Long-Term Visual SLAM With Bayesian Persistence Filter Based Global Map Prediction

Tianchen Deng, Hongle Xie, Jingchuan Wang, and Weidong Chen

***Abstract*— With the rapidly growing demand for accurate localization in real-world environments, visual SLAM has received significant attention in recent years. However, those existing methods still suffer from the degradation of localization accuracy in long-term changing environments. To address these problems, we propose a novel long-term SLAM system with map prediction and dynamics removal. First, a visual point cloud matching algorithm is designed to efficiently fuse 2D pixel information and 3D voxel information. Second, each map point is classified into three types: static, semi-static, and dynamic, based on the Bayesian persistence filter. Then we remove the dynamic map points to eliminate the influence of those map points. We can obtain a global predicted map by modeling the time series of semi-static map points. Finally, we incorporate the predicted global map into a state-of-art SLAM method, achieving an efficient visual SLAM system for long-term dynamic environments. Extensive experiments are carried out on a wheelchair robot in an indoor environment over several months. The results demonstrate that our method has better map prediction accuracy and achieves more robust localization performance.**

I. Introduction

Over the past two decades, visual Simultaneous Localization And Mapping (SLAM) has made significant progress and has a wide range of applications, such as in intelligent wheelchairs and VR. Therefore, there are increasing demands for accurate and robust visual SLAM systems to operate in real-world environments over an extended period of time. However, most existing visual SLAM systems assume a static environment and build a constant map, which is unsuitable for the dynamics in environments. Long-term visual SLAM is becoming an intractable difficulty due to many challenges in dynamic environments and long-term operations.

Most existing visual SLAM systems are usually designed and evaluated for a single operation. However, in real-world deployment, robots usually operate at a region day after day with the requirement of reusing an accurate map in each operation. As the scene changes in real life, the robots have to face the fact that the previous observations are incomplete and uncertain, leading to poor and incorrect performance in localization.

The authors are with Institute of Medical Robotics and Department of Automation, Shanghai Jiao Tong University, and Key Laboratory of System Control and Information Processing, Ministry of Education, Shang hai 200240, China. This work is supported by the National Natural Science Foundation of China under Grant U1813206, the National Key R&D Program of China under Grant 2020YFC2007500, the Science and Technology Commission of Shanghai Municipality, China under Grant 20DZ2220400, and the Pudong New Area Science & Technology Development Fund (PKX2021-R01). (E-mail: dengtianchen@sjtu.edu.cn; xiehongle@sjtu.edu.cn; jchwang@sjtu.edu.cn; *corresponding author: wdchen@sjtu.edu.cn.

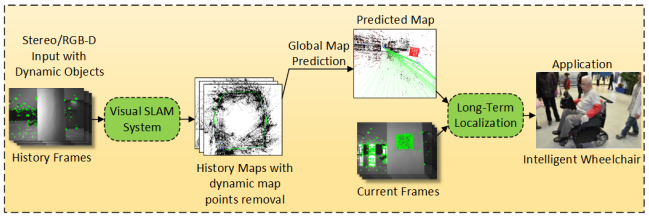

Fig. 1. The proposed long-term visual SLAM system can remove dynamics in environments and predict the future map, resulting in an excellent performance in localization for intelligent wheelchair.

In order to eliminate the influence of dynamic objects, we classify most dynamics into two types: (a)Dynamic: These are the objects that rarely appear and move fast, such as humans moving around. These dynamics usually occur in a short timescale during a single robot traversal. (b)Semi-Static: These are the objects that move slowly or seldom change its position. These dynamics usually happen across a longer timescale that can only be detected by multiple visits to the same place. In this paper, we focus on both dynamic and semi-static objects in long-term environments and eliminate their influence with different methods.

For the dynamic objects in real-world environments, removing the dynamic parts is one of the possible approaches to achieve reliable localization. [1] presents Depth-Edge SLAM system which only uses depth edge points for frame-to-KeyFrame registration. [2] presents DynaSLAM, a visual SLAM system that adds the capabilities of dynamic object detection and background inpainting. [3] presents a novel approach to segment and track multiple moving objects in real dynamic environments and detected objects are classified into stationary and moving objects using the ML-RANSAC algorithm. [4] presents a novel static point cloud map construction algorithm, called Removert, for use within dynamic urban environments. Leaving only static points and excluding dynamic objects is a critical problem in various robust robot missions in changing outdoors. [5] proposes point-correlation SLAM system which utilizes the correlation between map points to separate static and dynamic points and create a sparse graph using Delaunay triangulation. [6] proposes a practical application of persistent filter to recognize the dynamic map points in environments and remove them. Furthermore, some methods estimate both the motion of camera and the dynamic objects. [7] presents ClusterVO, a stereo visual odometry which simultaneously optimizes the

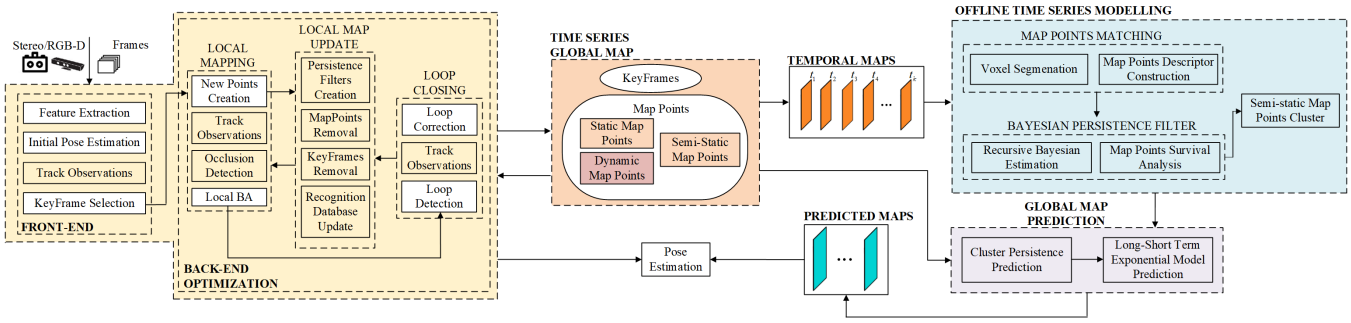

Fig. 2. The architecture of our system. In Front-End module and Back-End Optimization module, each map point is equipment with its own persistence filter. With the persistence filter, dynamic map points are removed and the recognition database is updated. The offline time series modelling module contains the map point matching module, persistence filter module, and semi-static map points cluster module. With the offline modelling module and global map prediction module, we can finally get the predicted maps and achieve long-term real-time localization for our intelligent wheelchair.

poses of the camera and multiple moving objects, regarded as clusters of point landmarks.

For the semi-static objects in real-world environments, there are usually two types of them within a scene: periodic and abrupt. Periodic changes can be both natural, e.g. sunlight changes, or artificial, e.g. turning on or turning off the lights regularly. Periodic changes include the changes that occur slowly and continuously, e.g. the weakening of lighting conditions. Abrupt changes usually occur for once and drastically change the environment. The solution for these semi-static objects is to predict the changes in the environment. [8] is our prior work that we make predictions for some semi-static objects with a fixed set of states, such as the 'opened' and 'closed' states of the door. Then we use the predicted map fused with current observation to maintain a global map in the changing environment. [9], [10] present a new approach for long-term mapping in dynamic indoor environment. They use the spectral model to represent arbitrary timescales of environments dynamics with low memory requirements. Their system can accurately predict the future state of the map point when facing periodic changes. [11] is a recent work that models persistence in semi-static environments, and incorporates the feature persistence estimation into a graphical mapping techniques.

In addition, topological map-based methods are another approach for long-term visual SLAM in dynamic environments. In [12], they propose a topological local-metric framework (TLF), aiming to deal with environmental changes, erroneous measurements and achieve constant complexity. They organize the sensor data collected by the robot in a topological graph, and the geometry is only encoded in the edge to relax the global consistency.

In this paper, we propose a long-term visual SLAM system with dynamic map point removal and global map prediction. Our system is based on ORB-SLAM [14], [15] system with the Bayesian persistence filter [11]. For the dynamic objects in the environment, we use the persistence filter to remove dynamic map points. As for the semi-static objects in the environment, we propose a global map prediction method to model the time series of these map points and predict their future states. In order to obtain a more accurate time series of map points, a map point matching algorithm is proposed which fuses 2D pixel metric information and 3D voxel information, making it more efficient and accurate. Then, by using the Bayesian persistence filter and survival analysis, each map point in the temporal map is classified into static and semi-static based on their persistence probability. The future state of each semi-static map point is predicted according to its previous observations and the predicted maps are built for localization. Furthermore, we extended the persistence filter with a long-short term exponential model to make the predicted map more accurate. The main contributions of this paper are as follows:

- We propose an efficient and accurate segmentation method to separate static, semi-static, and dynamic map points, which can eliminate the influence of dynamics in real-world environments.
- Our method provides a long-term global map prediction method with a map point matching algorithm which can provide us with a global map which fits the current environment.
- A long-term visual SLAM system (LTVS) with Bayesian persistence filter (BPF) is proposed to improve the robustness and accuracy of pose estimation in long-term dynamic environments. Extensive experiments are carried out to demonstrate the effectiveness and accuracy of our system.

II. System Overview

In this section, we present a brief introduction to our system. The architecture of our system is shown in Fig. 2. Those white blocks in Fig. 2 are the same in ORB-SLAM2 [14]. Our system contains several modules which are shown as follows.

A. Front-End and Back-End Optimization

In the front-end module, for each incoming stereo/RGB-D frame, ORB features are extracted, and the global map is initialized. The track observations module is used to track the status of each map point in the current frame. When a KeyFrame arrives, the local mapping module is activated to add map points to the map. Whenever a new map point is

created, the local map update module is activated to create a new persistence filter and assign it to the map point. Furthermore, occlusion detection is employed to avoid wrong input for the persistence filter. For each frame arrival, the persistence probability of all map points is updated for the current time stamp. With the persistence filter, the map point whose persistence probability is under a certain threshold is considered a dynamic map point and is removed from the map. The KeyFrame and recognition database is then updated in the local map update module. As for the loop closing module, it is used to detect loop closures and refine localization.

B. Time Series Global Map

Different from the map built by ORB-SLAM, the time series global map is built to save the previous observations and obtain the temporal maps of the environment. In our algorithm, each map M_i consists of the KeyFrames K_i and the map points Mp_i. Each map point Mp_i stores the following:

- The 3D position O_i in the world coordinate system.
- Viewing direction n_i, which means the ray that joins the point Mp_i with the optical center of the KeyFrames K_i that observe it.
- Survival-time $T_i \in [0, \infty)$ which represents the time when the map point Mp_i ceases to exist.
- Persistence probability of the map point P_i which is the output of the persistence filter. It will be used to classify the map points into static, semi-static, and dynamic.
- A representative map point descriptor $\mathbf{D}_i$ which is different from the original ORB descriptor. The specific description of this descriptor is shown in Section V.
- The maximum and minimum observation distance d_{max}, d_{min}, according to the characteristic of sensor.

C. Offline Time Series Modelling

There are three parts in this module: map points matching module, Bayesian persistence filter module, and semi-static map points cluster module. The Map points matching module is used to get the accurate time series of map points. After the long-term operation in a region, the temporal maps of different time t_k are constructed using the front-end module and back-end optimization module. The voxel segmentation is then operated on the temporal maps to obtain the voxel information of each map point. And then, we fuse the voxel information with the 2D ORB descriptors and create a new 3D map point descriptor $\mathbf{D}_i$. With this carefully designed descriptor, the time series of map points are constructed. Then, with the Bayesian persistence filter and survival analysis, we can model the time series of semi-static map points and get the survival function of the map point. Semi-static map point clustering is performed to reduce the prediction time. Hamming and Euclidean distance is used to describe the distance in temporal and spatial.

D. Global Map Prediction

In this part, we predict the future state of the semi-static map points and obtain the predicted global map. The persistence of map points clusters are calculated and the predicted maps are constructed with long-short term exponential model prediction.

III. Bayesian Persistence Filter

In static environments, objects in environments are fixed for all time due to the static-world assumption. Therefore, 3D map construction only requires when map points are added to it as they are observed. However, for semi-static or dynamic objects, the state of objects varies in time due to the change of environments, and the static-world assumption no longer fits these scenes. Therefore, we incorporate the Bayesian persistence filter into our system for global map maintenance and update, including adding the newly detected map points into the map and removing map points that no longer exist.

Following the survivability method in [11], we briefly formulate our method. Here, each map point Mp_i has its "survival-time", i.e., $T_i \in [0, \infty)$, which represents the time map point Mp_i survive. And the persistence variable δ_i^t

$$\begin{aligned} T_i &\sim p_{T_i}(\cdot) \\ \delta_i^t | T_i &= \begin{cases} 1, & t \leq T_i \\ 0, & t > T_i \end{cases} \end{aligned} \tag{1}$$

where δ_i^t is a Boolean random variable representing whether map point Mp_i exists or not at time t, and $p_{T_i}(\cdot)$ denotes some prior distribution over the survival time T_i. We are interested in estimation for each map point Mp_i, its persistence $P(\delta_i^t = 1 | Y^{1:N})$, where $Y^{1:N}$ are all the map points detection collected until time t_N.

A. Map Point Persistence

For the i_{th} map point in map M, its survival time T_i means that the map point Mp_i should be expected to no longer exist for $t > T_i$. Hence, the persistence of the map point can be defined as:

$$P(\delta_i^t | Y_i^{1:N}) = P(T_i \geq t | Y_i^{1:N}) = \frac{P(Y_i^{1:N} | T_i \geq t_n) P(T_i \geq t)}{P(Y_i^{1:N})} \tag{2}$$

where $Y_i^{1:N}$ denotes the observation sequence of map point Mp_i. $Y_i^{1:N} \triangleq \{y_i^{t_k}\}_{k=1}^N$ and $y_i^{t_k}$ is the random variable describing the state of a observation of a map point Mp_i at the possible observation time t_k. Then, using Bayes' Rule, we can obtain the persistence of a map point at time t.

Then, we estimate the likelihood and the evidence of the map point Mp_i to estimate the persistence probability.

B. Detection Likelihood

In order to obtain the persistence of a map point, we have to calculate the individual detection likelihoods $P(y_i | T_i)$. With a expected senor, this would be simply formalized as: $P(y_i | T_i) = 1$ if $t < T_i$ and $P(y_i | T_i) = 0$ if $t > T_i$. However, in the real-world environment, false detection and

missing detection often occur. And in order to deal with this, we introduce two parameters into our algorithm: the probability of missed detection P_M and the probability of false detection P_F. Then, the formulation of the map point detection likelihood is:

$$P(y_i|T_i) = \begin{cases} P_M^{1-y_i^t}(1-P_M)^{y_i^t}, & T_i \geq t \\ P_F^{y_i^t}(1-P_F)^{1-y_i^t}, & T_i < t \end{cases} \quad (3)$$

Furthermore, if $t \in [t_N, \infty)$, then we can obtain

$$P(Y_i^{1:N}|T_i \geq t) = P(y_i^{1:N}|t_N) = \prod_{k=1}^{N} P_M^{1-y_i^{t_k}}(1-P_M^{y_i^{t_k}}) \quad (4)$$

Hence, the likelihood at time t_{N+1} can be defined recursively. Our work capitalizes on the conditional independence of map point detection given map point persistence. The likelihood is constant over the intervals in the set, and the integral is simplified by defining: $t_0 \triangleq 0, t_{N+1} \triangleq \infty$. Then, following [11], the evidence is calculated as :

$$P(Y_i^{1:N}) = \sum_{t=0}^{N} P(Y_i^{1:N}|T_i)[F_{T_i}(t_{N+1}) - F_{T_i}(t_N)] \quad (5)$$

where $F_{T_i}(t)$ is the cumulative distribution function of P_{T_i} which is described in (1). Then, the evidence can be recursively defined as:

$$P(Y_i^{1:N+1}) = L(Y_i^{1:N}) + P(Y_i^{1:N}|T_i)[1 - F_{T_i}(t_N)] \quad (6)$$

where $L(Y_i^{1:N})$ is the lower partition sum of evidence. $L(Y_i^{1:N+1})$ can also be calculated recursively, resulting in a complete algorithm, which we refer to as the BPF.

C. Survival Analysis

For dynamic map points in the environments, we use the persistence filter to evaluate the persistence of map points and remove dynamic map points. Upon having no information of the map point's activity pattern, we use the survival analysis algorithm [11] to obtain the general-purpose survival priors of the map points. The original probability density function $p_T(\cdot)$ is defined as :

$$p_T(t) = \lambda_T(t)e^{-\Lambda_T(t)} \quad (7)$$

$\lambda_T(\cdot)$ is the hazard function and $\Lambda_T(\cdot)$ is the cumulative hazard function. The principle of maximum entropy is introduced into our method with exponential distribution. The probability density function is simply defined as $p_T(t) = \lambda e^{-\lambda t}$. The parameters of the BPF are the hyperparameters in our system. We tune the parameters according to the category of dynamic objects in the environment. Moreover, we finetune the parameters of BPF in the training dataset to get better performance and adapt to the real-world environment.

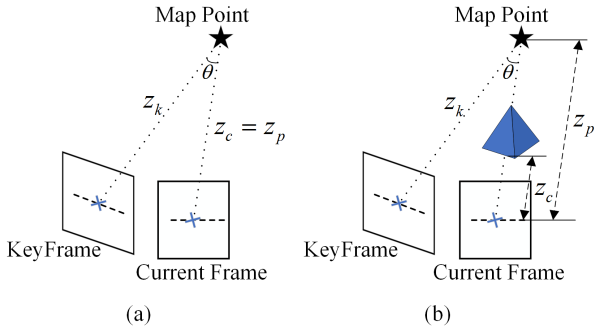

Fig. 3. Occlusion Detection. Every map point Mp_i is projected into the current frame using its 3D position O_i and camera pose. The depth in current frame of the projected keypoint is z_c. The projected depth is z_p. If the difference between z_p and z_c is greater than the threshold z_{th} and the angle θ between KeyFrame and current frame is greater than θ_{th}, we define this map point as occluded.

IV. Dynamic Map Points Removal

In this section, we introduce our dynamics removal algorithm in detail. We incorporate the Bayesian persistence filter into Front-End and Back-End optimization modules to evaluate the persistence probability of map points and remove the dynamic map points. In order to accurately use the persistence filter, we have to track observations of each map point.

A. Track Observations

In the track observations module, we can track all the observations of the corresponding map points to update the persistence filter. Whenever a new frame arrives, we match the keypoints in the current frame with the candidate map point. The maximum and minimum observation distance d_{max} d_{min} are used to select the candidate map points which are in the field of view. Then we project the candidate map points on the current frame and employ ORB descriptors to match them with the keypoints. If the map point is matched, it is defined as "survive" in this timestamp, and we assign hit observation to our persistence filter. Otherwise, the map point is defined as "death" in this timestamp, and we assign missed observation to our persistence filter. However, according to the characteristic of stereo/RGB-D camera, the observation occlusion is really common in the real-world environment, which can really make our persistence filter ineffective. It will assign many wrong observations to our persistence filter, leading to many erroneous deletions of static map points, thus reducing the localization ability of our long-term visual SLAM system. In order to deal with it, we design an algorithm for occlusion detection.

B. Occlusion Detection

For each candidate map point Mp_i, it is projected into the current frame using its depth and camera pose and find the corresponding keypoint in the KeyFrame. Then we calculate the angle θ between KeyFrame and current frame. We also calculate the projected depth z_p and compare it with the depth z_c of keypoint in current frame (from the depth

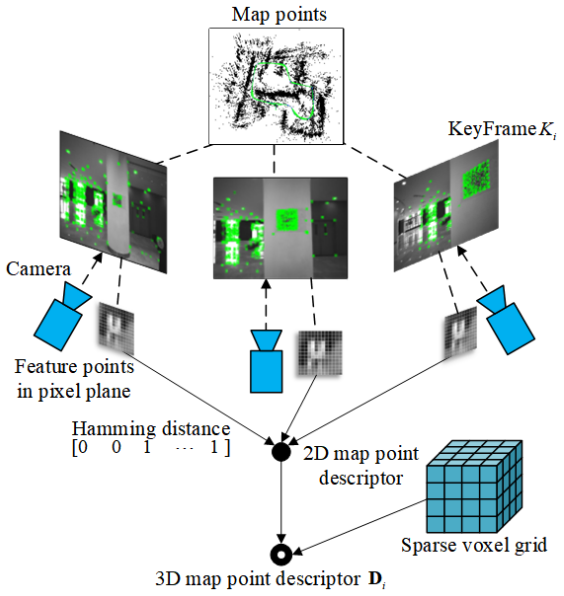

Fig. 4. The architecture of our 3D map points descriptor.

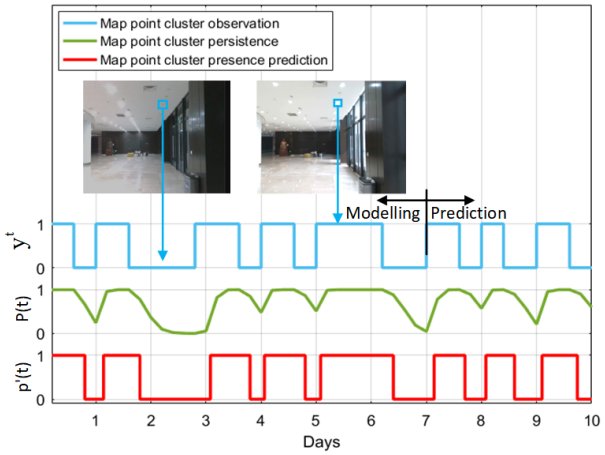

Fig. 5. The survival analysis and prediction of a map point cluster. The observations of map point cluster y^t during the data collection session (blue) are sent to the persistence filter and survival analysis part. Then we can obtain the persistence probability $P(t)$ of the map point cluster (green), and predict the state $p'(t)$ of the map point cluster for a given time (red).

measurement). If the difference between z_p and z_c is greater than the threshold z_{th} and the angle θ is greater than the threshold θ_{th}, the map point is defined as occluded. The detailed information of our method is shown in Fig. 3. The angle threshold is set to 30° and the depth threshold is set to 0.4 m.

C. Local Map Update

In the persistence filters creation module, we create persistence filters for each map point. Whenever a map point is created, a persistence filter is initiated and assigned to it. With the track observations module, the persistence probability of map points is updated in each time stamp. If the persistence probability of a map point is lower than the threshold p_{th}, we remove it from the global map in the map points removal module. In order to get the persistence of the map point, we have to evaluate the persistence of the map point through several frames, so we cannot define it as a dynamic map point immediately. Each map point corresponds with some keypoints in the KeyFrames which observe it. When we remove a map point, we set the corresponding keypoints to unavailable. If there are not enough keypoints in KeyFrame, the KeyFrame will also be removed in the KeyFrame removal module. And then, in the recognition database update module, we update the visual words which are used by the loop detection thread to avoid picking the wrong candidates in the loop closing module.

V. Semi-static Map Points Prediction

In the real-world environment, service robots should be able to operate autonomously in dynamic and daily changing environments over an extended period of time. In this scene, semi-static objects are very common such as the moving chairs or the influence of light conditions. Most existing approaches either rely on the assumption of a static environment or handle particular types of dynamic objects, leading to poor and inaccurate performance in dynamic settings. Our proposed method model the time series of these map points, predict the future state of these semi-static map points and provide the predicted map which fits the current environment. With our method, we can get a more accurate map of the environment and better localization performance.

A. Map Points Matching with 3D Descriptor

In this section, we introduce a novel map point matching algorithm with 3D map point descriptor, which can make our matching algorithm more robust, efficient, and accurate. The architecture of the algorithm is shown in Fig. 4. We make voxel segmentation for each temporal map at first to get the voxel information of the map points. For each map point, we will search for all KeyFrames that have observed this map point and find the minimal Hamming distance with respect to all other associated descriptors in the KeyFrames. Then, we extend the 2D descriptor with 3D voxel information and save the voxel ID information into 2D descriptor, which is called the 3D map point descriptor $\mathbf{D}_i$. When we match different map points in different maps, we match the voxel ID at first, and then match 2D descriptor of map points which are in the same voxel. If the map points can not match, we expand the range of voxel matching and decrease the confidence of map points matching accordingly. If the map points can not match up eventually, we define it "invisible", and the matching result y_t is set to zero. Otherwise, we define it "visible" and set y_t to one. The visible state of every map point is saved in a vector $Y_i^{1:N} = \{y_i^{t_1}, y_i^{t_2}, \ldots, y_i^{t_N}\}$ which is the same as $Y_i^{1:N}$ in section III, to represent the previous observations of them. Hence, the state vector is a sequence of Boolean random variables $Y^{1:N} \subseteq \{0, 1\}$ and the map point is observed at $\{t_i\} \subset [0, \infty)$.

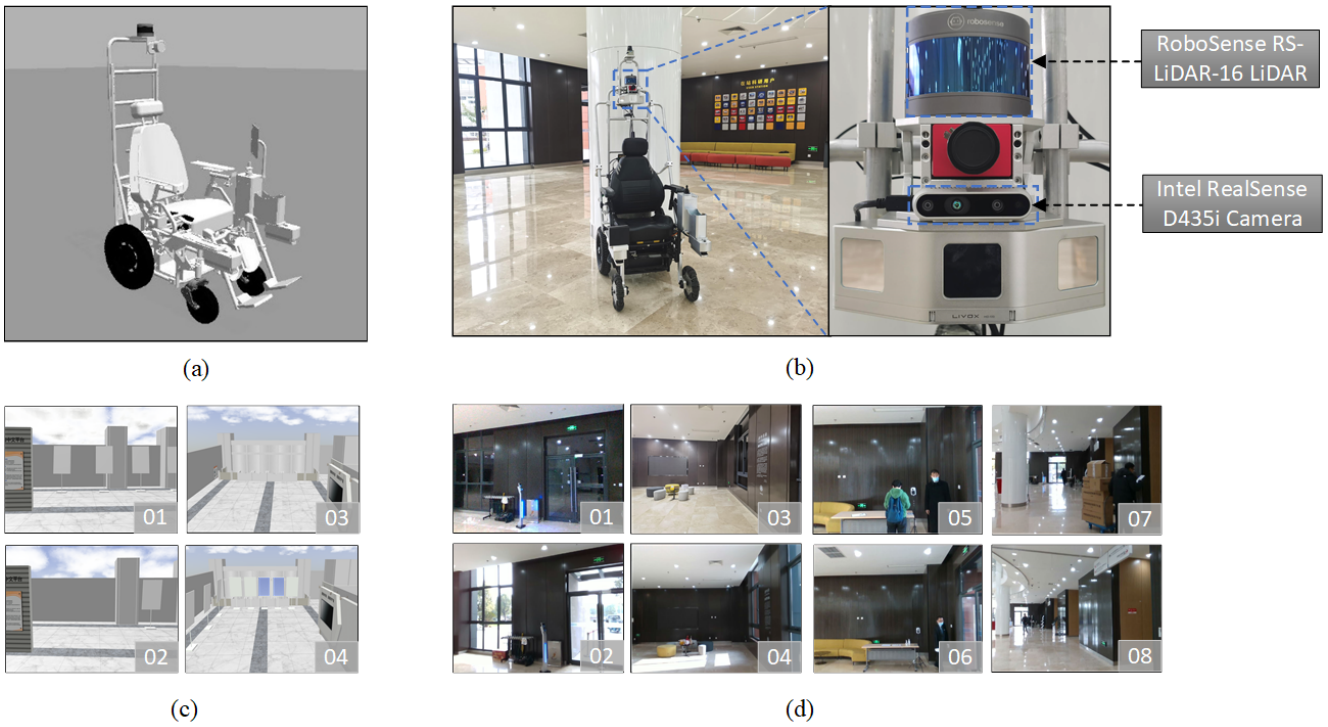

Fig. 6. Experiments scenarios for our intelligent wheelchair. (a) is a simulation of our JiaoLong intelligent wheelchair. (b) is the prototype of our JiaoLong intelligent wheelchair. (c) is the sample images of our simulation scenes and (d) is the images of the real-world experiment scene under different time and different lighting conditions. The upper and lower images in each column show approximately the same place in different data sequences, the scene is changed. (c) are the images of gazebo datasets, and (d) are the images of real-world datasets.

B. Semi-static Map Points Clustering

With the Bayesian persistence filter, we can get the semi-static map points in the global map. In order to reduce the time and memory consumption of global map prediction, time series-based map point clustering is performed first. The semi-static map points are clustered according to their 3D position and time series. D_H represents the Hamming distance of the time series of two clusters. $D_H = \sum_{i=1}^{N} Y_a^i \oplus Y_b^i$, where $Y_a^{1:N}, Y_b^{1:N}$ represents time series of two different map points clusters, Y_a^i, Y_b^i represents the ith state of the time series, and $\oplus$ represents the Exclusive or. D_E represents the Euclidean distance of two clusters. If D_H and D_E are both lower than the threshold, which means the distance in temporal and spatial of the two clusters are close enough, then the two clusters will be merged. Otherwise, the two clusters will not be merged. After that, we will find the minimal Hamming distance with respect to all other map points in the cluster and use this map point to represent the cluster.

TABLE I

COMPARISON OF THE ABSOLUTE TRAJECTORY ERROR (ATE) IN DIFFERENT DATASETS

Methods	Datasets-Low		Datasets-Medium		Datasets-Gazebo	
	RMSE	Max	RMSE	Max	RMSE	Max
ORB-SLAM2	0.4294	1.0744	0.5827	1.3235	0.1242	0.4061
ORB-SLAM3	0.4162	0.6578	0.4728	0.8865	0.0684	0.1392
DVO SLAM	0.7342	1.6831	0.8754	1.8754	0.1562	0.6850
Depth-Edge	0.4327	0.9792	0.5342	1.5839	0.1048	0.3245
DynaSLAM	0.4037	0.9062	0.5061	1.2672	-	-
Point-Correlations	0.3931	0.6972	0.4521	0.8963	**0.0504**	0.1467
LTVS-BPF w/o MPR	0.4221	0.9197	0.4724	0.9721	0.0624	0.1895
LTVS-BPF w/o GMP	0.4323	0.9124	0.4533	0.8876	0.0682	0.1963
LTVS-BPF w/o OD	0.4778	1.4021	0.5659	1.7435	0.1321	0.4156
LTVS-BPF w/o 3DM	0.4342	1.0231	0.4625	0.9531	0.0663	0.1593
LTVS-BPF	**0.3741**	**0.7063**	**0.4324**	**0.8021**	0.0551	**0.1355**

C. Global Map Prediction

After that, with Bayesian persistence filter and survival analysis, we can obtain the accurate persistence probability of map points clusters and model the survival function of the clusters.

Then, the persistence probability is used to predict the state of map points clusters for a given time. We introduce long-short term exponential model [10] to fuse the last observation and the persistence of clusters and predict the state. We define the threshold of map points cluster's survival time γ as: $\gamma^{-1} \leftarrow \frac{1}{m+1}(m\gamma^{-1} + \frac{|y^t - y^{t_l}|}{t - t_l})$ where y^t represents the state of map points cluster at time t, which is the same in Section III, m represents the number of observations. $p'(t)$ is the prediction of the state, $p'(t) = y^{t_l} e^{\frac{t_l - t}{\gamma}} + P(t)(1 - e^{\frac{t_l - t}{\gamma}})$ where $P(t)$ is the persistence probability of map points cluster, the same in section III. And t represent the current

TABLE II

COMPARISON OF THE ABSOLUTE TRAJECTORY ERROR (ATE) ON THE TUM BENCHMARK IN LOW DYNAMIC ENVIRONMENTS

Sequences		RMSE of trajectory alignment [m]						
		ORB-SLAM2	ORB-SLAM3	DVO SLAM	Depth-Edge	DynaSLAM	Point-Correlations	LTVS-BPF
Low dynamic	fr2/desk_with_person	0.0062	0.0068	0.1037	0.0484	-	0.0075	**0.0063**
	fr3/sitting_static	0.0073	0.0072	0.0119	-	-	0.0091	**0.0061**
	fr3/sitting_xyz	0.0111	0.0096	0.2420	0.0397	0.015	0.0096	**0.0087**
	fr3/sitting_halfsphere	0.0234	0.0247	0.2198	0.0432	0.017	0.0235	**0.0168**
	fr3/sitting_rpy	0.0211	0.0203	0.1756	-	-	0.0225	**0.0127**

TABLE III

COMPARISON OF THE ABSOLUTE TRAJECTORY ERROR (ATE) ON THE TUM BENCHMARK IN HIGH DYNAMIC ENVIRONMENTS

Sequences		RMSE of trajectory alignment [m]						
		ORB-SLAM2	ORB-SLAM3	DVO SLAM	Depth-Edge	DynaSLAM	Point-Correlations	LTVS-BPF
High dynamic	fr3/walking_halfsphere	0.3666	0.3165	1.0136	0.0489	**0.025**	0.0354	0.171
	fr3/walking_static	0.0331	0.0283	0.7515	0.0261	0.0108	0.0128	**0.0106**
	fr3/walking_xyz	0.4216	0.3157	1.3830	0.0601	**0.015**	0.0874	0.1583
	fr3/walking_rpy	0.1861	0.1678	1.2922	0.1791	**0.035**	0.1608	0.1121

TABLE IV

PREDICTIVE CAPABILITY.

Methods	Accuracy (%)			Time Consumption (s)
	one day	one week	one month	
LTVS-ARMA	95.37	**91.25**	88.21	5362.9035
LTVS-FreMen	91.42	90.57	88.33	3347.7561
LTVS-BPF	**95.80**	91.15	**91.35**	**1155.9178**

time and t_l represent the time of last observation. When $|t-t_l| \ll \gamma$, the expression $e^{\frac{t_l-t}{\gamma}}$ is close to 1, which means that the expected occupancy would be the same as the one recently observed. And when $|t-t_l| \gg \gamma$, the expression $e^{\frac{t_l-t}{\gamma}}$ is close to 0, which weakens the effect of the latest observation.

The performance of our semi-static map point cluster prediction is shown in Fig. 5. The first seven days are the data collection session. We use the data of the first seven days to model the time series of map points clusters, then predict the future states of map points clusters in the last several days. If the prediction value is higher than the threshold (0.5) we set, it is set to one, which means it survives at the current time, otherwise, it is set to zero, which means it is no longer alive at the current time.

VI. SIMULATION AND REAL-WORLD EXPERIMENTS

A. Simulation

The simulation scene and the model of our intelligent wheelchair are shown in Fig. 6. The model of our intelligent wheelchair is equipment with 3D lidar and realsense camera (Stereo). Simulations are done in Gazebo using an indoor museum environment. In this environment, there are periodical and gradually incremental changes, such as the lighting conditions and the periodical placed boards. We operate the intelligent wheelchair in the simulation scene across days (where the interval is 3h) and collect the temporal maps of the environments. The previous seven days are used as training datasets and the next several days are the testing datasets. For comparison, our results contain the following state-of-art methods: VINS-Fusion [19] (without IMU), ORB-SLAM2 [14], ORB-SLAM3 [15] (without IMU), LTVS with FreMen map prediction method [9], LTVS with ARMA map prediction method [8]. The localization results of our method are shown in Table I. The best results for each sequence are **bold**. The sequence we used is twelve hours after the training datasets. Compared with other static visual SLAM methods, our global map prediction method can predict the future state of these semi-static map points and get the predicted map which is more suitable to the current environment. In addition, our method is also better than other map prediction-based methods, because our method considers the occlusion and error detection of the map points, and the global map prediction is more accurate. The absolute trajectory error of our system is also decline, which means better localization performance in the simulation scene.

B. TUM Datasets Evaluations

We use TUM datasets to demonstrate the effectiveness of our system on dynamics removal. TUM datasets [18] are the indoor sequences from the Technical University of Munich (TUM) RGB-D benchmark. The TUM datasets contain both the static and dynamic scenarios in indoor environments. We divided the environments of TUM benchmark into three types: static, low dynamic, and high dynamic, according to the part of the FOV covered by moving objects. The low dynamic environment sequences are the sitting and desk-person sequences, and the high dynamic environment se-

quences are the walking sequences. We run our dynamic map points removal method on the sequences of TUM datasets and measure the absolute trajectory error (ATE) of different methods. Our method is performed on a desktop computer equipped with Intel Core i7-8750 (2.2 GHz) CPU and 8GB of RAM. For comparison, our results contains the following state-of-art methods: ORB-SLAM2 [14], ORB-SLAM3 [15], dense VO (DVO) [17], Depth-Edge [1], DynaSLAM [2], Point-Correlations [5]. The results of the comparison of ATE of our method and other visual SLAM methods are shown in Table II and Table III.

For low dynamic environments, most of the influence of moving objects can be eliminated. Our method is more accurate compared to those methods with static world assumption [14], [15]. In addition, our system is also better than those methods that consider moving objects. Those methods mistakenly remove some static objects or aggressively segment images. The learning-based methods [1], [2] mask the entire image region of the person. However, most of the body is static in low dynamic environments. As a result, less environment information is used by these methods, and the number of available static map points for pose estimation is not enough, resulting in poor localization performance.

For high dynamic environments, the features on people's bodies are all moving rapidly and should be removed. Our method is better than those methods that assume a static world. When compared with those learning-based methods, our method is not good enough. Those methods use human shape prior, and the entire image region of the person is masked using learning techniques. They are specifically designed for TUM dynamic dataset. Those learning-based methods only utilizing static information perform better in high dynamic sequences. In walking-static sequence, our method performs better than other methods. In this sequence, the camera position is relatively static, and our method can get more accurate poses by removing the dynamic map points and KeyFrames.

C. Real-world Experiments

In this section, we introduce our real-world experiments platform and datasets. JiaoLong intelligent wheelchair is used as our experiments platform. It is equipped with RoboSense RS-LiDAR-16 LiDAR and Intel RealSense D435i camera. An industrial personal computer MIC-770 with Intel Core i7-9700TE CPU is also employed, with Ubuntu and ROS. To demonstrate the effectiveness and general applicability of our system, we collect our own real-world datasets. The experiment scenarios and the detailed information of our intelligent wheelchair are shown in Fig. 6.

For the real-world datasets, we collect the datasets from Nov. 2021 to Feb. 2022, from 9 AM to 9 PM (where the interval is three hours), seven days per week. Our wheelchair is operated day after day in an indoor hall environment. During data collection, there are different types of dynamic objects and changing lighting conditions. The stereo images are captured with 640×480 resolution at 30 Hz, and our localization system is operating at about 10 Hz. Here, the

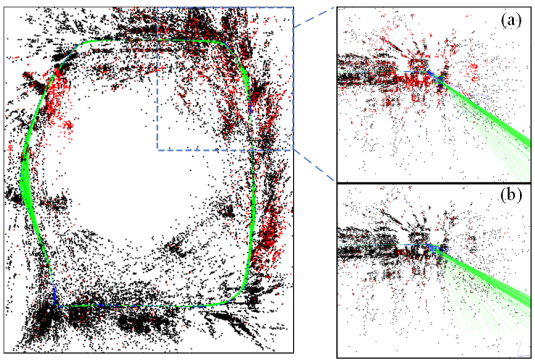

Fig. 7. The result of map points matching. The map points in red are the mismatched points. Figure (a) and (b) on right are the amplification of the circled area. Figure (b) shows the result of map points matching with our matching algorithm.

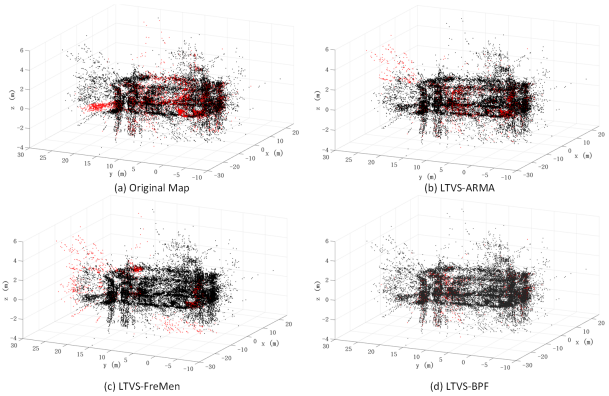

Fig. 8. The original map without map prediction and the predicted maps of different prediction methods. The red map points are the error map points which are the missed map points or the false map points compared with the current map.

data of the first week is used as the training dataset, and the rest of the data in the following months are used as the testing datasets. For the training dataset, there is not a long gap between two sessions, about three hours between two sessions. We also consider the problem that there may be bad localization performance in some places. Suppose the localization performance of our system is bad in some places. In that case, we use the local mapping thread to get the accurate map of the environment and merge the local map into the global map, following the method in ORB-SLAM3 [15]. Furthermore, we also use the re-localization algorithm to get the corresponding on photometric and use the ICP algorithm to get the matching of map point cloud of different sessions. Then we can get an accurate initial pose of the robots. To better evaluate the localization performance, the testing datasets are collected in different speeds: low-speed (0.3 m/s, 10 deg/s), medium-speed (0.8 m/s, 15 deg/s). The long-term datasets will be open sourced in github https://github.com/dtc111111/LTVS.

The ground truth of our real-world datasets is obtained by the 3D LiDAR (RoboSense RS-LiDAR-16 LiDAR) with LIO-SAM method [16].

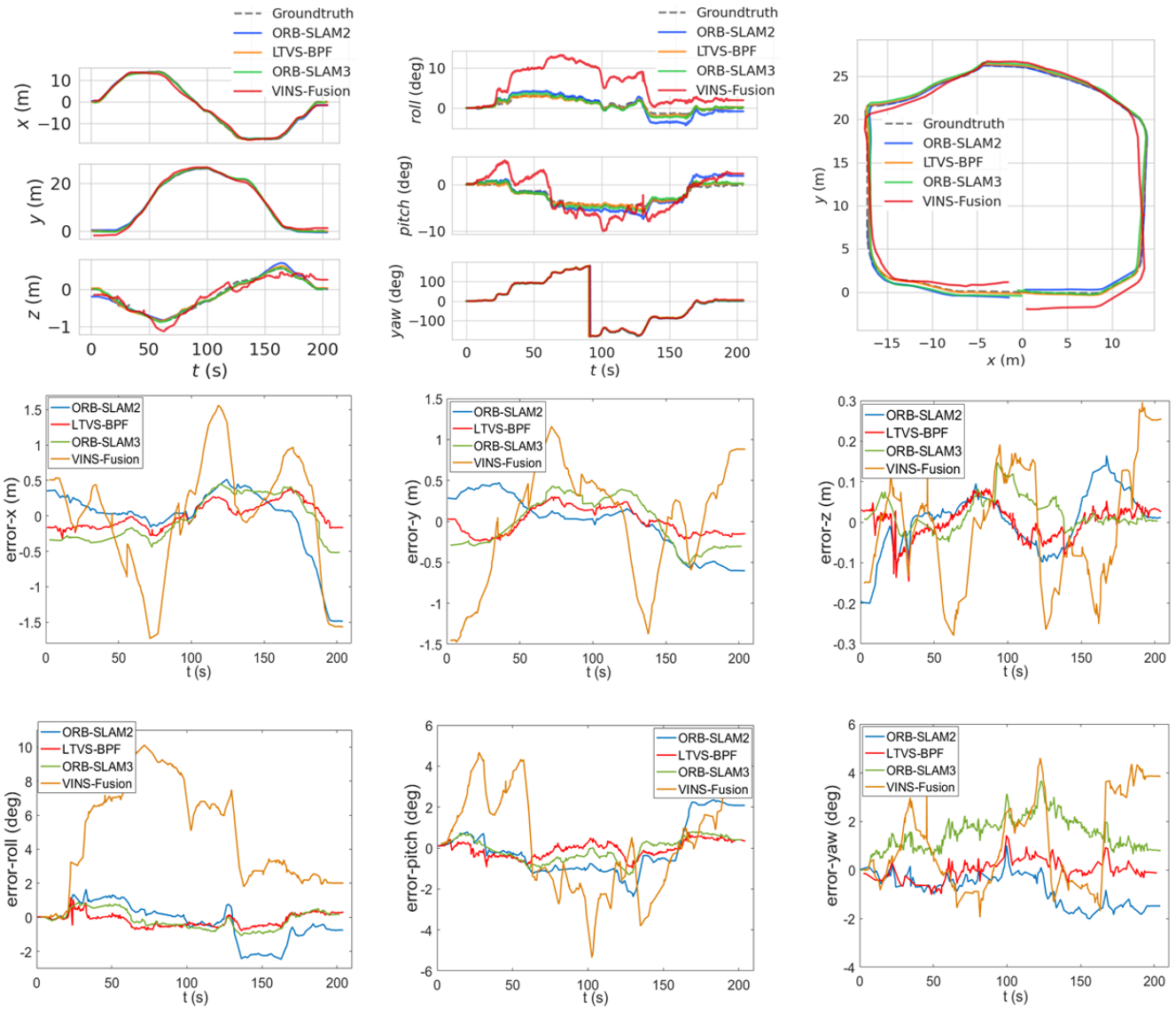

Fig. 9. The first row is the comparison of different visual algorithms of estimated trajectories on real-world datasets. The second row and third row are the XYZ and RPY trajectory error of different visual methods on real-world datasets.

D. Map Points Matching Accuracy

Experiments are carried out to verify the effectiveness and accuracy of our map points matching algorithm compared with the algorithms that do not use the 3D voxel information. The result of our experiment is shown in Fig. 7. If the ORB distance or the spatial distance of the pairs of matched map points is higher than the threshold, they will be defined as the mismatched map points and painted in red in Fig. 7. For the original map points method, some map points are mistakenly matched with the map points that are close in ORB distance and far away from them in Euclid Distance. Therefore, there are many mismatched map points, which decreases the accuracy of map point matching. Our method, which makes use of the voxel information, provides better results in map point matching.

E. Predictive Capability

Predictive capability [9] is defined to evaluate the map prediction performance of our system. Predictive Capability is defined as: $\delta_m = (n_{11} + n_{00})/(n_{11} + n_{10} + n_{01} + n_{00})$ where $n_{11}, n_{10}, n_{01}, n_{00}$ represent the map points that are observed in both the predicted map and the current map, the predicted map alone, the current map alone, and neither of them. Experiments are carried out in one day, one week, and one month after the training datasets. As summarized in Table IV, we evaluate the predictive capability of three algorithms at three different times, and the time consumption of time series modeling is recorded, where our method with the Bayesian persistence filter outperforms other prediction methods. The time consumption in Table IV is the offline time consumption. The predicted global maps of different prediction methods are shown in Fig. 8. The red map points in the original map are the semi-static map points. The error

TABLE V

COMPARISON OF THE ABSOLUTE TRAJECTORY ERROR (ATE) AND CORRECTNESS IN SEQUENCES WITH DIFFERENT TIME AFTER THE LAST OBSERVATION.

Sequences	ORB-SLAM2			LTVS-FreMen			LTVS-ARMA			LTVS-BPF		
	RMSE	Max	$CR^{\varepsilon,\phi}$	RMSE	Max	$CR^{\varepsilon,\phi}$	RMSE	Max.	$CR^{\varepsilon,\phi}$	RMSE	Max	$CR^{\varepsilon,\phi}$
One day later	0.4563	0.7044	99.99%	0.4583	0.7343	99.99%	0.4325	0.7911	99.99%	**0.4219**	**0.7038**	**99.99%**
One week later	0.4935	1.2742	95.5 %	0.4938	1.5531	95.15%	**0.4537**	1.3019	**98.05%**	0.4603	**1.0018**	97.76%
One month later	0.8473	2.0697	83.27%	0.5902	1.4737	91.33%	0.5849	1.5261	90.05%	**0.5255**	**1.6270**	**92.15%**
Two months later	0.8515	2.0021	81.17%	0.6311	1.8519	87.78%	0.6129	1.9085	88.87%	**0.5805**	**1.8562**	**90.17%**

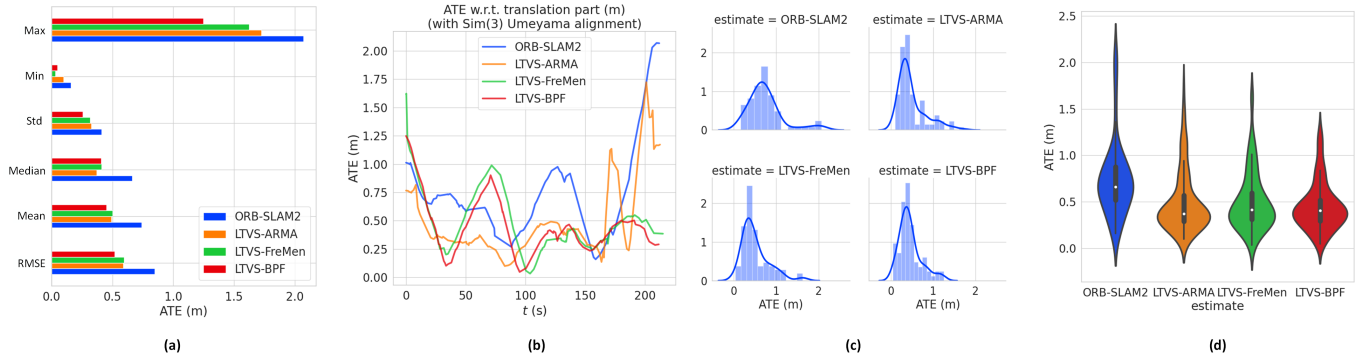

Fig. 10. Related statistics of ATE (absolute trajectory error) of our method and other visual SLAM on the testing datasets one month after the training datasets. (a) The chart of different parameters of ATE. (b) The comparison chart of ATE. (c) The histogram of ATE. (d) The violin-histogram of ATE.

map points in predicted maps contain two types: missed and false map points. The missed map points are in the current map but not in the predicted map. The false map points are in the predicted map but not in the current map. Because of the semi-static map points segmentation and dynamic map points removal, the predicted map of our method is more accurate than other prediction methods. For the margin of the map points cloud, false and missed detection of our camera always occurs. Because our method considers the false and missed detection and eliminates their influence, the prediction accuracy is better than other methods.

F. Localization Performance

For real-world long-term deployment, estimation failures or wrong poses are severe, and they may occur commonly because of the scene changes. Therefore, inspired by [20], we set Correctness $c^{\varepsilon,\phi}(p_i)$ and Correct Rate (CR) to evaluate the correctness and robustness of our system. For each pose estimate p_i at time t_i, we obtain the correctness of the estimate by its ATE and absolute orientation error(AOE): if ATE$(p_i) \leq \varepsilon$ and AOE$(p_i) \leq \phi$, then $c^{\varepsilon,\phi}(p_i) = 1$, otherwise $c^{\varepsilon,\phi}(p_i) = 0$. For a sequence from t_{min} to t_{max}, and given an estimated trajectory $t_i, p_{i k=0,\ldots,N}$, define:

$$CR^{\varepsilon,\phi} = \frac{\sum_{k=0}^{N}(min(t_{i+1} - t_i, \delta) \cdot c^{\varepsilon,\phi}(p_i))}{t_{max} - t_{min}} \tag{8}$$

where $t_{N+1} \doteq t_{max}$, δ is a parameter to determine how long a correct pose estimation is valid. δ is larger than the time cycle of pose estimation and much smaller than the time span of the sequence. The Correct Rate is set to evaluate the robustness over the whole period of data. The results of localization performance of our system in different speed-level datasets are listed in Table I and Fig. 9. We make comparisons with some advanced frameworks in real-world datasets to verify the effectiveness of our system. Our proposed dynamic culling scheme performs better than other methods. It can better eliminate the influence of those dynamic map points. Our method also provides a more accurate global map that fits the current environment well, resulting in better performance in localization.

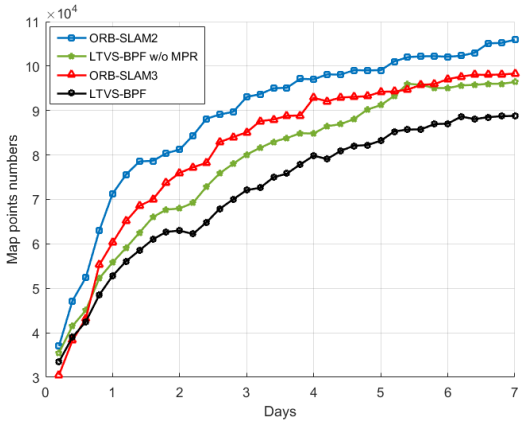

Fig. 11. The number of map points of the global map across days.

In order to evaluate those modules in the system, we conduct an ablation study to find which technology has played a role in improving localization accuracy. We conduct

different ablation studies: our method without map points removal (MPR) algorithm, without global map prediction (GMP), without occlusion detection (OD), and without 3D matching (3DM). From these results, we can see that our system (LTVS-BPF) with all those modules achieves the best localization performance. Every module contributes to the final performance, and without each module may decrease the localization accuracy. We can see that in medium-speed datasets, the dynamic map points module plays a more important role, and in low-speed datasets, the global map prediction module plays a more important role. In medium-speed datasets, there are more dynamic objects in the environment, and the moving speed of these dynamic objects is relatively great.

In Fig. 10 and Table V, there are the results of different systems' localization performance at different times after training datasets. In our experiments, the ATE threshold ε is set to 1 m, and the AOE threshold is 30°. The results in Table. V indicates that our system has better correctness and robustness in localization compared with static methods. When compared with other map prediction methods, our method has a more accurate predicted map. In our system, we consider the missed detection and false detection of our sensor using the persistence filter, which can provide us with accurate segmentation of semi-static map points and avoid modeling some static map points.

G. Memory Consumption

Experiments on the memory consumption of our system are also carried out. The comparison of different methods is shown in Fig. 11. The relationship between the number of map points and the days of the global map construction is nearly linear initially, and the difference between our method and other visual SLAM methods is unclear. However, the advantage of the proposed method is more evident with time growth. We can see that the number of map points of our systems increases more slowly than other systems with the daily running in the dynamic environments, thanks to the dynamic map points removal and the map points matching method. After the long-term operation of our robots, there is a sound reduction in map size. Therefore, our system is more suitable than the baseline method in long-term real-world environments.

VII. Conclusion

In this paper, we propose a novel long-term visual SLAM system with global map prediction and dynamics removal based on the Bayesian persistence filter. We demonstrate the utility of our system which overcomes difficulties caused by dynamic objects and periodic changes in long-term environments. With the persistence filter, we remove the dynamic map points in the global map. Using the proposed map points matching algorithm and survival analysis algorithm, we get the time series of map points cluster and obtain their survival function. Then we predict the future state of the semi-static map points cluster and obtain the predicted global map which fits the current environment. The proposed framework achieves good localization performance and memory consumption both in simulation and real-world experiments. The results indicate the effectiveness and robustness of our system in localization and mapping in long-term environments.

References

[1] S. Li and D. Lee, "RGB-D SLAM in dynamic environments using static point weighting," *IEEE Robot. Autom. Lett.*, 2017, pp. 2263-2270.

[2] B. Bescos, J. M. Fácil, J. Civera and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.*, 2018, pp. 4076-4083.

[3] M. S. Bahraini, M. Bozorg, and A. B. Rad, "SLAM in dynamic environments via ML-RANSAC," *Mechatronics*, vol. 49, pp. 105–118, 2018.

[4] G. Kim and A. Kim, "Remove, then revert: Static point cloud map construction using multiresolution range images," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 10 758–10 765.

[5] W. C. Dai, Y. Zhang, P. Li, Z. Fang, and S. Scherer, "RGB-D SLAM in dynamic environments using point correlations," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022, pp. 373-389.

[6] Z. Hashemifar and K. Dantu, "Practical persistence reasoning in visual SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 7307-7313.

[7] J. Huang, S. Yang, T. J. Mu and S. M. Hu, "ClusterVO: Clustering moving instances and estimating visual odometry for self and surroundings," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2165-2174.

[8] B. Song, W. Chen, J. Wang and H. Wang, "Long-term visual inertial SLAM based on time series map prediction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 5364-5369.

[9] T. Krajník, J. P. Fentanes, J. M. Santos, and T. Duckett, "FreMEn: Frequency map enhancement for long-term mobile robot autonomy in changing environments," *IEEE Trans. Robot.*, vol. 33, no. 4, pp. 964-977, Aug. 2017.

[10] N. Hawes et al., "The STRANDS project: long-term autonomy in everyday environments," *IEEE Robot. &Auton. Maga.*, vol. 24, no. 3, pp. 146-156, Sept. 2017.

[11] D. M. Rosen, J. Mason and J. J. Leonard, "Towards lifelong feature-based mapping in semi-static environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1063-1070.

[12] L. Tang, Y. Wang, X. Ding, H. Yin, R. Xiong, and S. Huan, "Topological local-metric framework for mobile robots navigation: a long term perspective," in *Autom. Robot.*, 2020, pp. 10 758–10 765.

[13] F. Nobre, C. Heckman, P. Ozog, R. W. Wolcott, and J. M. Walls, "Online probabilistic change detection in feature-based maps," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 3661–3668.

[14] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, 2017.

[15] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, 2021.

[16] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.

[17] C. Kerl, J. Sturm and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 3748-3754.

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

[19] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors", *arXiv preprint: 1901.03638*, 2019.

[20] X. Shi et al., "Are we ready for service robots? The OpenLORIS-Scene datasets for lifelong SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 3139-3145.