15
Dynamic and Deformable SLAM

Lukas Schmid, Jose Maria Martinez Montiel, Shoudong Huang,
Daniel Cremers, Jose Neira, and Javier Civera

Historically, SLAM systems have always faced the challenge of environments where not only the sensor but also the scene is in motion. To address this challenge (probably the first challenge that SLAM faced), data association tries to identify which sensor data correspond to entities that present a coherent change of pose with respect to the sensor location and which appear to independently change their pose. This is much in the same way that stars apparently move in unison in the sky, while planets wander, thus leading to their name[1].

Similarly, the methods described in this book so far have addressed the problem of estimating the robot pose and a representation of the environment based on observations during an exploratory trajectory. However, an important assumption was that the environment is *static*, meaning that the scene remains unchanged and only the robot is moving. This assumption has several important implications. For example, Chapter 1 showed that if detected features remain in the same place, re-detecting them can directly be used to better constrain the robot pose. Conversely, Chapter 5 demonstrated how dense scene models can efficiently be reconstructed by *fusing* several posed observations if the underlying environment remains the same.

Nevertheless, real-world scenarios are often highly *dynamic*, thus violating the rigidity assumption. For example, imagine what information an autonomous vehicle requires to drive safely through a city, where the environment includes not only a static background but also a variety of moving elements such as other vehicles, cyclists, and pedestrians. Or consider a home or service robot, mapping an environment where humans are frequently moving around the robot and impart changes on the scene, such as modifying objects or re-arranging a room. Or the even more extreme case of a surgical robot navigating through the human body with no static background at all and the entire scene is constantly deforming. Such factors make both *state estimation* of the robot and *representation* of the environment challenging problems in *dynamic* and *deformable* scenes.

To address these limitations, this chapter analyses the challenges of *dynamic and deformable SLAM*. We first define and categorize different kinds of dynamic effects

[1] The Greek term $\pi\lambda\alpha\nu\eta\tau\eta\varsigma$ (planētēs) means wanderer.

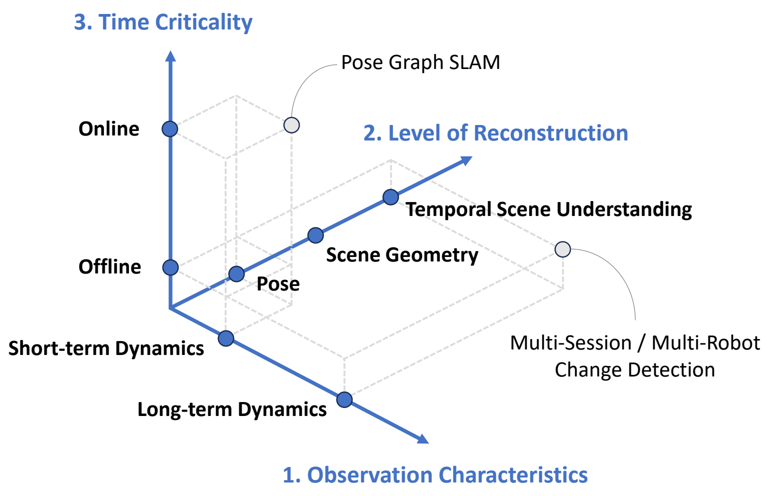
3. Time Criticality
Pose Graph SLAM
Online
2. Level of Reconstruction
Temporal Scene Understanding
Scene Geometry
Offline
Pose
Short-term Dynamics
Multi-Session / Multi-Robot Change Detection
Long-term Dynamics
1. Observation Characteristics

Figure 15.1 Overview of main considerations for dynamic SLAM: 1) which dynamic effects are present in the observation of the robot, 2) at which level of detail is the robot is rquired to understand the environment to accomplish its tasks, and 3) when and how fast does the robot need process this information. For example, typical pose graph SLAM can estimate the robot pose in real-time by rejecting dynamic parts as outliers. On the other hand, multi-session change detection aims to build a detailed understanding of the evolution of a scene from irregular observations (sessions). Note that while some solution approaches can be well characterized in the space spanned by these axes, others may fill in an 'area' or 'volume' by addressing several aspects, such as both short and long-term dynamics, simultaneous state-estimation and scene representation, some components running online and some offline, as well as combinations across axes.

and objectives of dynamic SLAM in Section 15.1. We then introduce prevalent problems and current solution approaches for short and long-term dynamic SLAM in Section 15.2 and Section 15.3, respectively. We address the deformable SLAM case in Section 15.4. Finally, Section 15.5 revisits remaining challenges and outlines directions for future research in the rapidly expanding field of *dynamic and deformable SLAM*.

15.1 Characterizing the Dynamic SLAM Problem

Central considerations when modeling dynamic environments are 1) which dynamic effects are present in the observation of the robot, 2) at which level of detail the robot is rquired to understand the environment to accomplish its tasks, and 3) when and how fast the robot needs process this information. While, ideally, a generalist robot will be able to build a rich description of any dynamic scene in real-time, oftentimes, it may be sufficient and more efficient to address only the subset of problems that is relevant to a given application. An overview of these directions is shown in Figure 15.1, and each axis is further detailed below.

15.1.1 Characterizing Dynamics

Our approach to characterizing dynamics is based on how they manifest in the *observations of the robot.* We follow the definition of [700] and distinguish between *short-term* and *long-term* dynamics, referring to more *transient* and more *abrupt* characteristics in the *observations* of the robot, respectively. The key idea is that, fundamentally, almost all physical processes are continuous at short enough time scales. How different the observations of the robot are, and therefore whether the change characteristics are those of short or long-term dynamics, thus does not depend on the absolute duration of a phenomenon, but on the *relation* between the rate of change in the scene and the rate of observation by the robot. This relation is demonstrated in Figure 15.2, including several examples to illustrate how this applies to a broad range of time scales.

While the above definition is general and applies to vast time scales, it is important to point out that in most robotic applications, the relevant rate of change (humans or objects move typically at speeds $\sim 10^{-1}$ to 10^{2}m/s) is of the same order of magnitude as the rate of observation (most robotics sensors have a rate of $\sim 10^{0}$ to 10^{3}Hz). In this case, the distinction often simplifies to whether the robot is *currently observing* the motion or not. Then, *short-term* dynamics refer to all motion that is happening *within view* of the robot, where continuous observation of dynamic entities will result in *transient* change characteristics. For example, consider the smooth trajectory of a continuously observed rolling ball or the coherent motion of a human walking around. We will discuss techniques to address *short-term* dynamics in Section 15.2.

On the other hand, *long-term* dynamics refer to dynamics happening *outside* the view of the robot. While the motion is still continuous locally, the robot will only observe the result of this motion that accumulated in between measurements, which is therefore typically of a more *abrupt* nature. For example, consider the case where a person picks up a bottle and places it again next to its initial position. If the robot only observes the table with the bottle before and after the person interacts with it, although the true motion was continuous, from the perspective of the robot, it will look like the bottle teleported between observations. An extreme example of this is the case of multi-session or multi-robot mapping, where the scene may be static during individual visits, but changes tend to accumulate between visits when the robot is not observing the scene.

As a result, an important implication of this view is that the difference between short and long-term dynamics depends on the *observation* of the robot and is not an inherent property of the environment or moving object, illustrated in Figure 15.3. However, the probability of long-term dynamics occurring is oftentimes corre-

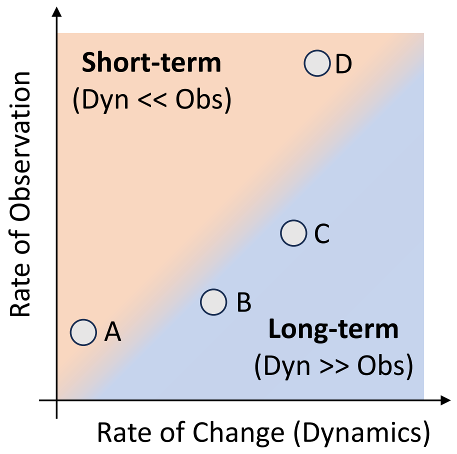

Figure 15.2 Definition of short vs. long-term dynamics: the dynamic characteristics apparent to a robot are given by the relation between rate of change in the scene and the rate of observation by the robot, not by absolute time spans. Note that in most robotics domains, where the sensing rate is sufficient to capture continuous motion when in view, this simplifies to objects moving *within view* of the robot (short-term) vs. changing *outside* the view of the robot (long-term). Examples: A) Daily measurements of plant growth. Although the rate of observation is very low, the plants also only grow a small amount. As a result, appropriate methodologies to track the plants are similar to those for tracking people [181]. B) Daily patroling of a building. Although the rate of observation is comparable, parts of the scene will have changed completely and long-term methods such as mutli-session change detection are appropriate [573]. C) Object rearrangement behind the back of the robot. Although the robot may only be gone for a minute, the object configuration may have changed substantially between observations [981]. D) The Hawk-Eye System can track Tennis balls moving at up to 250 km/h by observing them at 340 Hz, leading to short-term dynamic observations and milimeter tracking accuracy [682].

lated with the time between observations. We will overview methods for *long-term* dynamics in section 15.3.

Finally, the cases discussed above typically assume that there is *some static background*, with various dynamic entities in the scene. However, in some relevant use cases like endoscopic surgery and medical robotics, *all of the scene* is deforming and moving around the robot, typically within the category described above as *short-term* dynamics. This setting where there is *no static* part of the scene is commonly referred to as *deformable SLAM*, which we discuss in Section 15.4.

15.1.2 Terminology

It is important to point out that in the community, a variety of further terms are used to describe related concepts to the ones introduced above. Instead of *short-*

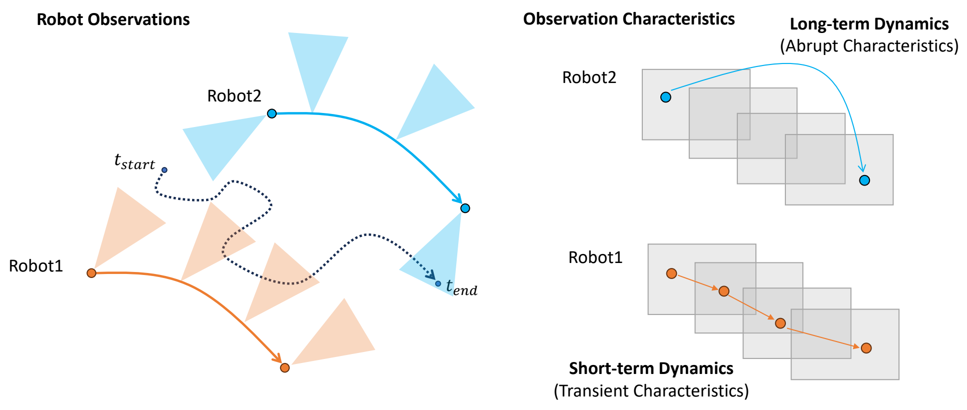

Figure 15.3 Role of the robot as observer. The same physical motion of an object (dotted line) can reflect as *short-term* dynamic for Robot1 and *long-term* dynamic for Robot2. Similarly, multi-session or multi-robot mapping can include cases where objects are moving while being observed (short-term dynamics), or changed outside of the view of the robot, most typically in between sessions or visits (long-term dynamics).

term dynamics, used terms include *dynamic* and *high-dynamic*, typically referring to objects that are currently moving (potentially within view of the robot).

On the other hand, to refer to *long-term* dynamic effects, terms including *semi-static*, *quasi-static*, or *low-dynamic* are used. These typically refer to objects that could move but are static while being observed, or changes that occur between multiple visits, which is a special case of long-term dynamics.

Beyond dealing with dynamic effects in 3D SLAM, certain methods aim to reconstruct a history, evolution, or higher-level understanding of the scene and its dynamics. This is often referred to as *spatio-temporal* or *4D* (being 3D space + time[2]) scene models.

In addition, the terms *lifelong*, *persistent*, and *continuous* are sometimes used to refer to SLAM, mapping or localization systems that can address variable environments, often through operation over long time spans. Finally, note that not only geometric dynamics affect sensor readings, but also texture, illumination, and other changes (such as day/night or seasonal changes), in particular from visual sensors [1167, 1014].

15.1.3 Degrees of Dynamism

While the above characterization may seem to imply a clear classification into three cases, it is important to note that in practice the boundaries can be fuzzy,

[2] Note that this is different from the 4D or 3+1D used in radar SLAM (Chapter 9), where the fourth dimension refers to velocity measurements by the radar.

numerous different effects can combine, and each effect can be present in different intensities. For example, a scene may contain a single rigid moving object (*e.g.,* a car or box [698]), a single moving and deforming object (*e.g.,* a person [457]), only long-term changes (*e.g.,* regular patrolling of a building [329]), up to many moving and changing entities at the same time (*e.g.,* robots with humans in a home [981]), or even no static entities at all (*e.g.,* endoscopic surgery [900]). While, ideally, a generalist robot will be able to estimate all of these effects jointly, the problem becomes exponentially more complex the more effects are considered. Thus, prior knowledge about which dynamic effects can be expected or are relevant for the task at hand can enable the design of more specialized and thus better performing and more tractable solutions for a given scenario.

15.1.4 State Estimation vs. Scene Representation

As introduced in Chapter 5, SLAM is the dual process of estimating the *robot pose* as well as a *representation* of the environment. The same considerations apply to SLAM in dynamic environments. If only estimating the robot state is sufficient for an application, it is oftentimes adequate to treat dynamic observations as outliers or noise. In this case, similar to SLAM in static scenes, accurate results can be achieved by ignoring or rejecting the corresponding features or measurements and tracking or localizing with respect to the static background. We will discuss considerations to this effect, tailored to short-term and long-term dynamics, in Section 15.2.1 and Section 15.3.1, respectively. A special case is the setting of deformable SLAM, where there are *no static* parts to localize against. We will address this setting in Section 15.4.

Conversely, if a dense map is desired, reconstructing a moving and changing scene requires that the robot can *detect and represent* some or all motion and changes in the scene. We will discuss representations and techniques for short-term dynamics in Section 15.2.3, long-term dynamics in Section 15.3.2, and first unified dynamic SLAM methods in 15.3.3. Beyond reconstruction, we will briefly introduce methods for *higher-level* understanding of dynamic and changing scenes in Section 15.3.4.

15.1.5 Online vs. Offline Methods

The final axis we consider is the time criticality of a perception system, meaning whether dynamics must be handled in real-time or whether it suffices to process the data offline. Since most robotic applications require that robots can interact with their environment immediately, especially if the scene is dynamic, this chapter primarily focuses on real-time methods. However, in certain applications such as regular monitoring of a building, it may suffice to collect the sensor data and then process it offline. Similar to (offline) global bundle adjustment vs. (online)

SLAM, this has the advantage of additional computational resources, which facilitates processing at higher resolutions and generally can achieve more accurate results. An additional notable difference to online methods is the fact that all data is already available. Specifically, to detect moving or changed objects in frame F_t at time t, all future measurements $F_{t+1,\ldots,T}$ are already available which can provide essential information not available to online methods. While this chapter focus primarily on online methods, we briefly introduce the relevant offline problems of map cleaning and change detection in Section 15.3.2 and Non-Rigid Structure from Motion (NRSfM) in Section 15.4.1.

Finally, while part two of the book was structured by sensing modality, the implications of dynamic scenes for SLAM are often similar for many sensing modalities. We therefore structure this chapter according to different dynamics characteristics. Nonetheless, each section will discuss considerations for different sensing modalities, where so far predominantly visual, LiDAR, and proprioceptive sensors have found notable use for dynamic and deformable SLAM.

15.2 Dynamic SLAM

In the following sections we further discuss the implications of individual short-term dynamic objects for *SLAM*. Historically, first approaches used data association to identify measurments that can be explained by the sensor moving and which ones appear to independently change their pose. Initial techniques included statistical tests [803], as well as classical algorithms such as RANSAC [230] and the Hough transform [1077]. These methods allowed simply discarding measurements of dynamic objects as outliers. Later, efforts turned to not simply ignoring dynamic objects, but also trying to reconstruct the environment regions occluded by them, either by interpolating missing data [1075], recovering data from alternative sensor poses where no occlusion occurred [82], or by image 'inpainting' using machine learning [84]. The result is much more useful for localization because there are no 'holes' in the rendered map. We will address dynamic objects as outliers in Section 15.2.1.

An alternative also pursued is to track the dynamic objects as part of the SLAM framework. In computer vision, this problem is known as multi-body structure from motion, an extension to classical SFM. The mathematical foundations for the solution, as well as practical algorithms have much in common with the SLAM pespective [841]. This was initially deemed unfeasible due to high computational cost [1149]. Recently, this has become feasible [83], and it might be necessary in order for the robot to avoid collisions with dynamic objects. We discuss dynamic object tracking in Section 15.2.2.

To provide an even more detailed understanding of dynamic scenes, one can further aim to reconstruct the scene in a dense manner, including the dynamic

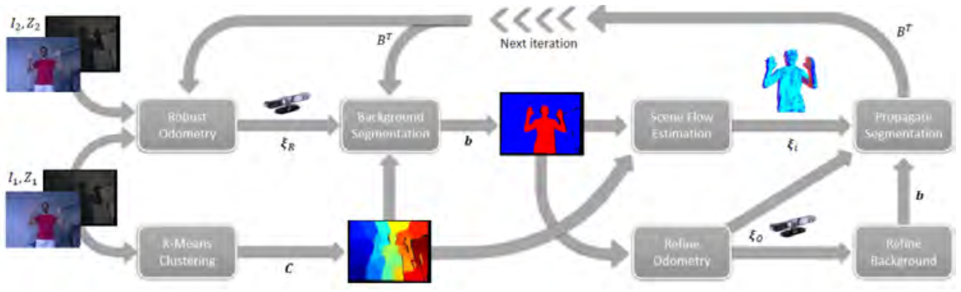

Figure 15.4 In [517] a method is introduced that jointly estimates the camera motion and the 3D scene flow for moving objects from an RGB-D video sequence. Initial robust odometry is used to segment the clustered frame into dynamic and static (background) based on the residuals of each cluster. The segmentation can then be used to further refine the odometry. (©2017 IEEE)

objects. Having access to detailed movin object models can be useful for applications such as manipulation, where the robot needs to understand the object shape and motion in order to interact with it. We discuss dense dynamic SLAM in Section 15.2.3.

15.2.1 Dynamic Object Removal

One of the simplest, but still reasonable approaches to SLAM in partially dynamic scenes consists in discarding sensor measurements of the dynamic parts of the scene and removing such parts from the map states. The rationale is twofold. First, persistent objects and landmarks are oftentimes the main focus of mapping, dynamic entities being in many occasions of limited interest (such as pedestrians passing by). Second, scene rigidity is assumed by most geometric estimation techniques, and hence dynamic parts can be detected and classified as spurious measurements to rigid models. To this end, a large variety of approaches has been proposed. We summarize the main families of methods below. Note that modern dynamic SLAM methods oftentimes employ combinations of these principles for best performance.

Proprioceptive Sensing. An essential property of proprioceptive sensing is that, by definition, it does not rely on observations external to the robot. For this reason, it can also not be influenced by external effects, such as motion in the scene. Since inertial and internal odometric measurements carry information of the robot states they help disambiguating the motion corresponding to the robot and that corresponding to dynamic parts of the scene from relative measurements [1030]. We refer the reader to Chapter 11 and Chapter 12 for details on how to fuse inertial or kinematic measurements with exteroceptive ones, respectively.

Robust Estimation. Most scenes and most sensors provide a set of measurements that is significantly larger than the minimal set for geometric estimation.

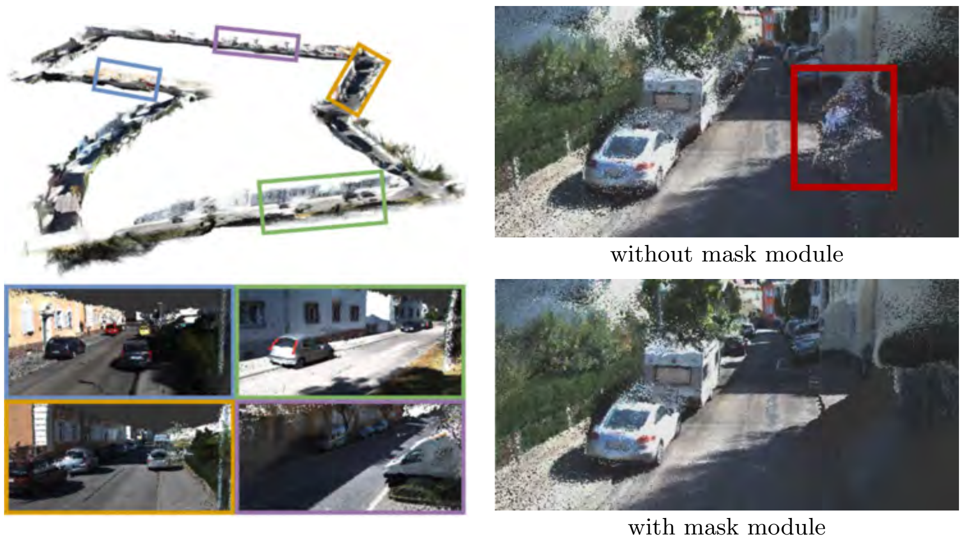

Figure 15.5 In MonoRec [1184] a deep network is trained to recover a dense reconstruction from a single moving camera. Moving objects are filtered out by a *mask module* that taps into a brightness consistency cost volume to identify structures (colors across subsequent frames) that are not compatible with the dominant ego motion. (©2021 IEEE)

In these cases, and for low rates of dynamic content, redundancy can be used to identify measurements of dynamic objects as outliers to rigid geometric models. Two methods are mainly used for that. First, RANSAC [336] is commonly used to initialize the state estimates by selecting a minimal sample set with the largest consensus among the whole measurement set. However, this may fail in cases of large occlusions where the largest consensus set may be that of a moving object. Second, after initialization and during iterative optimization, an alternative to the typical L2 norm is to use robust cost functions. These have sub-quadratic growth for large residuals, which are assumed to be outliers, and hence their influence on the states is reduced. For further details on RANSAC and robust losses, we refer the reader to Chapter 3.

Multi-View Motion Cues. In addition to discarding dynamic content at initialization and reducing or even canceling its influence by robust cost functions, another common approach is removing dynamic states once there is sufficient evidence of non-rigidity from its measurements. Evidence is typically implemented as a threshold on the accumulated number of high-residual measurements. Figure 15.4 shows an example of this where RGB-D images are clustered into regions and each region is classified between dynamic and static (background) based on the residuals for that region.

Semantic Information. Deep neural networks for semantic segmentation can be used to pre-segment image areas that correspond to potentially dynamic objects. While this may certainly help, closed-set vocabularies may be unable to segment all the objects in an image. Additionally, objects being potentially dynamic does not imply that they are moving in a particular moment, such as parked cars. For this reason, combining semantic segmentation of potentially dynamic objects with multi-view geometric checks may be convenient [51]. For example, the method MonoRec [1184] advocates a deep network for dense reconstruction from a single moving camera that combines a brightness cost volume computed from a set of consecutive warped frames with a mask module that is trained to filter out moving objects. These objects are determined from training data, but also from the color inconsistency across the warped frames, shown in Figure 15.5.

Domain Knowledge. Where available, additional prior and domain knowledge can be integrated to disambiguate features of the static scene and dynamic outliers. A common example of this is a no-side-slip constraint for autonomous vehicles. Since we know that a car cannot move sideways, all features that suggest a side-ways motion can be removed as dynamic outliers.

Motion-aware Sensors. Visual and LiDAR sensors typically provide a capture of the scene at a given time, and thus do not provide direct measurements of the motion of dynamic objects. Other sensors, such as event cameras provide a continuous stream of measurements at high temporal resolution. Furthermore, radars and certain LiDARs use the Doppler effect to directly measure the velocity of each observed point. While dynamic SLAM methods using these sensors are actively being investigated, access to such information could be an important cue to identify dynamic objects and their motion. For more details on event cameras and radars, we refer the reader to Chapter 10 and Chapter 9, respectively.

15.2.2 Dynamic Object Tracking

Dynamic object removal, as detailed in the previous section, might be convenient if the goal is only to estimate the robot state. However, in many other cases, it may be necessary to track the 3D motion of these dynamic entities, for example in order to navigate without collisions. Note that this problem is highly related or sometimes also referred to as Multi-Object Tracking (MOT) or Multi-Instance Dynamic SLAM (MID). The problem of 2D and 3D object tracking has accrued large interest in the computer vision and robotics communities [1241]. As a result, the body of literature is vast and cannot be exhaustively covered in this chapter. We here thus focus on selected techniques most relevant to SLAM.

The joint problem of simultaneous localization, mapping and dynamic object tracking can be formulated as follows. The central idea is that each dynamic object in the scene can be modeled as a *rigid moving body.* The factor-graph representation of the static SLAM problem presented in Chapter 7 can therefore be extended to account for individually moving objects, shown in Figure 15.6. As before, most approaches follow a parallel-tracking-and-mapping approach. For each incoming

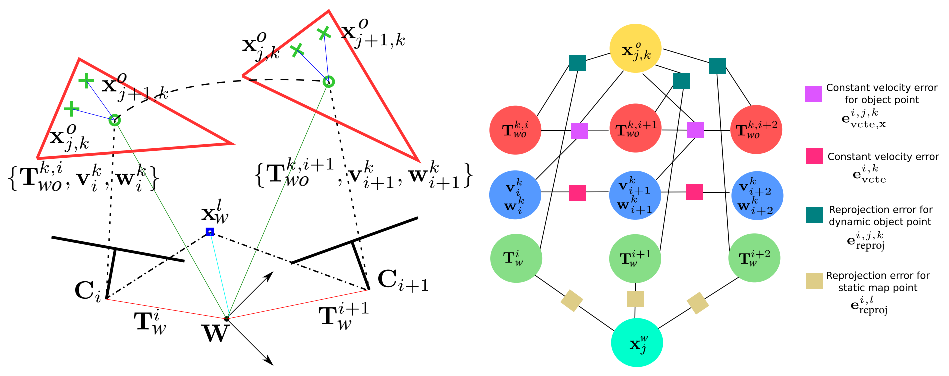
$\mathbf{x}^o_{j,k}$
$\mathbf{x}^o_{j+1,k}$
$\{\mathbf{T}^{k,i}_{wo}, \mathbf{v}^k_i, \mathbf{w}^k_i\}$
$\{\mathbf{T}^{k,i+1}_{wo}, \mathbf{v}^k_{i+1}, \mathbf{w}^k_{i+1}\}$
$\mathbf{x}^l_W$
$\mathbf{C}_i$
$\mathbf{C}_{i+1}$
$\mathbf{T}^i_w$
$\mathbf{T}^{i+1}_w$
$\mathbf{W}$
$\mathbf{T}^{k,i}_{wo}$
$\mathbf{T}^{k,i+1}_{wo}$
$\mathbf{T}^{k,i+2}_{wo}$
$\mathbf{v}^k_i$ $\mathbf{w}^k_i$
$\mathbf{v}^k_{i+1}$ $\mathbf{w}^k_{i+1}$
$\mathbf{v}^k_{i+2}$ $\mathbf{w}^k_{i+2}$
$\mathbf{T}^i_w$
$\mathbf{T}^{i+1}_w$
$\mathbf{T}^{i+2}_w$
$\mathbf{x}^w_j$
Constant velocity error for object point $\mathbf{e}^{i,j,k}_{\text{vcte,x}}$
Constant velocity error $\mathbf{e}^{i,k}_{\text{vcte}}$
Reprojection error for dynamic object point $\mathbf{e}^{i,j,k}_{\text{reproj}}$
Reprojection error for static map point $\mathbf{e}^{i,l}_{\text{reproj}}$

Figure 15.6 Overview of SLAM with dynamic object tracking [83]. Left: rigid body motion assumption. The camera pose $\boldsymbol{T}^i_w$ is estimated with respect to a static background, while the dynamic object k is modeled as a rigid body with respect to the world frame $\mathcal{F}^w$. Note that all feature points $\boldsymbol{x}^{j,k}_o$ on each object k adhere to its rigid body motion $\Delta\boldsymbol{T}^k_{i,i+1}$. Right: resulting factor graph representation of the problem. The camera poses are connected through odometry factors, while the dynamic object poses are connected through constant velocity factors. (©2021 IEEE)

frame, the tracking thread extracts salient features (typically ORB [958]). To assign points to individual objects, oftentimes semantic segmentation masks are extracted.

Let $\boldsymbol{T}^i_w \in \mathrm{SE}(3)$ be the rigid transformation representing the pose of the camera at time t_i in a world reference frame $\mathcal{F}^w$ and $\boldsymbol{x}^w_j \in \mathbb{R}^3$ the coordinates corresponding to rigid map points j. Each dynamic object k is represented, at time t_i, by its six DoF transformation $\boldsymbol{T}^{k,i}_{wo} \in \mathrm{SE}(3)$ of its object frame $\mathcal{F}^o$ with respect to the world, and linear and angular velocities $\boldsymbol{v}^w_{k,i}, \boldsymbol{\omega}^w_{k,i} \in \mathbb{R}^3$ in the object frame $\mathcal{F}^o$. Map points on dynamic objects are represented in the local object frame as $\boldsymbol{x}^o_{j,k} \in \mathbb{R}^3$. This results in the modified reprojection error corresponding to $\boldsymbol{x}^o_{j,k}$

$$\boldsymbol{e}^{i,j,k}_{\text{reproj}} = \boldsymbol{z}_{ij} - \pi\left(\boldsymbol{T}^{i\;-1}_w \boldsymbol{T}^{k,i}_{wo} \bar{\boldsymbol{x}}^o_{j,k}\right), \tag{15.1}$$

where we used $\bar{\boldsymbol{x}}^o_{j,k} \in \mathbb{R}^4$ as the homogeneous coordinates of point $\boldsymbol{x}^w_j$, and overloaded the projection function $\pi(\cdot)$ for points in homogeneous coordinates.

To further constrain the motion of each dynamic object k through time, a constant velocity model for linear and angular veolicty is frequently assumed between consecutive observations of dynamic objects. This can be formalized as the following error term to minimize

$$\boldsymbol{e}^{i,j,k}_{\text{vcte}} = \begin{bmatrix} \boldsymbol{v}^w_{k,i+1} - \boldsymbol{v}^w_{k,i} \\ \boldsymbol{\omega}^w_{k,i+1} - \boldsymbol{\omega}^w_{k,i} \end{bmatrix} \tag{15.2}$$

In order to couple object velocities and poses, the following error term is added

$$\boldsymbol{e}^{i,j,k}_{\text{vcte},\boldsymbol{x}} = \left(\boldsymbol{T}^{k,i+1}_{wo} - \boldsymbol{T}^{k,i}_{wo}\Delta\boldsymbol{T}^k_{i,i+1}\right)\bar{\boldsymbol{x}}^o_{j,k} \tag{15.3}$$

where the increment in the pose from t_i to t_{i+1} is computed from the constant velocity model as

$$\Delta \boldsymbol{T}^k_{i,i+1} = \begin{bmatrix} \mathrm{Exp}\left(\boldsymbol{\omega}^w_{k,i}\,(t_{i+1} - t_i)\right) & \boldsymbol{v}^w_{k,i}\,(t_{i+1} - t_i) \\ \mathbf{0} & 1 \end{bmatrix} \tag{15.4}$$

Figure 15.6 illustrates these modeling assumptions (left) and the resulting factor graph (right) to optimize.

This general formulation has found notable success in a number of works. For example, earlier methods such as Multimotion Visual Odometry (MVO) [534] detect tracklets and segment them into clusters such that the rigid-body-motion constraint is satisfied for each cluster. Visual Dynamic Object-aware SLAM (VDO-SLAM) [1265], Dynamic SLAM [456], and DynaSLAM II [83] introduce the factor-graph-based approach and add motion consistency factors between observations. This has been further generalized in AirDOS [904] to capture *articulated* objects consisting of several connected rigid bodies such as human skeletons.

15.2.3 Dense Dynamic SLAM

Similar to *static* dense SLAM, discussed in Chapter 5, *dynamic* dense SLAM aims to additionally estimate a dense reconstruction of the scene, however, now including dynamic entities. To achieve this, most approaches separate the process into a *localization* and a *mapping* step.

The localization step aims to track the robot pose with respect to the static background, which can be achieved using any of the sparse dynamic SLAM methods discussed in dynamic object removal (Section 15.2.1) and MOT (Section 15.2.2). Alternatively, also other techniques such as dense tracking applied to the background once dynamic points in the input have been removed can be employed.

In the mapping step, the goal is to build a representation of both the static background and the dynamic entities. An essential design decision is the choice of representation, as this defines which quantities need to be estimated to perform continuous updates of the representation through time. Naturally, reconstructing the background is identical to static dense mapping, however, the same or similar representations can also be used to model dynamic entities. Notice that the representation of the background and dynamic entities do not have to be the same, although in practice this is oftentimes the case for ease of interpretation and processing of the robot map. In this section, we discuss common groups of approaches, starting from Moving Object Segmentation (MOS) and sensor-level representations in Section 15.2.3.1, to parametric representations if the target object or category is known in Section 15.2.3.2, to reconstructing arbitrary moving objects in Section 15.2.3.3.

15.2.3.1 Moving Object Segmentation

A minimal solution to address dense dynamic SLAM is to detect all dynamic parts, oftentimes all points or pixels, in the input data. This problem is often called Moving Object Segmentation (MOS). Similar to dynamic object removal (Section 15.2.1), once these points are detected, they can be ignored to avoid inconsistency and artifacts in the static background reconstruction, or alternatively, be stored as low-level representation of dynamic entities in the scene. To detect dynamic or inconsistent points, similar ideas as before can be applied. However, a central difference is that for dense SLAM methods, also the dynamic point detection needs to be dense, meaning one needs to classify *each* point in the sensor data and not just selected features, as illustrated in Figure 15.7. Therefore, the majority of approaches rely on using geometric consistency and/or semantic information as main motion cues for dense segmentation. In the former, registration residuals can be densely computed during camera-to-model tracking to classify areas of the input image with high residuals as dynamic [989], or one can use the consistency between individual measurements [1249] or measurements fused in the map [980] to identify inconsistent and thus dynamic points. Oftentimes, these detections are complemented with additional clustering steps to deal with noise in the sensing data [989, 1249, 980]. In the latter case, dense semantic segmentation can provide useful information on which parts of the input data reflect classes or instances that are potentially dynamic, such as people, that can be masked out [946]. While semantic segmentation of camera images and LiDAR point clouds has been widely studied and can easily be re-purposed to mask out dynamic classes, classifying dynamic pixels or points directly using deep learning methods on individual or sequences of measurements has become a growing field of research [878, 765, 670]. The learning-based approach has the advantage of being able to incorporate priors about typical moving object shapes, motion patterns, and sensor characteristics which can improve performance in the target domain, but may not generalize to out-of-domain settings.

15.2.3.2 Parametric Models

While discarding dynamic measurements and reconstructing the static background may suffice for certain applications, others, such as manipulation or human-robot-interaction, may require robots to also estimate detailed representations of the dynamic entities in the scene. In these cases, the tracking-by-detection methods of Section 15.2.3.1 have limited expressiveness since they operate on the level of noisy and partial measurements of dynamic regions in each frame.

If the type of objects of interest is known, their shape can oftentimes be represented using a *parametric* model. Given this structure, each measurement can be treated as an observation of the true underlying parameters to optimize for them. Similar to MOT (Section 15.2.2), these parameters can be added as variables to a factor graph with model-specific observation factors and estimated jointly with the

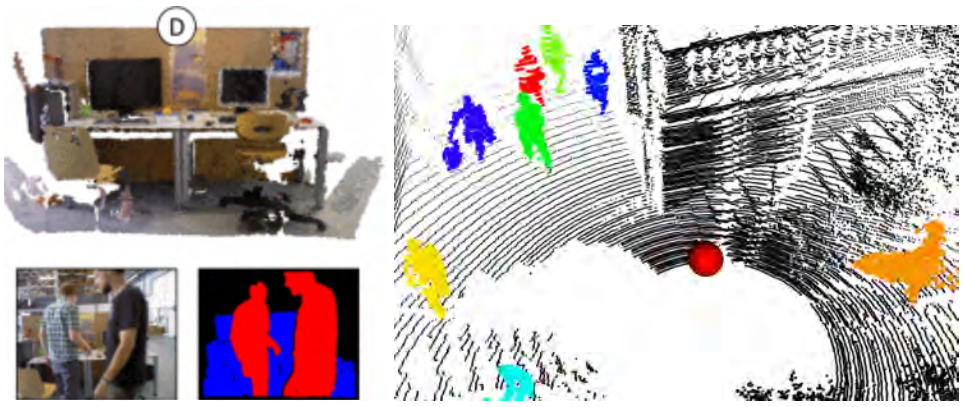

Figure 15.7 Dense moving object segmentation (MOS). Left: StaticFusion [989] reconstructs the static background (top) from RGB-D images by segmenting each frame (bottom) into dynamic (red) and static (blue) parts based on tracking residuals. Right: Dynablox [980] densely segments LiDAR scans into dynamic (color) and static (black) points using volumetric map consistency. ([989] ©2018 IEEE, [980] ©2023 IEEE)

sensor motion and background. This approach has found particular interest in capturing and tracking human motion, where humans can be represented as skeleton models [904] or dense SMPL [701] meshes [948, 457, 458], among many others.

15.2.3.3 Simultaneous Tracking and Reconstruction

Alternatively, the area of simultaneous tracking and reconstruction aims to reconstruct arbitrary moving objects. In most cases, one can make the simplifying assumption that each moving object is a *rigid body*. The goal is then to track and reconstruct a single or multiple *rigid moving objects*. The central idea is to create a separate dense representation of each object as well as the background. To achieve this, most approaches follow four main steps [967, 963, 1203, 933]. First, object detection is performed in the input images, typically combining semantic or instance segmentation with geometric refinement and/or geometric motion cues. Second, the camera motion is tracked with respect to the background. Third, each moving object is tracked with respect to the current image. For steps two and three, several techniques are admissible, where typically *direct methods* that minimize the projected depth and photometric residuals of each object are used, since these tracking methods can directly operate on the dense representations of each object or the background. Finally, the measurements of each object and the background are fused into the respective models to incrementally estimate a dense reconstruction of each object. Most prominently, objects are represented as TSDF volumes (for more details on TSDF see Section 5.2.2) for their capacity to fuse measurements with sensing noise [967, 963, 1203, 933]. An example of this is shown in Figure 15.8.

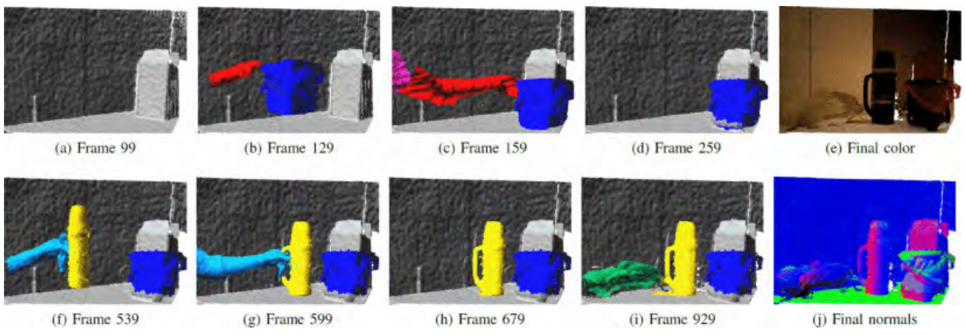

Figure 15.8 Simultaneous tracking and reconstruction with Co-Fusion [967]. Three objects were sequentially placed on a table: First a small bin (blue label), a flask (yellow) and a teddy bear (green). The results show that all objects were successfully segmented, tracked and modeled. (©2017 IEEE) TODO: Reuse permission

Alternatively, DirectTracker [394] uses a sparser point-cloud-based reconstruction to jointly estimating the object grouping and respective rigid body motions for each object.

The simultaneous tracking and reconstruction approach has the large advantage of being more general as no prior object models are needed. However, the many steps required can be computationally expensive, especially for large scenes and in the presence of many dynamic objects. A further limitation is the fact that errors in object tracking can lead to misaligned measurements being fused into the object representations, which can degrade the reconstruction and further inhibit accurate tracking in the future, where, for example, key-point-based tracking methods such as BundleTrack [1176] can be more robust.

The assumption of objects being rigid can further be relaxed to articulated objects by estimating transforms for each rigid part connected to a complete object [982], or to general deformable objects. Examples of the latter include DynamicFusion [808] and KillingFusion[1022], which will be covered in more detail in Section 15.4. An example of a learning-based approach is AnyCam [1186] shown in Figure 15.9, which makes use of a transformer-based neural network in order to jointly estimate the camera motion and a 4D reconstruction of a dynamic scene with non-rigidly moving objects from a casual monocular video.

Finally, the additional reconstructed objects could also be used to improve the tracking of the robot's ego-motion. However, this only appears to improve the robot state estimates in corner cases, where tracking against the background is insufficient, for example when large parts of the robot's view are occluded (>70%) and good motion models for the moving objects are available [698].

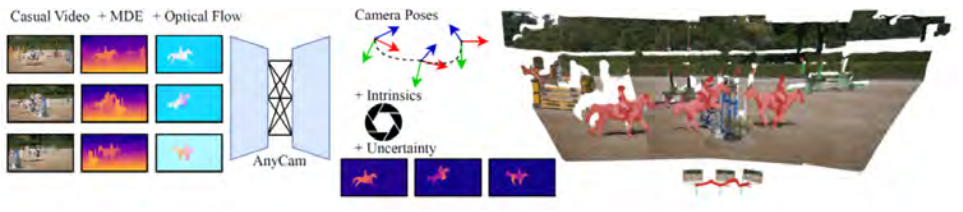

Figure 15.9 AnyCam [1186] is a transformer-based neural network that jointly estimates the camera motion and a 4D non-rigid reconstruction from a casual input video. (©2025 IEEE)

15.3 Long-term and Changing SLAM

The previous section addressed considerations to handle *short-term* dynamic observations. In this section, we will address the challenges and implications of *long-term* dynamics on SLAM.

In general, there are several notable challenges for long-term dynamic SLAM. First, in contrast to short-term dynamics, the observed changes patterns are more abrupt, and can in principle grow *arbitrarily large* (especially the longer a robot does not observe a space the more likely it is to be ever more different from when the robot last observed it). This makes *data association* significantly more difficult. For example, whereas short-term motion is, by definition, small between observations and data association can be addressed through tracking, data association across long-term dynamics may require solving a potentially global re-identification and matching problem. Furthermore *perceptual aliasing* is exacerbated, as, for example, a unique landmark (or several if they are feature points on a single object) can be observed in several positions because it has moved.

Second, because the magnitude of changes is larger, many effects that can potentially be neglected in the short-term case become significant in the long-term case. This includes geometric changes, from motion (such as objects being added, moved, or removed from the scene [979]) to structural changes (for example, consider a construction site that will look fundamentally different at each visit [1062]), as well as appearance changes from illumination (such as from different weather, time of day, or seasons [1167]) to changes in texture or reflectivity of surfaces.

Finally, because long-term dynamic observations typically coincide with longer elapsed times and farther robot motion, large changes in view point as well as potential sensor degradation or change between observations can occur.

To address these challenges, we will first briefly consider implications for state estimation in Section 15.3.1, followed by offline approaches for persistent scene representation in Section 15.3.2. Extensions thereof for online SLAM and unified short and long-term dynamic SLAM will be covered in Section 15.3.3. Finally, we will introduce temporal scene understanding models in 15.3.4, which aim to go beyond capturing spatial variations to reconstructing temporal patterns.

15.3.1 Lifelong SLAM

A central capacity of lifelong SLAM systems is the ability to accurately localize the robot in spite of the challenges pointed out above. This problem is conceptually very similar to the *place recognition* and *loop closure detection* problems, which addresses the related problem of identifying whether two views are of the same place in spite of long-term changes. In this section, we expand these ideas for continuous long-term localization. While a range of approaches exist, we will summarize only a few of the most relevant ones.

15.3.1.1 Parallel and Summary Maps

A first group of approaches to this problem, similar to the case of dynamic object removal in Section 15.2.1, leverages the fact that the robot pose can often be estimated from a subset of inlier points that are matched. However, instead of removing points as outliers, the idea here is to keep adding additional feature points to the map if their appearance has changed to the extent that they can no longer be matched to their previous observations. Typically, individual visits to a scene are referred to as *experiences* or *sessions*, where the changes within each experience are small enough to enable tracking of the robot pose [227, 858]. If the current experience can be localized with respect to previous ones, this implies that the map is expressive enough to localize the robot, and no further data needs to be added. If localization is weak or fails, the new data is added to the map. This has the advantage that new points are only added when needed and the complexity of the map thus reflects the variation of appearance in the scene [227]. The localization performance with respect to a single reference frame can further be improved by adding cross-experience constraints. For example, experience $\mathcal{E}_A$ may not be able to sufficiently localize w.r.t. $\mathcal{E}_C$, but if both can localize to an intermediate experience $\mathcal{E}_B$, the constraints $\mathcal{E}_C \rightarrow \mathcal{E}_B \rightarrow \mathcal{E}_A$ can refine the map coherence and localization accuracy [304, 786, 858]. To address the problems of many experiences leading to a possibly large map and potentially misleading features (such as from moved objects) remaining in the map, maps are sometimes optimized and *summarized* offline, running complete bundle adjustment between all experiences and retaining only the features most likely to aid future localization [786, 304].

15.3.1.2 Appearance Invariant Representations

Instead of mapping features for each appearance, recent work focuses on feature detection, description, and matching algorithms that allow a single feature to be matched across varying appearances. This has primarily been made possible through advances in deep learning, that allow i) training such methods on large datasets covering a range of different conditions and ii) taking much more context into account (*e.g.,* up to entire images compared to pixel neighborhoods for classical methods) [976, 42, 275].

A special kind of appearance invariant feature to highlight is semantic information. For example, if one can identify that an object is a chair in different visual settings, the fact that it is a chair is not going to change over time. Therefore, the presence of specific objects and their configuration can be used for localization across large visual changes, such as aerial and ground viewpoints [377].

Once such appearance invariant features and descriptors are extracted, they can be integrated into the SLAM pipeline as landmarks to facilitate map optimization and long-term localization. While deep learning-based approaches have shown strong performance, they typically require a GPU which may not be available on all robots and may be computationally more expensive than classical features. Finally, while this can address some challenges related to appearance, outlier matches such as objects that have moved or changed still need to be identified and rejected.

15.3.1.3 Memory, Scaling, and Marginalization

A fundamental problem for robots that operate over long times, and potentially indefinitely, is that ever more information is collected by the robot and added to the map, eventually resulting in computer memory and processing time limitations.

Forgetting. One strategy to partially mitigate this is to incorporate mechanisms for *forgetting*. Typically, this is implemented with a weight or probability of persistence of map points that diminishes over time, such that points that have not been observed for a long time have a lower probability of corrupting current localization as outliers and eventually are pruned from the map [942, 270]. Alternatively, *map summarization* [304, 786] can be run, oftentimes as an offline process, to retain only the features most likely to support future localization. One advantage of this approach is that it provides an ordering of importance such that only the top-N points can be kept subject to a memory budget constraint. This allows a robot to keep as much information around as it can fit in its memory and only start forgetting afterwards.

Memory Management. Instead of forgetting, another option is to dynamically save and load map features in and out of the memory of the robot. An example of this is shown in RTAB-Map [624], where nodes can be moved from short-term to working to long-term memory and back, ensuring that the number of nodes currently being processed is small enough for real-time computation. This prevents the need to completely remove nodes, but incurs additional computational cost for memory management and may eventually run out of capacity.

Marginalization. In addition to the number of points in the map, pose-graph-based SLAM methods optimize over the history of robot poses, which usually grows linearly with elapsed time. To avoid infinite growth in the number of nodes and factors, the process of *marginalization* aims to remove old nodes *without* losing information. The central idea is to replace a set of nodes (or other information, more generally) with a single node whose properties reflects or summarizes the information previously stored in the replaced nodes. In probabilistic models, this is the *marginal distribution* over the removed nodes (see *e.g.,* [263], Sec. 5.3 for more details). This allows effective compression of the graph and reducing the size of the problem without discarding nodes completely. Nonetheless, in practice, some information is lost when marginalizing nodes. Most importantly, the topology of the marginalized nodes can no longer be changed. For example, if there is uncertainty whether certain factors, such as observations or more importantly loop closure detections, are inliers or noise, marginalizing them will freeze their current assignment into a single node which cannot be undone, even if future measurements arise that would contradict or disambiguate this assignment. In summary, marginalizing previous measurements provides an effective strategy to reduce the size of the problem while retaining information, but reduces the flexibility of the optimization by "baking in" errors that can not easily be corrected afterwards. Thus, when to marginalize is an important open question.

15.3.2 Map Cleaning and Change Detection

Even with eventual forgetting of points or objects, detection-based methods such as the ones introduced in Section 15.3.1 tend to accumulate outdated measurements (for example of objects that have since moved). Specifically, when a detector fires it is evidence that an object or feature is present and can be added to the map. However, the inverse is not necessarily true. This manifests in the *absence of evidence* vs. *evidence of absence* problem. For example, consider a robot that visits a room and observes a chair in a corner (positive detection). If the robot later revisits the room and does not observe the chair, this might be the result of the robot not looking into the same corner and the chair might still be there (absence of evidence), or of the robot looking into the corner and establishing that the chair has disappeared (evidence of absence, or *negative* detection). In addition, since detectors are imperfect, it is also possible that the robot looked into the corner but did not recognize the chair, *e.g.,* due to illumination changes, in which case the conclusion that the chair is now absent would also be incorrect.

The establishment of such negative observations, or changes, is the goal of *change detection.* Historically, this has mostly been addressed in an offline or post-processing setting, since the goal is oftentimes to retain a static map after data collection by a robot. We will further detail this setting in this section and discuss recent extensions to online SLAM in Section 15.3.3.

15.3.2.1 Map Cleaning

The goal of map cleaning is the removal of dynamic and spurious measurements from a map in order to only retain the high-quality, static, and persistent features of

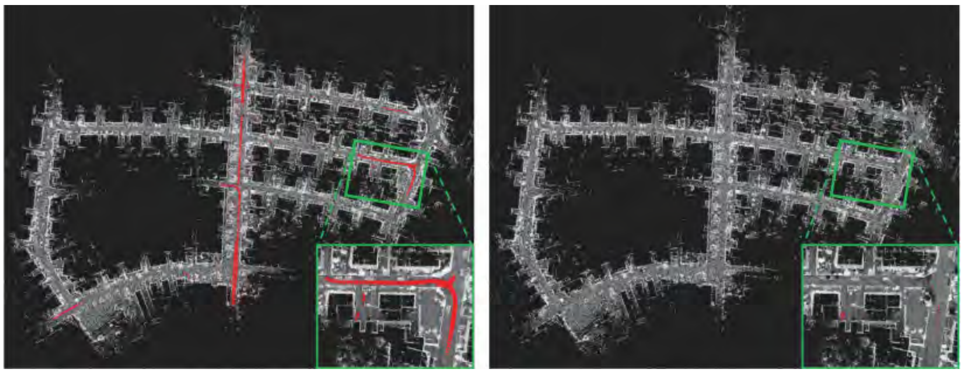

Figure 15.10 Example of *map cleaning* by ERASOR [669]. Path artifacts from measurements on dynamic objects (red) are largely detected and removed from the static map. (©2021 IEEE) TODO: Reuse permission

a scene. The most common use case is to filter out spurious points and observations of short-term dynamic objects from a mapping session [572, 669], but the same techniques can also be applied to long-term and multi-session mapping [889].

Conceptually, the underlying detection mechanisms and principles are very similar to *dense dynamic point detection discussed* in Section 15.2.3.1. However, for offline processing, additional resources are available. In a first step, globally consistent robot poses are estimated through robust bundle adjustment, where dynamic and spurious points are ignored as outliers [572, 669]. Once the sensor poses are fixed, all measurements are revisited to ensure consistency and remove spurious data. Similar to Section 15.2.3.1, prominent methods include fusing all measurements in a *volumetric map* of the scene (typically through ray-casting into an occupancy or TSDF voxel grid), where the consistency of measurements can be checked in each map cell [978]. While this considers all data, it can also lead to large memory consumption and possibly long processing times. To avoid this, *visibility*-based methods, instead of a globally fused map, select a subset of nearby measurements to compute the consistency of points across them, oftentimes directly on the sensing data [572, 669, 889].

A notable advantage of offline processing is that all measurements are available to use in map cleaning, and that a variety of additional steps, such as ground plane segmentation [669] or iterative algorithms to incrementally add high-confidence detection [572], can be performed. An example of this is shown in Figure 15.10.

15.3.2.2 Change Detection

Historically, *change detection* is a term that has been used in a variety of communities. Most prominently, *2D* or *image-based* change detection has been widely studied in computer vision. In this setting, the goal is to identify differences between two images of the same scene or even from the same view-point, with applications in background subtraction, surveillance, medical imaging, and many others [907]. Additionally, image-based change detection from satellite data is an active area of research in remote sensing [46].

In robotics, the ability to detect changes between images is also highly useful. However, in the context of mapping, where the goal is to estimate the underlying *3D structure* of the scene from all observations, change detection can directly be performed in 3D on the level of submaps or sessions. This is especially useful if measurements are eventually marginalized or fused in the map and may no longer be available. Although interest in image-based change detection for robotics has recently re-surfaced with advances in high quality novel view synthesis methods [714], we will primarily discuss 3D and mapping-based change detection approaches in this section.

It is important to note that the distinction between map cleaning and change detection can be fuzzy, since they address related problems. In the most typical case, change detection is posed in a *multi-session* mapping scenario (for example, consider a robot that scans a room in the morning and again in the evening), where each session allows the reconstruction of an internally consistent (possibly cleaned) map. The goal is then to identify and capture changes between the two visits on the scene level. To achieve this, oftentimes detailed scene representations are required and *dense* reconstruction methods are used. In addition, meaningful change representation frequently requires object-level reasoning, where an object can be considered a *unit of coherent change*. Most prominently, this includes objects (such as mugs or chairs) being added to, removed from, or moved around the scene, which will be the focus of this section. However, this could in principle also include many different and non-geometric changes, such a cushion being deformed, a wall being painted differently, or button being switched on or off.

Geometric Change Detection. As in earlier sections in this chapter, a number of cues can be used to contrast measurements, which can also be applied for change detection [573]. To contrast maps directly, however, *volumetric* representations such as occupancy or TSDF maps have proven most useful for change detection [329, 956]. This is primarily due to the fact that, in contrast to solely *dense surface* representations, they explicitly differentiate between *unobserved* and observed to be *free* space, which is essential to disambiguate the *absence of evidence* problem. During change detection, the two volumetric maps can be compared against each other in a process called *volumetric*, *map*, or *scene differencing*, where surfaces previously observed to be free must have newly appeared, and old surfaces now observed to be free must have disappeared. Finally, areas only observed in one of the sessions can be included to complete the map, and surfaces observed twice can be fused to increase the accuracy of the reconstruction.

To create consistent maps and avoid false positive detections, one first needs to accurately register all submaps or sessions into a common reference frame. Since there may be odometry drift or inaccurate state estimates within each session, it is often not sufficient to just rigidly align the individual sessions, but best results are achieved by jointly optimizing for alignment between sessions, typically through inter-session constraints and bundle adjustment [329, 956, 573]

Semantic Information. In addition to geometric checks, semantic information is frequently incorporated for change detection for a number of reasons. First, since the semantic label of an object is appearance invariant, this can provide a powerful abstraction to compare submaps created under different conditions. Note that the use of semantics does not eliminate the problem of appearance changes, but simply offloads them to a detection network which can be trained on large datasets, or specialized detectors for different sensors and conditions can be used. Second, semantics can provide valuable non-geometric information, for example a sheet of paper being removed from a table will not change the geometric surface, but may be easy to detect semantically. Finally, the goal of change detection is often to identify objects as a *unit of coherent change*. Since coherent motion is also an essential aspect of human scene understanding, there is large overlap between this definition of an object and most closed-set semantic labels (for example mugs, chairs, and similar objects often tend to move as a unit). To achieve best performance, many systems combine geometric and sematic information [632, 979].

Semantic Consistency. The problem of *semantic consistency* arises from the combination of the facts that scenes typically change in *coherent units*, but robots can generally only obtain *partial measurements* of the scene. For example, consider the case where a robot observes a table. Later that day, it returns and sees that the table has disappeared, but half of the view is occluded. If change detection is carried out on the observations or maps of that scenario, only the observed half of the table will be removed leaving half a table floating around, whereas for a human this outcome would seem rather implausible (*i.e.,* we have a strong prior that tables tend to move as a unit). This problem is illustrated in Figure 15.11. To avoid such artifacts, one can forget previous maps and map every time from scratch, which will ensure semantic consistency but also prevent the robot from ever building more complete models of the scene. More commonly, change detection can be performed on the level of objects or semantic classes to try to prevent such artifacts from forming. Finally, future observations may also remove artifcats as changes once they are observed, but the presence of artifacts in the map may reduce the overall map fidelity and limit its use for planning and autonomy.

15.3.2.3 Object Understanding through Changes

Object Detection from Changes. The inherent relation between changes and semantic objects can be used to make deductions in both directions. While we have seen above how semantic information can be used improve change detection, the fact that something has changed can also be used as a cue for object detection. To this end, typically, changes in a dense reconstruction are detected to identify all moving

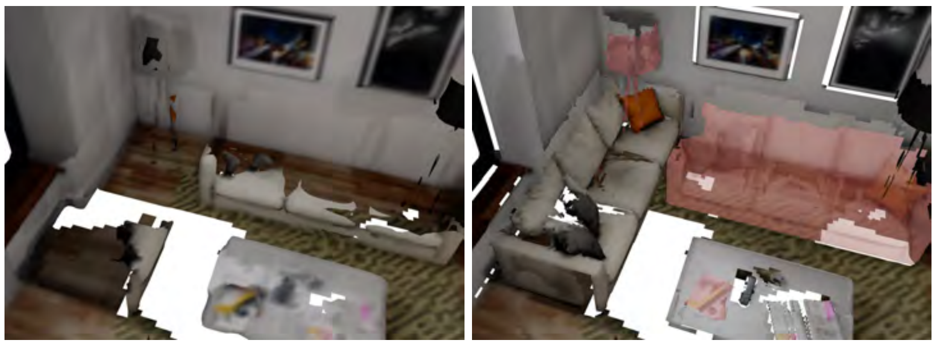

Figure 15.11 Illustration of the *semantic consistency* problem. In this scene, the sofa and lamp were switched and objects on the table were changed between two robot observations. The left image shows the reconstruction when all measurements are fused, whereas on the right semantic-aware change detection is performed [979]. Since the sofa was only partially observed on the second visit, we see parts of it remain on the left whereas the entire sofa is removed on the right (shaded red). Furthermore, conflicting measurements of free space and the sofa are fused on the left, leading to only partial reconstruction of the sofa in the new place. Finally, objects with small geometric changes such as the journal on the table are not recognized and merged with other observations. (©2022 IEEE)

surfaces. These can then be spatially clustered to identify likely objects [329, 35, 335]. This complementarity between geometry and semantics is particularly useful if no semantic object detector is present, or to complement semantic segmentation by detecting objects that were not correctly segmented. However, as before, geometric object segmentation alone is susceptible to partial observations. For example, if two touching objects were both removed they will likely be under-segmented into a single object. Similarly, an object partially observed through occlusions will likely be over-segmented into several parts.

Object Instance Re-localization. In order to build an object-centric understanding of the scene through time and not only an estimate of the current or static state, the goal of object instance re-localization is to track objects through long-term dynamic observations. Conceptually, this problem and most solution approaches are again similar to the ones for *place recognition* and *loop closure* detection. However, there are a number of additional considerations that make this a challenging problem.

First, by the abrupt nature of long-term dynamics, objects can, in principle, change arbitrarily much and move arbitrarily far between observations. Of course, based on the time elapsed and the motion model assumed for an object, the last observed state of an object may be an informative cue to associate it to current observations [956, 102], but this may in general not always be the case.

Second, since an individual object is much smaller than a scene, there is oftentimes less variety in geometry and texture, and thus many fewer features that uniquely describe it. This challenge is exacerbated for small objects by the fact that if they are not observed up closely, fewer and lower resolution measurements are available to extract features from.

Third, since there may be several instances of identical or similar object types in a given environment, *perceptual aliasing* is a major challenge. This further raises a more philosophical question of what it means for an object to be a unique instance and whether they should be individually tracked. For example, consider a meeting or seminar room with tens or hundreds of identical chairs. In this case, it may be hard to tell which chair is which and how each of them was moved over time. However, this might also not be relevant information for most tasks, and it might be preferable to fuse all observations of any chair into a single representative model that is replicated many times. On the other hand, if you consider the case of a coffee mug, there may be numerous identical mugs in an office or building, but people might care a great deal to keep track of their own mug. One option to address the challenge of perceptual aliasing is to take the uniqueness of solutions into account, *i.e.,* unique matches of a highly distinct object can directly be accepted, whereas for objects with several similar match candidates, a *multi-hypothesis* approach or adequate *uncertainty measure* can be employed.

Finally, one can consider the object instance re-localization problem in a *closed set* or *open set* setting. In the former, one assumes that the set of objects in a space is fixed, and re-localization essentially becomes an optimal matching problem. In the latter and more general case, objects can also be added or removed from the scene, which raises the additional question of "when are object observations similar enough to be the same", reflecting the fact that there is always the alternative explanation of a new but similar object having appeared in the scene.

To work towards these challenges, prevalent solution approaches primarily focus on descriptor extraction and matching. One family of methods focuses on extracting a set of keypoints on each object and descriptors for each object, which can be directly used to identify correspondences and solve for 6DoF transforms. As a natural first step, such keypoints could directly be visual features from the SLAM frontend [755]. Alternatively, 3D keypoints and descriptors such as Fast Point Feature Histograms (FPFH) [965] and Signature of Histograms of OrienTations (SHOT) [970] have found significant use for changed object re-localization. These have the advantage that they can be extracted from the dense 3D reconstruction of a session and are insensitive to variations in appearance. However, since they only represent a local patch of geometry, they tend to be less descriptive and often require further geometric verification [359]. Recently, deep-learning-based local shape (and sometimes appearance) descriptors achieve the best performance since they can be trained to emphasize the most informative aspects of a shape or object [1144]. An example of this is shown in Figure 15.12.

This has also given rise to a second family of approaches that compute a *single descriptor per object* instead of several keypoints [956, 1302]. This method is

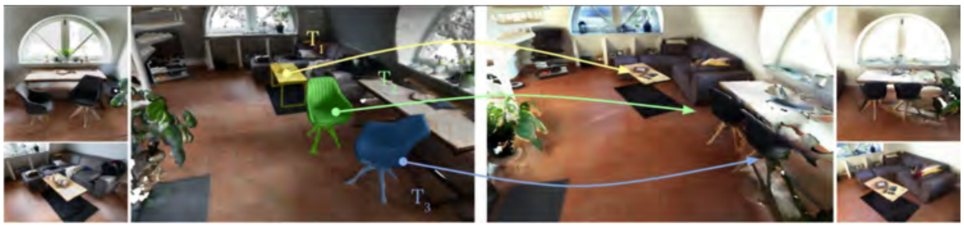

Figure 15.12 Object instance re-localization task in the RIO dataset [1144]. The goal is to identify which object instances correspond between an initial scan (left) and later re-scan (right), and estimate the 6-DoF poses $\boldsymbol{T}_1$, $\boldsymbol{T}_2$, and $\boldsymbol{T}_3$ between each object and its new location. (©2019 IEEE) TODO: Reuse permission

advantageous as keeping track of a large database of keypoints and exhaustively matching them can be computationally expensive. An important aspect of these methods is to construct or train the networks such that they are SE(3)-*invariant* or SE(3)-*equivariant* to allow matching across different viewpoints and registration of the matched objects, respectively.

Finally, once the many partial observations of changed objects can be re-localized and registered, this not only provides valuable insights about the history of each object, but also allows the reconstruction of ever more complete object models and richer scene descriptions over time [359, 1302].

15.3.3 Change-aware SLAM

While multi-session mapping provides an exemplary case of long-term dynamic observations, it is by far not the only setting where they can occur. In practice, such changes may occur over much shorter time scales. For example, consider a robot that moves from a living room to the kitchen to fetch something. If the living room changes in between, it does not matter whether the robot returns a week or only few minutes later. At its limit, this can be as fast as a robot or sensor just looking away and finding the scene changed when looking back. The goal of change-aware SLAM is to be able to detect and represent such long-term changes *during online operation* of the robot or system.

Challenges. Compared to *multi-session* mapping, where all changes are assumed to occur between sessions, changes can now occur at any point between observations. As a consequence, assumptions about stationarity can only be made within a short window, and less data is available to fuse and disambiguate noisy observations. Furthermore, the perceptual similarity between inaccurate relative localization and changes in the observed scene can make it difficult to disentangle the two problems. As an example, if the robot's pose estimate is offset by a meter, the surfaces of previously mapped objects may appear to be absent in the current, offset view, even though that is not the case. This interplay between change detection and localization, *i.e.,* the classification into inliers and outliers for each localization constraint, is further exacerbated through the influence of one's solution on the other's. This potential problem is well illustrated if we consider the two extremal solutions; if all measurements are considered unchanged inliers in spite of possibly large changes in the scene, the resulting registration solution may be inaccurate and represent an "average" of true and false measurements. On the other hand, if the entire previous scene is considered changed and now absent, the current open-loop estimate is trivially optimal and the resulting pose estimates and scene reconstruction may not be complete and consistent. Finally, change detection has to operate continuously and is based on the partial data collected by the robot so far. This requires that change detection can run at higher rates in order for the robot to have an up-to-date understanding of the environment, and ideally that false inlier/outlier change decisions can be corrected as more data is collected.

Object-level Change-aware SLAM. Since global reasoning about changing scenes on low-level sensor data quickly becomes intractable, the vast majority of change-aware SLAM methods first locally summarize measurements. Due to the nature of most considered changes happening at the level of rigid semantic objects, object-level representations are an intuitive choice [979, 898, 353, 899, 981]. In this setup, an object-level scene representation is extracted, where localization and reasoning about changes happens also on the level of each object, presenting a much more manageable problem than reasoning about individual sensor measurements, and, by construction, enforcing semantic consistency of the scene representation.

Local Consistency. An important consideration in change-aware SLAM is how to disambiguate noisy measurements from actual changes in the scene, and consequently, how these measurements are integrated into the map. For example, if change detection is run on the raw data of every frame, some noisy points may penetrate into existing objects and falsely mark them as changed. On the other hand, if more conservative change detection rules are employed, it is easy to get false negatives and fuse together data that are not observations of the same object, leading to corrupted and inconsistent reconstruction (see Figure 15.11). To overcome this, the central idea is to use *local consistency* to establish sequences of measurements that can be guaranteed to be change-free. At a minimum, this is the case when measurements are tracked frame-to-frame [979, 1134]. Intuitively, this corresponds to the fact the scene cannot change unobservedly while the robot is observing it. In practice, through adequate assumptions on the rate of change in the scene and the odometry error being low over short times, this can be extended to a sliding [353] or active [981] window.

Change-aware Mapping. Combining these ideas, a first real-time change-aware mapping system is presented in Panoptic Multi-TSDFs [979]. Assuming globally consistent poses are given, the Panoptic Multi-TSDF frontend estimates a set of semantically and locally consistent submaps by only updating submaps that can be tracked frame to frame, called the *active* submaps, and for which the semantics of the measurements match the target submap. By differencing the active to overlapping inactive submaps, changes can be efficiently detected on this higher-level abstraction of submaps, and previous submaps can be discarded or merged if they disagree or agree with the current submaps, respectively. This allows retaining information from previous submaps where they match, and by removing entire submaps enforces semantic consistency by construction. An alternative approach is presented only shortly after in POCD [898], where incoming frames are also segmented into objects, but then are tracked and registered against the current map. This has the advantage that some level of odometry noise can be tolerated and corrected. To handle noisy change observations, a probabilistic persistence model for each object is created that is updated when an object is observed, or an object that should have been observed is not registered, respectively.

Globally Consistent Change-aware SLAM. In large-scale scenes and when no global poses are available, these have to be jointly estimated with the scene changes. For object-level representations, this can directly be done using a factor-graph-based approach with the partial, locally consistent object observations acting as landmarks. To deal with the problems of data association in change detection, NeuSE [353] presents an approach to learn highly descriptive and SE(3)-equivariant neural object representations. These latent descriptors capture the complete shape from partial observations and allow for a direct association and registration of object instances. As a result, such associations can be used as loop closure constraints and spatially-inconsistent or missing associations used as a change signal. Alternatively, POV-SLAM [899] presents an approach using the expectation minimization (EM) algorithm to iterate between optimizing the poses of the robot and objects given the associations between object observations, and optimizing the object observations and persistence probabilities given the spatial layout of the scene. This allows it to continuously refine both localization and change detection, especially when new observations are made in the future. Instead, Khronos [981] creates candidate association factors between nearby objects and uses robust factor graph optimization via graduated non-convexity [1218] to determine the inlier associations. Furthermore, the (assumed static) background deformation is incorporated as extra factors to aid spatial optimization, and a deformable geometric change detection step is performed after loop closure to resolve the evidence of absence vs absence of evidence problem.

Unified Short and Long-term Dynamic SLAM. Finally, the notion of local consistency can be used to combine short-term dynamic object tracking with change-aware methods in order to capture various dynamics patterns at the same time, shown in Figure 15.13. A first formulation of this is presented in Changing-SLAM [1134]. Changing-SLAM utilizes a sparse SLAM formulation building on top of ORB-SLAM [793], where in each frame feature points are grouped into objects by semantic masks. These points are then tracked frame-to-frame using a Kalman

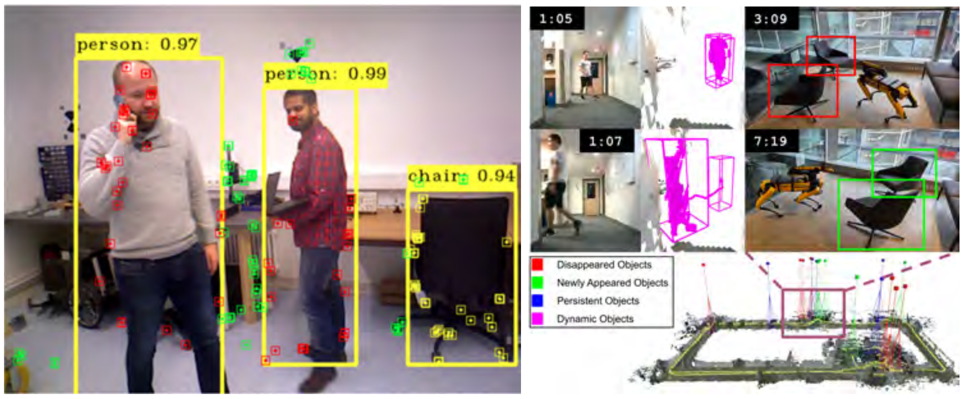

Figure 15.13 Unifying short and long-term dynamic SLAM. Changing-SLAM [1134] (left, ©2023 Springer) TODO: Reuse permission [OA] tracks keypoints for short-term dynamic objects (red), the background (green), and associates potential long-term dynamic objects (yellow). Alternatively, Khronos [981] (right) densely detects and reconstructs short-term motion (purple) and long-term changes (removal and addition shown in red/green) during online operation.

filter to capture short-term motion. Object points that are not tracked (long-term changes) are associated globally to nearby objects of the same semantic class. Finally, a persistence score is estimated for each object such that objects that have not been detected are forgotten and removed from the map.

A first method for dense spatio-temporal metric-semantic SLAM is presented in Khronos [981]. Following similar ideas, Khronos uses a volumetric local map to extract reconstructions of the background and and tracked static and moving objects. After optimizing for globally consistent poses after loop closure, an additional change detection step estimates when each object change occurred to estimate the history and evolution of the state of the scene.

15.3.4 Temporal Scene Understanding

When mapping an environment, the goal is to build a digital representation and understanding of the scene. As discussed before, when mapping dynamic environments, the desired level of understanding can vary from building a model of the *persistent* parts of the scene, excluding motion and changes, to building a detailed *4D* reconstruction and understanding of what moved and changed, when, and how. However, such an understanding can go beyond capturing the individual moving and changing entities to estimating the underlying *temporal variation patterns* and predict the future evolution of the scene. In this section, we briefly overview three key families of temporal models.

Maps of Dynamics. Where *dynamic maps* focus on detecting and tracking moving objects, Maps of Dynamics (MoD) take these trajectories or other motion information as input to estimate the *typical motion patterns* in the scene [618]. Such patterns can be constant (such as the dominant motion direction in a one-way road), or also time dependent (for example, people tend to move into the office in the morning and move out again in the evening). Understanding these patterns can provide valuable information for navigating dynamic scenes or human motion prediction.

Periodic Events. A special but commonly found kind of dynamics pattern are periodic or cyclical events, such as for example daily or yearly variations of a scene. If one assumes that the state of a scene is governed by an underlying hidden process that is periodic, observations of the scene can be treated as observations of the periodic process and used to estimate its properties. This idea has been pioneered by Krajnik *et al.* [611], who propose *frequency maps* to analyze the different periodic processes of observations in the time domain through spectral analysis in the frequency domain by means of the Fourier transform. By only storing the dominant modes of the frequency spectrum, a compact map representation is achieved that can capture periodic events of different (and arbitrarily long) frequencies. Since the complete 4D model of the scene is defined by these modes, once they are estimated from past observations, all future states of the scene can be predicted, which can improve future localization and path planning [610].

Learning-based Methods. Beyond structured or manually designed statistical models of dynamics, learning-based methods can also be used to predict future variations in the scene. Typically, object-level changes or motion are predicted using scene graphs as compact representations of objects and their relations to their surroundings [857, 700, 402]. The learning-based approach has several advantages, including higher flexibility to represent complex dynamics patterns and the ability to combine semantic priors (*e.g.,* that dining tables tend to change around noon due to lunch) with observations from a given environment. This can lead to better prediction performance compared to more static co-occurrence priors or pure frequency-based models [857, 402], and improve active and pro-active robot behavior in the future [857, 700]. As a disadvantage, more data may be necessary to train more complex models, where change data may be scarce and imbalanced with respect to static observations of the scene.

15.4 Deformable SLAM

Deformable SLAM, or SLAM in deformable environments, refers to SLAM in environments *without a static background*, *i.e.,* in which there is not any static part. A particularly relevant case is in robotic minimally invasive surgery scenarios, where the only available sensor is a monocular camera that can only observe the deformable organs inside the human body [900]. Since the lack of any static parts makes the deformable SLAM problem severly underconstrained, it is a more challenging problem than static and dynamic SLAM.

In the following, we start by discussing the differences between deformable SLAM and Non-Rigid Structure from Motion (NRSfM) in Section 15.4.1. We then first present the simpler case where depth information, for example from an RGB-D camera or stereo cameras, is available for deformable SLAM in Section 15.4.2. Second, we will then introduce the pipeline for deformable SLAM for the challenging case of monocular cameras in Section 15.4.3, followed by details about map initialization and map extension in Section 15.4.4.

15.4.1 NRSfM vs. Deformable SLAM

Similarly as rigid visual SLAM is connected with SFM, non-rigid SLAM has a strong connection with its non-rigid counterpart NRSfM.

15.4.1.1 Non-Rigid Structure from Motion (NRSfM)

NRSfM is a classical topic in computer vision which addresses the monocular 3D modeling in deforming environments, and provides the theoretical foundations for understanding the 3D modeling of a deforming scene from a video sequence. Each 3D point in the scene has a distinct 3D position in every frame, therefore, the map does not represent a single position but rather a 3D trajectory for each map point. As a consequence, the monocular problem is severely under-constrained and additional priors are compulsory to constrain the solution. Popular options are *temporal smoothing* priors, which assume that the trajectory of a 3D point is smooth over time, and *spatial* priors that assume that the deformation of a 3D point is similar to that of its neighbors.

The classical formulation of NRSfM assumes a static monocular camera observing a moving object without any static background. Most of the video frames observe the full object extent, *i.e.,* most of the images overlap while observing the same deforming object. Typical cases are a moving hand, a waving flag, or a jumping person, all in front of a camera. It is important to note that despite the camera being fixed, the object motion includes a significant rigid motion which is never explicitly computed, but included in the non-rigid scene motion with respect to the fixed camera (see Figure 15.14(a)). Although the rigid motion is not computed, it implies a significant parallax that makes the estimation problem well conditioned.

Shape from Template. A first family of NRSfM are Shape from Template (SfT) methods. They assume that the textured 3D shape-at-rest of the observed object is available as a prior, which is then used to recover the deformation, thus significantly simplifying the problem. This textured 3D shape-at-rest of the object is called the *template*. These methods depend on the deformation model of the template. On one hand, there is the analytic isometric deformation model which assumes that the geodesic distance between points in the surface is preserved. This has proven

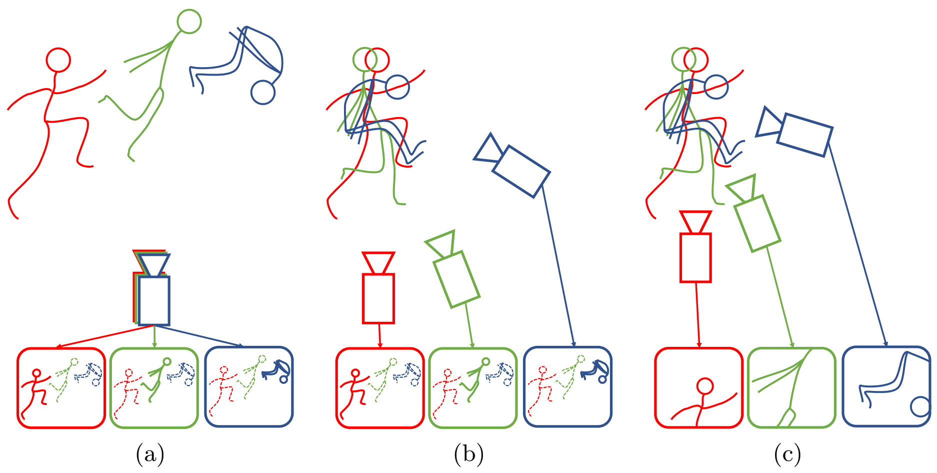

Figure 15.14 Sketch of NRSfM vs. Deformable SLAM in 3 frames. The scene *does not include any static background*. (a) The classical NRSfM assumes a fixed monocular camera while the 3D object undergoes a non-rigid motion which includes also a rigid motion. (b) Deformable SLAM assumes that the scene object mainly undergoes a non-rigid motion while the moving camera focuses on the rigid motion despite the absence of a static background. (c) Deformable SLAM also often assumes that the camera is in a close-up, where the deforming scene is only partially observed, *i.e.,* the camera undergoes an *exploratory trajectory*.

to be well-posed and quickly evolved to stable and real-time SfT solutions [235, 66, 206]. On the other hand, there are energy-based methods [971, 28, 810, 627, 429], which jointly minimize the deformation energy w.r.t. the shape-at-rest and the reprojection error for the image correspondences. These methods have further been embedded within sequential data association with robust kernels to detect and reject outliers [627, 628, 429].

Non-Rigid Structure from Motion (NRSfM). Early approaches to deal with the fully-fledged NRSfM problem without the strong prior of SfT assumed an orthographic camera, which is only valid for a moving object far from the camera. These earliest methods were proposed in [119]. This seminal work gave rise to methods based on a low dimensional basis to compute the 3D trajectories from the frames of a sequence [843, 781, 248]. They include regularizers either spatial in [248, 376], temporal in [30], spatio-temporal in [27, 403, 404], or more recently, topological priors [994].

A perspective camera model is a must for the close-up sequences that are typical in deformable SLAM, such as in medical endoscopy. The isometry assumption, first proposed in SfT methods, has also produced excellent results in NRSfM [1080, 1131, 204, 205, 849, 850]. Particularly useful for SLAM is the isometric [849], a local method that is able to naturally handle occlusions and missing data preva-

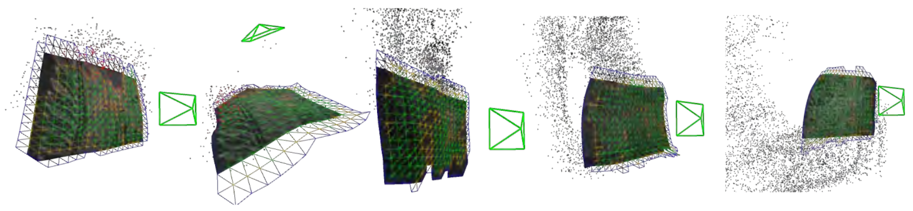

Figure 15.15 Typical exploratory trajectory in Deformable SLAM [628]. The camera pose (green frustum) is estimated together with a local deformable template (mesh) of the derforming scene. Global map points are shown in black. (©2021 IEEE)

lent in deformable SLAM applications. In the transition between SfT and NRSfM, Agudo *et al.* [29] propose a deformation model based on Navier's equations and a FEM model processing the video frames in a EKF-SLAM sequential manner.

15.4.1.2 Deformable SLAM

In contrast, deformable SLAM targets a variation of the above-mentioned problems, where the scene is deforming and the camera is moving. The goal is then to estimate both the history of deformations of the observed scene and the camera trajectory. The hypothesis is that all rigid components of the deformation are included in the camera trajectory (see Figure 15.14(b)). The disentanglement of the rigid and non-rigid components of the motion is an ill-posed problem, as highlighted by the so-called 'Floating Map Ambiguity' [629]. However, this separation can be achieved by introducing priors on the displacements of the deformable object.

An important SLAM challenge is the fact that the camera is typically undergoing an *exploratory trajectory*, meaning not all the frames observe the same scene region (see Figure 15.14(c)). It is also frequent that the camera is close to the deforming surface, hence the orthographic camera assumption is no longer valid. Figure 15.15 displays this typical close-up in a sequence observing a deforming mandala [628].

The final prevalent assumption is that the process has to be causal and in real-time, thus to produce the map and camera pose at time k, only the frames from 1 to k can be processed. This is in sharp contrast to NRSfM where processing of all images is performed in batch mode.

Given the difficulties involved, we will first talk about deformable SLAM with depth information in Section 15.4.2 and then come back to deformable SLAM using a monocular camera in Section 15.4.3.

15.4.2 Deformable SLAM with Depth Information

3D sensors such as RGB-D or stereo cameras provide depth. This makes the estimation overconstrained, which greatly reduces the complexity of the problem compared to the pure monocular approach. DynamicFusion [808] is the seminal work

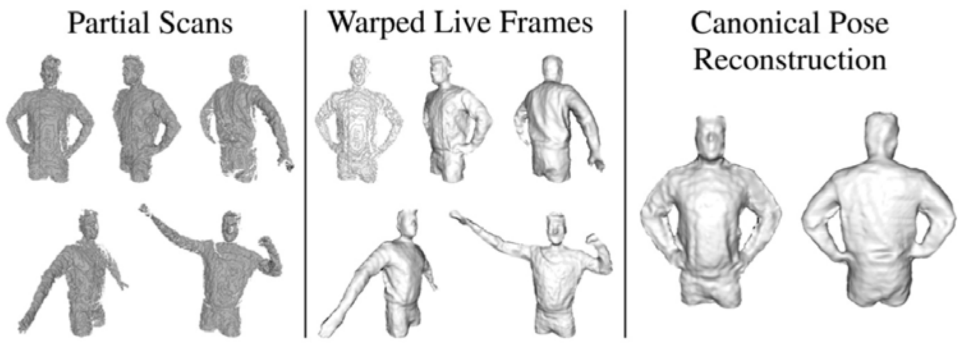

Figure 15.16 KillingFusion [1022] is a method that recovers a non-rigidly moving object observed in a moving RGB-D camera. By using Signed Distance Function (SDF) and a killing regularizer, it is able to recover structures that can undergo topological changes such as the merging of the hands with the torso. (©2017 IEEE)

of deformable 3D SLAM from RGB-D. It builds a canonical map of the scene, *i.e.,* its shape at rest, and deforms it in order to explain the current depth observation combining a form of a Embedded Deformation graph model (ED) [1054] with an as-rigid-as-possible regularizer [1031]. ED models build a discretization of the deformations space in a graph structure, speeding up dense reconstructions. DynamicFusion defines the foundations for subsequent works like VolumeDeform [512], that included photometric information to improve the results. Later, KillingFusion [1022] proposed to enforce deformations to be smooth and nearly isometric. An example of this is shown in Figure 15.16. All of the methods above represent the map as SDF, which scale poorly with the size of the map and limit the application of these methods for exploration. Surfelwarp [372] proposes a surfel representation to be used instead of the classical SDF to improve scalability. In the medical arena, MIS-SLAM [1029] presented deformable SLAM for a stereo system based on as-rigid-as-possible deformations combined with the ED model. Also in stereo medical applications, Zhou *et al.* [1289] propose the Expectation Maximization and Dual Quaternion (EMDQ) algorithm combined with SURF features to track the camera motion and estimate tissue deformation between video frames.

15.4.3 Pipeline for Deformable SLAM using Monocular Cameras

The pipeline reproduces the typical tracking at frame rate and the mapping at keyframe rate of visual SLAM, but adapted to the deforming case as proposed in DefSLAM [628].

Tracking. Deformable tracking is similar to SfT methods. The shape-at-rest is assumed available for the local area that is being explored, as firstly proposed in [627], with a deformation energy inspired in the physics based elastic model.

Later, Gomez *et al.* [429] propose a deformable tracking system based on spatial, as-rigid-as-possible, and viscous temporal regularizers. These methods ended up integrated into full deformable SLAM systems in DefSLAM[628] and in NR-SLAM [938], respectively.

Mapping. The deformable mapping thread takes point tracks along the sequence as input and is in charge of recomputing the templates at keyframe rate. DefSLAM [628], the first ever deformable SLAM, uses isometric NRSfM [849] to perform this computation. The NRSfM processes a subset of covisible keyframes to produce a template per map keyframe. These templates are aligned with the previous templates computed at earlier stages by previous local mapping operations, assuming a planar topology for the scene. To relax the assumption of planar topology, NR-SLAM [938] introduces the concept of the *dynamic deformable grap* to cope with any scene topology. Regarding the deformation model, the authors go one step further than [429] and propose the intuitive visco-elastic deformation model, which includes both temporal and spatial regularization. This model is brought into the local mapping deformable bundle adjustment which is at the core of the deformable mapping, replacing the isometric NRSfM of DefSLAM.

Feature Matching. Regarding feature matching, DefSLAM inherits the ORB [958] features from ORB-SLAM [793]. However, they prove to be inadequate descriptors for close frame tracking in medical scenes, where Lukas-Kanade (LK) optical flow [715] shows better performance [399] due to its invariance to affine lighting changes. In addition, the authors exploit the ORB descriptor combined with a DBoW bag of words [370] for place recognition to achieve camera relocalization after tracking losses. LK tracking is also the method for feature tracking in NR-SLAM [938].

15.4.4 Initialization and Map Extension

Initialization from scratch is always a challenging step for any monocular visual SLAM system. The challenge is even more pronounced in the deformable case. DefSLAM [628, 399] relies on the isometric NRSfM algorithm [849], that is able to compute the templates for the first keyframes from scratch. The only issue is how to compute the initial matches between the keyframes, which is addressed by assuming a planar scene. In the case of NR-SLAM [938], an initial guess for the deformable bundle adjustment is needed, where the initial guess map and camera motion are computed using rigid SfM. The dynamic deformation graph is initialized from a segmentation of the image optical flow between the initial keyframes.

When the camera explores a new region, new points have to be added to the map to expand the map to cover the new areas. In the case of DefSLAM, the isometric NRSfM naturally handles this extension provided that matches in the keyframes can be found for these new points. In the case of NR-SLAM, the fact that camera poses are available is exploited to triangulate the new map points from their matches in the already located frames. To this end, a model selection approach is proposed where map points are triangulated with both a rigid and a non-rigid algorithm, and then the right model for the newly explored region is selected.

15.5 Summary, Challenges, and Future Directions

SLAM in dynamic, changing, and deforming scenes is a highly active and developing area of research. In this chapter, we aimed to provide an overview of key challenges and solution approaches in the field. However, we inevitably could not cover all work that is ongoing, and we hope this serves as inspiration to read more deeply into some of the topics and contribute to the field in the future. In spite of notable advances, many open problems remain. We highlight some of them and other promising future directions below.

State Estimation in Extreme Environments. Modern SLAM systems are already quite robust to changes and dynamics in the environment, even without explicitly modeling them. Nonetheless, integrating dynamic object tracking for state estimation has shown to improve robustness and accuracy if there are large occlusions [698, 458], large fractions of moving objects [904], and strong motion priors [458, 83]. However, these advantages come at increased computational cost and currently each of these challenges is addressed separately. SLAM solutions that are efficient and robust to a diverse range of extreme conditions, or 'corner cases', remain an open problem.

Differentiable Scene Representations. Differentiable rendering models such as NeRF and 3D Gaussian Splatting (3DGS) as introduced in Chapter 14, although still recent, have already shown a lot of promise as representation for dynamic scenes. Initial works demonstrate their capacity for high-fidelity reconstruction of dynamic and deforming objects [1189] as well as for change detection [714]. Nonetheless, this is still a developing field with open challenges in memory and computation cost for real-time SLAM, partial and noisy observations from mobile sensors, as well as scaling and map management for spatially and temporally consistent mapping.

Data Association and Deep Learned Priors. The fundamental challenge of data association is still a hard and relevant problem, especially for partial observations, occlusions, as well as long-term place recognition and instance re-localization. To overcome this, there appears to be a trend toward deep learned methods that incorporate priors to complete and associate objects from partial observations [353, 1302, 1204]. In particular, geometric deep learning techniques to extract SE(3)-invariant or SE(3)-equivariant features hold promise to address these challenges. However, there are still open questions with respect to generalization, reliability, and scalability for mobile deployment.

Lifelong Operation and Continual Learning. A further open problem is the scalability and performance of SLAM methods for lifelong operation. This includes a range of perceptual challenges from appearance to geometric changes, as well as possible sensor degradation over time. Furthermore, if a robot can operate for long durations, it should also be able to continually learn [1138] to adapt and improve over time. Finally, both of these directions will need some form of memory management to decide which information to retain, how to summarize or marginalize, and when and what to forget.

Deformable SLAM. The research on SLAM in deformable environments is still at its earlier stage. There are still many open research questions [499]. For example, under what conditions is it possible to accurately estimate both the camera trajectory and the deformation of the map? Can we clearly distinguish the rigid motion of the deformable object and the rigid motion of the camera? Can we provide accurate estimates of the uncertainties of the deformable SLAM results? Are there local minima in deformable SLAM? How can we know whether the obtained result is the global minimum or not?

Towards Spatio-Temporal AI. Beyond SLAM in dynamic and deformable scenes, spatio-temporal AI aims to build a holistic scene understanding that goes beyond localization and reconstruction, but captures and learns temporal patterns. Ideally, such a map or 'world model' will combine semantic priors and observations from the robot to provide a profound and mechanistic understanding of a scene and its dynamics. This will allow estimating a consistent and complete history of the past, predicting future outcomes, and facilitate high-level and long-term reasoning and decision making.