Three-Dimensional Extended Object Tracking and Shape Learning Using Gaussian Processes

Murat Kumru and Emre Özkan, *Member, IEEE*

***Abstract*—In this study, we investigate the problem of tracking objects with unknown shapes using three-dimensional (3D) point cloud data. We propose a Gaussian process-based model to jointly estimate object kinematics, including position, velocity and orientation, together with the shape of the object in an online fashion. We describe the unknown shape by a radial function in 3D, and induce a correlation structure via a Gaussian process. Furthermore, we propose an efficient algorithm to reduce the computational complexity of working with 3D data. This is accomplished by casting the tracking problem into projection planes which are attached to the object's local frame. The proposed methods provide an analytical expression for the object shape together with confidence intervals. The confidence intervals, which quantify the uncertainty in the shape estimate, can later be used for solving the gating and association problems inherent in object tracking. The performance of the methods is demonstrated both on simulated and real data. The results are compared with an existing random matrix model, which is commonly used for extended object tracking in the literature.**

***Index Terms*—Extended Object Tracking, Gaussian Processes, Shape Learning, Point Cloud Data.**

I. Introduction

Object tracking can be described as making inference about the unknown kinematics of an object using sequentially available noisy sensor data. The problem is explicitly referred to as *point object tracking* when the number of measurements returned from the object is limited to be at most one per sensor scan. On the other hand, it is dubbed as *extended object tracking* (EOT) when the object potentially originates multiple measurements at a single scan. In the latter case, the measurements not only convey information about the kinematics, e.g., position, velocity and orientation, but they also naturally reveal characteristics of the latent shape/extent. In order to systematically assimilate this information, a solid body of EOT literature relying on various extent representations has been developed, [1]. These representations exhibit significant variance in their compactness and expressive power. For example, a group of EOT algorithms imposes simple shape models such as a circle, a rectangle, or an ellipse. These essentially achieve extent modeling with only a few parameters at the cost of limited potential for shape description. A substantial fraction of studies in robotics and autonomous driving resides in this category as they utilize the bounding box model. [2] considers pedestrian tracking by processing a partition of 3D point cloud data. They model the extent of a pedestrian by a bounding box. In [3], tracking for autonomous driving in urban settings is addressed. They formulate the problem in two-dimensional (2D) motion space regarding the bounding box model for objects. Another popular line of research, named random matrix approach, approximates the object extent by an ellipse, [4]–[7]. Random hyper-surface models (RHM), on the other hand, formulates the EOT problem via a more flexible extent representation for star-convex objects, [8], [9]. The model is based on the Fourier series expansion of the spatial extent, and the coefficients of the expansion are estimated together with the kinematics. A specific adaptation of RHM to tackle people tracking using depth data is presented in [10], and therein 3D shape is approximated as a cylinder. From a similar perspective, a more general tracking framework based on the assumption that 3D object surface can be constructed by some transformations, e.g., translation, rotation, of a plane curve is proposed in [11]. However, this approach necessitates a special formulation of the recursive estimator in accordance with the particular transformation considered. This devalues the virtue of the model for a standard tracking application as there is typically no prior information about the object shape.

With its favorable analytical properties and close connections to the Bayesian paradigm, a Gaussian process (GP) facilitates modeling of unknown functions. With this in mind, the authors describe the latent extent of star-convex objects by GP in [12] and [13]. These models estimate the pose of the object while learning its arbitrary shape simultaneously. This approach is applied to the multiple object tracking problem in [14]. Several adaptations of the GP model have also been investigated for specific application settings. Multiple sensor fusion problem in automotive scenarios is addressed in [15]. [16] suggests a tracking filter processing measurements from a high-resolution automotive radar; it is built upon the GP extent representation. [17] focuses on object classification making use of the extent estimates produced by a GP based tracking algorithm. In [18], measurement models to leverage both negative and positive information, i.e., where the object should and should not exist, are developed with the aim of effectively exploiting the laser range scanners in tracking. These models rely on the GP description of the object extent.

An alternative approach for tracking is to make use of a grid representation, which approximates the continuous space as a collection of small-sized units. For instance, so-called occupancy grids, which were initially motivated by the mapping problem, have been adapted to tracking of dynamic objects. In [19] and [20], local occupancy grids, which are fixed to the local coordinate frame of the object, are utilized for tracking arbitrarily shaped objects. Although this representation can potentially lead to a rich description of

The authors are with the Department of Electrical and Electronics Engineering, Middle East Technical University, 06531, Ankara, Turkey (e-mail: kumru@metu.edu.tr; emreo@metu.edu.tr).

the latent shape, it is limited by the inherent assumption that the individual cells are mutually independent. In particular, this assumption possibly causes the inference to disregard the consistent spatial patterns which are locally intrinsic in the underlying object shape. Similarly, processing environmental representations expressed by grids, neural networks are also applied to object tracking, [21], [22]. All of these methods come with a fundamental trade-off between spatial resolution and memory consumption/computational load. Consequently, a vast majority of the existing literature alleviates these issues by formulating the tracking problem in 2D space. Besides, another significant inconvenience associated with the grid-based approach is the selection of grid size and cell resolution without having a priori information about the object.

All of the approaches discussed so far prescribes object tracking to infer some latent variables which are meant to express the object extent, e.g., Fourier coefficients, random matrices, occupancy grids. In contrast, it is also possible to refer to an intuitive extent description which is formed as a collection of point measurements. Using this notion, [23] suggests to jointly estimate the self-motion and the track's motion using the Iterative Closest Point (ICP) algorithm and a Kalman filter where the appearance of the object is stored as an aggregation of LIDAR measurements and corresponding features. [24] basically combines LIDAR data and color information, and offers an inference method, named annealed dynamics histograms, based on the iterative sampling of the state space. They also do not model the shape explicitly instead integrate measurements over time to obtain a point cloud representation of the shape. Likewise, the object extent is expressed as an accumulated point cloud in [25]. As a principal difference, they formulate tracking as a batch optimization on a sliding window of measurements rather than applying a standard Bayesian estimator. These methods facilitate joint tracking and shape learning of arbitrary objects in the presence of continuously available, high-precision and informative measurements. However, as they do not feature a principled representation of the underlying shape, there arise robustness issues with the sparsity of the measurements due to increasing distance, change of the vantage point and occlusions. In addition, dependence on the ICP algorithm to align point clouds renders the tracking performance sensitive to the initialization errors. Lastly, the storage and computational requirements scale with the size of the object extent.

Motivated by several applications, such as medical, robotics, computer vision and mechanical production, 3D modeling of objects has been an active research topic for many years. The proposed algorithms achieve 3D shape representations by means of various tools, e.g., occupancy maps, polygon meshes, implicit surfaces or collection of point clouds. A complete review of this broad field is well beyond the scope of this paper; instead, we focus our attention to those exploiting a GP-based approach. Being a non-parametric model, GP establishes a convenient basis for probabilistic modeling of an arbitrary 3D shape. In this regard, the central idea of learning an implicit surface description by a GP was first suggested in [26]. With a particular focus on the robotic grasping problem, [27] and [28] consider how to exploit information collected by various sensors, such as visual, haptic and laser, relying on this shape representation. Both of these studies make use of grid structures in the Cartesian coordinates. [29] examines systematic ways to incorporate prior knowledge about the object shape into the mentioned model. In this study, an iterative rendering procedure is employed to reconstruct the corresponding surface representation which is essentially embedded as the solution of the implicit function. Note that all of these studies address the modeling of stationary objects where full information about the true pose is available. Besides, none of these efforts accounts for the constraints of an online tracking problem as they either require batch processing of the measurements, [29], or maintain the implicit function on a 3D grid, [27], [28], which do not provide an efficient base for a recursive application.

In this study, we consider the problem of tracking 3D objects while simultaneously learning their arbitrary shapes using point cloud measurements. Estimating arbitrary shapes from noisy 3D point cloud data is a challenging task, and the problem gets even more severe when the objects are in motion. This is mainly due to the inherent interdependence between the pose and the shape description. Therefore, reliable estimation of one necessitates precise information about the other. That being said, we formulate the problem as a joint estimation of both motion and shape variables where we can explicitly account for the correlation in a stochastic framework. In this pursuit, we first derive two novel probabilistic representations of the 3D surface, which can be utilized alternatively. For the first one, the unknown 3D surface is described by a radial function defined for all spherical angles. The second approach exploits the correspondence between a 3D shape and its projections onto multiple planes, and thus expresses the original 3D surface by a collection of 2D contours of these projections. Then, the shape representations are achieved by casting the above descriptions into the probabilistic domain by GP modeling without imposing any parametric form. By doing so, we attain a flexible basis to estimate various different-shaped objects. Moreover, measurement models are developed based on an efficient approximation of the GP. Finally, the kinematics and the object shape are jointly inferred by an extended Kalman filter regarding a unified state space model.

In this document, we extend our previous work on 3D object tracking, [30], and present rigorous derivations of the models discussed. Additionally, the performance of the proposed algorithms is comprehensively evaluated on both simulated and real measurements in a comparative manner.

The rest of this document is organized as follows. We introduce a 3D extent description relying on the radial function in Section II. The basics of the GP regression together with an efficient recursive approximation are briefly reviewed in Section III. A GP model for the given extent representation is developed in Section IV. Section V derives a unified state space model comprising of both the kinematic and extent variables, and the inference method regarding this model is described in Section VI. Subsequently, we propose the alternative approach relying on the object projections in Section VII. The performance of the suggested algorithms are demonstrated in Section VIII. Section IX concludes the article.

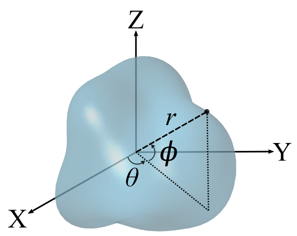


Fig. 1: Object extent description in spherical coordinates.

II. Extent Model for 3D Objects

To facilitate effective shape[1] learning, a suitable description of the object extent which meets the requirements of the problem is to be adopted. Specifically, it is required to have high representational power to be able to express the latent extent of an arbitrary object. In addition, it should be sufficiently compact so that it will enable an efficient implementation of the online tracking algorithm.

In this regard, we model the object shape in spherical coordinates by means of a radial function $f(\theta, \phi)$. The arguments of this function are the azimuth, $\theta \in [-\pi, \pi]$, and the elevation angles, $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, and the output, r, is the distance between the center of the object and the point on the surface at the corresponding spherical angle pair, i.e., $r = f(\theta, \phi)$. Fig. 1 illustrates the representation for an arbitrary object.

Notice that this representation summarizes the 3D shape exclusively by the external boundary (surface) of the object considering that the point cloud measurements are merely originated from the surface. Additionally, it implicitly assumes that the latent shape is star-convex[2]. This assumption does not introduce a strict limitation as star-convex shapes present an adequately broad class for object tracking applications.

The main concern of the upcoming sections is to construct a unified state space model to serve as a basis for the joint estimation of the kinematics and the shape of the object. The corresponding state vector includes both the kinematics and a parametric description of the given extent model. This description will basically be obtained by developing a GP model for the radial function. In particular, we adopt a recursive approximation of GP modeling to avoid associated computational difficulties. This approach primarily accomplishes a probabilistic representation of the latent extent in a principled manner. It conveniently accounts for the inherent spatial correlation within the object surface. Besides, the GP model is able to maintain the local uncertainty information of the extent which becomes vital for robust tracking and shape learning in scenarios including occlusions and sparse sampling.

The next section first briefly introduces the standard GP regression and then elaborates on its recursive approximation.

[1]Throughout this paper, we deliberately use the terms, extent and shape interchangeably to contribute to a common understanding between researchers from different fields, such as tracking, computer vision and robotics.

[2]A set $\mathcal{S}$ is star-convex with respect to the origin if each line segment from the origin to any point in $\mathcal{S}$ is fully contained in $\mathcal{S}$.

III. Gaussian Processes

A Gaussian Process (GP) is a stochastic model which specifies a probability distribution in the function space for a function $f(\cdot)$, [31]. We hereby engage a GP to model the radial function expressing the unknown extent of the object. A GP is uniquely defined by the mean $\mu(u)$ and the covariance function $k(u, u')$ defined as

$$\mu(u) = \mathbb{E}[f(u)], \tag{1a}$$

$$k(u, u') = \mathbb{E}[(f(u) - \mu(u))(f(u') - \mu(u'))^\top]. \tag{1b}$$

The corresponding GP model is denoted as

$$f(u) \sim \mathcal{GP}(\mu(u), k(u, u')),$$

where u is the input of the function.

A GP can also be interpreted as a collection of random variables, any finite number of which have a joint Gaussian distribution that is consistent with the specified mean and covariance functions. The joint distribution of the function evaluations at the inputs, $u_1, ..., u_N$, reads as

$$\begin{bmatrix} f(u_1) \\ \vdots \\ f(u_N) \end{bmatrix} \sim \mathcal{N}(\boldsymbol{\mu}, K), \text{ where } \boldsymbol{\mu} = \begin{bmatrix} \mu(u_1) \\ \vdots \\ \mu(u_N) \end{bmatrix}, \tag{2a}$$

and

$$K = \begin{bmatrix} k(u_1, u_1) & \dots & k(u_1, u_N) \\ \vdots & & \vdots \\ k(u_N, u_1) & \dots & k(u_N, u_N) \end{bmatrix}. \tag{2b}$$

A. Gaussian Process Regression

Prior belief about the unknown function encoded by the GP can be conveniently conditioned on the information provided by observations. For this purpose, a noisy observation m can be described as the true function output perturbed by an independent Gaussian noise e,

$$m = f(u) + e, \quad e \sim \mathcal{N}(0, R). \tag{3}$$

Assume that we seek for the refined distribution of the function values $\mathbf{f} \triangleq [f(u_1^\mathbf{f}) \; \dots \; f(u_{N^\mathbf{f}}^\mathbf{f})]^\top$ at the inputs $\mathbf{u^f} \triangleq [u_1^\mathbf{f} \; \dots \; u_{N^\mathbf{f}}^\mathbf{f}]^\top$. Available measurements are denoted by $\mathbf{m} \triangleq [m_1 \; \dots \; m_N]^\top$ which are originated from the inputs $\mathbf{u} \triangleq [u_1 \; \dots \; u_N]^\top$. The GP model (2) together with the measurement model (3) leads to the following joint distribution,

$$\begin{bmatrix} \mathbf{m} \\ \mathbf{f} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{u}, \mathbf{u}) + I_N \otimes R & K(\mathbf{u}, \mathbf{u^f}) \\ K(\mathbf{u^f}, \mathbf{u}) & K(\mathbf{u^f}, \mathbf{u^f}) \end{bmatrix}\right), \tag{4a}$$

$$\text{where } K(\mathbf{u}, \mathbf{u^f}) = \begin{bmatrix} k(u_1, u_1^\mathbf{f}) & \dots & k(u_1, u_{N^\mathbf{f}}^\mathbf{f}) \\ \vdots & & \vdots \\ k(u_N, u_1^\mathbf{f}) & \dots & k(u_N, u_{N^\mathbf{f}}^\mathbf{f}) \end{bmatrix}, \tag{4b}$$

I_N indicates an N-by-N identity matrix, and $\otimes$ is the Kronecker product. Notice that the mean function is set to be identically zero for the sake of brevity. For an arbitrary mean function, the derivation of the regression closely follows this specific case, [31].

By regarding the joint Gaussian distribution, the conditional distribution $p(\mathbf{f}|\mathbf{m})$ is derived as

$$p(\mathbf{f}|\mathbf{m}) \sim \mathcal{N}(A\mathbf{m}, P), \tag{5a}$$

where

$$A = K(\mathbf{u}^{\mathbf{f}}, \mathbf{u})K_y^{-1}, \tag{5b}$$
$$P = K(\mathbf{u}^{\mathbf{f}}, \mathbf{u}^{\mathbf{f}}) - K(\mathbf{u}^{\mathbf{f}}, \mathbf{u})K_y^{-1}K(\mathbf{u}, \mathbf{u}^{\mathbf{f}}), \tag{5c}$$
$$K_y = K(\mathbf{u}, \mathbf{u}) + I_N \otimes R. \tag{5d}$$

B. Recursive Gaussian Process Regression

The GP regression necessitates to process all available information in a single batch as the complete measurement vector $\mathbf{m}$ and the corresponding covariance matrix K_y appear in (5). While this attribute can be interpreted to be the primary strength of GP modeling since it enables to draw conclusions directly from the observations, it also poses some computational problems for certain settings. Specifically, in object tracking, the aim is to compute the posterior density $p(\mathbf{f}|m_{1:k})$ at time k using measurements which are acquired sequentially in time. For this problem, online inference can be achieved by a recursive algorithm which efficiently updates the posterior by considering only the newly available measurements. In this respect, the standard GP regression is not applicable due to its increasing needs for computational sources and memory storage with the accumulation of measurements over time. Therefore, we hereby rely on an approximation of the GP, that basically summarizes the original model at a finite set of basis points at which the model is maintained recursively. The approximation was initially proposed in [32], [33], and then applied to the object tracking problem in [12].

In this approach, the objective is to derive a formulation of the posterior distribution $p(\mathbf{f}|m_{1:N})$ that enables recursive implementation. To this end, the posterior is first expanded as the collection of the following terms by applying the Bayes' law iteratively.

$$p(\mathbf{f}|m_{1:N}) \propto p(m_N|\mathbf{f}, m_{1:N-1})p(\mathbf{f}|m_{1:N-1}), \tag{6a}$$
$$\propto \cdots \underbrace{p(m_k|\mathbf{f}, m_{1:k-1}) \cdots p(\mathbf{f})}_{p(\mathbf{f}|m_{1:k})}. \tag{6b}$$

At this point, $\mathbf{f}$ is assumed to provide the sufficient statistics for m_k. Under this assumption, m_k conditioned on $\mathbf{f}$ becomes independent from all previous measurements, $m_{1:k-1}$, i.e.,

$$p(m_k|\mathbf{f}, m_{1:k-1}) \approx p(m_k|\mathbf{f}). \tag{7}$$

Notice that the assumption becomes exact if the inputs of m_k form a subset of the inputs of $\mathbf{f}$. Moreover, it can be claimed to a reasonable approximation when the distance between the inputs of m_k and the inputs of $\mathbf{f}$ is sufficiently small compared to the characteristic lengthscale of the covariance function. In this study, we want to model the unknown radial function whose input is the spherical angle pair. As the set of the possible input values has a well-defined boundary, it is possible to sufficiently sample this set by a finite number of basis points which can be located equidistantly.

The above assumption leads to a setting where we essentially treat $\mathbf{f}$ to be the latent variable and the measurements provide noisy observations of it. Accordingly, once the measurement likelihood and the initial prior densities are defined, it is possible to apply recursive Bayesian inference for $\mathbf{f}$. With this in mind, we simply refer to the underlying GP model to offer these densities in a principled way. At first, the joint distribution of the measurement m_k and $\mathbf{f}$ is revealed by the definition of the GP as

$$\begin{bmatrix} m_k \\ \mathbf{f} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(u_k, u_k) + R & K(u_k, \mathbf{u}^{\mathbf{f}}) \\ K(\mathbf{u}^{\mathbf{f}}, u_k) & K(\mathbf{u}^{\mathbf{f}}, \mathbf{u}^{\mathbf{f}}) \end{bmatrix}\right). \tag{8}$$

Then, the joint distribution together with (5) computes the following likelihood and prior densities,

$$p(m_k|\mathbf{f}) = \mathcal{N}(m_k; H_k^{\mathbf{f}}\mathbf{f}, R_k^{\mathbf{f}}), \tag{9a}$$
$$p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, P_0^{\mathbf{f}}), \tag{9b}$$

where

$$H_k^{\mathbf{f}} = H^{\mathbf{f}}(u_k) = K(u_k, \mathbf{u}^{\mathbf{f}})[K(\mathbf{u}^{\mathbf{f}}, \mathbf{u}^{\mathbf{f}})]^{-1}, \tag{9c}$$
$$R_k^{\mathbf{f}} = R^{\mathbf{f}}(u_k) = k(u_k, u_k) + R - K(u_k, \mathbf{u}^{\mathbf{f}})[K(\mathbf{u}^{\mathbf{f}}, \mathbf{u}^{\mathbf{f}})]^{-1}K(\mathbf{u}^{\mathbf{f}}, u_k), \tag{9d}$$
$$P_0^{\mathbf{f}} = K(\mathbf{u}^{\mathbf{f}}, \mathbf{u}^{\mathbf{f}}). \tag{9e}$$

The structure of (9) allows us to construct the following state space model to which a standard Kalman filter can be applied for recursive inference, [12].

$$\mathbf{x}_{k+1}^{\mathbf{f}} = \mathbf{x}_k^{\mathbf{f}}, \tag{10a}$$
$$m_k = H^{\mathbf{f}}(u_k)\ \mathbf{x}_k^{\mathbf{f}} + e_k^{\mathbf{f}}, \quad e_k^{\mathbf{f}} \sim \mathcal{N}(0, R^{\mathbf{f}}(u_k)), \tag{10b}$$
$$\mathbf{x}_0^{\mathbf{f}} \sim \mathcal{N}(\mathbf{0}, P_0^{\mathbf{f}}), \tag{10c}$$

where $\mathbf{x}_k^{\mathbf{f}} \triangleq \mathbf{f} = [f(u_1^{\mathbf{f}}) \ \dots \ f(u_{N^{\mathbf{f}}}^{\mathbf{f}})]^\top$.

The benefits of having such a state space model for the object extent are twofold: first, presumed dynamical characteristics of the extent can easily be encoded into the model; second, it can simply be augmented by another state space model to obtain a unified representation.

In this study, we consider the following dynamical model to express the evolution of the unknown object shape:

$$\mathbf{x}_{k+1}^{\mathbf{f}} = F^{\mathbf{f}}\mathbf{x}_k^{\mathbf{f}} + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, Q^{\mathbf{f}}), \tag{11a}$$

where

$$F^{\mathbf{f}} = e^{-\alpha T}I, \quad Q^{\mathbf{f}} = (1 - e^{-2\alpha T})K(\mathbf{u}^{\mathbf{f}}, \mathbf{u}^{\mathbf{f}}). \tag{11b}$$

This model introducing a forgetting factor α with process noise $\mathbf{w}_k$ basically accounts for the possible changes in the object extent. Therefore, it potentially facilitates tracking of nonrigid objects. In addition, it enables recovery from erroneously integrated shape information which might occur due to temporal errors in the pose estimates, especially during the initial phases of the tracking.

IV. GP Modeling of Object Extent

In this section, the radial function which expresses the object extent is to be modeled via a GP. By doing so, we will be able to facilitate effective shape learning in the probabilistic framework by using incomplete and noisy point measurements.

GP lends itself conveniently to extent modeling since it is naturally able to describe the spatial correlation between different sections the object surface. In addition, it maintains local uncertainty information associated with the object surface which is vital for accurate gating and association of the measurements leading to robust tracking performance.

As a GP model is uniquely defined by its mean and covariance functions, the main focus of this discussion is to properly construct these functions regarding the characteristics of the extent representation. Note that we hereby put forward a generic approach to be able to apply to arbitrarily shaped objects; however, prior knowledge about the object shape can also be systematically incorporated by adjusting these functions accordingly, as in [29].

As discussed earlier, the output of the radial function is the distance r, and the input is the pair of azimuth and elevation angles (θ, ϕ), i.e., $r = f(\theta, \phi)$. For notational simplicity, the pair (θ, ϕ) is assigned to $\boldsymbol{\gamma}$, i.e., $\boldsymbol{\gamma} \triangleq (\theta, \phi)$ and $r = f(\boldsymbol{\gamma})$. Therefore, we denote the mean and covariance functions as $\mu(\boldsymbol{\gamma})$ and $k(\boldsymbol{\gamma}, \boldsymbol{\gamma}')$, respectively, and $f(\boldsymbol{\gamma}) \sim \mathcal{GP}(\mu(\boldsymbol{\gamma}), k(\boldsymbol{\gamma}, \boldsymbol{\gamma}'))$ indicates the GP model.

A. Mean Function

The mean function is modeled to be an unknown constant having a normal distribution, i.e.,

$$\mu(\boldsymbol{\gamma}) = r, \quad \text{where } r \sim \mathcal{N}(\mu_r, \sigma_r^2). \tag{12}$$

Then, we integrate out the uncertainty in the mean and obtain the GP model in the form of

$$f(\boldsymbol{\gamma}) \sim \mathcal{GP}(\mu_r, k(\boldsymbol{\gamma}, \boldsymbol{\gamma}') + \sigma_r^2). \tag{13}$$

B. Covariance Function

Selection of the covariance function for a GP is of great importance since it basically determines the characteristics of the functions to be learned. In this application, it is required to conform to the fundamentals of 3D object geometry as it encodes the spatial correlation between points on the extent.

The design of the covariance function is initiated from the exponentiated quadratic function, which is accepted to be the de facto choice in various fields, [31], as

$$k(\boldsymbol{\gamma}, \boldsymbol{\gamma}') = \sigma_f^2 e^{-\frac{d^2(\boldsymbol{\gamma}, \boldsymbol{\gamma}')}{2l^2}}, \tag{14}$$

where σ_f^2 stands for the prior variance, l is the lengthscale and $d(\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ calculates the relative distance between two inputs.

The unconventional aspect of the employed covariance function is the formulation of the distance, $d(\boldsymbol{\gamma}, \boldsymbol{\gamma}')$. It is simply utilized to imply higher correlation for closer regions compared to those which are rather separated. An immediate option for the distance definition could be the Euclidean distance, i.e., $d(\boldsymbol{\gamma}, \boldsymbol{\gamma}') = \|\boldsymbol{\gamma} - \boldsymbol{\gamma}'\|$. However, being inconsistent with the basics of the spherical geometry, it leads to erroneous correlation patterns for the extent defined in the spherical coordinates. As a simple example, lets consider $\boldsymbol{\gamma} = \left(0, \frac{\pi}{2}\right)$ and $\boldsymbol{\gamma}' = \left(\pi, \frac{\pi}{2}\right)$ both pointing to the upper pole of a sphere. For these inputs, the Euclidean distance is computed as π which is also equal to the distance for any two spherical angles pointing exactly to opposite directions, e.g., the upper and the lower poles, i.e., $\boldsymbol{\gamma} = \left(0, \frac{\pi}{2}\right)$ and $\boldsymbol{\gamma}' = \left(0, -\frac{\pi}{2}\right)$. To overcome this problem, we set $d(\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ to be the angle of the shortest arc connecting the input points on a sphere. The analytical expression for this definition can be written as

$$\begin{aligned} d(\boldsymbol{\gamma}, \boldsymbol{\gamma}') = \arccos\big(&\cos(\phi)\cos(\phi')\cos(\theta)\cos(\theta') \\ &+ \cos(\phi)\cos(\phi')\sin(\theta)\sin(\theta') + \sin(\phi)\sin(\phi')\big), \end{aligned} \tag{15}$$

where $\boldsymbol{\gamma} = (\theta, \phi)$ and $\boldsymbol{\gamma}' = (\theta', \phi')$. Notice that with this formulation, the distance takes values within the interval $[0, \pi]$, and any coincident angle pair is mapped to 0 while opposite directions compute π.

Finally, the total covariance function is attained as

$$\begin{aligned} k_{total}(\boldsymbol{\gamma}, \boldsymbol{\gamma}') &= k(\boldsymbol{\gamma}, \boldsymbol{\gamma}') + \sigma_r^2, \\ &= \sigma_f^2 e^{-\frac{d^2(\boldsymbol{\gamma}, \boldsymbol{\gamma}')}{2l^2}} + \sigma_r^2. \end{aligned} \tag{16}$$

V. State Space Model

In this section, we will develop a state space model to be regarded by the inference method which realizes object tracking. This model is based on the state vector involving both the kinematics and the extent representation of the object. In this setting, joint estimation of this aggregated state variables will be accomplished by a single inference algorithm. In other words, the idea of leveraging the latent shape information for object tracking is basically realized by this formulation.

The state vector is defined as $\mathbf{x}_k \triangleq \begin{bmatrix} \bar{\mathbf{x}}_k^\top & \mathbf{f}_k^\top \end{bmatrix}^\top$ where $\bar{\mathbf{x}}_k^\top \triangleq \begin{bmatrix} \mathbf{c}_k^\top & \mathbf{v}_k^\top & \mathbf{q}_k^\top \end{bmatrix}^\top$ includes the kinematic variables and $\mathbf{f}_k$ indicates the extent; $\mathbf{c}_k$ is the center of the object and $\mathbf{v}_k$ stands for the velocity of the center; $\mathbf{q}_k$ is the unit quaternion vector, i.e., $\mathbf{q}_k = [q_{0k} \; q_{1k} \; q_{2k} \; q_{3k}]^\top$ and $\|\mathbf{q}_k\| = 1$, expressing the orientation of the local frame with respect to the global frame.

The definition of the state vector makes use of two distinct coordinate frames as shown in Fig. 2. The first one is the global coordinate frame which is fixed to the sensor; the second one is the local coordinate frame which is anchored at the center of the object. As the local frame performs exactly the same motion with the object, it allows to describe the extent in a consistent manner. Accordingly, the extent information is maintained in the local coordinate frame while the motion dynamics are estimated in the global coordinate frame.

An overview of the state space model is given by the following set of equations,

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, Q_k), \tag{17a}$$

$$\mathbf{m}_{k,l} = \mathbf{h}_{k,l}(\mathbf{x}_k) + \mathbf{e}_{k,l}, \quad \mathbf{e}_{k,l} \sim \mathcal{N}(\mathbf{0}, R_{k,l}), \tag{17b}$$

$$\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, P_0), \tag{17c}$$

where the process, $\mathbf{w}_k$, and the measurement noise, $\mathbf{e}_{k,l}$, are modeled as i.i.d. Gaussian random variables. The following subsections will introduce the details of these equations starting from the derivation of the measurement model.

A. Measurement Model

In general, we assume that there are multiple point measurements returned from an object at time k, which can be represented by the set $\{\mathbf{m}_{k,i}\}_{i=1}^{n_k}$. A single measurement can be expressed as

$$\mathbf{m}_{k,l} = \mathbf{c}_k + \mathbf{p}((\theta,\phi)_{k,l}) f((\theta,\phi)_{k,l}) + \bar{\mathbf{e}}_{k,l}, \qquad \bar{\mathbf{e}}_{k,l} \sim \mathcal{N}(\mathbf{0}, \bar{R}). \tag{18}$$

$\mathbf{c}_k$ is the center of the object at time k; $(\theta,\phi)_{k,l}$ is the spherical angle pair corresponding to the point on the object surface that originates $\mathbf{m}_{k,l}$; $\mathbf{p}((\theta,\phi)_{k,l})$ is the unit-length vector in the direction specified by $(\theta,\phi)_{k,l}$; $f(\cdot)$ is the radial function; and $\bar{\mathbf{e}}_{k,l}$ stands for the zero-mean Gaussian measurement noise with covariance $\bar{R}$.

Notice that for the measurement $\mathbf{m}_{k,l}$, the underlying spherical angles $(\theta,\phi)_{k,l}$ are not immediately available; instead, they can be expressed as a function of the measurement and the object pose. To write this function, we first need an intermediate representation of the original measurement $\mathbf{m}_{k,l}$ resolved in the local coordinate frame. It is simply obtained by successive transformations of translation and rotation as

$$\mathbf{m}_{k,l}^L(\mathbf{c}_k, \mathbf{q}_k) = \underbrace{R_G^L(\mathbf{q}_k)}_{Rotation} \underbrace{(\mathbf{m}_{k,l} - \mathbf{c}_k)}_{Translation}. \tag{19}$$

$R_G^L(\mathbf{q})$ is the rotation matrix from the global frame to the local frame defined by

$$R_G^L(\mathbf{q}) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix}. \tag{20}$$

The transformations referred in (19) are illustrated in Fig. 2. $\mathbf{m}_{k,l}^L$ can be interpreted as a pseudo-measurement in the local frame, and it will only be exploited to find out the spherical angle pair, $\boldsymbol{\gamma}_{k,l} \triangleq (\theta_{k,l}, \phi_{k,l})$, associated to $\mathbf{m}_{k,l}$. Please note that the figure depicts the object by a spherical shape to provide a straightforward description although the procedure applies to any arbitrarily shaped object. Then, $\boldsymbol{\gamma}_{k,l}$ is easily computed by converting $\mathbf{m}_{k,l}^L$ into the spherical coordinates.

With the aim of attaining a more precise formulation, the measurement equation in (18) can be rewritten to explicitly indicate that the spherical angle pair is a function of $\mathbf{c}_k$ and $\mathbf{q}_k$ as implied by (19),

$$\mathbf{m}_{k,l} = \mathbf{c}_k + \mathbf{p}_{k,l}(\mathbf{c}_k) f\left(\boldsymbol{\gamma}_{k,l}(\mathbf{c}_k, \mathbf{q}_k)\right) + \bar{\mathbf{e}}_{k,l}. \tag{21}$$

Additionally, the unit-length vector $\mathbf{p}_{k,l}(\mathbf{c}_k)$, starting from the object center pointing towards the measurement, is defined by

$$\mathbf{p}_{k,l}(\mathbf{c}_k) = \frac{\mathbf{m}_{k,l} - \mathbf{c}_k}{\|\mathbf{m}_{k,l} - \mathbf{c}_k\|}. \tag{22}$$

Finally, the GP representation as specified in (10) is substituted for the radial function resulting in the following measurement model:

$$\begin{aligned} \mathbf{m}_{k,l} &= \mathbf{c}_k + \mathbf{p}_{k,l}(\mathbf{c}_k)\left[H^{\mathbf{f}}\left(\boldsymbol{\gamma}_{k,l}(\mathbf{c}_k,\mathbf{q}_k)\right)\mathbf{f}_k + \mathbf{e}_{k,l}^{\mathbf{f}}\right] + \bar{\mathbf{e}}_{k,l} \\ &= \underbrace{\mathbf{c}_k + \tilde{H}_l(\mathbf{c}_k,\mathbf{q}_k)\mathbf{f}_k}_{=\mathbf{h}_{k,l}(\mathbf{x}_k)} + \underbrace{\mathbf{p}_{k,l}(\mathbf{c}_k)\mathbf{e}_{k,l}^{\mathbf{f}} + \bar{\mathbf{e}}_{k,l}}_{=\mathbf{e}_{k,l}} \\ &= \mathbf{h}_{k,l}(\mathbf{x}_k) + \mathbf{e}_{k,l}, \quad \mathbf{e}_{k,l} \sim \mathcal{N}(\mathbf{0}, R_{k,l}), \end{aligned} \tag{23}$$

where

$$\tilde{H}_l(\mathbf{c}_k,\mathbf{q}_k) = \mathbf{p}_{k,l}\ H^{\mathbf{f}}\left(\boldsymbol{\gamma}_{k,l}(\mathbf{c}_k,\mathbf{q}_k)\right), \tag{24a}$$

$$R_{k,l} = \mathbf{p}_{k,l}\ R_{k,l}^{\mathbf{f}}\ \mathbf{p}_{k,l}^\top + \bar{R}, \tag{24b}$$

$$\mathbf{p}_{k,l} = \mathbf{p}_{k,l}(\mathbf{c}_k), \quad R_{k,l}^{\mathbf{f}} = R^{\mathbf{f}}\left(\boldsymbol{\gamma}_{k,l}(\mathbf{c}_k,\mathbf{q}_k)\right). \tag{24c}$$

B. Process Model

In this subsection, the process model describing the evolution of the states over time is discussed to complete the state space model. For this purpose, the following linear Gaussian model is considered:

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, Q), \tag{25a}$$

$$\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, P_0), \tag{25b}$$

where $\mathbf{w}_k$ denotes the Gaussian noise with covariance Q. Recall that the state vector is formed by concatenating the kinematics and the extent representation, i.e., $\mathbf{x}_k \triangleq \left[\bar{\mathbf{x}}_k^\top \quad \mathbf{f}_k^\top\right]^\top$. It is obvious that the dynamics of these two state components does not interact with each other. Therefore, the process model comprises of two independent subsystems and can explicitly be written as

$$F = \begin{bmatrix} \bar{F} & 0 \\ 0 & F^{\mathbf{f}} \end{bmatrix}, \quad Q = \begin{bmatrix} \bar{Q} & 0 \\ 0 & Q^{\mathbf{f}} \end{bmatrix}, \tag{26a}$$

$$\boldsymbol{\mu}_0 = \begin{bmatrix} \bar{\boldsymbol{\mu}}_0 \\ \boldsymbol{\mu}_0^{\mathbf{f}} \end{bmatrix}, \quad P_0 = \begin{bmatrix} \bar{P}_0 & 0 \\ 0 & P_0^{\mathbf{f}} \end{bmatrix}. \tag{26b}$$

While $F^{\mathbf{f}}$ and $Q^{\mathbf{f}}$ matrices, which model the evolution of the extent, are specified in (11), the details of the motion dynamics designated by $\bar{F}$ and $\bar{Q}$ will be exposed in Section VIII. Finally, the initial covariance of the extent, $P_0^{\mathbf{f}}$, is determined by the underlying GP model as presented in (9).

VI. Inference

Having developed the state space model, the last step is to design an effective inference method to realize object tracking using point cloud measurements. While there are various standard techniques to recursively compute the posterior distribution of the state vector, we employ an extended Kalman filter (EKF) due to the nonlinearities in the measurement model.

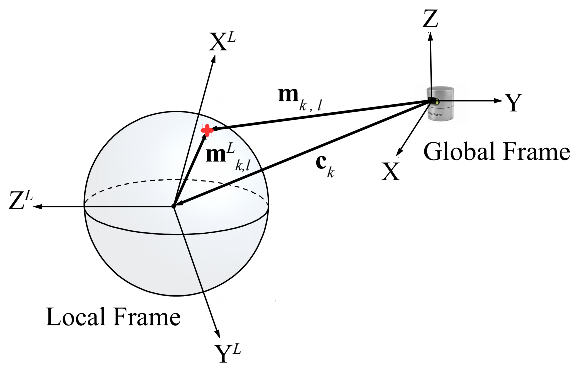


Fig. 2: Illustration of the coordinate frames and the vectors regarded in the measurement model.

To be able to process multiple measurements $\{\mathbf{m}_{k,l}\}_{l=1}^{n_k}$ in a single recursion at time k, we first need to slightly modify the state space model. To this end, the following measurement vector is created by concatenating the measurements together,

$$\mathbf{m}_k = \left[\mathbf{m}_{k,1}^\top, \ \ldots, \ \mathbf{m}_{k,n_k}^\top\right]^\top. \tag{27}$$

Then, the corresponding measurement equation can be simply written as

$$\mathbf{m}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{e}_k, \quad \mathbf{e}_k \sim \mathcal{N}(\mathbf{0}, R_k), \tag{28a}$$

$$\mathbf{h}_k(\mathbf{x}_k) = \left[\mathbf{h}_{k,1}(\mathbf{x}_k)^\top, \ \ldots, \ \mathbf{h}_{k,n_k}^\top(\mathbf{x}_k)\right]^\top, \tag{28b}$$

$$R_k = \operatorname{diag}\left[R_{k,1}, \ \ldots, \ R_{k,n_k}\right]. \tag{28c}$$

Notice that R_k is formed as a block diagonal matrix by considering that the noise coupled to the individual measurements are mutually independent.

Consequently, the state space model considering the complete set of measurements reads as

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, Q), \tag{29a}$$

$$\mathbf{m}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{e}_k, \quad \mathbf{e}_k \sim \mathcal{N}(\mathbf{0}, R_k), \tag{29b}$$

$$\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, P_0). \tag{29c}$$

The EKF regards the above representation to recursively compute the estimate of the state vector, $\hat{\mathbf{x}}_k$. Note that the gradient of the measurement function $\frac{\partial \mathbf{h}_k(\mathbf{x}_k)}{\partial \mathbf{x}_k}$ can be derived analytically which is to be utilized in the measurement update phase of each recursion.

VII. 3D Extent Tracking Using Projections

In the first part of this study, we developed a tracking algorithm which is essentially based on the radial function representation, $f(\theta, \phi)$, of the underlying 3D object shape. This function is further approximated via some basis points at which the shape information is accumulated during inference. Notice that as there are two input arguments of the radial function, the basis points are required to cover a two-dimensional space at a sufficient density to be able to capture the characteristics of the object shape. Also note that the computational load and the memory storage scale with the number of basis points since they are included in the state vector and updated at each recursion. A naive attempt to utilize fewer basis points for more efficient implementation will naturally result in a degraded representational power, potentially missing salient features of the 3D extent which might in turn deteriorate tracking accuracy.

Having said that, in this section we will seek for an alternative algorithm with improved computational properties. This second approach essentially retains the basic structure of the previous one; however, it fundamentally differs in the description of the object shape. In particular, multiple projections of the object are exploited to express the original 3D extent. Accordingly, the problem is reformulated as tracking the object while simultaneously learning the contours of its projections. This will eventually enable us to radically lower the number of basis points without compromising the representational power. The next section presents the alternative extent model in details.

A. Projection Model

It is a long-standing idea to exploit projections, silhouettes or images for expressing the corresponding 3D shape, [34], [35]. Being inspired by these methods, we suggest to model the object extent using projections onto several planes. Fig. 3 illustrates the idea for an example object with cone shape. In this case, the object is projected onto three orthogonal planes and the contours of these projections are essentially utilized to represent the original 3D shape. In this exposition, we assume that three orthogonal projections can sufficiently approximate the 3D shape; however, the number of projections can be increased to be able to generalize to a broader class of objects. For a systematic discussion on the objects which are exactly reconstructable from projections and the minimum number of projections necessary for reconstructing such objects, interested readers can refer to [36].

The contour of each projection can be described by a radial function in polar coordinates, i.e., $r = f(\theta)$, as shown in Fig. 3. The radial function maps the polar angle, θ, to the radial distance, r, between the projection center and the contour. Notice that having only one input argument, this function can possibly be approximated by a less number of basis points leading to a tracking algorithm demanding less computational sources.

The rest of the derivation closely follows the first algorithm. The unknown radial function on each projection plane is modeled by a GP, i.e., $f(\theta) \sim \mathcal{GP}(\mu(\theta), k(\theta, \theta'))$, whose mean function is taken to be constant $\mu(\theta) = \mu_r$, and the covariance function is defined as

$$k(\theta, \theta') = \sigma_f^2 e^{-\frac{2\sin^2\left(\frac{\theta-\theta'}{2}\right)}{l^2}} + \sigma_r^2. \tag{30}$$

Notice that the exponential term is structured to assure the periodicity of $f(\cdot)$ such that $f(\theta)$ and $f(\theta + 2\pi)$ are perfectly correlated since they basically correspond to the same point on the projection contour.

Further Discussion: Expressing the 3D shape in terms of a collection of projection contours enables us to introduce separate probabilistic models for each contour to account for application-specific knowledge about the objects. For example, many targets in driving environments, such as cars, vans and bicycles, possess two lines of symmetry, i.e., they are left-right and back-forward symmetric, in their projections onto the ground plane. In this case, to encode this information into the corresponding GP model, the covariance function can be designed as

$$k(\theta, \theta') = \sigma_f^2 e^{-\frac{\sin^2(\theta-\theta')}{2l^2}} + \sigma_r^2. \tag{31}$$

As the covariance function is periodic with π, the learned contours will be symmetric as intended.

B. State Space Model

In this subsection, the state space model relying on the extent description obtained by projection contours is to be constructed. The state vector is defined as $\mathbf{x}_k \triangleq \left[\bar{\mathbf{x}}_k^\top \quad \mathbf{f}_k^\top\right]^\top$ where $\bar{\mathbf{x}}_k^\top \triangleq \left[\mathbf{c}_k^\top \quad \mathbf{v}_k^\top \quad \mathbf{q}_k^\top\right]^\top$ includes the object kinematics

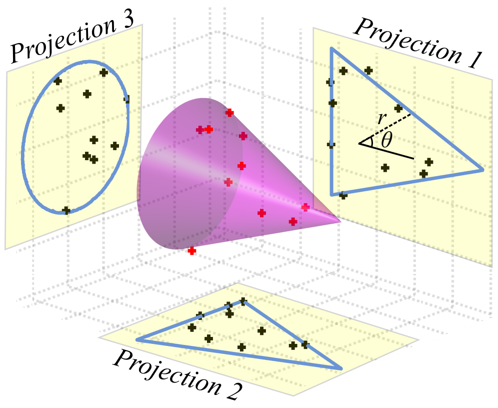


Fig. 3: Illustration of a cone-shaped object and the corresponding projection contours on three orthogonal planes. Point cloud measurements and their projections are shown by red and black plus signs, respectively.

and the extent is indicated by $\mathbf{f}_k \triangleq \begin{bmatrix} \mathbf{f}_k^{1\top} & \mathbf{f}_k^{2\top} & \mathbf{f}_k^{3\top} \end{bmatrix}^\top$ as a collection of the projection contours. More specifically, $\mathbf{f}_k^j$ is the parameterized description of the GP model for the radial function specifying the contour of the projection on the j^{th} plane. $\mathbf{c}_k$ is the center of the 3D object and $\mathbf{v}_k$ stands for the velocity of the center; $\mathbf{q}_k$ is the unit quaternion vector.

1) *Measurement Model:* The measurement model makes use of the local and global coordinate frames as defined earlier. While the object motion is to be tracked in the global frame, the shape is described in the local frame. In particular, the projection planes are fixed to the local frame so that the projections of the object onto the planes are kept unchanged at any time. By doing so, it is enabled to accumulate extent information over these planes by learning the latent contours of the projections.

$\{\mathbf{m}_{k,i}\}_{i=1}^{n_k}$ denotes 3D point cloud measurements acquired at time k. Firstly, each measurement $\mathbf{m}_{k,l}$ is transformed into the local frame to obtain $\mathbf{m}_{k,l}^L$ by (19). Thereafter, these local measurements are to be projected onto each plane to establish a relation between the projected measurements and the projection contour on the corresponding plane. As an example, let $\mathbf{m}_{k,l}^L$ be projected onto the j^{th} plane by

$$\mathbf{m}_{k,l}^j = P_j \; \mathbf{m}_{k,l}^L, \tag{32}$$

where $\mathbf{m}_{k,l}^j \in \mathbb{R}^2$ denotes the projection of $\mathbf{m}_{k,l}^L$, and $P_j \in \mathbb{R}^{2x3}$ is the projection matrix.

Similarly to the former case, the projected measurement can be described as

$$\begin{aligned} \mathbf{m}_{k,l}^j = \mathbf{p}_{k,l}(\mathbf{c}_k, \mathbf{q}_k) f^j(\theta_{k,l}(\mathbf{c}_k,\mathbf{q}_k)) + \bar{\mathbf{e}}_{k,l}, \\ \bar{\mathbf{e}}_{k,l} \sim \mathcal{N}(0, \bar{R}), \end{aligned} \tag{33}$$

where $f^j(\cdot)$ is the radial function expressing the contour of the projection on the j^{th} plane; $\mathbf{p}_{k,l}(\mathbf{c}_k, \mathbf{q}_k)$ is the unit-length vector pointing from the projection center towards the measurement; and $\bar{\mathbf{e}}_{k,l}$ is the Gaussian measurement noise with covariance matrix $\bar{R}$. Notice that unlike (21), the center position is not superposed in (33) as the projection is specified to be centered at the origin of the corresponding plane. Therefore, the polar angle $\theta_{k,l}$ associated with the projected measurement can be computed as

$$\theta_{k,l}(\mathbf{c}_k, \mathbf{q}_k) = \angle \mathbf{m}_{k,l}^j. \tag{34}$$

Besides, the unit-length vector $\mathbf{p}_{k,l}$ is obtained by

$$\mathbf{p}_{k,l}(\mathbf{c}_k, \mathbf{q}_k) = \frac{\mathbf{m}_{k,l}^j}{\|\mathbf{m}_{k,l}^j\|}. \tag{35}$$

The next step is to plug the GP representation for the radial function into (33) as

$$\mathbf{m}_{k,l}^j = \tilde{H}_l(\mathbf{c}_k, \mathbf{q}_k) \; \mathbf{f}_k^j + \tilde{\mathbf{e}}_{k,l}, \;\; \tilde{\mathbf{e}}_{k,l} \sim \mathcal{N}(0, \tilde{R}_{k,l}), \tag{36}$$

where

$$\tilde{H}_l(\mathbf{c}_k, \mathbf{q}_k) = \mathbf{p}_{k,l}(\mathbf{c}_k, \mathbf{q}_k) H^{\mathbf{f}}(\theta_{k,l}(\mathbf{c}_k, \mathbf{q}_k)), \tag{37a}$$

$$\tilde{\mathbf{e}}_{k,l} = \mathbf{p}_{k,l}(\mathbf{c}_k, \mathbf{q}_k) \; \mathbf{e}_{k,l}^{\mathbf{f}} + \bar{\mathbf{e}}_{k,l}, \tag{37b}$$

$$\tilde{R}_{k,l} = \mathbf{p}_{k,l} R_{k,l}^{\mathbf{f}} \mathbf{p}_{k,l}^\top + \bar{R}, \tag{37c}$$

$$R_{k,l}^{\mathbf{f}} = R^{\mathbf{f}} \left(\theta_{k,l}(\mathbf{c}_k, \mathbf{q}_k)\right). \tag{37d}$$

Note that the projected measurements are not necessarily located on the contour, instead some of them may fall within the interior of the projection area as depicted in Fig. 3. Accounting for this observation, the measurement model is modified as

$$\mathbf{m}_{k,l}^j = s_{k,l} \tilde{H}_l(\mathbf{c}_k, \mathbf{q}_k) \; \mathbf{f}_k^j + \tilde{\mathbf{e}}_{k,l}, \tag{38}$$

where $s_{k,l} \in [0, 1]$ is a random scaling factor. We approximate s as a Gaussian random variable, i.e., $s_{k,l} \sim \mathcal{N}(\mu_s, \sigma_s^2)$, since a Kalman filter will be employed for inference.

Considering the characteristics of the scaling factor, the measurement model can be rewritten as

$$\begin{aligned} \mathbf{m}_{k,l}^j &= \underbrace{\mu_s \tilde{H}_l(\mathbf{c}_k, \mathbf{q}_k) \; \mathbf{f}_k^j}_{=\mathbf{g}_{k,l}^j(\mathbf{x}_k)} + \underbrace{(s_{k,l} - \mu_s) \tilde{H}_l(\mathbf{c}_k, \mathbf{q}_k) \; \mathbf{f}_k^j + \tilde{\mathbf{e}}_{k,l}}_{=\mathbf{e}_{k,l}^j} \\ &= \mathbf{g}_{k,l}^j(\mathbf{x}_k) + \mathbf{e}_{k,l}^j, \quad \mathbf{e}_{k,l}^j \sim \mathcal{N}(0, R_{k,l}^j), \end{aligned} \tag{39}$$

where

$$R_{k,l}^j = \sigma_s^2 \tilde{H}_l \; \mathbf{f}_k^j \; \mathbf{f}_k^{j\top} \tilde{H}_l^\top + \tilde{R}_{k,l}, \tag{40a}$$

$$\tilde{H}_l = \tilde{H}_l(\mathbf{c}_k, \mathbf{q}_k). \tag{40b}$$

Then, the expression for the projected measurement is substituted into this equation as

$$P_j R_G^L(\mathbf{q}_k)(\mathbf{m}_{k,l} - \mathbf{c}_k) = \mathbf{g}_{k,l}^j(\mathbf{x}_k) + \mathbf{e}_{k,l}^j. \tag{41}$$

Finally, collecting the terms on one side of the equation, we end up with the following implicit measurement model:

$$\begin{aligned} \mathbf{0} &= \underbrace{P_j R_G^L(\mathbf{q}_k)(\mathbf{m}_{k,l} - \mathbf{c}_k) - \mathbf{g}_{k,l}^j(\mathbf{x}_k) - \mathbf{e}_{k,l}^j}_{\mathbf{h}_{k,l}^j(\mathbf{m}_{k,l}, \mathbf{x}_k, \mathbf{e}_{k,l}^j)} \\ &= \mathbf{h}_{k,l}^j(\mathbf{m}_{k,l}, \mathbf{x}_k, \mathbf{e}_{k,l}^j). \end{aligned} \tag{42}$$

This measurement model together with the process model introduced in Section V-B establishes the state space model.

C. Inference

Similar to the former case, an EKF is employed to realize recursive inference. To process all measurements instantaneously at the update phase of the filter, the complete measurement equation is written as

$$\mathbf{0} = \mathbf{h}_k(\mathbf{m}_k, \mathbf{x}_k, \mathbf{e}_k), \quad \mathbf{e}_k \sim \mathcal{N}(0, R_k), \tag{43}$$

where

$$\mathbf{h}_k(\mathbf{m}_k, \mathbf{x}_k, \mathbf{e}_k) = \left[\mathbf{h}_k^{1\top},\ \mathbf{h}_k^{2\top},\ \mathbf{h}_k^{3\top}\right]^\top, \tag{44a}$$

$$\mathbf{h}_k^j = \mathbf{h}_k^j(\mathbf{m}_k, \mathbf{x}_k, \mathbf{e}_k^j) = \left[\mathbf{h}^{j}{}_{k,1}^{\top},\ \ldots, \mathbf{h}^{j}{}_{k,n_k}^{\top}\right]^\top, \tag{44b}$$

$$R_k = \text{diag}\left[R_k^1,\ R_k^2,\ R_k^3\right], \tag{44c}$$

$$R_k^j = \text{diag}\left[R_{k,1}^j,\ \ldots, R_{k,n_k}^j\right] \text{ for } j \in \{1,2,3\}. \tag{44d}$$

VIII. Results

In this section, the performance of the proposed algorithms is evaluated on both simulated and real measurements in Section VIII-A and VIII-B, respectively. To be able to present the results in a comparative manner, we also consider a random matrix-based extended tracker, denoted as RM, as proposed in [5]. Throughout this section, we will refer to the first proposed method as 'GPEOT' (short for GP-based extended object tracker); while 'GPEOT-P' will stand for the second approach considering the projections.

Please recall that the developed algorithms regard the kinematic state which was defined as $\bar{\mathbf{x}}_k^\top \triangleq \begin{bmatrix}\mathbf{c}_k^\top & \mathbf{v}_k^\top & \mathbf{q}_k^\top\end{bmatrix}^\top$. For all of the experiments, we employ a dynamic model for the kinematics which basically relies on the well-known almost constant velocity model for the position and velocity, and it models the orientation to be perturbed by an additive Gaussian noise,

$$\bar{\mathbf{x}}_{k+1} = \bar{F}\bar{\mathbf{x}}_k + \bar{\mathbf{w}}_k, \quad \bar{\mathbf{w}}_k \sim \mathcal{N}(\mathbf{0}, \bar{Q}), \tag{45a}$$

$$\bar{\mathbf{x}}_0 \sim \mathcal{N}(\bar{\boldsymbol{\mu}}_0, \bar{P}_0), \tag{45b}$$

where

$$\bar{F} = \begin{bmatrix}\bar{F}_c & 0\\ 0 & \bar{F}_q\end{bmatrix},\ \bar{F}_c = \begin{bmatrix}1 & T\\ 0 & 1\end{bmatrix} \otimes I_3,\ \bar{F}_q = I_4, \tag{46a}$$

$$\bar{Q} = \begin{bmatrix}\bar{Q}_c & 0\\ 0 & \bar{Q}_q\end{bmatrix},\ \bar{Q}_c = \begin{bmatrix}\frac{T^3}{3} & \frac{T^2}{2}\\ \frac{T^2}{2} & T\end{bmatrix} \otimes \begin{bmatrix}\sigma_c^2 & 0 & 0\\ 0 & \sigma_c^2 & 0\\ 0 & 0 & \sigma_c^2\end{bmatrix},$$
$$\bar{Q}_q = \sigma_q^2\ I_4. \tag{46b}$$

σ_c^2 and σ_q^2 are process noise variances for the center and the quaternions, respectively.

A. Experiments with Simulated Measurements

To demonstrate the performance of the algorithms, various simulation experiments are conducted. Section VIII-A1 examines the setting where the point cloud measurements are simulated in MATLAB® for several dynamic objects with basic shapes. In this case, the measurements are randomly sampled from the objects' surfaces. In Section VIII-A2, we make use of a specialized sensor simulation environment, namely Blensor, [37], that generates measurements for realistic vehicle models.

1) MATLAB *Simulations:* In this subsection, the algorithms process point cloud measurements which are generated in MATLAB. In particular, the measurements are originated from random sources on the object surface and perturbed by additive Gaussian noise. Three different-shaped objects, e.g., cube, ellipsoid and cone, are tracked during the experiments. The dimensions of the objects are as follows: the length of the edge of the cube is 3 m, the semi-axes of the ellipsoid are (2.5, 1, 1) m in length, and the base radius and height of the cone are 1.5 m and 4 m, respectively.

The overall performance is evaluated based on the Intersection-Over-Union (IOU) measure given by

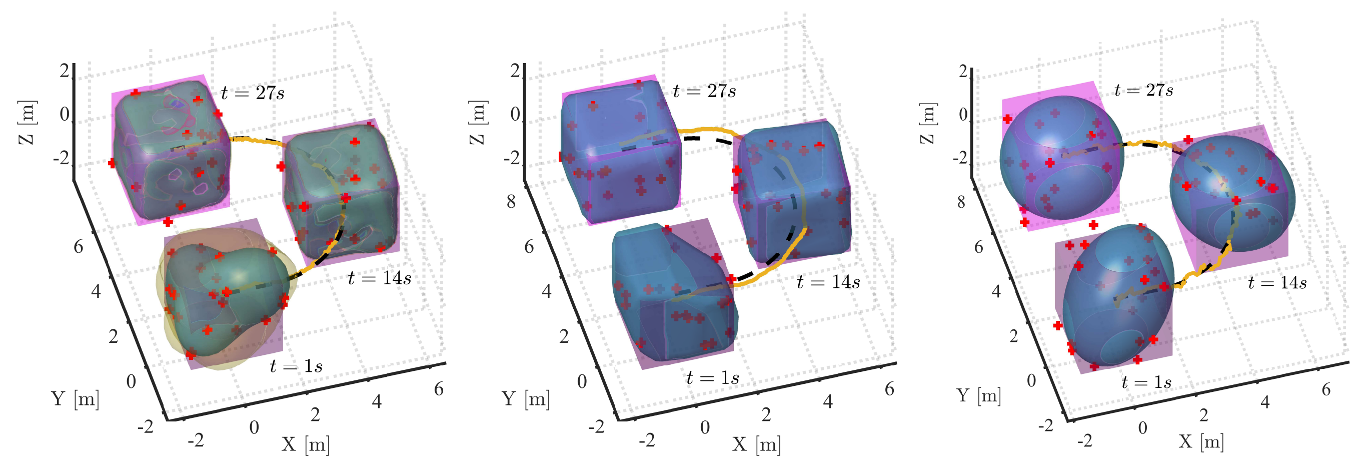
$$\text{IOU}(S_{true}, \hat{S}) = \frac{\text{volume}(S_{true} \cap \hat{S})}{\text{volume}(S_{true} \cup \hat{S})}, \tag{47}$$

where S_{true} is the true object shape, and $\hat{S}$ stands for the estimate. Notice that IOU simultaneously accounts for the quality of the estimates of the kinematics and the extent. In other words, an algorithm needs to produce accurate tracking outputs together with precise shape description to attain high IOU scores. Also note that in our discussion we deliberately exclude the RMSE measure for the position estimates since the suggested shape models do not imply a unique center definition; instead, different center positions with compatible radial functions can accurately represent the same object.

Two different scenarios are studied in the simulations: a linear motion and a u-turn. In the first case, objects move along a linear trajectory at a constant speed of 0.5 m/s. The second experiment starts with a linear section, then it makes a u-turn and ends with a final linear portion. Throughout the trajectory, the linear speed is again kept constant at 0.5 m/s. At each instant, 20 point measurements are originated from random sources which are sampled from a uniform distribution defined over the object surface. Each point measurement is perturbed by i.i.d. Gaussian noise with covariance $0.1^2 I_3$. The measurements are produced at 10 Hz, hence the sampling time of all algorithms is set to $T = 0.1$.

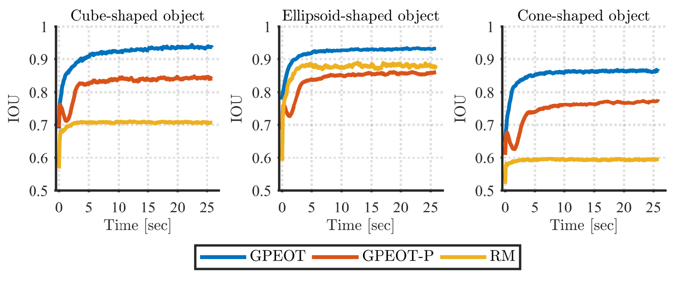
For GPEOT, the process noise standard deviations are set to $\sigma_c = 0.1$ and $\sigma_q = 0.005$, and $\alpha = 0.0001$ is used for the forgetting factor in the extent dynamics; the hyper-parameters of the GP model are set to $\mu_r = 0.3$, $\sigma_f = 1$, $\sigma_r = 0.2$, $l = \pi/8$; $\bar{R} = 0.1^2 I_3$ is used for the measurement noise variance; and the extent is represented by 642 basis points which are evenly spaced with respect to their spherical angles. For GOEOT-P, the process noise standard deviations are set to $\sigma_c = 0.1$ and $\sigma_q = 0.01$, and $\alpha = 0.0001$ is used for the forgetting factor in the extent dynamics; the hyper-parameters of the GP model are set to $\mu_r = 0$, $\sigma_f = 1$, $\sigma_r = 0.2$, $l = \pi/5$; $\bar{R} = 0.1^2 I_2$ is used for the projected measurement noise variance; each projection contour is represented by 50 basis points which are equidistantly located in $[0, 2\pi]$; and the parameters of the scaling factor are set to $\mu_s = \frac{5}{6}$ and $\sigma_s^2 = \frac{1}{18}$. Besides, we manually optimized the parameters of the RM model to obtain a competent performance in both scenarios: The scaling factor and the extension time constant are set to 1/3 and 1, respectively.

Due to page limitations, we hereby present some instances of our findings as representative examples. Typical results for

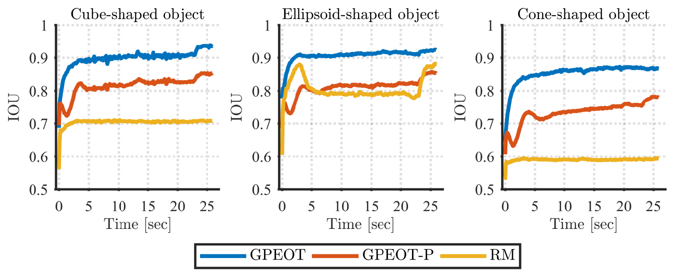


(a) GPEOT (b) GPEOT-P (c) RM

Fig. 4: Typical results for the cube-shaped object during the u-turn experiment. (Blue and magenta surfaces represent the estimated and true extent, respectively. In Fig. (a) yellow surface indicates the confidence interval of one standard deviation. Red plus signs are the point measurements. Solid yellow and dashed black curves are the estimated and true trajectory, respectively.)



(a) Linear motion experiments



(b) U-turn experiments

Fig. 5: Intersection-Over-Union (IOU) plots. (The signals are averaged over 100 MC runs.)

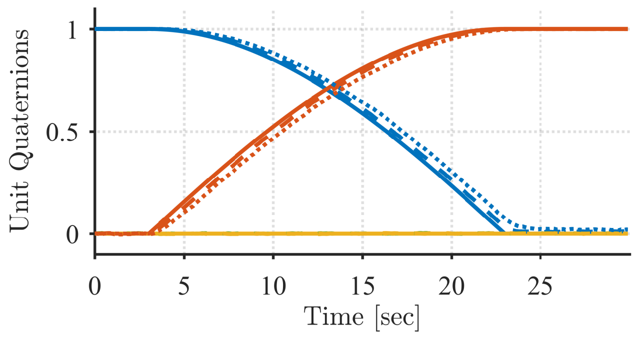


Fig. 6: True and estimated unit quaternions for the cube-shaped object during the u-turn experiment. (The estimates are averaged over 100 MC runs. Solid lines indicate the true orientation; dashed and dotted lines stand for the estimates of GPEOT and GPEOT-P, respectively. Color code is blue: q_0, green: q_1, yellow: q_2, red: q_3.)

the cube-shaped object performing a u-turn maneuver are illustrated in Fig. 4. Remember that GPEOT-P originally estimates the latent shape by learning the associated projection contours; therefore, to be able to visualize and interpret the results in 3D, we implemented a simple 3D reconstruction algorithm. The algorithm basically starts from a conservative estimate of the underlying 3D volume and refines the estimate by carving out the sections that are inconsistent with the projections. Fig. 4b exhibits the reconstructed shapes as estimates.

All simulations are repeated 100 times with different realizations of the measurement noise and the measurement origins. Fig. 5 exhibits the IOU plots obtained by averaging these Monte Carlo (MC) runs. For all experiments, GPEOT is observed to outperform the other algorithms with respect to the IOU measure. GPEOT and GPEOT-P produce successful results for all three shapes while RM model shows satisfactory performance only for the ellipsoid object. It is an expected finding since both of the GP-based approaches are flexible methods to represent any arbitrary star-convex shape, whereas RM essentially models the underlying shape by an ellipsoid. Additionally, the proposed algorithms are shown to be robust enough to handle the model mismatch in kinematics occurring in the u-turn scenario as the constant velocity model is no longer valid for this motion pattern. A particular reason for their robustness is that they can competently track the orientation of the objects (see Fig. 6).

Computation time: Both of the proposed algorithms are basically realized by an EKF, hence the estimates are recursively updated using newly available measurements at each

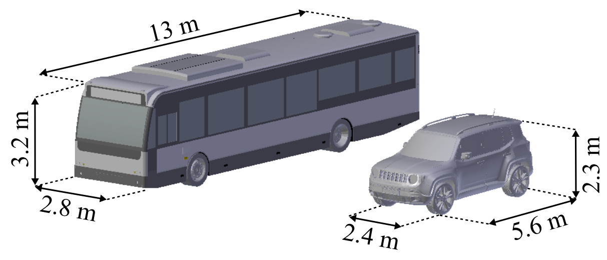


Fig. 7: Realistic vehicle models utilized in the Blensor experiments.

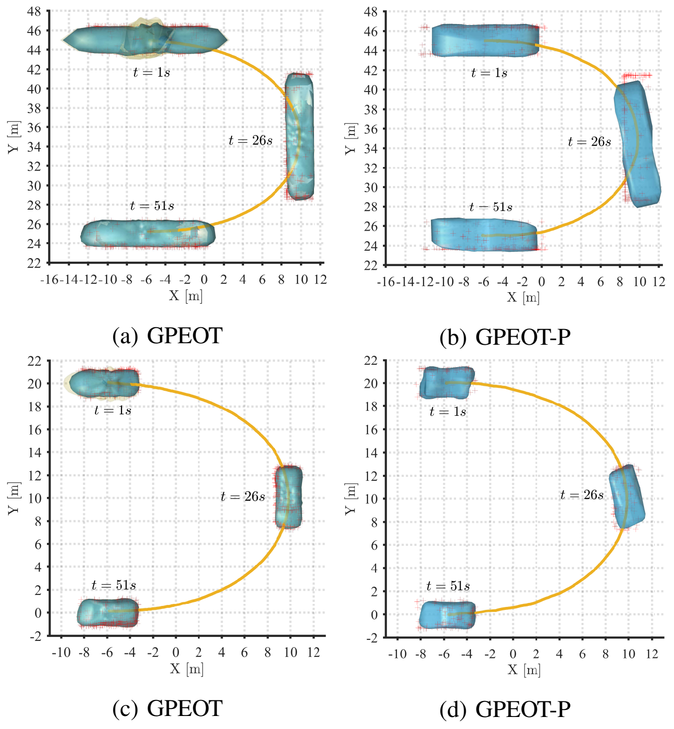


Fig. 8: Results obtained during Blensor simulations. (In Figs. (a) and (b), the bus is observed by two sensors at (0, 60, -5) and (0, 15, 5). In Figs. (c) and (d), the jeep is observed by two sensors at (0, 30, -5) and (0, -10, 5).)

time step. Therefore, the computational requirements do not increase over time and are basically determined by the size of the state vector and the number of the measurements. The state dimension in GPEOT is $\dim(\mathbf{x}_k) = \dim(\mathbf{c}_k) + \dim(\mathbf{v}_k) + \dim(\mathbf{q}_k) + \dim(\mathbf{f}_k) = 652$, and in GPEOT-P, it is $\dim(\mathbf{x}_k) = \dim(\mathbf{c}_k) + \dim(\mathbf{v}_k) + \dim(\mathbf{q}_k) + \dim(\mathbf{f}_k^1) + \dim(\mathbf{f}_k^2) + \dim(\mathbf{f}_k^3) = 160$. We utilize a naive implementation of EKF for each method without exploiting any code optimization methods. Note that the partial derivatives used in the measurement update phase of the filter are analytically available. All simulations are conducted in MATLAB 2017a on a standard laptop with Intel Core i7-6700HQ 2.60 Hz CPU using 16 GB of RAM. Average computation time for an update is recorded as 37.3 ms for GPEOT, 8.2 ms for GPEOT-P and 0.2 ms for RM model.

2) Blensor Simulations: To qualify the representational power of the suggested algorithms, additional experiments

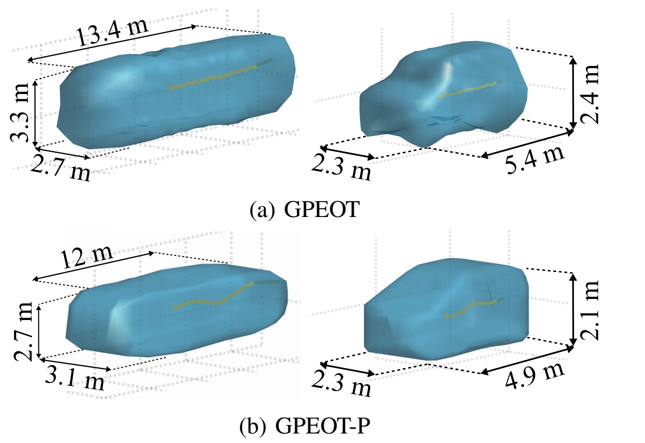


Fig. 9: Close-up views of the extent estimates obtained at the last instant of the Blensor simulations.

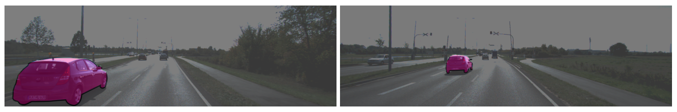
are conducted in Blensor which is a high fidelity sensor simulation environment. In these experiments, we consider realistic models of two different types of vehicles, namely a bus and a jeep, which are depicted in Fig. 7. In the scenario, each vehicle makes a u-turn while being observed by two Velodyne HDL-64E2 LIDAR sensors. The parameters of the algorithms are kept the same as in the previous subsection except the lengthscale of GPEOT is set to $l = \pi/12$.

The overview of the tracking outputs is shown in Fig. 8. Both of the proposed methods can successfully track two different vehicles. Furthermore, the shape estimates obtained at the last instant of the experiments are demonstrated by some close-up views in Fig. 9. While GPEOT is able to capture a highly detailed representation of the underlying object extent, GPEOT-P achieves a satisfactory but rather rough shape estimate. Besides, GPEOT-P slightly underestimates the size of the object due to the mismatch between the specified and true values of the scaling factor used in the measurement model.

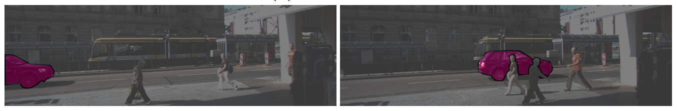
B. Experiments with Real Data

In this section, the performance of the algorithms is assessed on real data. To this end, we hereby exploit the Kitti tracking benchmark, [38]. The benchmark consists of various records of real-world traffic scenarios captured by several sensor modalities mounted on an ego vehicle. We form two scenarios of different vehicles by extracting the corresponding sequences of point measurements acquired by a Velodyne HDL-64E laser scanner. The same sets of parameters as in the previous subsections are utilized except the lengthscale of GPEOT is set to $l = \pi/14$.

The scenarios are visualized in Fig. 10. The first scenario takes place on a highway where the ego and the target vehicle move in the same direction, and the target pulls consistently ahead in time. Figs. 11 and 12 demonstrate that both of the algorithms accomplish successful tracking. Throughout the experiment, the sensor can only observe the back and right side of the target, thus the uncertainty of the extent on the observed portion decreases in time while a high uncertainty is properly associated with the unobserved section as explicitly

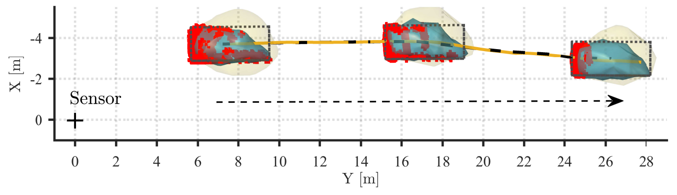


(a) Scenario 1

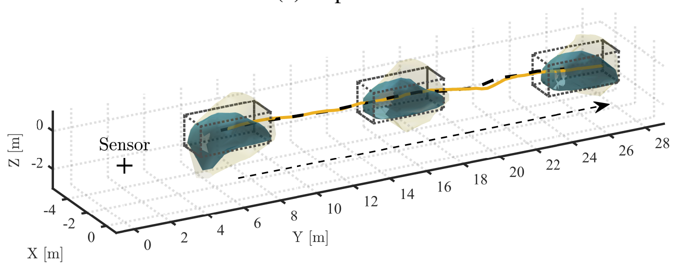


(b) Scenario 2

Fig. 10: Example views captured by a camera mounted next to the laser scanner on the ego vehicle. Left and right images depict the initial and the intermediate frames of the scenarios, respectively. Note that the highlighted vehicles are tracked by the proposed algorithms using only point cloud measurements.

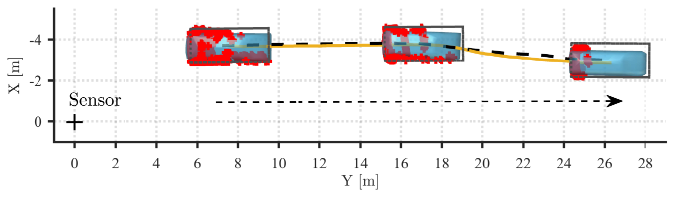


(a) Top view

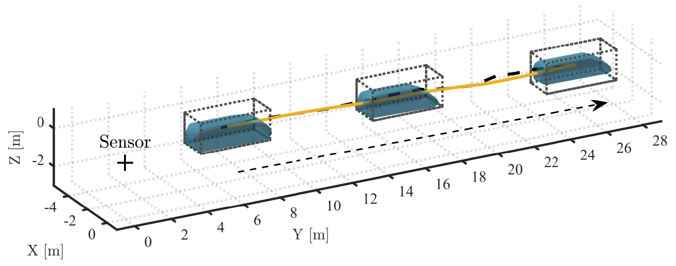


(b) Diagonal view

Fig. 11: Scenario 1: GPEOT results. (Blue and yellow surfaces indicate the estimated extent and the predicted uncertainty of one standard deviation, respectively. Dashed bounding box denotes the ground truth annotation of the target. Measurements are shown only in (a) by red plus signs. Solid yellow and dashed black curves are the estimated and true trajectory, respectively. Dashed black arrow is the direction of the target.)

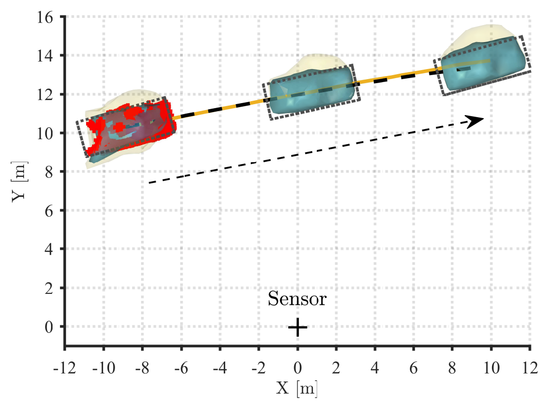


(a) Top view

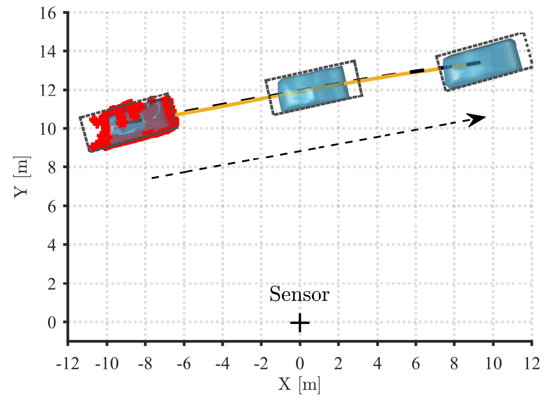


(b) Diagonal view

Fig. 12: Scenario 1: GPEOT-P results.



(a) GPEOT



(b) GPEOT-P

Fig. 13: Scenario 2. (Measurements are only visualized for the first frame.)

shown in Fig. 11. The GPEOT-P implementation uses the symmetric covariance function in (31) for the projection onto the ground plane. Note that considering solid amount of empirical evidence, this is a reasonable assumption for many targets in driving settings. The implementation inherently assumes that the corresponding projection contour is periodic with π so that the radial function takes exactly same values for $f(\theta)$ and $f(\theta+\pi)$. Accordingly, the reconstructed shape estimates accurately captures the appearance on the unobserved section of the object as seen in Fig. 12a.

In the second scenario, the ego vehicle waits stationary at a road junction while the target vehicle crosses the street. The experiment imposes two main challenges: First, the target is temporarily occluded by pedestrians and a column of a building; second, there are respectable number of 3D point measurements returned from the driver and the interior structure of the target vehicle. The tracking outputs of the algorithms are presented in Fig. 13. Note that GPEOT-P makes use of the symmetric covariance function for the ground projection as in the previous case. Both of the methods achieve accurate tracking and prove their robustness against occlusions and interior measurements.

IX. Conclusion

A new approach is proposed which is capable of processing 3D point cloud data for tracking objects with unknown shapes. The method can exploit the full potential of the information hidden in point cloud measurements by estimating the object's shape simultaneously with its kinematic state including the position, velocity and orientation. The proposed model is flexible to express and learn a large variety of shapes which may co-exist in a surveillance region. An alternative efficient implementation of the method is also derived, which reduces the computational requirements by utilizing plane projections. The algorithms are efficient in the implementation such that an extension to multi-target tracking framework is also possible. The methods provide an analytical expression of the object's shape, and this information can later be used for online identification and classification purposes in the future.

References

[1] K. Granstrom, M. Baum, and S. Reuter, "Extended object tracking: Introduction, overview and applications," *J. Adv. Inf. Fusion*, vol. 12, no. 2, pp. 139–174, Dec. 2017.

[2] L. E. Navarro-Serment, C. Mertz, and M. Hebert, "Pedestrian detection and tracking using three-dimensional ladar data," *Int. J. Robot. Res.*, vol. 29, no. 12, pp. 1516–1528, May 2010.

[3] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2-3, pp. 123–139, Apr. 2009.

[4] J. W. Koch, "Bayesian approach to extended object and cluster tracking using random matrices," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 3, pp. 1042–1059, Jul. 2008.

[5] M. Feldmann, D. Franken, and W. Koch, "Tracking of extended objects and group targets using random matrices," *IEEE Trans. Signal Process.*, vol. 59, no. 4, pp. 1409–1420, Apr. 2011.

[6] U. Orguner, "A variational measurement update for extended target tracking with random matrices," *IEEE Trans. Signal Process.*, vol. 60, no. 7, pp. 3827–3834, Jul. 2012.

[7] J. Lan and X. R. Li, "Tracking of extended object or target group using random matrixPart II: Irregular object," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Singapore, Jul. 2012.

[8] M. Baum and U. D. Hanebeck, "Random hypersurface models for extended object tracking," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Ajman, United Arab Emirates, Dec. 2009.

[9] ——, "Shape tracking of extended objects and group targets with star-convex RHMs," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Chicago, IL, USA, Jul. 2011.

[10] F. Faion, M. Baum, and U. D. Hanebeck, "Tracking 3D shapes in noisy point clouds with random hypersurface models," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Singapore, Aug. 2012.

[11] F. Faion, A. Zea, J. Steinbring, M. Baum, and U. D. Hanebeck, "Recursive Bayesian pose and shape estimation of 3D objects using transformed plane curves," in *Proc. IEEE ISIF Workshop Sens. Data Fusion: Trends, Solutions, Appl. (SDF)*, Bonn, Germany, Dec. 2015.

[12] N. Wahlström and E. Özkan, "Extended target tracking using Gaussian processes," *IEEE Trans. Signal Process.*, vol. 63, no. 16, pp. 4165–4178, Apr. 2015.

[13] E. Özkan, N. Wahlström, and S. J. Godsill, "Rao-Blackwellised particle filter for star-convex extended target tracking models," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Heidelberg, Germany, Jul. 2016.

[14] T. Hirscher, A. Scheel, S. Reuter, and K. Dietmayer, "Multiple extended object tracking using Gaussian processes," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Heidelberg, Germany, Jul. 2016.

[15] M. Michaelis, P. Berthold, D. Meissner, and H.-J. Wuensche, "Heterogeneous multi-sensor fusion for extended objects in automotive scenarios using Gaussian processes and a GMPHD-filter," in *Proc. IEEE ISIF Workshop Sens. Data Fusion: Trends, Solutions, Appl. (SDF)*, Bonn, Germany, Oct. 2017.

[16] K. Thormann, M. Baum, and J. Honer, "Extended target tracking using Gaussian processes with high-resolution automotive radar," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Cambridge, UK, Jul. 2018.

[17] B. Tuncer, M. Kumru, E. Özkan, and A. A. Alatan, "Extended object tracking and shape classification," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Cambridge, UK, Jul. 2018.

[18] S. Lee and J. McBride, "Extended object tracking via positive and negative information fusion," *IEEE Trans. Signal Process.*, vol. 67, no. 7, pp. 1812–1823, Feb. 2019.

[19] J. Aue, M. R. Schmid, T. Graf, J. Effertz, and P. Muehlfellner, "Object tracking from medium level stereo camera data providing detailed shape estimation using local grid maps," in *Proc. IEEE Intell. Veh. Symp.*, Gold Coast, QLD, Australia, Jun. 2013.

[20] M. Schütz, N. Appenrodt, J. Dickmann, and K. Dietmayer, "Multiple extended objects tracking with object-local occupancy grid maps," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Salamanca, Spain, Jul. 2014.

[21] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Proc. AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, Feb. 2017.

[22] J. Dequaire, P. Ondrúška, D. Rao, D. Wang, and I. Posner, "Deep tracking in the wild: End-to-end tracking using recurrent neural networks," *Int. J. Robot. Res.*, vol. 37, no. 4-5, pp. 492–512, Apr. 2018.

[23] F. Moosmann and C. Stiller, "Joint self-localization and tracking of generic objects in 3D range data," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Karlsruhe, Germany, Oct. 2013.

[24] D. Held, J. Levinson, S. Thrun, and S. Savarese, "Robust real-time tracking combining 3D shape, color, and motion," *Int. J. Robot. Res.*, vol. 35, no. 1-3, pp. 30–49, Jan. 2016.

[25] S. Kraemer, M. E. Bouzouraa, and C. Stiller, "Simultaneous tracking and shape estimation using a multi-layer laserscanner," in *Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, Yokohama, Japan, Mar. 2017.

[26] O. Williams and A. Fitzgibbon, "Gaussian process implicit surfaces," *Proc. of Gaussian Processes in Practice Workshop*, 2007.

[27] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 2011.

[28] J. Mahler, S. Patil, B. Kehoe, J. Van Den Berg, M. Ciocarlie, P. Abbeel, and K. Goldberg, "GP-GPIS-OPT: Grasp planning with shape uncertainty using Gaussian process implicit surfaces and sequential convex programming," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, May 2015.

[29] W. Martens, Y. Poffet, P. R. Soria, R. Fitch, and S. Sukkarieh, "Geometric priors for Gaussian process implicit surfaces," *IEEE Robot. and Autom. Lett.*, vol. 2, no. 2, pp. 373–380, Apr. 2017.

[30] M. Kumru and E. Özkan, "3D extended object tracking using recursive Gaussian processes," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Cambridge, UK, Jul. 2018.

[31] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[32] M. F. Huber, "Recursive Gaussian process regression," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Vancouver, Canada, May 2013.

[33] ——, "Recursive Gaussian process: On-line regression and learning," *Pattern Recogn. Lett.*, vol. 45, pp. 85–91, Aug. 2014.

[34] A. Rivers, F. Durand, and T. Igarashi, "3D modeling with silhouettes," *ACM Trans. Graph.*, vol. 29, Jul. 2010.

[35] W. Niem, "Robust and fast modeling of 3D natural objects from multiple views," in *Proc. SPIE: Image and Video Process. II*, Feb. 1994.

[36] A. Laurentini, "How many 2D silhouettes does it take to reconstruct a 3D object?" *Comput. Vis. Image Underst.*, vol. 67, no. 1, pp. 81–87, Jul. 1997.

[37] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree, "Blensor: Blender sensor simulation toolbox," in *Proc. Int. Symp. Visual Computing*, Las Vegas, NV, USA, May 2011.

[38] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR)*, Providence, RI, USA, Jun. 2012.