ with K clusters.
Initialize θ as described in text;
repeat
  Let $\mathcal{X}_{\boldsymbol{q}} = \{i|\theta(i) = \boldsymbol{q}\}$ be the $\boldsymbol{q}$-th cluster;
  Compute the representation center for each cluster: $y_{\mathcal{X}_{\boldsymbol{q}}} = \{\text{majority}\{(\mathcal{X}_{\boldsymbol{q}})_1\}, \ldots, \text{majority}\{(\mathcal{X}_{\boldsymbol{q}})_m\}\}$, where $\{(\mathcal{X}_{\boldsymbol{q}})_m\}$ is the set of clustering results of $\mathcal{X}_{\boldsymbol{q}}$ according to θ_m;
  for all landmark i **do**
    Re-assign $\theta(i) \leftarrow \text{argmin}_{\boldsymbol{q}} D(y^i, y_{\mathcal{X}_{\boldsymbol{q}}})$, where $D(y^i, y_{\mathcal{X}_{\boldsymbol{q}}})$ is the Hamming distance between vector y^i and $y_{\mathcal{X}_{\boldsymbol{q}}}$; only valid values in the sparse vector y^i are considered;
  end for
until θ does not change.

3.3 SLAM for clusters and camera ego-motion

3.3.1 Noise-aware cluster shape initialization

The initialization process for a new cluster $\boldsymbol{q}$ aims to estimate the positions of its landmarks $\hat{\boldsymbol{X}}_t^{\boldsymbol{q},i}$ (note the difference between $\hat{\boldsymbol{X}}_t$ and the back-projected single-frame position $\boldsymbol{X}$, where the former represents estimated state using all historical frames up to frame t). Like the reconstruction pipelines in Ref. [43], this process contains two operations, registration and integration, and we consider uncertainty of frame observations in both. In our implementation, we maintain a Gaussian mixture $\mathcal{G}_t^{\boldsymbol{q},i}$ for each landmark i and regard $\hat{\boldsymbol{X}}_t^{\boldsymbol{q},i}$ as the mean for all components in the mixture.

Frame-to-model registration. When the first frame of a cluster $\boldsymbol{q}$ is encountered in frame t, we initialize the local coordinates of this cluster by assigning $\boldsymbol{T}_t^{\boldsymbol{qc}} = \boldsymbol{I}$ for integration. For subsequent frames, frame-to-model registration is executed to obtain the transformation $\boldsymbol{T}_t^{\boldsymbol{qc}}$ which best aligns points $\boldsymbol{X}_t^{\boldsymbol{c},i}$ from the current local coordinates of the frame $\boldsymbol{P}_t^{\boldsymbol{c}}$ with the points $\hat{\boldsymbol{X}}_{t-1}^{\boldsymbol{q},i}$ in the previously constructed coordinate system, with acceptable noise Σ_g^i:

$$\begin{aligned}\boldsymbol{T}_t^{\boldsymbol{qc}} &= \underset{\boldsymbol{T}}{\text{argmax}} \prod_i \sum_{g \in \mathcal{G}_{t-1}^{\boldsymbol{q},i}} \mathcal{N}\left(\boldsymbol{T}\boldsymbol{X}_t^{\boldsymbol{c},i} - \hat{\boldsymbol{X}}_{t-1}^{\boldsymbol{q},i}; \boldsymbol{0}, \Sigma_g^i\right) \\ &\approx \underset{\boldsymbol{T}}{\text{argmax}} \prod_i \max_{g \in \mathcal{G}_{t-1}^{\boldsymbol{q},i}} \mathcal{N}\left(\boldsymbol{T}\boldsymbol{X}_t^{\boldsymbol{c},i} - \hat{\boldsymbol{X}}_{t-1}^{\boldsymbol{q},i}; \boldsymbol{0}, \Sigma_g^i\right)\end{aligned} \tag{3}$$

where $\mathcal{N}(x; \mu, \Sigma)$ is the probability density function of the multivariate normal distribution. The above equation is equivalent to minimizing the following energy:

$$\begin{aligned}\boldsymbol{T}_t^{\boldsymbol{qc}} &= \underset{\boldsymbol{T}}{\text{argmin}} \sum_i \min_{g \in \mathcal{G}_{t-1}^{\boldsymbol{q},i}} \left(\frac{1}{2}\left\|\boldsymbol{T}\boldsymbol{X}_t^{\boldsymbol{c},i} - \hat{\boldsymbol{X}}_{t-1}^{\boldsymbol{q},i}\right\|_{\Sigma_g^i}^2 - \boldsymbol{C}_g^i\right) \\ \boldsymbol{C}_g^i &\triangleq \frac{1}{2}\log|\Sigma_g^i| + \log|\Sigma_g|\end{aligned} \tag{4}$$

where g traverses each component of the Gaussian mixture $\mathcal{G}_{t-1}^{\boldsymbol{q},i}$, and Σ_g is its g-th covariance component. $\Sigma_g^i = \boldsymbol{R}\Sigma_t^{\boldsymbol{c},i}\boldsymbol{R}^{\mathrm{T}} + \Sigma_g$. $\boldsymbol{C}_g^i$ is the constant factor introduced by maximum-likelihood estimation [44]. This formulation can be viewed as a weighted ICP algorithm, with the weight considering the uncertainty of both the frame and the model.

Frame integration and shape refinement. After the transformation matrix for the latest frame t has been robustly estimated, we update the Gaussian mixture $\mathcal{G}_{t-1}^{\boldsymbol{q},i}$ by inserting a new component with covariance $\Sigma_{g'} = \boldsymbol{R}_t^{\boldsymbol{qc}} \Sigma_t^{\boldsymbol{c},i} (\boldsymbol{R}_t^{\boldsymbol{qc}})^{\mathrm{T}}$ weighted by $1/|\Sigma_{g'}|$, and remove the component with the least weight from the mixture if the size of $\mathcal{G}_{t-1}^{\boldsymbol{q},i}$ exceeds 3. This strategy ensures registration is considered using one of the most reliable measurements. Then, we integrate the observation into the landmark position $\hat{\boldsymbol{X}}_t^{\boldsymbol{q},i}$ using:

$$\hat{\boldsymbol{X}}_t^{\boldsymbol{q},i} = \underset{\boldsymbol{X}}{\operatorname{argmin}} \frac{\left\| \boldsymbol{X}_t^{\boldsymbol{c},i} - \boldsymbol{T}_t^{\boldsymbol{cq}} \boldsymbol{X} \right\|_{\Sigma_t^{\boldsymbol{c},i}}^2}{|\Sigma_{g'}|} + \sum_g \frac{\left\| \hat{\boldsymbol{X}}_{t-1}^{\boldsymbol{q},i} - \boldsymbol{X} \right\|_{\Sigma_g}^2}{|\Sigma_g|} \tag{5}$$

where the first term is used to add current observations and the other terms are for previous observations.

Eqs. (4) and (5) can be solved efficiently using the Gauss–Newton method and QR decomposition, respectively. In practice, we choose to fix Σ_g^i in Eq. (4) during each iteration to make registration easier to solve. We compare such a probabilistic form of registration and integration to the traditional point-to-point registration in Section 4.4. The final initialized cluster poses and landmark positions are $\boldsymbol{T}^{\boldsymbol{qc}} = \boldsymbol{T}_T^{\boldsymbol{qc}}$ and $\hat{\boldsymbol{X}}^{\boldsymbol{s},i} = \hat{\boldsymbol{X}}_T^{\boldsymbol{s},i}$, respectively, where T is the index of the last frame.

3.3.2 Decoupled factor graph optimization

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

$$\boldsymbol{E}_{\boldsymbol{q}} \triangleq \sum_{i,t} \rho \left(\left\| \boldsymbol{x}_t^i - f' \left((\boldsymbol{P}_t^{\boldsymbol{c}})' \boldsymbol{P}_t^{\boldsymbol{q}} \hat{\boldsymbol{X}}^{\boldsymbol{q},i} \right) \right\|_{\Sigma_t^i}^2 \right) \tag{6}$$

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

We considered three candidate optimization strategies. Firstly, *fully-coupled* optimization tries to solve $\boldsymbol{E} = \sum_{\boldsymbol{q}} \boldsymbol{E}_{\boldsymbol{q}}$ w.r.t. variables of all clusters jointly. Secondly, *decoupled* optimization follows three steps: (1) solve $\boldsymbol{E}_{\boldsymbol{q}}$ $(\boldsymbol{q} \neq \boldsymbol{0})$ for $\bigcup_{\boldsymbol{q} \neq \boldsymbol{0}} \{ \bigcup_i \hat{\boldsymbol{X}}^{\boldsymbol{q},i}, \bigcup_t \boldsymbol{T}_t^{\boldsymbol{cq}} \}$ by regarding $(\boldsymbol{P}_t^{\boldsymbol{c}})' \boldsymbol{P}_t^{\boldsymbol{q}}$ as a single variable; (2) solve $\boldsymbol{E}_{\boldsymbol{0}}$ to obtain the camera ego-motion $\bigcup_t \boldsymbol{P}_t^{\boldsymbol{c}}$ and static landmark positions $\bigcup_i \hat{\boldsymbol{X}}^{\boldsymbol{0},i}$; (3) composte the ego-motion and these transformations to generate the motion of clusters as $\bigcup_{\boldsymbol{q},t} \boldsymbol{P}_t^{\boldsymbol{q}}$. Thirdly, *semi-decoupled* optimization adds an additional step to the *decoupled* strategy by solving the whole objective function $\boldsymbol{E} = \sum_{\boldsymbol{q}} \boldsymbol{E}_{\boldsymbol{q}}$ again only for camera motion and *static* landmarks with other state variables fixed.

We plot the sparsity pattern of Hessian matrices of *decoupled* and *fully-decoupled* optimization methods in Fig. 2 for comparison. Let N be the number of landmarks (# Landmarks), T be the number of frames (# Frames), and Q be the number of clusters (# Clusters). *Decoupled* optimization solves Q sub-problems of size $(N/Q+T)^2$ while *fully-coupled* optimization method solves one problem with size $(N + TQ)^2$, which is much larger. However, this analysis does not reflect the matrix sparsity, so an experimental comparison (Section 4.4) was performed. We found empirically that the *decoupled* strategy is sufficiently efficient, and we adopt this method in our final algorithm.

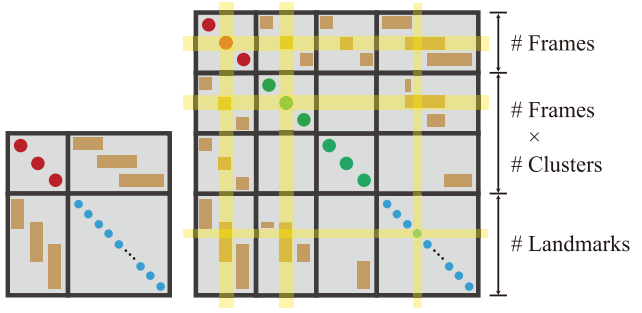


Fig. 2 Comparison of sizes and sparsity patterns of Hessian matrix for *decoupled* and *fully-coupled* factor graph optimization. Red: camera pose block. Green: cluster motion block. Blue: landmark position block. Brown rectangles show a possible sparsity pattern of observations. The intersection of the transparent yellow rectangles denotes the 9 Hessian blocks filled for one observation using the Gauss–Newton method.

4 Evaluation

4.1 Datasets and parameter setup

Our experiments were performed on two synthetic datasets, SUNCG [45] and CARLA [46], and one real-world dataset, KITTI [16].

4.1.1 Synthetic datasets

The SUNCG dataset [45] provides 3D models for constructing indoor scenes, and we built 3 scenes with 2 sequences for each. In these sequences, 2–5 objects as well as the stereo camera are dynamic, with their motions generated manually with 6 degrees of freedom. The CARLA simulator [46] is used

for generating outdoor car-driving scenes. We used its engine to simulate streets with multiple driving vehicles and generated 4 (2 short, CARLA-S1/2 + 2 long, CARLA-L1/2) sequences for experiments.

Ground-truth landmarks were extracted by random sampling of the vertices of these models, and we added a maximum of 1.5 pixels noise to both u, v coordinates to simulate noisy landmark observations on these stereo frames. The synthetic stereo camera had a resolution of 1280×720 and a horizontal field-of-view of 90°, with its baseline set to 10 cm for indoor SUNCG and 50 cm for outdoor CARLA, respectively.

Sampled frames from the generated sequences are shown in Fig. 3 and detailed statistics are listed in Table 1. In order to facilitate future research, we have made this dataset publicly available to the community at `https://cg.cs.tsinghua.edu.cn/people/~huangjh/page/clusterslam-dataset/`.

4.1.2 Real-world dataset

We used selected KITTI raw sequences [16] in which most cars are moving, in order to show the effectiveness of our algorithm. We detected and described landmarks using the state-of-the-art Superpoint [39] network. A similar step as used in Ref. [20] was performed in the frontend to find associations and generate landmark tracklets.

Fig. 3 Frames from our synthetic dataset. First 6 images: indoor sequences rendered from SUNCG. Last 6 images: outdoor sequences simulated by CARLA.

Table 1 Statistics of our synthetic datasets. "# Inst." denotes the number of moving objects. Travel distance shows the length of the camera trajectory, the unit of which is meter

Sequence	# Frames	# Inst.	# Landmarks	Travel distance
SUNCG-1-1	190	2	748	1.94
SUNCG-1-2	250	2	2595	21.10
SUNCG-2-1	300	3	381	6.03
SUNCG-2-2	200	3	370	6.01
SUNCG-3-1	200	5	554	3.61
SUNCG-3-2	200	5	620	11.37
CARLA-S1	200	5	2402	120.92
CARLA-S2	200	8	4179	164.70
CARLA-L1	750	14	13,600	480.87
CARLA-L2	600	17	10,486	367.62

4.1.3 Parameters and hardware setup

Since both the baseline and scales were different in indoor and outdoor scenes, we used two sets of parameters for these two scenarios, respectively. For indoor scenes, these parameters remain as presented in Section 3. For outdoor scenes, we adjusted ϵ to 90.0 to allow for the change in stereo baseline and larger vehicles, and raised the size of each chunk to 200 to maintain the density of $\boldsymbol{D}$ (i.e., to provide sufficient pairwise distances for intra-chunk clustering). We utilized the g^{2}o framework [47] to implement least-squares optimizations. All experiments for the backend were executed on an Intel Core i7-8700K, 32 GB RAM computer with a GTX 1080 GPU.

4.2 Full backend performance

4.2.1 Baselines

To compare the full backend performance, three candidate baseline methods were built. The first used full bundle adjustment, where BA was performed on all visible landmarks assuming them to be static. The second used progressive DCS (dynamic covariance scaling), which goes beyond full BA, using a robust dynamic covariance scaling kernel [1] to determine dynamic objects one by one during each iteration. Specifically, landmarks with average error larger than $\chi_N^2 + 0.3(\chi_M^2 - \chi_N^2)$ were marked as dynamic and introduced at the next iteration (χ_N^2 represents the smallest reprojection error and χ_M^2 the largest). The algorithm repeatedly segments one consensus object after each iteration, until the number of outliers is smaller than 10. The third used semantic segmentation, where a mask R-CNN model trained on the MS-COCO dataset [12] was employed for instance-level segmentation of each input frame. These predicted labels were used to vote for the final labeling of each landmark through a recursive Bayesian. In conclusion, these three categories represent the classical strategy, a robust strategy, and sequential tracking-by-detection methods respectively, as discussed in Section 2.

4.2.2 Evaluation criteria

We use the following metrics to quantitatively compare the performance: (1) $\log \chi^2$, as the logarithm

of reprojection error in BA, reflecting agreement of the optimization results with the input constraints, (2) RMSE, the pointwise root mean square error in 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 rotational/translational relative pose error w.r.t. the ground truth motions, showing the quality of 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 [48]; these two metrics reflect the performance of segmentation, (8) runtime of the whole backend when all frames of the sequence are considered as a batch.

4.2.3 Results and analysis

Quantitative comparisons on all synthetic sequences are averaged and listed in Table 2, with variations of our methods presented together but further discussed in Section 4.4. While our method requires more processing time, it outperforms other methods for tracking and mapping. Although mask R-CNN [12] is better than ours for segmenting individual objects in 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. Progressive DCS is fastest due to a better Gauss–Newton quadratic approximation of its cost function [1].

In terms of quality, 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 accurate kinematics (there is a low signal-to-noise ratio). Despite the higher clustering accuracy of semantic segmentation than our method, its estimation of motions and landmark positions is affected by their imprecise masks, since an inaccurate mask at the border of an object may erroneously categorize nearby landmarks, influencing backend performance. We show visual comparisons of different clustering results on sample frames (Fig. 4) and motion trajectories (Fig. 5): our method produces more accurate results.

4.2.4 Performance on KITTI

Our algorithm is also tested on KITTI and the estimated trajectories are further smoothed by applying Gaussian interpolation [49] to reduce jitter. We compare and list the results in Table 3 with a visualized sample shown in Fig. 6. In addition to a better camera ego-motion, our algorithm can further detect and track multiple moving cars.

We also compared our ego-motion estimation performance with two dynamic SLAM systems, DynSLAM [7] and Li et al. [10]. Results are given in Table 4 using ATE as metric. In general, our backend can reasonably identify the static scene, leading to accurate camera tracking. We outperform DynSLAM by a large margin because DynSLAM uses simple frame-to-frame odometry which is inherently prone

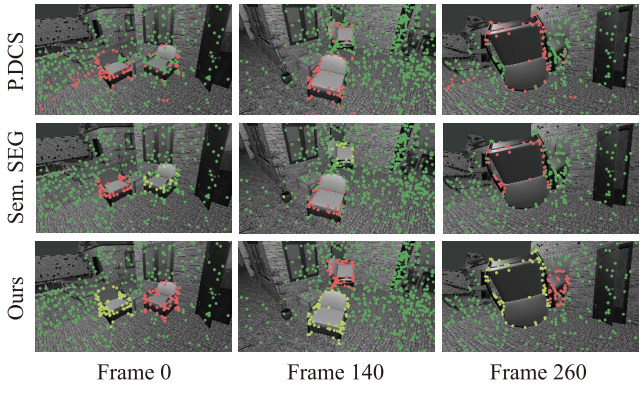


Fig. 4 Visual comparisons on a SUNCG sequence. Landmarks are colored according to cluster.

Table 2 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 [12] prediction time is excluded.

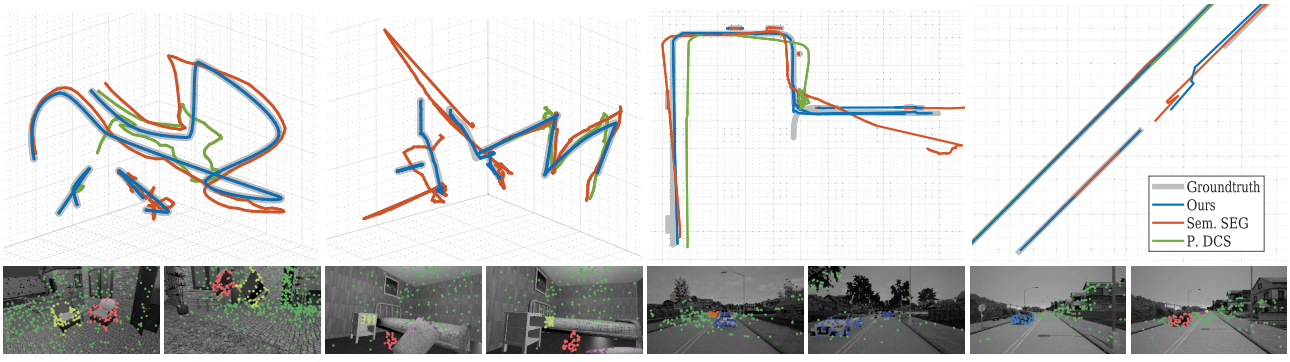


Fig. 5 Trajectories recovered from synthetic sequences. Resolutions of the major grids (solid lines) are 1, 1, 20, and 5 m, respectively. Disconnected trajectories indicate multiple dynamic objects.

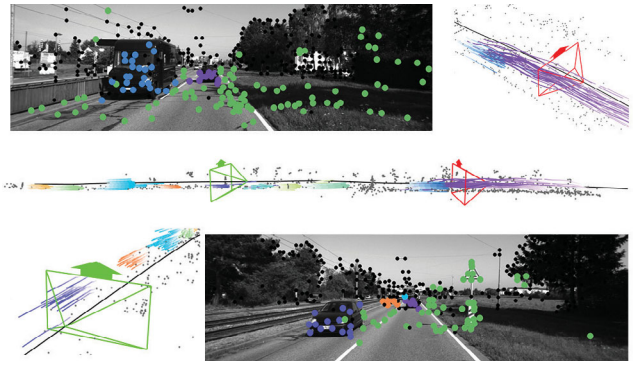


Fig. 6 Our results on the KITTI dataset. Middle: Overview of the street with the camera trajectory (black) and multiple clusters colored by index and time. Top, bottom: Sample images and visualized point clouds in perspective view.

to drift. We perform on par with Li et al. [10] in ego-motion estimation even without a semantic detection module thanks to our robust motion clustering mechanism. However, for sequences like 1003-0047 with slow movement over long distances, it is hard to detect subtle motions and these motions eventually corrupt odometry estimation in our system.

Table 3 Ego motion comparison on KITTI sequences

	0926-0013			0926-0015			0926-0017		
	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**

Table 4 Camera ego-motion comparison on selected KITTI raw sequences. The error is ATE in meters

Sequence	DynSLAM [7]	Li et al. [10]	Ours
0926-0009	7.51	1.14	**0.92**
0926-0014	5.98	**0.51**	0.81
0926-0051	10.95	**0.76**	1.19
0926-0101	10.24	5.30	**4.02**
0929-0004	2.59	**0.40**	1.12
1003-0047	9.31	**1.03**	10.21

4.3 Real-time SLAM system performance

The evaluation in Section 4.2 runs our algorithm using the entire video sequence as a single batch. In real-time scenarios, it is intractable to run batch optimization on all acquired frames. We now consider implementation strategies to build an online version of our system which can run at 7 Hz for outdoor cases.

4.3.1 Implementation

Following Ref. [20], we restrict the number of input frames to our backend algorithm to a small window of 30 recent frames and optimize within the window periodically. Clusters detected in different runs of the backend are associated by counting co-existing landmarks. Pairs of historical 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 historical and new trajectories to connect them. Experiments show that this implementation takes 80 and 21 ms on average for every iteration of optimization for indoor and outdoor sequences, respectively, due to their different numbers of landmarks.

4.3.2 Comparisons and analysis

We compared accuracy and speed of our online method, denoted "Ours (RT)", to our batch version (denoted as "Ours"), stereo ORB-SLAM2 [18], its variant proposed by Murali et al. [50], DynSLAM [7], and CarFusion [21]. Table 5 shows the effectiveness of our method in precisely acquiring trajectory and clustering. The criteria used are the same as Section 4.2.

Table 5 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.

ORB-SLAM2 [18] and Murali et al.'s system [50] are only designed for ego-motion tracking, which perform on a par with ours for indoor cases. But for outdoor sequences where a large part of the scene is dynamic, as at road junctions, they fail to precisely track the ego-motion, causing large trajectory error or even tracking loss.

Both DynSLAM [7] and CarFusion [21] determine segmentation through an external deep network instead of optimization. DynSLAM furthermore does not perform backend optimization and suffers from cumulative drift. CarFusion addresses a different kind of input, multiple video sequences captured alongside the road, so we provided it with groundtruth ego-motion to avoid tracking failure, allowing us to concentrate on assessing the motions of the dynamic objects. We find its performance depends on the precision of car keypoint detection: inaccurate detections may conflict with intra-frame smoothing constraints and generate undesirable results.

4.4 Ablation study

Ablation studies were performed and are now discussed for the modules presented in Section 3.

4.4.1 Formulation of motion distance

We evaluated the effectiveness of Eq. (1) by validating the necessity of its formulation, specifically: (1) the noise-aware formulation by switching the Mahalanobis form of the first term to Euclidean and replacing the weighted average l_*^{ij} by an unweighted average (denoted, **w/o U**); (2) the second, vision based prior term by removing it (denoted **w/o I**). Results are provided in Table 2, with other parts of the algorithm unchanged. For outdoor sequences, **w/o U** requires more iterations to converge, and therefore is slower. Since most metrics except running time are worse with these changes, we prove the utility of both forms in improving the final quality of the backend.

4.4.2 Alternative clustering methods

We compared both the clustering accuracy and β_{VI} of our agglomerative clustering method with spectral clustering (SC) and affinity propagation (AP) methods. All three methods do not need a feature vector for each element and therefore are suitable for our problem with the dense pairwise matrix form. We eliminated all non co-visible landmarks and built a dense motion distance matrix $\boldsymbol{D}$ for comparison. In this experiment, we generated landmark observations on synthetic scans with a standard deviation of noise varying from 0 to 3 pixels to test the robustness.

Their performance w.r.t. the standard deviation of noise is shown in Fig. 7. We found that clustering accuracy of AP drops quickly when noise becomes severe. Even for low noise, SC is not as stable as our method (it is not fully accurate even if the input observations are noise-free). For a motion distance matrix $\boldsymbol{D}$ of size 260×260, the average running time of our approach is only 6 ms while AP takes 13 ms and SC takes 180 ms on average.

4.4.3 Chunk size

We used the two long CARLA sequences (CARLA-L1 and CARLA-L2) to investigate how the length of chunks affects β_{VI} and run time. We adjusted the chunk size from 30 to 300 and give the results in Fig. 8. In general, smaller chunks present better clustering results but reduce the number of observations (challenging the density of $\boldsymbol{D}$). It also increases the total running time since *inter-chunk* merging requires more computation.

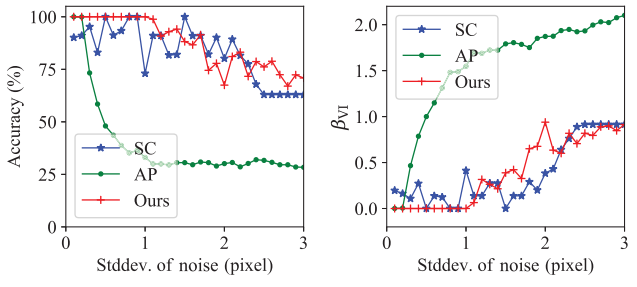


Fig. 7 Accuracy and variation of information for different clustering methods with respect to noise.

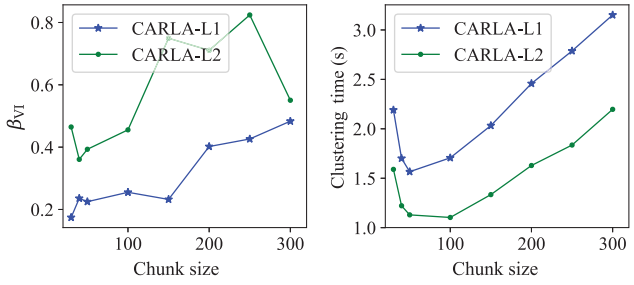


Fig. 8 Chunk size versus quality of clustering and run time for two long CARLA sequences.

4.4.4 Noise-aware cluster initialization

Table 6 shows our results compared to a traditional point-to-point ICP approach, which neglects uncertainty in both registration and integration phases. Our method works better especially on these outdoor sequences, since the uncertainty of the disparity becomes more important as the baseline increases. It is also notably slower than traditional ICP due to the requirement to compute all covariances of observations and the replacemenet of the SVD by Gauss Newton optimization. However, the time taken to initialize all poses is not a bottleneck in the backend.

4.4.5 Decoupled optimization

The three alternative optimization strategies in Section 3.3.2 were tested; their RMSE and run time per iteration are given in Table 7. All optimizers ran for 20 iterations. The results show that the *decoupled* strategy is the fastest, with no obvious difference in quality to the *semi-decoupled* strategy. The *fully-coupled* strategy obtains better results indoors but pose estimation for these dynamic clusters and the camera ego-motion may interfere with each other, producing unexpected results especially on noisy outdoor sequences.

Table 6 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 7 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**

5 Limitations

Despite the general applicability of our approach, there are several limitations worth noticing. Firstly, our backend algorithm relies on the quality of landmark extraction and association from the frontend. Although false associations may be numerically filtered out by the robust kernel, excessive errors can cause unexpected results from our backend. Secondly, our algorithm may fail to detect dynamic objects with insufficient landmarks, since their recovered trajectories are more severely affected by each single noisy landmark.

6 Conclusions

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

Appendix A Derivations

A.1 Noise-aware distance term d^{ij}

We start by defining the affinity probability of landmarks i and j as the product of 3D geometric probability $\mathcal{P}_{3D}$ and vision based prior probability $\mathcal{P}_{2D}$ with balancing factor 2α:

$$\mathcal{P}^{ij} = \mathcal{P}_{3D}(\mathcal{P}_{2D})^{2\alpha} \tag{7}$$

For $\mathcal{P}_{3D}$, we assume that the distance between the two endpoints should stay unchanged in different frames, i.e., $l_t^{ij} = l_{t'}^{ij}, \forall t, t'$, and utilize a maximum likelihood estimate of the optimal length l_*^{ij} considering the uncertainty of each l_t^{ij}:

$$\begin{aligned} l_*^{ij} &= \underset{l}{\operatorname{argmax}} \exp(-\frac{1}{2}\sum_t \| l_t^{ij} - l \|^2_{\sigma_t^{ij}}) \\ &= \underset{l}{\operatorname{argmin}} \sum_t \| l_t^{ij} - l \|^2_{\sigma_t^{ij}} \\ &= \sum_t (\frac{1}{\sigma_t^{ij}} \cdot l_t^{ij}) \Big/ \sum_t \frac{1}{\sigma_t^{ij}} \end{aligned} \tag{8}$$

If the deviations of the observed lengths w.r.t. the optimal length conform to their noise distribution, the likelihood should be large and vice versa. Hence, the 3D geometric probability is summarized

through maximum-a-posteriori calculation over all observations in co-visible frames, whose total number is denoted ψ^{ij} ($\mathcal{N}$ is the normal distribution):

$$\mathcal{P}_{3D} = \prod_t \mathcal{N}(l_t^{ij}; l_*^{ij}, \sigma_t^{ij})^{\frac{1}{\psi^{ij}}} \tag{9}$$

The prior probability $\mathcal{P}_{2D}$ is defined as follows:

$$\mathcal{P}_{2D} = C_{2D} \exp\left(-\frac{1}{2} \max_t \| \boldsymbol{x}_t^i - \boldsymbol{x}_t^j \|^2_{\Sigma_t^{ij}}\right) \tag{10}$$

where C_{2D} normalizes the probability; Σ_t^{ij} is the uncertainty in landmark distance in image space and we assume this uncertainty stays unchanged in different frames t.

The noise-aware distance term d^{ij} is therefore taken as the negative logarithm of the affinity probability $\mathcal{P}^{ij}$ to avoid numerical underflow:

$$\begin{aligned} d^{ij} &= -\log \mathcal{P}^{ij} \\ &= -\log \mathcal{P}_{3D} - 2\alpha \log \mathcal{P}_{2D} \\ &= \frac{1}{2} \operatorname*{avg}_t \left(\left\| l_t^{ij} - l_*^{ij} \right\|^2_{\sigma_t^{ij}} + \log \sigma_t^{ij} \right) \\ &\quad + \alpha \max_t \left\| \boldsymbol{x}_t^i - \boldsymbol{x}_t^j \right\|^2_{\Sigma_t^{ij}} + \underbrace{\frac{1}{2} \log 2\pi + 2\alpha C_{2D}}_{\text{constant}, \forall i,j} \end{aligned} \tag{11}$$

We ignore the trailing common constant in d^{ij} since it does not affect the output of the clustering algorithm which relies only on relative ordering of values.

A.2 Approximate variance σ_t^{ij}

The variance of length σ_t^{ij} is approximated by the error propagation theorem, as the variance of $h(\boldsymbol{x})$ can be deduced from the variance of $\boldsymbol{x}$, denoted Σ_x, through a first-order approximation:

$$\Sigma_{h(\boldsymbol{x})} \approx \boldsymbol{J}_h \Sigma_{\boldsymbol{x}} \boldsymbol{J}_h^{\mathrm{T}} \tag{12}$$

with $\boldsymbol{J}_h$ being the Jacobian of h w.r.t. $\boldsymbol{x}$. Such an approximation is the basis for both the stereo back-projection uncertainty and Eq. (2).

In Eq. (2), we define function h as the Euclidean distance:

$$h\left(\begin{bmatrix} \boldsymbol{X}_t^{\times,i} \\ \boldsymbol{X}_t^{\times,j} \end{bmatrix}\right) = \left\| \boldsymbol{X}_t^{\times,i} - \boldsymbol{X}_t^{\times,j} \right\| = l_t^{ij} \tag{13}$$

Its Jacobian can be computed as

$$\boldsymbol{J}_h = \frac{1}{l_t^{ij}} \begin{bmatrix} \boldsymbol{X}_t^{\times,i} - \boldsymbol{X}_t^{\times,j} \\ \boldsymbol{X}_t^{\times,j} - \boldsymbol{X}_t^{\times,i} \end{bmatrix}^{\mathrm{T}} \tag{14}$$

The covariance of argument $\boldsymbol{x}$ is

$$\Sigma_{\boldsymbol{x}} = \begin{bmatrix} \Sigma_t^{\times,i} & \boldsymbol{0} \\ \boldsymbol{0} & \Sigma_t^{\times,j} \end{bmatrix} \tag{15}$$

We can then obtain $\sigma_t^{ij} = \Sigma_h$ by substituting Eqs. (14) and (15) into Eq. (12).

Appendix B List of abbreviations

- SLAM: Simultaneous localization and mapping
- AuT: Autonomous things
- MAP: Maximum-a-posteriori
- HAC: Hierarchical agglomerative clustering
- IVC: Iterative voting consensus
- ICP: Iterative closest point
- BA: Bundle adjustment
- DCS: Dynamic covariance scaling
- RMSE: Root mean square error
- ATE: Absolute trajectory error
- RPE: Relative pose error
- RT: Real-time
- MLE: Maximum likelihood estimate

Acknowledgements

This work was supported by the National Key Technology R&D Program (Project No. 2017YFB1002604), the Joint NSFC-DFG Research Program (Project No. 61761136018), and the National Natural Science Foundation of China (Project No. 61521002).

References

[1] Agarwal, P.; Tipaldi, G. D.; Spinello, L.; Stachniss, C.; Burgard, W. Robust map optimization using dynamic covariance scaling. In: Proceedings of the IEEE International Conference on Robotics and Automation, 62–69, 2013.

[2] Carlone, L.; Censi, A.; Dellaert, F. Selecting good measurements via $\ell 1$ relaxation: A convex approach for robust estimation over graphs. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2667–2674, 2014.

[3] Kim, D. H.; Kim, J. H. Effective background model-based RGB-D dense visual odometry in a dynamic environment. *IEEE Transactions on Robotics* Vol. 32, No. 6, 1565–1573, 2016.

[4] Bescos, B.; Facil, J. M.; Civera, J.; Neira, J. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters* Vol. 3, No. 4, 4076–4083, 2018.

[5] Rünz, M.; Agapito, L. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In: Proceedings

of the IEEE International Conference on Robotics and Automation, 4471–4478, 2017.

[6] Runz, M.; Buffier, M.; Agapito, L. MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality, 10–20, 2018.

[7] Barsan, I. A.; Liu, P.; Pollefeys, M.; Geiger, A. Robust dense mapping for large-scale dynamic environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, 7510–7517, 2018.

[8] Xu, B.; Li, W.; Tzoumanikas, D.; Bloesch, M.; Davison, A.; Leutenegger, S.; MID-fusion: Octree-based object-level multi-instance dynamic SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation, 5231–5237, 2019.

[9] Paull, L.; Huang, G.; Seto, M.; Leonard, J. J. Communication-constrained multi-AUV cooperative SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation, 509–516, 2015.

[10] Li, P. L.; Qin, T.; Shen, S. J. Stereo vision-based semantic 3D object and ego-motion tracking for autonomous driving. In: *Computer Vision – ECCV 2018. Lecture Notes in Computer Science, Vol. 11206*. Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y. Eds. Springer Cham, 664–679, 2018.

[11] Jaimez, M.; Kerl, C.; Gonzalez-Jimenez, J.; Cremers, D. Fast odometry and scene flow from RGB-D cameras based on geometric clustering. In: Proceedings of the IEEE International Conference on Robotics and Automation, 3992–3999, 2017.

[12] He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 2961–2969, 2017.

[13] Chen, L. C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A. L. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 40, No. 4, 834–848, 2018.

[14] Lenz, P.; Ziegler, J.; Geiger, A.; Roser, M. Sparse scene flow segmentation for moving object detection in urban environments. In: Proceedings of the IEEE Intelligent Vehicles Symposium, 926–932, 2011.

[15] Huang, J.; Yang, S.; Zhao, Z.; Lai, Y.-K.; Hu, S.-M. Clusterslam: A slam backend for simultaneous rigid body clustering and motion estimation. In: Proceedings of the IEEE International Conference on Computer Vision, 5875–5884, 2019.

[16] Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research* Vol. 32, No. 11, 1231–1237, 2013.

[17] Alcantarilla, P. F.; Yebes, J. J.; Almazán, J.; Bergasa, L. M. On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, 1290–1297, 2012.

[18] Mur-Artal, R.; Tardos, J. D. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics* Vol. 33, No. 5, 1255–1262, 2017.

[19] Kundu, A.; Krishna, K. M.; Jawahar, C. Realtime multibody visual SLAM with a smoothly moving monocular camera. In: Proceedings of the IEEE International Conference on Computer Vision, 2080–2087, 2011.

[20] Judd, K. M.; Gammell, J. D.; Newman, P. Multimotion visual odometry (MVO): Simultaneous estimation of camera and third-party motions. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 3949–3956, 2018.

[21] Dinesh Reddy, N.; Vo, M.; Narasimhan, S. G. CarFusion: Combining point tracking and part detection for dynamic 3D reconstruction of vehicles. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1906–1915, 2018.

[22] Strecke, M.; Stuckler, J. Em-fusion: Dynamic object-level slam with probabilistic data association. In: Proceedings of the IEEE International Conference on Computer Vision, 5865–5874, 2019.

[23] Saputra, M. R. U.; Markham, A.; Trigoni, N. Visual SLAM and structure from motion in dynamic environments. *ACM Computing Surveys* Vol. 51, No. 2, 1–36, 2018.

[24] Costeira, J. P.; Kanade, T. A multibody factorization method for independently moving objects. *International Journal of Computer Vision* Vol. 29, No. 3, 159–179, 1998.

[25] Li, T.; Kallem, V.; Singaraju, D.; Vidal, R. Projective factorization of multiple rigid-body motions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–6, 2007.

[26] Fischler, M. A.; Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* Vol. 24, No. 6, 381–395, 1981.

[27] Azartash, H.; Lee, K.; Nguyen, T. Q. Visual odometry for RGB-D cameras for dynamic scenes. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 1280–1284, 2014.

[28] Xu, X.; Cheong, L.F.; Li, Z. Motion segmentation by exploiting complementary geometric models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2859–2867, 2018.

[29] Vidal, R.; Ma, Y.; Soatto, S.; Sastry, S. Two-view multibody structure from motion. *International Journal of Computer Vision* Vol. 68, No. 1, 7–25, 2006.

[30] Vidal, R.; Hartley, R. Three-view multibody structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 30, No. 2, 214–227, 2008.

[31] Ilg, E.; Mayer, N.; Saikia, T.; Keuper, M.; Dosovitskiy, A.; Brox, T. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In: Proceedings of the IEEE International Conference on Computer Vision, 2462–2470, 2017.

[32] Xie, Z.-F.; Guo, Y.-C.; Zhang, S.-H.; Zhang, W.-J.; Ma, L.-Z. Multi-exposure motion estimation based on deep convolutional networks. *Journal of Computer Science and Technology* Vol. 33, No. 3, 487–501, 2018.

[33] Zhang, C. C.; Liu, Z. L. Prior-free dependent motion segmentation using Helmholtz–Hodge decomposition based object-motion oriented map. *Journal of Computer Science and Technology* Vol. 32, No. 3, 520–535, 2017.

[34] Isack, H.; Boykov, Y. Energy-based geometric multi-model fitting. *International Journal of Computer Vision* Vol. 97, No. 2, 123–147, 2012.

[35] Fan, R. C.; Zhang, F. L., Zhang, M.; Martin, R. R. Robust tracking-by-detection using a selection and completion mechanism. *Computational Visual Media* Vol. 3, No. 3, 285–294, 2017.

[36] Yuan, G.; Sun, P. H.; Zhao, J.; Li, D. X.; Wang, C. W. A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review* Vol. 47, No. 1, 123–144, 2017.

[37] Guha, S.; Rastogi, R.; Shim, K. CURE: An efficient clustering algorithm for large databases. *ACM SIGMOD Record* Vol. 27, No. 2, 73–84, 1998.

[38] Sokal, R. R. A statistical method for evaluating systematic relationship. *University of Kansas Science Bulletin* Vol. 28, 1409–1438, 1958.

[39] DeTone, D.; Malisiewicz, T.; Rabinovich, A. SuperPoint: Self-supervised interest point detection and description. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 337, 2018.

[40] Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2003.

[41] Defays, D. An efficient algorithm for a complete link method. *The Computer Journal* Vol. 20, No. 4, 364–366, 1977.

[42] Nguyen, N.; Caruana, R. Consensus clusterings. In: Proceedings of the IEEE International Conference on Data Mining, 607–612, 2007.

[43] Newcombe, R. A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A. J.; Kohi, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. KinectFusion: Real-time dense surface mapping and tracking. In: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality, 127–136, 2011.

[44] Cao, Y. P.; Kobbelt, L., Hu, S. M. Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras. *ACM Transactions on Graphics* Vol. 37, No. 5, Article No. 171, 2018.

[45] Song, S.; Yu, F.; Zeng, A.; Chang, A. X.; Savva, M.; Funkhouser, T. Semantic scene completion from a single depth image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1746–1754, 2017.

[46] Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning, 1–16, 2017.

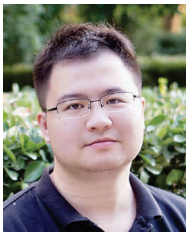
[47] Kümmerle, R.; Grisetti, G.; Strasdat, H.; Konolige, K.; Burgard, W. G^2o: A general framework for graph optimization. In: Proceedings of the IEEE International Conference on Robotics and Automation, 3607–3613, 2011.

[48] Meilă M. Comparing clusterings by the variation of information. In: *Learning Theory and Kernel Machines. Lecture Notes in Computer Science, Vol. 2777.* Schölkopf, B.; Warmuth, M.K. Eds. Springer Berlin Heidelberg, 173–187, 2003.

[49] Ravankar, A.; Ravankar, A.; Kobayashi, Y.; Hoshino, Y.; Peng, C. C. Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. *Sensors* Vol. 18, No. 9, 3170, 2018.

[50] Murali, V.; Chiu, H.-P.; Samarasekera, S.; Kumar, R. T. Utilizing semantic visual landmarks for precise vehicle navigation. In: Proceedings of the IEEE International Conference on Intelligent Transportation Systems, 1–8, 2017.

Jiahui Huang received his B.S. degree in computer science and technology from Tsinghua University in 2018. He is currently a Ph.D. candidate in computer science in Tsinghua University. His research interests include computer vision, robotics, and computer graphics.



Sheng Yang received his Ph.D. degree in computer science from Tsinghua University in 2019. He is currently a software engineer in Alibaba. His research interests include SLAM, robotics, and computer graphics.

Zishuo Zhao is an undergraduate student in the Institute for Interdisciplinary Information Sciences in Tsinghua University. His research interests include computer graphics, computational geometry, operations research, and algorithm theory.

Yu-Kun Lai received his bachelor and Ph.D. degrees in computer science from Tsinghua University in 2003 and 2008, respectively. He is currently a professor in the School of Computer Science & Informatics, Cardiff University. His research interests include computer graphics, geometry processing, image processing and computer vision. He is on the editorial boards of *Computer Graphics Forum* and *The Visual Computer*.

Shi-Min Hu is currently a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing. He received his Ph.D. degree from Zhejiang University in 1996. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He has published more than 100 papers in journals and refereed conferences. He is the Editor-in-Chief of *Computational Visual Media* (Springer), and on the editorial boards of several journals, including *Computer Aided Design* (Elsevier) and *Computers & Graphics* (Elsevier).