TwistSLAM++: Fusing multiple modalities for accurate dynamic semantic SLAM

Mathieu Gonzalez[1], Eric Marchand[2], Amine Kacete[1] and Jérome Royan[1]

***Abstract*— Most classical SLAM systems rely on the static scene assumption, which limits their applicability in real world scenarios. Recent SLAM frameworks have been proposed to simultaneously track the camera and moving objects. However they are often unable to estimate the canonical pose of the objects and exhibit a low object tracking accuracy. To solve this problem we propose TwistSLAM++, a semantic, dynamic, SLAM system that fuses stereo images and LiDAR information. Using semantic information, we track potentially moving objects and associate them to 3D object detections in LiDAR scans to obtain their pose and size. Then, we perform registration on consecutive object scans to refine object pose estimation. Finally, object scans are used to estimate the shape of the object and constrain map points to lie on the estimated surface within the bundle adjustment. We show on classical benchmarks that this fusion approach based on multimodal information improves the accuracy of object tracking.**

***Index Terms*— SLAM, Localization, Mapping**

I. INTRODUCTION

The goal of visual Simultaneous Localization and Mapping (SLAM) is to estimate the pose of a camera moving in space while simultaneously building a map of the environment. Classical approaches [1] assume the scene to be static, a condition that is rarely met in real world scenarios. To solve this problem some systems propose to mask out dynamic objects in images [2]. While this method enables SLAM in dynamic scenarios, it also loses an important piece of information for some applications. Indeed autonomous vehicles or augmented reality may need an estimate of the trajectory and pose of objects in the scene. Moreover some approaches mask out a priori dynamic objects that are in reality static (e.g. parked cars) which can hurt camera pose estimation accuracy. To solve this problem, systems such as [4], [5], have been designed to track both the camera and all moving objects. They show that they can accurately estimate camera poses in dynamic scenarios while estimating the trajectory and velocities of moving objects. However those approaches are often based solely on RGB information and are less precise than camera pose estimation. Furthermore, those approaches suffer from tracking drift that can not be corrected by loop closure. Finally they do not have access to the canonical pose of the object but rather to its relative pose with respect to its initial pose. This can be limitating as relative pose alone is insufficient for some use cases, such as augmented reality applications that need an estimate of the object pose to seamlessly overlay virtual content on it. To solve those problems we propose to update our previous work TwistSLAM [5] by integrating a 3D object detector based on LiDAR information. This detector is used to predict the pose and size of 3D bounding boxes corresponding to potentially moving objects in the scene. Associating detections to tracked objects allows us to have access to their canonical pose, i.e. their pose with respect to an a priori known object coordinate frame. Furthermore, we use consecutive poses to constrain the displacement of objects, thus reducing the drift. The obtained bounding boxes are then used to associate 3D LiDAR points to tracked clusters which serves two purposes. First they allow us to improve object tracking by feeding successive scans to a generalized ICP algorithm. The computed pose is then used as a constraint in the bundle adjustment (BA). Second, inspired by the work of DSP-SLAM [3], we use scans to fit a deep-learnt signed distance function (SDF) [6] that represents the object geometry. However contrary to DSP-SLAM we do not use the SDF to estimate the object pose as we already have a good estimate of it, but we rather use it to constrain the 3D map points of clusters to lie on the estimated mesh, similarly

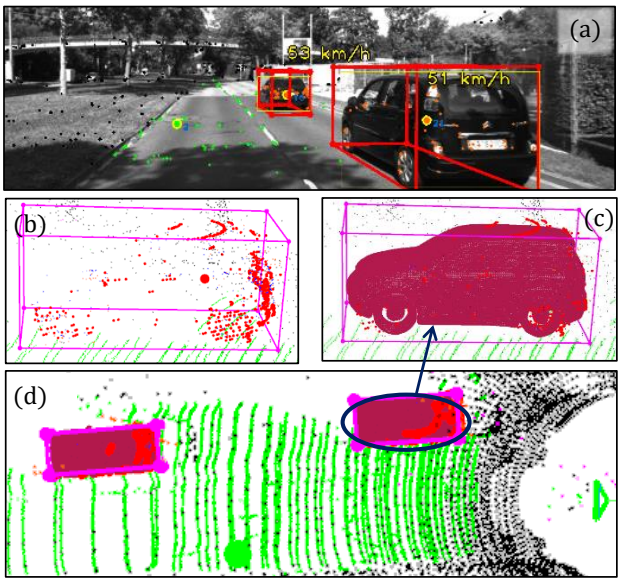

Fig. 1. In our SLAM system we track object (here cars) moving in the scene. We show here (a) a frame with tracked cars, their speed and reprojected bounding boxes. (b) the bounding box and clustered LiDAR points (red) for the closest car. (c) the reconstruction of the car, using the approach of DSP-SLAM [3]. (d) the map seen from above, with LiDAR points (black), road LiDAR points (green), tracked cars and the camera frustum (right).

[1] Mathieu Gonzalez, Amine Kacete and Jerome Royan are with the Institute of Research and Technology b<>com, Rennes, France, {mathieu.gonzalez,amine.kacete, jerome.royan}@b-com.com

[2] Eric Marchand is with Univ Rennes, Inria, IRISA, CNRS, Rennes, France, Eric.Marchand@irisa.fr

to our previous work [7] which was restricted to planes.
To summarize, our contributions are:

- A semantic SLAM system that can robustly estimate the pose of a camera in dynamic scenes.
- A SLAM framework that can track multiple moving objects and estimate their canonical pose.
- A SLAM system able to fuse 3D object pose estimation, object tracking and 3D registration results from LiDAR scans to reduce tracking drift.
- A SLAM framework that uses the 3D reconstruction of object from LiDAR data to constrain the geometry of map points.

We evaluate our approach on sequences from the KITTI tracking datasets. We compare our results with state of the art dynamic SLAM systems DynaSLAM 2 [4] and TwistSLAM [5] and show that we improve object tracking robustness and accuracy within our SLAM system.

The rest of the paper is described as follows. First we describe related work on dynamic SLAM, object based SLAM and LiDAR based SLAM. Then, we rapidly recall the work of TwistSLAM and present the novelties of our approach: the use of a LiDAR based object detector, followed by scan registration and SDF fitting. Finally we demonstrate the benefits of our approach on multiple sequences from a public dataset.

II. Related work: Dynamic and object based SLAM

In this section we first present some semantic dynamic SLAM systems that tackle the problem of dynamic objects by tracking them [8], [4], [9], [10], [11] or use them as high level landmarks [12], [11]. For a additional resources on classical and semantic SLAM we refer the reader to [13], [14], [5]. We also present SLAM systems that make use of LiDAR information and fuse multiple modalities to improve the accuracy and robustness of camera tracking.

DynaSLAM II [4] uses semantic information to detect objects. Object 3D points are represented in the object reference frame and used to estimate the object pose at all time by minimizing their reprojection error. TwistSLAM, [5] creates a map of clusters corresponding to objects in the scene. Static objects are used for camera tracking while potentially dynamic objects are tracked. Furthermore, by using mechanical links between clusters, TwistSLAM constrains the velocity of objects to be coherent with the structure of the scene, improving object pose estimation.

Some approaches propose to detect objects in the scene to use them as high level landmarks. The first object based SLAM systems such as [12], [15] require a specific object pose estimation algorithm [16], [17] which limits their applicability in real world scenarios. More recent ones, however, are based on generic object detectors. Those approaches use quadrics [18] or 3D bounding boxes [11] to represent objects. Some even more recent approaches have represented the geometry of objects more accurately using learning based approaches. NodeSLAM [19] optimizes detected object poses and shape, represented by an autoencoder. Object poses are then used in the SLAM system to estimate the camera pose. DSP-SLAM [3] optimizes the latent code of a deep learning based SDF [6] and uses it to estimate the pose of the object and to reconstruct the object shape. Object poses are then used to constrain camera pose estimation in the BA.

Finally, some approaches [20], [21] have been using LiDAR scans instead of images as an input for SLAM: [20] is a full SLAM system based only on LiDAR data, which represents the map using a set of surfels. 3D LiDAR points are transformed to the image plane using a spherical projection, yielding a so-called *vertex map*. This map is used, together with a *normal map* to estimate the updated current pose using point to plane registration after finding associations in the image plane. The current scan is then fused with the map to update it. Finally loop closure is performed. Virtual views are generated with the surfel map to compute the alignment with the current scan. After a verification step, a pose graph optimization is performed and used to update the surfel map. Some approaches [22], [23], [24], [25] have also injected semantic information into LiDAR based SLAM systems, to improve pose estimation, for example by masking out moving objects. SuMa++ [22] improves on [20] by integrating a CNN to segment LiDAR scans [26]. This allows them to obtain a higher level map. Furthermore semantic information is used to detect and remove surfels belonging to dynamic objects. It is also used to guide the ICP by weighting associated points.

III. TwistSLAM++: Multimodal object tracking

In this section we present our approach for which we show a pipeline in figure 2. Following the idea of the algorithms TwistSLAM [5] and S^{3}LAM [7] we use a panoptic neural network [27] to create a map of clusters corresponding to objects in the scene. Using points extracted from a priori static clusters (e.g. road, house, vegetation) we track the camera. Then, we use the points from remaining potentially dynamic clusters to track the objects. As we estimate the geometry of some objects (e.g. a plane for the road) we are able to constrain the velocity of tracked clusters with mechanical links. To improve this approach we chose to use LiDAR scans in several ways. First we feed them to a 3D object detection network that estimates the pose and size of objects in the scene. Second we use successive LiDAR scans corresponding to objects and register them to compute their relative pose. We inject both detected and registered poses as constrains in the BA, the first one being free from any drift and the second one more accurate. Third, we follow the work of [3] and use DeepSDF [6] to fit a SDF to objects using LiDAR points. The SDF is then used in the BA to constrain the SLAM map points to lie on the object surface, thus improving the estimated map.

A. Clusters creation

To obtain a complete semantic map in which objects are uniquely identified we estimate the panoptic segmentation of images, using [27]. Similarly to [7] we fuse 2D observations of a single 3D point to obtain its class and id. Doing so we obtain a semantic map and create a set of K clusters $\mathcal{O} =$

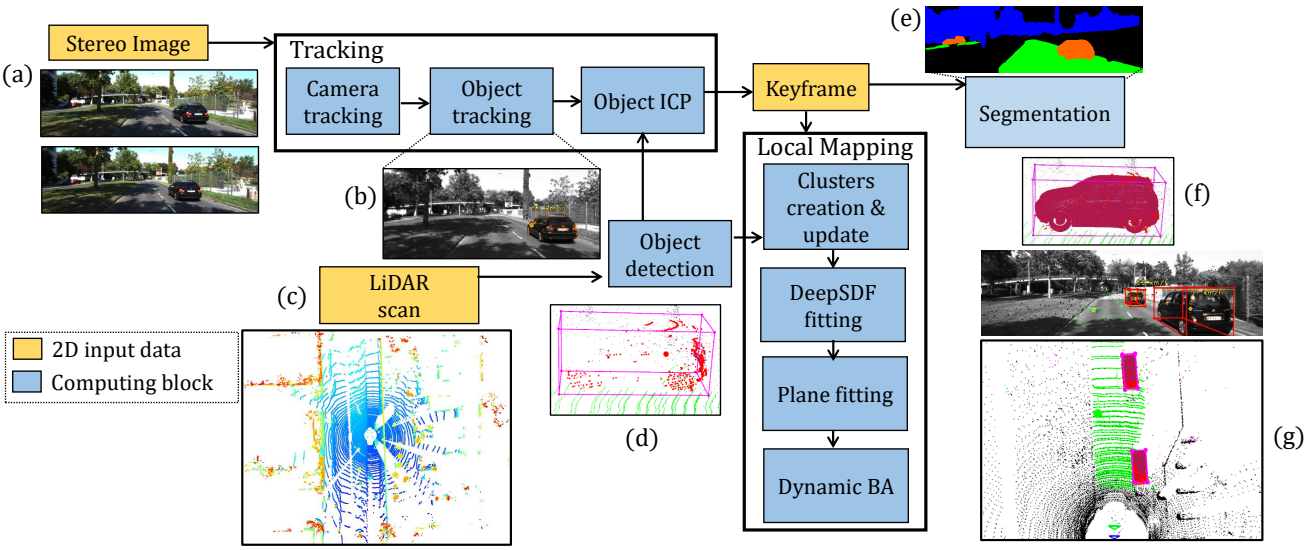

Fig. 2. The pipeline of our approach: (a) keypoints are extracted from stereo images and used for camera tracking and (b) object tracking. (c) LiDAR scans are fed to a 3D object detector, allowing us to obtain (d) the 3D bounding box of objects in the scene and to cluster LiDAR points, which are then used in an ICP algorithm. (e) selected keyframes are segmented to create clusters that are augmented with clustered LiDAR points. (f) Object LiDAR points are then used to fit a per object sdf that constrains the geometry of clusters. (g) The trajectory and geometry of clusters is refined in the BA.

$\{O_k, k \in [1, K]\}$. A cluster is a set of 3D points corresponding to a single object, grouped according to their class and instance id. We split the set of clusters into two parts: static clusters $\mathcal{S}$ (such as *road, building, ...*) and a priori dynamic clusters $\mathcal{D}$ (such as *car, bike, human, bus, ...*). A static cluster contains 3D points $\{{}^w\mathbf{X}\}$ expressed in the world frame. On the other hand each dynamic cluster contains a set of 3D points $\{{}^o\mathbf{X}\}$ expressed in the object coordinate frame, a set of poses $\{{}^w\mathbf{T}_o\}$ and a set of twists expressed in the world coordinate frame $\{{}^w\boldsymbol{\xi}_o\}$ representing the cluster trajectory and velocity through time. The pose transforms points from the object coordinate frame to the world coordinate frame:

$$ {}^w\bar{\mathbf{X}} = {}^w\mathbf{T}_o \, {}^o\bar{\mathbf{X}} \tag{1} $$

where $\bar{\mathbf{X}}$ denotes the homogeneous coordinates. The twist, of object o at the i^{th} timestamp, express in world coordinates transforms a pose between two timestamps:

$$ {}^w\mathbf{T}_{o_{i+1}} = \exp({}^w\boldsymbol{\xi}_{o_i}) {}^w\mathbf{T}_{o_i} \tag{2} $$

where $\exp(.)$ is the exponential map [28] of $se(3)$. For simplicity in the remainder of this paper we will omit the object index k. For some clusters corresponding to a priori chosen classes (such as the road or the facade of a building) we estimate a 3D plane, represented by $\pi = (a, b, c, d)^\top$ with $||\pi||^2 = 1$, using its 3D points $\{{}^w\mathbf{X}\}$. The plane follows the following equation: $\pi^\top {}^w\mathbf{X} = 0$ and we estimate it using a SVD in a RANSAC loop.

B. Dynamic SLAM

Using static clusters we can robustly track the camera:

$$ E({}^{c_i}\mathbf{T}_w) = \sum_{j \in \mathcal{S}} \rho(||{}^i\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w, {}^w\mathbf{X}_j)||_{\Sigma_{i,j}^{-1}}) \tag{3} $$

where ${}^i\mathbf{x}_j$ is the 2D keypoint corresponding to the observation of ${}^w\mathbf{X}_j$ in the i^{th} frame, p is the pinhole camera projection function, ρ is a robust cost function (in our case Huber) [29] and $\Sigma_{i,j}$ is the covariance matrix of the reprojection error. The camera pose is correctly estimated, even in dynamic scenes as it is only based on static parts of the scene.

C. Dynamic object tracking

To track objects we match keypoints extracted from the current and the previous frame. This can be a challenging task as we do not initially know the movement of objects. To facilitate matching of keypoints and robustify it we estimate the optical flow between consecutive frames with a CNN [30]. Keypoints are searched in areas defined by their previous position updated with the optical flow. Furthermore we ensure that enough keypoints are available for tracking by extracting more keypoints from areas defined by objects bounding boxes. The keypoints are then used either to create new 3D points with stereo triangulation, which are added to existing clusters or used to create new clusters, or used to track the existing cluster. The assumption in TwistSLAM [5] is that many moving clusters can be represented as being linked to a static parent cluster with a specific mechanical link. To track the object we thus optimize the twist ${}^w\boldsymbol{\xi}_{o_i}$:

$$ E({}^w\boldsymbol{\xi}_{o_i}) = \sum_j \rho(||{}^i\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w \exp(\boldsymbol{\Pi} {}^w\boldsymbol{\xi}_{o_i}) {}^w\mathbf{T}_{o_{i-1}}, {}^o\mathbf{X}_j)||_{\Sigma_{i,j}^{-1}}) \tag{4} $$

where $\boldsymbol{\Pi}$ is the projection operator that removes twists degrees of freedom, which constrains the twist according to the mechanical link, ρ is the Huber robust estimator [29] and $\Sigma_{i,j}$ is the covariance matrix of the reprojection error, which we estimate using the median absolute deviation (MAD) [29]. For further information about the development of the projection operator we refer the reader to [5]. We then update the object pose using equation (2).

We optimize this cost function with the Levenberg-Marquardt algorithm on matches found between consecutive frames. We then use the estimated pose to project 3D

map points into the current frame, find new matches and optimize again the cost function.

D. Injecting LiDAR scans in TwistSLAM

To improve the accuracy of object tracking we propose to use LiDAR scans, taken at each timestamp and processed in multiple ways. LiDAR scans are loaded by the tracking thread and tranformed from the LiDAR to the camera coordinate frame.

The first way we process the scans is by using a 3D object detection network (namely 3DSSD [31]). For each scan this network yields a set of detected objects, with their corresponding size and pose, denoted ${}^{c_i}\mathbf{T}^d_{o_i}$ where d stands for detection for the estimation at the i^{th} frame. We associate clusters created in our SLAM system to object detections by minimizing the 3D distance between the box center and the cluster centroid. A detection is valid if its distance to the cluster centroid is lower than 2 meters. We then use consecutive detections to compute the relative twist ${}^w\boldsymbol{\xi}^d_{o_i}$ linking two poses:

$$ {}^w\boldsymbol{\xi}^d_{o_i} = \log({}^{c_{i+1}}\mathbf{T}^d_{o_{i+1}} ({}^{c_i}\mathbf{T}^d_{o_i})^{-1}) \tag{5} $$

The estimation of this twist has the advantage of being free from any drift. It can thus be used to limit the drift accumulated during tracking, similarly to the action of a loop closing step for camera tracking. The drawback however is that it is more noisy than keypoints based tracking. Indeed, the detector was trained to detect 3D objects rather to accurately estimate their pose

We show in section III-F how to inject this estimated twist in the BA to improve object tracking.

One of the main limitations of TwistSLAM [5] is the lack of canonical pose for objects. Indeed, when an object is first created, its pose is initialized with an identity matrix for the rotation, and the centroid of the cluster for the translation. Thus this pose does not relate to the pose of objects in their canonical coordinate frame. Using 3D object detection we can estimate the initial object pose. This initial pose is then updated by object tracking and by the BA, using estimated twists. We also fuse the estimated dimensions of the detection using the median of each dimension for robustness. Both the pose and the dimensions allow us to estimate a 3D bounding box for clusters.

Using this box we can associate 3D LiDAR points to clusters. We thus obtain at each timestamp a precise 3D scan of each object in the scene. We apply a generalized ICP algorithm [32] to compute the transformation between consecutive timestamps, that we denote ${}^{o_{i+1}}\mathbf{T}^r_{o_i}$ where r stands for registration. This transformation can be decomposed as:

$$ {}^{o_{i+1}}\mathbf{T}^r_{o_i} = {}^{o_{i+1}}\ \mathbf{T}^r_w \ ({}^{o_i}\mathbf{T}^r_w)^{-1} \tag{6} $$

which allows us to compute the corresponding twist in the world coordinate frame:

$$ {}^w\boldsymbol{\xi}^r_{o_i} = \log({}^{o_{i+1}}\mathbf{T}^r_{o_i}) \tag{7} $$

This twist has the advantage of being accurate compared to keypoint based twists, particularly when keypoints are difficult to extract (e.g. on small, far or textureless objects). We show in section III-F how to inject it in the global BA.

Finally, to improve the estimation of plane parameters we propose to use LiDAR points that are more precise, denser and cover more space than triangulated points. We transform LiDAR scans to world coordinates using the estimated camera camera pose and project them into the segmented image. If their class is a priori planar and their score higher than a threshold, we append them to the cluster and use them for plane fitting. We apply this strategy to the class *road*.

E. Estimating clusters geometry

The geometry of objects is an important property that we can inject in a SLAM system to improve the accuracy of 3D mapping. To estimate it we use clustered LiDAR points, that are precise and apply the method developed in DSP-SLAM [3]. DSP-SLAM uses DeepSDF [6] to represent the geometry of an object with a signed distance function generated from its latent code vector:

$$ G({}^o\mathbf{X}, \mathbf{z}) = s \tag{8} $$

where s is the SDF value computed at the 3D points position ${}^o\mathbf{X}$ and $\mathbf{z} \in \mathbb{R}^{64}$ is the latent code representing the object shape. They optimize the latent code, object pose and scale so that the generated geometry tightly fits the object scan. Doing so it is possible to reconstruct a realistic watertight mesh and use the object poses as constrains in the BA. As we already have a good estimate of the object canonical poses we propose to apply their algorithm on LiDAR points not to refine the pose but rather to refine the SLAM 3D points. We use clustered LiDAR scans to fit the latent code $\mathbf{z}$ similarly to DSP-SLAM. However contrary to DSP-SLAM we keep the object pose and scale fixed, their values being set using our own estimate of the object pose and length. Then we seek to constrain 3D map points so that they lie on the object estimated surface. As we have an estimate of the SDF value and of its derivative with respect to the 3D points position we can apply a gradient descent algorithm to project points on the surface. At the k^{th} step of the algorithm we have:

$$ {}^o\mathbf{X}^{(k+1)} = {}^o\mathbf{X}^{(k)} - \alpha^{(k)} \frac{\partial G({}^o\mathbf{X}^{(k)}, \mathbf{z})}{\partial {}^o\mathbf{X}^{(k)}} \tag{9} $$

where $\alpha^{(k)}$ is the step size, the point initial value is ${}^o\mathbf{X}^{(0)} = {}^o\mathbf{X}$ and the derivative of G is obtained through back propagation. This process is repeated for 10 steps to obtain projected points that we denote ${}^o\bar{\mathbf{X}}$. Projected points will then be used as anchors in the bundle adjustment to constrain 3D points to be coherent with the estimated geometry. LiDAR scans, which are usually more precise than points triangulated from stereo images are thus used to constrain the map.

F. Dynamic Bundle Adjustment

In TwistSLAM [5] the bundle adjustment is used to refine all object and camera poses as well as all static and dynamic point positions. Furthermore it links consecutive poses so that their twists follow a constant velocity model. Doing so dynamic points are used to improve camera

pose estimation. In our new BA we improve upon [5] by adding new regularization terms, taking into account LiDAR scans processed in three ways: using a 3D object detection network, an iterative closest point algorithm and a deep signed distance function fitting.

Our bundle adjustment cost function can be written as follows:

$$E(\{{}^{w}\boldsymbol{\xi}_{o}, {}^{c}\mathbf{T}_{w}, {}^{w}\mathbf{X}, {}^{o}\mathbf{X}\}) = \sum_{i,j} e^{i,j}_{stat} + \sum_{i,j} e^{i,j}_{dyna} + \sum_{i} e^{i}_{const} + \sum_{i} e^{i}_{reg} + \sum_{i} e^{i}_{det} + \sum_{j} e^{j}_{geo} \tag{10}$$

where $e^{i,j}_{stat}$ is the classical static reprojection error:

$$e^{i,j}_{stat} = \rho(||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w, {}^{w}\mathbf{X}_j)||_{\Sigma^{-1}_{i,j}})$$

$e^{i,j}_{dyna}$ is a dynamic reprojection error:

$$e^{i,j}_{dyna} = \rho(||{}^{i}\mathbf{x}_j - p({}^{c}\mathbf{T}_w \exp(\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i}){}^{w}\mathbf{T}_{o_i}, {}^{o}\mathbf{X})||_{\Sigma^{-1}_{i,j}})$$

where $\Sigma^{-1}_{i,j}$ is estimated using the MAD as in equation (4). e^{i}_{const} is a constant velocity model that penalizes twists variations by linking 3 consecutive poses:

$$e^{i}_{const} = \rho(||\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}} - \mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}||_{\mathbf{W}_{const}})$$

where $\mathbf{W}_{const}$ is a diagonal weight matrix used to balance the errors, tuned experimentally, ${}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}$ is the twist linking the poses $\exp(\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i}){}^{w}\mathbf{T}_{o_i}$ and $\exp(\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}}$ and ${}^{w}\tilde{\boldsymbol{\xi}}_{o_i}$ is the twist linking the poses $\exp(\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_{i-1}}){}^{w}\mathbf{T}_{o_{i-1}}$ and $\exp(\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i}){}^{w}\mathbf{T}_{o_i}$. Those twists are computed using the logmap from $\mathrm{SE}(3)$ to $\mathrm{se}(3)$ defined in [28] and can be written for ${}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}$ as:

$${}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}} = \log(\exp((\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}})(\exp(\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i}){}^{w}\mathbf{T}_{o_i})^{-1})$$

The error e^{i}_{det} penalizes the difference with twists estimated from the object detection network:

$$e^{i}_{det} = \rho(||\mathbf{\Pi}^{w}\boldsymbol{\xi}^{d}_{o_{i+1}} - \mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}||_{\mathbf{W}_{det}})$$

where $\mathbf{W}_{det}$ is a diagonal weight matrix. Similarly, the error e^{i}_{reg} penalizes the difference with twists estimated by registering consecutive point clouds:

$$e^{i}_{reg} = \rho(||\mathbf{\Pi}^{w}\boldsymbol{\xi}^{r}_{o_{i+1}} - \mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}||_{\mathbf{W}_{reg}})$$

where $\mathbf{W}_{reg}$ is a diagonal weight matrix.
Finally, the residual e^{j}_{geo} constrains points to lie on the estimated geometry surface by penalizing the difference between the position of 3D points and of their projected counterpart:

$$e^{j}_{geo} = \rho(||{}^{o}\tilde{\mathbf{X}}_j - {}^{o}\mathbf{X}_j||_{\mathbf{W}_{geo}}) \tag{11}$$

where $\mathbf{W}_{geo}$ is a diagonal weight matrix. To avoid corrupting map points due to wrong pose estimations or wrong projections, we only apply this constraint if its value is below some threshold. Note that we could also directly use the DeepSDF function $G({}^{o}\mathbf{X}, \mathbf{z})$ to compute the value of the residual and to obtain the jacobian via back propagation.
This equation refines all camera and object poses as well as all 3D points. To optimize it in real time we apply the Schur trick as the Hessian is sparse [4]. For the management of keyframes, we adopt the same strategy as TwistSLAM [5], with temporal and spatial keyframes.

IV. Experiments

In this section we present the experiments we conducted to test our approach. We evaluate both the accuracy of the camera pose estimation and of the object pose estimation.

A. Experiments details

Datasets. We evaluate our approach on the KITTI [33] tracking dataset as it contains both camera and object trajectories groundtruth. Points segmented by the network as the *unknown* class are considered to be static, as the dynamic classes are often correctly segmented.
Metrics. To evaluate the accuracy of camera pose estimation we compute the translation and rotation parts of the Relative Pose Error (RPE) [34], similarly to previous works. We also evaluate the object pose accuracy using the Absolute Translation Error (ATE) and RPE. Furthermore we evaluate precision of objects 3D bounding boxes estimations by computing the MOTP, similarly to [4] using KITTI evaluation tools. We evaluate the true positive rate (TP) and the MOTP using the projected 3D bounding box (2D), in bird view (BV) and in 3D. Those evaluations are done in the *easy* setting as in [4].

B. Camera pose estimation

In this subsection we evaluate the accuracy of our camera pose estimation which can be seen in table I. As we obtain almost exactly the same results as TwistSLAM we only show here the results on some sequences. This is not surprising as the sequences only exhibit a mild amount of dynamicity, that is well dealt even by non dynamic approaches [1]. Furthermore in this approach we rather focused on the accuracy of object tracking, that we improve compared to state of the art, as shown in the following paragraph.

C. Object pose estimation.

In this paragraph we evaluate the accuracy of our object pose estimation, we show the results in tables II and III. As we can see in table II we obtain better results in terms of object tracking accuracy for the ATE and RPE compared to [4] and [5]. We particularly improve the RPE for both the rotation and translation, which shows that adding constrains from processed LiDAR data reduces tracking drift. The TP and MOTP metrics in table III are improved compared to state of the art on average. On most sequences they show very similar scores to TwistSLAM, which can be expected as the MOTP metrics only require an overlap of 0.25 for a detection to be positive, thus an improvement of even tens of centimeters on the pose may not translate to new positive detections. A way to do so would be to decrease the number of points required for tracking to track objects for longer periods. This however is out of scope of our work as we focus on improving tracking accuracy. On some sequences however we obtain lower 3D and birdview MOTP for a similar 2D MOTP. Those differences are mainly due to wrong pose estimates that happen

TABLE I

CAMERA POSE ESTIMATION COMPARISON ON THE KITTI TRACKING DATASET.

seq	ORB-SLAM2 [1]		DynaSLAM [2]		DynaSLAM2 [4]		TwistSLAM [5]		Ours	
	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)
00	**0.04**	0.06	**0.04**	0.06	**0.04**	0.06	**0.04**	**0.05**	**0.04**	**0.05**
02	0.04	0.03	0.04	0.03	0.04	**0.02**	0.03	0.03	0.03	**0.02**
03	0.07	0.04	0.07	0.04	0.06	0.04	0.06	**0.02**	0.06	0.03
04	0.07	0.06	0.07	0.06	0.07	0.06	0.06	**0.04**	0.06	**0.04**
05	0.06	0.03	0.06	0.03	0.06	0.03	0.06	**0.02**	0.06	**0.02**
10	**0.07**	0.04	**0.07**	0.04	**0.07**	0.03	**0.07**	0.03	**0.07**	**0.02**
11	0.04	0.03	0.04	0.03	0.04	0.03	**0.03**	**0.02**	**0.03**	**0.02**
14	**0.03**	0.08	**0.03**	0.08	**0.03**	0.08	**0.03**	**0.06**	**0.03**	**0.06**
18	0.05	0.03	0.05	0.03	0.05	**0.02**	0.04	**0.02**	0.04	**0.02**
mean	0.052	0.044	0.052	0.046	0.051	0.041	**0.041**	**0.032**	**0.041**	**0.032**

when the cluster is first created far from the camera. Those differences can also be explained by the fact that TwistSLAM uses the grountruth initial bounding box of objects, which is noise-free while we use a 3D object detector. Finally on some sequences (such as 11/35 and 20/0) the additional LiDAR information allows us to improve tracking stability and thus to track on longer trajectories, increasing the MOTP.

In addition to the first figure, we also show qualitative results in figure 3.

V. CONCLUSION

In this paper we proposed TwistSLAM++, an improvement over our previous work TwistSLAM, able to track the camera in dynamic scenes and estimate the canonical pose of all potentially moving objects. By injecting LiDAR data in our pipeline, we estimate the canonical pose and size of objects using a 3D object detection network. Then, we use consecutive clustered LiDAR scans to accurately compute their relative pose using an ICP algorithm, allowing us to further constrain their movement. Finally, we use object scans to estimate the 3D geometry of objects and use it to constrain the 3D position of map points. We show that adding those constraints from a new sensor allows us to improve object pose accuracy compared to the state of the art.

REFERENCES

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

[2] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[3] J. Wang, M. Rünz, and L. Agapito, "Dsp-slam: Object oriented slam with deep shape priors," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 1362–1371.

[4] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "DynaSLAM II: Tightly-coupled multi-object tracking and SLAM," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5191–5198, 2021.

[5] M. Gonzalez, E. Marchand, A. Kacete, and J. Royan, "Twistslam: Constrained slam in dynamic environment," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6846 – 6853, 2022.

[6] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.

[7] M. Gonzalez, E. Marchand, A. Kacete, and J. Royan, "S^{3}LAM: Structured scene SLAM," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'22*, 2021.

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

[9] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu, "ClusterVO: Clustering moving instances and estimating visual odometry for self and surroundings," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020, pp. 2168–2177.

[10] J. Zhang, M. Henein, R. Mahony, and V. Ila, "VDO-SLAM: a visual dynamic object-aware SLAM system," *arXiv:2005.11052*, 2020.

[11] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D object SLAM," *IEEE Trans. on Robotics*, vol. 35, no. 4, pp. 925–938, 2019.

[12] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *IEEE Conf. on computer vision and pattern recognition*, 2013, pp. 1352–1359.

[13] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[14] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: A survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, pp. 1–11, 2017.

[15] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. M. M. Montiel, "Towards semantic SLAM using a monocular camera," in *2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2011, pp. 1277–1284.

[16] M. Rad and V. Lepetit, "BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth," in *IEEE Int. Conf. on Computer Vision*, 2017, pp. 3828–3836.

[17] M. Gonzalez, A. Kacete, A. Murienne, and E. Marchand, "L6dnet: Light 6 DoF network for robust and precise object pose estimation with small datasets," *IEEE Robotics and Automation Letters*, 2021.

[18] L. Nicholson, M. Milford, and N. Sünderhauf, "QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 1–8, 2018.

[19] E. Sucar, K. Wada, and A. Davison, "Nodeslam: Neural object descriptors for multi-view shape reconstruction," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 949–958.

[20] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments." in *Robotics: Science and Systems*, vol. 2018, 2018, p. 59.

[21] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, "Elastic lidar fusion: Dense map-centric continuous-time slam," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1206–1213.

[22] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4530–4537.

[23] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett, "Recurrent-octomap: Learning state-based map refinement for long-term semantic mapping with 3-d-lidar data," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3749–3756, 2018.

TABLE II

OBJECT POSE ESTIMATION COMPARISON ON THE KITTI TRACKING DATASET. ATE IS IN M, RPE_t IN M/M, RPE_R IN °/M

	DynaSLAM 2 [4]			TwistSLAM			TwistSLAM++		
seq / obj. id / class	ATE	RPE_t	RPE_R	ATE	RPE_t	RPE_R	ATE	RPE_t	RPE_R
03 / 1 / car	0.69	0.34	1.84	0.31	**0.10**	0.28	**0.23**	0.11	**0.19**
05 / 31 / car	0.51	0.26	13.5	0.58	0.35	**0.19**	**0.09**	**0.07**	0.28
10 / 0 / car	0.95	0.40	2.84	0.77	0.21	1.98	**0.05**	**0.10**	**0.96**
11 / 0 / car	1.05	0.43	12.51	0.17	**0.23**	0.23	**0.15**	0.28	**0.21**
11 / 35 car	1.25	0.89	16.64	0.10	0.03	0.11	0.11	**0.02**	**0.09**
18 / 2 / car	1.10	0.30	9.27	**0.21**	0.27	0.66	**0.29**	**0.09**	**0.32**
18 / 3 / car	1.13	0.55	20.05	0.15	0.21	0.56	**0.13**	**0.10**	**0.37**
19 / 63 / car	0.86	1.45	48.80	**0.28**	2.17	**1.08**	0.34	**0.21**	**0.31**
19 / 72 / car	0.99	1.12	3.36	0.16	0.05	**0.34**	**0.09**	**0.03**	0.37
20 / 0 / car	0.56	0.45	1.30	**0.17**	**0.20**	0.72	0.30	0.21	**0.35**
20 / 12 / car	1.18	0.40	6.19	**0.24**	**0.20**	1.54	0.80	0.54	**0.64**
20 / 122 / car	0.87	0.72	5.75	0.17	**0.02**	0.07	**0.16**	**0.02**	**0.06**
mean	0.93	0.61	11.83	0.26	0.32	0.68	**0.23**	**0.15**	**0.35**

TABLE III

OBJECT POSE ESTIMATION COMPARISON ON THE KITTI TRACKING DATASET. TP AND MOTP ARE IN %.

	DynaSLAM 2 [4]						TwistSLAM						TwistSLAM++					
seq / obj. id / class	2D TP	2D MOTP	BV TP	BV MOTP	3D TP	3D MOTP	2D TP	2D MOTP	BV TP	BV MOTP	3D TP	3D MOTP	2D TP	2D MOTP	BV TP	BV MOTP	3D TP	3D MOTP
03 / 1 / car	50.0	**71.79**	39.34	56.61	38.53	48.20	**58.02**	60.00	**58.02**	**60.00**	**58.02**	**45.00**	**56.79**	60.00	**56.79**	**60.00**	**56.79**	**60.00**
05 / 31 / car	28.96	**60.30**	14.48	**46.84**	11.45	34.20	**30.84**	35.00	**30.84**	35.00	**30.84**	**35.00**	30.00	26.64	16.32	14.95	16.10	14.49
10 / 0 / car	**81.63**	**73.51**	**70.41**	**47.60**	**68.37**	**40.28**	7.20	3.70	6.10	3.10	5.80	2.80	6.48	10.00	6.48	10.00	6.48	10.00
11 / 0 / car	**72.65**	**74.78**	**61.66**	**50.74**	**52.28**	**47.35**	29.61	32.50	29.61	32.50	29.61	32.50	26.82	30.00	26.82	30.00	26.82	30.00
11 / 35 car	53.17	65.25	19.05	31.95	6.35	26.02	**65.00**	**67.50**	**65.00**	**67.50**	**65.00**	**67.50**	**73.75**	**77.50**	**73.75**	**77.50**	**73.75**	**77.50**
18 / 2 / car	**86.36**	74.81	67.05	45.47	62.12	34.80	84.67	**87.50**	**84.67**	**87.50**	**84.67**	**87.50**	85.18	**87.50**	**85.18**	**87.50**	**85.18**	**87.50**
18 / 3 / car	**53.33**	**70.94**	21.75	**41.45**	16.84	**35.80**	28.19	30.00	**28.19**	30.00	**28.19**	30.00	21.83	25.00	21.83	25.00	21.83	25.00
19 / 63 / car	35.26	63.50	29.48	45.69	26.48	33.89	**65.93**	**70.00**	**65.93**	**70.00**	36.26	20.64	**65.93**	**70.00**	**65.93**	**70.00**	**65.93**	**70.00**
19 / 72 / car	**29.11**	**62.59**	**29.43**	**55.48**	**29.43**	**39.81**	16.92	20.00	16.92	20.00	16.92	20.00	5.38	10.00	5.38	10.00	5.38	10.00
20 / 0 / car	63.68	78.54	43.78	45.00	31.84	46.15	84.75	87.50	84.75	87.50	84.75	87.50	**93.22**	**97.50**	**93.22**	**97.50**	**93.22**	**97.50**
20 / 12 / car	**42.77**	**76.77**	**37.64**	**49.29**	**36.23**	**40.81**	14.24	17.5	13.91	17.45	13.04	17.25	32.75	37.5	32.75	37.5	32.75	37.5
20 / 122 / car	34.90	78.76	34.51	48.05	29.02	44.43	**84.94**	**87.50**	**84.94**	**87.50**	**84.94**	**87.50**	**84.94**	**87.50**	**84.94**	**87.50**	**84.94**	**87.50**
mean	**55.15**	**70.96**	39.05	47.01	34.08	39.31	45.53	49.89	47.41	49.84	44.84	43.18	49.42	51.6	**47.45**	**50.62**	**47.43**	**50.58**

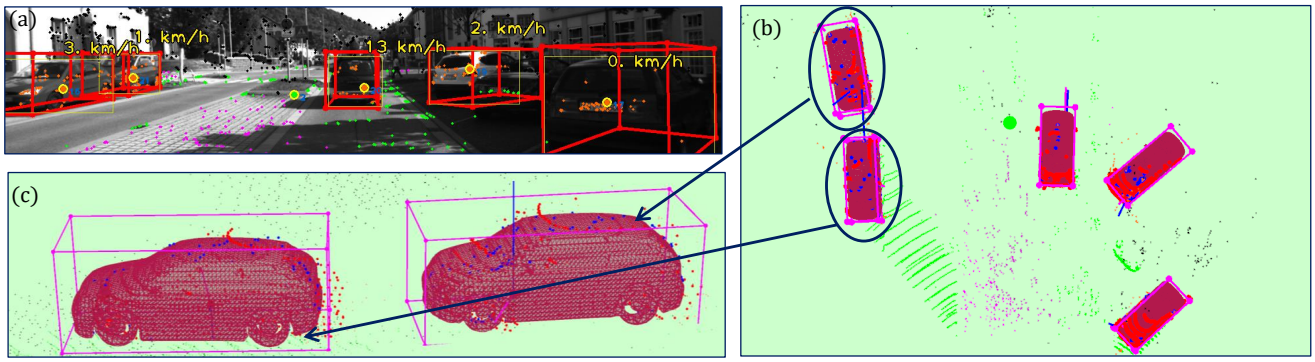

Fig. 3. (a) Frame with detected objects, bounding boxes and speed.(b) Map with tracked objects, seen from above. (c) Mesh of reconstructed cars with bounding boxes, LiDAR points (red) and projected points on the mesh (blue).

[24] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss, "Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6529–6536, 2021.

[25] J. Jeong, T. S. Yoon, and J. B. Park, "Multimodal sensor-based semantic 3d mapping for a large-scale environment," *Expert Systems with Applications*, vol. 105, pp. 1–10, 2018.

[26] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[27] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[28] J.-L. Blanco, "A tutorial on se (3) transformation parameterizations and on-manifold optimization," *University of Malaga, Tech. Rep*, vol. 3, p. 6, 2010.

[29] E. Malis and E. Marchand, "Experiments with robust estimation techniques in real-time robot vision," in *2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 223–228.

[30] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Eur. Conf. on Computer Vision*, 2020, pp. 402–419.

[31] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3d single stage object detector," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 040–11 048.

[32] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[33] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conf. on computer vision and pattern recognition*, 2012, pp. 3354–3361.

[34] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *2018 IEEE/RSJ Int Conf. on Intelligent Robots and Systems (IROS)*, 2018, pp. 7244–7251.