Online Stochastic Variational Gaussian Process Mapping for Large-Scale SLAM in Real Time

Ignacio Torroba\textsuperscript{1} Marco Cella\textsuperscript{1} Aldo Terán\textsuperscript{2} Niklas Rolleberg\textsuperscript{2} and John Folkesson\textsuperscript{1}

Abstract— Rao-Blackwellized particle filter (RBPF) SLAM solutions with Gaussian Process (GP) maps can both maintain multiple hypotheses of a vehicle pose estimate and perform implicit data association for loop closure detection in continuous terrain representations. Both qualities are of particular interest for SLAM with autonomous underwater vehicles (AUVs) in the open sea, where distinguishable features are scarce. However, the applicability of GP regression to parallel, real-time mapping in an RBPF framework remains limited by the size of the survey and the computational cost of the GP training. To overcome these constraints, in this letter we propose the adoption of Stochastic Variational GP (SVGP) regression to online mapping in combination with a novel, efficient particle trajectory storing in the RBPF. We show how the resulting RBPF-SVGP framework can achieve real-time performance in an embedded platform on two AUV surveys containing millions of points. We further test the framework on a live mission on an AUV and we make the implementation publicly available.

I. INTRODUCTION

Autonomous underwater vehicles (AUVs) are becoming standard tools for underwater exploration and seabed mapping in both scientific and industrial applications \cite{1, 2}. Their capacity to dive untethered allows them to reach areas inaccessible to surface vessels and to collect data more closely to the seafloor, regardless of the water depth. However, their navigation autonomy remains bounded by the accuracy of their dead reckoning (DR) estimate of their global position, severely limited in the absence of a priori maps of the area and GPS signal. Global localization systems equivalent to the later exists for the underwater domain, such as LBL or USBL. However they involve expensive external infrastructure and their reliability decreases with the distance to the AUV, making them unsuitable for deep sea surveys.

Simultaneous localization and mapping (SLAM) techniques can ameliorate these limitations by allowing the vehicle to autonomously adjust its DR estimate based on corrections against a map of the area being built at survey time. For bathymetric mapping, the de facto sensor in industry is the multibeam echosounder (MBES), which directly produces 3D representations of the seabed. Thus, the current state of the art SLAM solutions for deep sea navigation target AUVs equipped with an MBES. Two examples of these are \cite{3}, with a maximum a posteriori (MAP) approach to solving SLAM and \cite{4}, based on a Rao-Blackwellized Particle Filter (RBPF)

solution. The later methodology presents two advantages when compared to submap-based graph SLAM, as in the former. These two strong points of the RBPF are: i) its capacity to keep several hypotheses of the vehicle pose and ii) its ability to handle the data association problem implicitly. This second point is particularly important in bathymetric SLAM with MBES, where autonomous data association remains a pervasive challenge due to the scarcity of distinguishable features \cite{5}. These reasons motivated the recent work in \cite{6}, which presented an RBPF solution for bathymetric SLAM based on Gaussian Process (GP) models of the terrain. The use of GP regression (GPR) to represent bathymetry yields continuous and light maps with a readily available measure of map uncertainty. However, the challenge in the use of parallel GPR in online and large scale surveys in embedded hardware arises from their training complexity being cubic on the number of training inputs \cite{7}. To overcome this limitation, \cite{6} presented a GPU-targeted implementation of the GPR for a faster and more efficient learning. However, despite improvements in speed, their framework still required the bathymetry to be discretized a priori into tiles, each with their own GP, and it is unsuitable for embedded platforms.

To enable handling of surveys of millions of points without discretizing them while accomplishing real-time, in water, performance integrated on an AUV system on its payload computer, we propose in this work an RBPF solution based on Stochastic Variational Gaussian Process (SVGP) \cite{8} maps. Combining an adaption of SVGP to online mapping with a novel implementation of the handling of particles lineages, we show how our RBPF framework achieves real-time performance with datasets of millions of MBES data points in a NVIDIA Jetson Orin, with up to 100 particles. We assess the capabilities of the presented SLAM solution in three surveys.
collected with two AUVs, one of them live, and we make the framework publicly available to encourage further research.

II. RELATED WORK

The Rao-Blackwellized factorization of SLAM, first proposed in [9], divides its posterior into two separate problems, the vehicle localization and the map construction. Thus, the localization estimate can be computed via a particle filter, in which each particle represents a weighted hypothesis of the AUV pose [10]. Each particle also maintains its own map, from which expected measurements can be computed. By comparing them to the real ones, those particles whose maps resemble the real one more closely are resampled.

Regarding the filter structure, an efficient implementation of the particles’ maps is proposed in [11] based on the Distributed Particle Mapping (DPM) from [12]. Each particle’s trajectory is stored as a succession of the trajectory segments of all its ancestors, for which a so-called ancestry tree is built. Then, a common log of MBES measurements is shared across all particles in synchrony with their trajectories, such that a particle’s map can be reconstructed backtracing through the ancestry tree and transforming the measurements into the particle’s trajectory. A similar distributed approach has been adopted in this work.

Regarding the map representation, [13] employed 3D evidence grid maps, a type of Octree structure that permitted to accomplish real-time RBPF SLAM in underwater tunnels. Targeting open sea bathimetry instead, in [11] a 2D grid-based map is used to represent seabed depths. GP was originally used to regress seabed models within an RBPF framework in [4] with the approach later extended in [6]. Both works modelled bathymetric maps online through GPR and showed the advantages of using GP prediction to perform loop closures in unmapped areas. However, they both suffered from their inability to model the full survey with a single GP. In [4], the problem was sidestepped by retraining the GP on the latest patch of collected bathymetry at determined time intervals. This method, however, can suffer from “memory loss” in the kernel hyperparameters and therefore might not be able to handle big loop closures. Instead, in [6] the survey was divided into patches or tiles of predetermined size. Although effective, this divide-and-conquer approach involved discretizing the space similarly to grid-based mapping, hindering the continuous representation that GP is capable of. Furthermore, it implied equally fixed training times per GP, regardless of the bathymetry in the tile, data redundancies in the overlapping areas across tiles and the need to keep track of which GP to query when computing expected measurements.

[4], [6] remained therefore limited by the cubic complexity of the GP training on the number of training inputs \( N \). Other application-based solutions to adapt GP to large-scale mapping tasks [14] as well as sparse approximations to the GP kernel [15] have been presented in the literature. However, in this work we propose adopting the SVGP formulation from [8] to the online terrain reconstruction problem in order to handle large datasets efficiently.

III. SVGP BATHYMETRIC MAPPING

This section introduces large-scale bathymetric modelling with SVGP in an offline context.

A. Bathymetric surveying with a MBES

Autonomous bathymetric surveying tasks itself with creating a map of an unknown area of seabed \( E \) gathering MBES data with an AUV. The DR of the AUV can be modelled as a Gaussian distribution \( r_t \sim N(g(r_{t-1}, c_t), W) \), where \( r_t \) represents the AUV pose estimate at time \( t \). \( c_t \) and \( g() \) are the vehicle’s control input and its motion model respectively and \( W \) is the covariance of the zero-mean, additive Gaussian noise that parameterizes the uncertainty in the DR. Similarly, the MBES pings can be modelled as \( P_t \sim N(h(r_t, E_t), Q) \), where \( P_t = \{ p_i \}_{i=1}^n \) is a ping containing \( n \) 3D beams \( p_i = [x, y, z] \), \( h() \) is the MBES measurement model and \( Q \) parameterizes the measurements uncertainty. \( E_t = \{ e_i \}_{i=1}^m \) describes the patches of seabed, in the map frame, where the beams hit the ground at time \( t \).

After a bathymetric survey, a dataset of the form \( D = \{ p_i \}_{i=1}^N \) can be constructed, containing the \( N \) beams collected in time \( T \). In the reminder of this paper, we will alternatively use the notation \( D = \{ X, Y \} \) and the former when necessary to adapt to the standard notation in the GP literature.

B. Stochastic Variational Gaussian Process maps

If the seabed is assumed to be an unknown continuous function of the form \( f : X \rightarrow Y \) from which the MBES beams are noisy samples, SVGP can be used to regress the original \( f \) from the dataset \( D \). Applying Bayes rule, such regression problem takes the following form:

\[
p(f(X)|Y) = \frac{p(Y|f(X))p(f(X))}{p(Y)}
\]

where we seek to determine the posterior distribution that best fits the observations collected. The prior in Eq. [1] can be modelled through a GP as \( p(f(X)) \sim N(0, K_{NN}) \), parameterized be a zero mean and a kernel matrix \( K_{NN} \). For a selected GP kernel, the optimal set of its hyperparameters \( \theta_{kernel} \) can be learned maximizing the likelihood of collecting the \( N \) observations in \( D \) for the given model. However, such maximization involves the inversion of the \( K_{NN} \) matrix, which has historically prevented GP from being used in large mapping applications. In order to alleviate this constraint, several sparse formulations have been proposed [15] based on approximating the kernel via a set of \( S << N \) latent variables, normally referred to as inducing variables \( u \). These inducing variables correspond to a set of input locations \( u = f(Z) \). In our problem \( Z \) belongs in the same domain as \( X \). Furthermore, in order to handle intractable posteriors in Eq. [1] a Variational inference (VI) approach was proposed in [16] to approximate them instead through a user-defined variational term \( q() \). Minimizing the KL divergence between the target posterior and the selected \( q() \), \( KL[q(f(X), u)||p(f(X), u|Y)] \), is equivalent to maximizing its VI evidence lower bound (ELBO), which
yields the optimal set of hyperparameters \( \theta_{\text{kernel}} \). However, optimizing such ELBO remains subject to manipulating the \( N \) training points, limiting its applicability. To overcome this, the SVGP formulation [8] decouples the process complexity from the number of training points by parameterizing the variational term as \( q(f(X), u) = p(f(X)|u)q(u) \), with \( q(u) \sim \mathcal{N}(\mu, \Sigma) \). This results in the ELBO in Eq. 2, which can be factorized into \( N \) independent terms (we represent \( f(X) \) as \( f \) in the following for the sake of space):

\[
\mathcal{L} = \sum_i E_{q(u)p(f|u)} \left[ \ln p(y_i | f_i) \right] - KL[q(u) \parallel p(u)]
\] (2)

Thanks to this factorization, stochastic gradient descent (SGD) techniques can be applied to optimize the ELBO in Eq. 3 through minibatch estimates of size \( M \ll N \). Such minibatch estimator has the form:

\[
\hat{\mathcal{L}} = \frac{N}{M} \sum_i E_{q(u)p(f|u)} \left[ \ln p(y_i | f_i) \right] - KL[q(u) \parallel p(u)]
\] (3)

The ELBO in Eq. 3 can be derived with respect to the locations of the \( S \) inducing points \( Z_s \) and therefore these can be added as hyperparameters in the training process. Therefore the final set becomes \( \theta_{\text{SVGP}} = \{ \theta_{\text{kernel}}, Z_s \} \). Thus, the SVGP formulation has permitted learning offline map models from datasets containing millions of points, as in [10], [17].

IV. THE RBPF-SVGP ALGORITHM

An RBPF algorithm approximates the SLAM posterior by maintaining an estimate of the vehicle pose through a finite set \( R_t \) of \( J \) weighted particles, each containing its own map estimate \( \text{SVGP}_j^j \):

\[
R_t = \{(r_i^j, \text{SVGP}_j^j)\}_{j=1}^J
\] (4)

In order to be able model the particles maps with SVGP, in this section we present our approach to adapt SVGP to online bathymetry learning and its efficient integration in an RBPF framework to achieve real-time management of up to 100 particles in an embedded platform.

A. Online mapping with SVGP

In the previous section, the focus has been on offline SVGP mapping. In order to adapt it to online surveying, two minor modifications need to be adopted involving the construction of the dataset \( D \) and the initialization of the inducing locations \( Z_s \).

Regarding the dataset, so far we have assumed \( D \) to have a fixed size \( N \) and contain all the data collected from a survey of duration \( T \). However, in order to implement online SVGP training, the dataset must be increased with every new sample collected during the survey. Thus, we reformulate our dataset as \( D_t \), to denote its increasing size over time. We will show in our experiments how the SGD minibatch estimators can handle a growing dataset provided a sufficiently high rate of SVGP training iterations per new MBES added to \( D_t \).

Concerning \( Z_s \), their initial positions must now be selected carefully to account for the evolution of \( D_t \). Although the learning of \( Z_s \) will eventually result in their optimal positions being found across the dataset, this process is generally slower than the rate at which new data is collected with a moving vehicle. Therefore, the results will be suboptimal if all the \( Z_s \) locations are naively initialized over the data in \( D_t \) when the survey starts. We overcome this by uniformly distributing \( Z_s \) a priori over the area to survey, whose boundaries are known at the time of planning the mission.

B. SVGP minibatch construction

In an RBPF framework with SVGP maps, constructing minibatches for Eq. 3 entails randomly sampling from both the MBES log \( D_t \) and a particle’s history of trajectories \( h_j^t = \{r_j^t\}_{t=0}^T \), with both of them growing over time. Algorithm 1 describes the necessary steps involved in an SVGP training iteration for particle \( j \). Lines 3 – 7 construct a minibatch \( D_M \) and line 8 is an optimization step of Eq. 3. Line 4 randomly samples a beam \( p_b \) collected at time \( t_b \) <= \( t \) from the MBES log. Line 5 finds the corresponding particle’s pose at \( t_b \) and lines 6 – 7 transforms \( p_b \) to that pose and adds it to the minibatch. Instructions 4 – 7 cannot be parallelized since they involve accessing the common MBES log, which is unique due to memory constraints (note that line 8 is indeed parallel, see Section IV-D). Therefore they need to be very efficient in order to maximize SVGP training iterations across particles. Backtracing through an ancestry tree in line 5 as in [4], [6] is therefore unsuitable here. Thus we present a new implementation of the management of particles’ trajectories histories that sidesteps the need to explicitly keep track on a tree of the particles’ lineages upon resampling. Our C++ approach represents histories \( h_j^t \) as vectors of managed pointers to the vectors containing the segments of particles’ trajectories in between resamples. Our approach is illustrated in Fig. 2.

The left image in Fig. 2 depicts an example with three particles’ trajectories in 2D and three resampling steps, named \( LC_t \). In red, the trajectory of the winning particle after each resampling step. On the right, their three corresponding trajectory histories \( h_j^t \) as vectors of pointers evolving over time. During the time before the first loop closure, \( LC_{t_0} \), each particle stores its own trajectory as a vector of poses managed by a pointer. Once \( LC_{t_0} \) is triggered and particle 1 resampled,
the pointers in particle 0 and 2 delete their content and point at the trajectory segment in particle 1. After, a new pointer is added to each history vector and the process continues so that after LC2, each particles’ histories either contains or points at the three corresponding segments of trajectories between LCs. Thus, iterating through a particle’s history is trivial provided a common, auxiliary vector containing the length of the trajectory segments (as integers). Given a random beam at time $t_b$, the vector will signal at the corresponding pointer containing the particle’s pose at $t_b$, $r_{b,t}’$.

Additionally, as a result of this implementation computing the final RBPF trajectory can be done directly just by averaging all the particles histories. The need to explicitly keep track of their lineages has been sidestepped.

C. LC prompting and particle resampling

Given the set of $J$ particles $R_t$ in Eq. [4] the resampling procedure on the RBPF follows the steps in algorithm 2.

**Algorithm 2: LC prompting**

```
1 lc_promting($R_t, P_t, Q$):
2     $\hat{R}_t = \emptyset$
3     for $(r_i, SVGP^j_t)$ in $R_t$ do
4         $P_t^i = T(r_i^j) P_t$
5         $w_i = \mathcal{N}(P_t^i, SVGP^j_t(\hat{P}_t^{i,x,y}), SVGP^j_t(\hat{P}_t^{i,x,y}) + Q)$
6         add $(r_i^j, w_i)$ to $\hat{R}_t$
7     $R_t = \emptyset$
8     for $j$ in $J$ do
9         draw $j$ from $\hat{R}_t$ with probability $\propto w_i$
10        add $(r_j^i, SVGP^j_t)$ to $R_t$
11     return $R_t$
```

For each particle $j$, a new weight $w_i^j$ is computed in lines 4 – 6 comparing the depths of the latest ping $P_t$ against the predictions of that particle’s SVGP map. The weight will be proportional to the likelihood of the difference between the real and the expected ping depths being zero. The expected depths can be computed for a given set of 2D query locations $X = P^{i,x,y}_t$ via the posterior of each SVGP from Eq. [1] which is given by

$$p(f(X_s)|Y) \sim \mathcal{N}(K_sK_{ss}^{-1}\mu, (K_sK_{ss}^{-1}\Sigma(K_sK_{ss}^{-1})^T + K_{ss} - K_sK_{ss}^{-1}K_{ss}^T))$$  (5)

If the effective number of particles [18] is greater than half the number of particles, a systematic resampling [19] takes place in lines 8 – 10. The resampling step in line 10 might require copying particles. In our framework, this process entails two operations: i) copying $h_1$, which is extremely fast as it is a vector of pointers and ii) copying the SVGP map. The later involves not only the SVGP hyperparameters $\theta_{SVGP}$, but also the state of the particle’s optimizer, so that the training can continue in the new particle.

Finally, to save computational power and avoid GPU congestion, the LC prompting process in algorithm 2 occurs with a frequency $f_{LC}$, which is user defined but constrained by the number of particles and the hardware capacity. However, LC prompts are enabled only after every SVGP ELBO has converged, according to the criterion defined in Section V-B.

D. The RBPF parallel structure

As opposed to a custom-made implementation of the SVGP regression for our RBPF, our solution builds upon GPytorch [20]. The advantages are that GPytorch is open-source, general and community-based and maintained. The trade off rises from the so-called lazy management of the CUDA contexts carried out by Torch. Currently, GPytorch lazily instantiates a new CUDA context (600-800 MB) per SVGP created in a new process. This implies that a fully parallel implementation of the SVGP maps would be quickly limited by memory requirements. To overcome this, we have adopted a quasi-parallel implementation of the SVGP training. In this solution, a set of $B$ parallel nodes, each with its own CUDA context, hosts each $b$ SVGP maps, which run optimization steps recursively within the node. Thus, the total number of particles per node is $b = J/B$. An advantage of this implementation, however, is that the maximum number of SVGP maps accessing the GPU at the same time is now $B$, which can be limited by the user to prevent overloading it.

![Fig. 3: Schematic of the RBPF quasi-parallel structure, implemented in ROS](attachment:image)

Fig. 3 depicts a simplified graph of the system, implemented in ROS [21]. Each process from the set $O$ is a parallel ROS node, whose SVGP maps implement recursively
the key operations of SVGP training (line 8 on algorithm 1) and posterior sampling (line 5 in algorithm 2) when attempting a LC. The RBPF core node is written in C++ and both maintains the MBES log $D_t$ and computes the particles propagation whenever new data from the AUV is received. Furthermore, it stores the particles’ histories $H_t = \{h^j_t\}_{j=0}^J$ and keeps them synchronized with $D_t$ for an efficient minibatch construction when required by an SVGP map. Additionally, it handles the LC prompting, the weights computation based on the expected measurements from the SVGP maps and the particles resampling. All these task are implemented in parallel.

V. EXPERIMENTS

This section presents the evaluation methods designed and the datasets used to test the performance of the RBPF-SVGP framework proposed.

A. Datasets and vehicles

Three bathymetric surveys off the Swedish coast have been collected with two AUVs for testing the RBPF. A Kongsberg Hugin 3000 equipped with a MBES Kongsberg 2040 has collected mission 1 without any external navigation aid. Surveys 2 and 3 were executed by the AUV Lolo [22], with a R2Sonic MBES. While 1

![Fig. 4: The AUVs Hugin (left) and Lolo (right) being deployed for the surveys in the experiments.](image)

The size of the minibatch $M$ and the number of inducing points $S$ has been tuned for each mission, depending on the extension of the area to survey. The final parameters can be seen in Table I.

A key requirement for the RBPF resampling to be successful is that the SVGP maps resemble the true bathymetry close enough before attempting loop closures. For this, an exponential moving average is computed over the ELBO of each SVGP [24] at each optimization step in order to assess convergence at time $t$. However, in our online setup, convergence at time $t$ does not terminate the training, it only signals the RBPF core that the SVGP is ready to be used for LC prompting. This is due to the fact that in online training the SVGP ELBO can temporarily diverge when new data corresponding to previously unseen seabed relief is being added to $D_t$. This motivates continuously training the SVGP throughout the full mission. Fig. 6 showcases such phenomena in a particle’s SVGP ELBO from experiment 1.

C. Evaluation of the RBPF-SVGP framework

We now seek to assess the amenability of the proposed framework to real-time SLAM in large seabed areas in an embedded platform. For this, we have designed three sets of experiments carried out in the Jetson Orin. In experiment 1, we have gauged the effects of the parallelization of the SVGP training in the RBPF capacity to train as many SVGP maps as possible in real time during a very large survey with Hugin. We have evaluated the capabilities of the parallel structure proposed in terms of training iterations per particle across the mission and the resulting SVGP maps resemblance to the original bathymetry in terms of the consistency error metric from [25]. Experiment 2 shows the results from testing the same setup in Lolo live during a survey. In experiment 3, we have evaluated the performance of the RBPF in terms of the estimated vehicle trajectory compared against the ground truth computed from GPS measurements [26] on Lolo data.

VI. RESULTS

This section presents the results from testing the RBPF-SVGP framework on the terrain modelling and localization tasks on the experiments introduced above.

A. Experiment 1: online SVGP mapping parallelization

As mentioned in the previous section, for the RBPF to converge towards the true vehicle pose upon LC, the SVGP maps need to resemble the true underlying bathymetry closely enough. This requires that each SVGP achieves
enough training iterations in real time, which is the reason behind the highly efficient and parallel implementation of the RBPF presented. In order to measure the resemblance of the SVGP maps with respect to the raw bathymetry, we have tested the mapping-only capability of the SVGP-RBPF with the data from mission 1 and applied the error metric from [25] to assess it. For this type of experiment, the particles have been configured to follow the vehicles’ DR, with zero motion noise added and the LC prompting turned off.

Table II: Results from the parallel mapping-only experiments with the Hugin survey

| Experiment 1 | Particles (J) | Nodes (B) | Avg it/particle | Map RMSE |
|--------------|---------------|-----------|-----------------|----------|
| J            | B             |           |                 |          |
| 1            | 1             | 202.327   | 1.636           |          |
| 10           | 10            | 21.425    | 1.709           |          |
| 100          | 10            | 39.373    | 1.712           |          |
| 1000         | 10            | 3.714     | 1.995           |          |

The first row of images in Fig. 5 shows the results from this experiment with the configuration $J = 100$, $B = 10$ for one particle randomly picked at the end of the mission. Column a) depicts the original bathymetry map collected by Hugin with the DR trajectories in red. Columns b) and c) represent the final particle’s SVGP posterior mean and variance. The final locations of the inducing points are plotted over the variance map. Column d) illustrates the consistency error map from comparing a) and b) using the error metric from [25]. Notice that the peak errors correspond to either bathymetry areas with sharp changes, known to be difficult to model with GPR, or to noise in the MBES data.

To gauge the effects of the parallelization of the SVGP training, the Hugin survey has been replayed with four particles configurations, summarized in Table II. Namely: single particle ($J = 1$, $B = 1$), no parallelization ($J = 10$, $B = 1$), full parallelization ($J = 10$, $B = 10$) and quasi-parallelization ($J = 100$, $B = 10$). Notice how the fully parallel configuration has achieved roughly double the total amount of iterations in the mission than the fully sequential. Given that a 100% utilization of the GPU was achieved, we believe $\sim 400,000$ to be the maximum number of SVGP training iterations achievable in this hardware with the proposed framework, regardless of the degree of parallelization. Approximately the same amount has been achieved by the quasi-parallel implementation for a $\times 10$ larger number of particles, indicating maximum utilization of the GPU as well.

Regarding the resulting SVGP map, the fully-parallel configuration has achieved the same consistency RMS error than the sequential one for double the number of iterations per particle, which signals overfitting of the maps. We therefore adopt the $RMSE \sim 1.7m$ as the baseline and we see how a tenfold increment of the particles only causes an increase of 17.35% on the error, which hints at a good performance of our quasi-parallel implementation.

Finally, concerning the influence of the filter configuration in the temporal aspect of the online training, Fig. 6 illustrates the evolution of the ELBO of a particle randomly sampled from each configuration in Experiment 1. The initial bump caused by the AUV entering the flat area of the survey in
As in the previous experiment, due to technical problems with Lolo the mission was carried out on the surface. This, however, has allowed collecting GPS measurements that have been utilized as the ground truth trajectory. Nevertheless, the AUV’s DR used in the experiment has been estimated only from the velocities provided by the navigation system.

Fig. 8: DR (red) and RBPF (blue) trajectory errors against GT during experiment 3. Notice the two loop closures at \( t_{\text{step}} \sim 21000 \) and \( t_{\text{step}} \sim 28000 \).

Fig. 8 depicts the evolution over time of the DR (red) position error, in meters, against the GT position (from the GPS). In blue, the RBPF trajectory estimate during the mission, computed online from the average of the particles. The RBPF has registered two loop closures: a correct one at \( t_{\text{step}} \sim 21000 \) and an erroneous one at \( t_{\text{step}} \sim 28000 \). The high level of spurious measurements in the MBES data has led us to manually set the uncertainty of the MBES noise model (\( Q \)) to a high level in order to make the filter more robust against outliers, however resulting in only two loop closures detected.

VII. CONCLUSIONS AND FUTURE WORK

We have presented a Rao-Blackwellized particle filter with Stochastic Variational Gaussian Process maps for online, large-scale SLAM in an embedded platform. The key contributions proposed that have enabled our approach are: i) the modifications of the SVGP training process, which have
permitted to adapt SVGP to online regression on surveys with a mobile platform, ii) a new configuration of the RBPF particles’ lineage tracking, which has allowed parallel, real-time SVGP training in a platform with limited resources.

The proposed framework has been implemented and tested in a Jetson Orin and its performance has been assessed on the tasks of parallel, real-time mapping and localization in three challenging surveys collected with two AUVs, one of them live. When compared to the current state of the art solution for underwater RBPF SLAM from [6], our method is capable of handling larger surveys, both in terms of area covered and mission time, with a single SVGP per particle map and with larger sets of particles. Regarding its implementation, the SVGP regression module of our framework builds upon the open-source GPytorch library and it does not require any modification of the original SVGP formulation, facilitating its use. Additionally, the full framework has been released to encourage further research on the topic. 

The current major limitation of the implementation concerns the lazy initialization of the CUDA contexts per SVGP model carried out by Pytorch. This limits the number of fully parallel SVGP maps by the amount of available RAM of the host device. Although a quasi-parallel configuration has been proposed, overcoming this constraint would entail a full SVGP implementation in native CUDA. Nevertheless, when enough RAM is available, a fully-parallel setup can be achieved setting $B = J$. Additionally, we propose as further research the use of the particles’ ELBO value within the particles weighting process. It has been shown that the ELBO relates to the online state of the SVGP map and therefore factoring it in the particle weighting could potentially result in more accurate resampling.

Finally, although this work has targeted bathymetric SLAM, the proposed approach is not limited to it and it can be applied to different problems and domains.

ACKNOWLEDGMENT

The authors thank the Alice Wallenberg foundation for funding MUST, Mobile Underwater System Tools, project that provided the Hugin AUV.

REFERENCES

[1] Alastair GC Graham, Anna Wählín, Kelly A Hogan, Frank O Nitsche, Karen J Whyte, Rebecca T Totten, James A Smith, Claus-Dieter Hillenbrand, Lauren M Sinkins, John B Anderson, et al. Rapid retreat of thwaites glacier in the pre-satellite era. Nature Geoscience, pages 1–8, 2022.

[2] Ivan Stenius, John Folkesson, Srijanasha Bhat, Christopher Iliffe Sprague, Li Ling, Özer Ozkrahman, Nils Bore, Zheng Cong, Josefine Severholt, Carl Ljung, et al. A system for autonomous seaweed farm inspection with an underwater robot. Sensors, 22(13):5064, 2022.

[3] Ignacio Torroba, Nils Bore, and John Folkesson. Towards autonomous industrial-scale bathymetric surveying. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2019.

[4] Stephen Barkby, Stefan B Williams, Oscar Pizarro, and Michael V Jakuba. Bathymetric slam with no map overlap using gaussian processes. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2011.

[5] Jiarui Tan, Ignacio Torroba, Yiping Xie, and John Folkesson. Data-driven loop closure detection in bathymetric point clouds for underwater slam. arXiv preprint arXiv:2209.08578, 2022.

[6] Kristopher E Krasnosky and Christopher Roman. A massively parallel implementation of gaussian process regression for real time bathymetric modeling and simultaneous localization and mapping. 2022.

[7] Carl Edward Rasmussen. Gaussian processes in machine learning. In Summer school on machine learning, pages 63–71. Springer, 2003.

[8] James Hensman, Nicolò Fusi, and Neil D Lawrence. Gaussian processes for big data. arXiv preprint arXiv:1309.6835, 2013.

[9] Michael Montenero, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. Aaai/aaai, 593598, 2002.

[10] Ignacio Torroba, Christopher Iliffe Sprague, and John Folkesson. Fully-probabilistic terrain modelling and localization with stochastic variational gaussian process maps. IEEE Robotics and Automation Letters, 7(4):8729–8736, 2022.

[11] Stephen Barkby, Stefan B Williams, Oscar Pizarro, and Michael V Jakuba. A featureless approach to efficient bathymetric slam using distributed particle mapping. Journal of Field Robotics, 28(1):19–39, 2011.

[12] Austin I Eliazar and Ronald Parr. Hierarchical linear/constant time slam using particle filters for dense maps. Advances in neural information processing systems, 18, 2005.

[13] Nathaniel Fairfield, George Kantor, and David Wettergreen. Real-time slam with octree evidence grids for exploration in underwater tunnels. Journal of Field Robotics, 24(1-2):93–21, 2007.

[14] Shrihari Vasudevan, Fabio Ramos, Eric Nettleton, and Hugh Durrant-Whyte. Gaussian process modeling of large-scale terrain. Journal of Field Robotics, 26(10):812–840, 2009.

[15] Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. The Journal of Machine Learning Research, 6:1939–1959, 2005.

[16] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In Artificial intelligence and statistics, pages 567–574. PMLR, 2009.

[17] Andrew C Miller, Lauren Anderson, Boris Leistedt, John P Cunningham, David W Hogg, and David M Blei. Mapping interstellar dust with gaussian processes. arXiv preprint arXiv:2202.06797, 2022.

[18] Jun S Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. Statistics and computing, 6(2):113–119, 1996.

[19] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian non-linear state space models. Journal of computational and graphical statistics, 5(1):1–25, 1996.

[20] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. arXiv preprint arXiv:1809.11165, 2018.

[21] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In ICRA workshop on open source software, volume 3, page 5. Kobe, Japan, 2009.

[22] Clemens Deutsch, Lázaro Moratelli, Sebastian Thuné, Jakob Kuttenkeuer, and Filip Söderling. Design of an auv research platform for demonstration of novel technologies. In 2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV), pages 1–8. IEEE, 2018.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[24] Maximilian Balandat, Brian Karrer, Daniel R Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: Programmable bayesian optimization in pytorch. arXiv e-prints, pages arXiv–1910, 2019.

[25] Chris Roman and Hanumant Singh. Consistency based error evaluation for deep sea bathymetric mapping with robotic vehicles. In Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pages 3568–3574. IEEE, 2006.

[26] Edwin Olson and Michael Kaess. Evaluating the performance of map optimization algorithms. In RSS Workshop on Good Experimental Methodology in Robotics, volume 15, 2009.

[27] Christopher Iliffe Sprague, Özer Ozkrahman, Andrea Munafò, Rachel Marlow, Alexander Phillips, and Petter Ogren. Improving the modularity of auv control systems using behaviour trees. In 2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV), pages 1–6. IEEE, 2018.

https://github.com/ignaciobt/UWExploration/