2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Light-Weight Pointcloud Representation with Sparse Gaussian Process

Mahmoud Ali and Lantao Liu

***Abstract*— This paper presents a framework to represent high-fidelity pointcloud sensor observations for efficient communication and storage. The proposed approach exploits Sparse Gaussian Process to encode pointcloud into a compact form. Our approach represents both the free space and the occupied space using only one model (one 2D Sparse Gaussian Process) instead of the existing two-model framework (two 3D Gaussian Mixture Models). We achieve this by proposing a variance-based sampling technique that effectively discriminates between the free and occupied space. The new representation requires less memory footprint and can be transmitted across limited-bandwidth communication channels. The framework is extensively evaluated in simulation and it is also demonstrated using a real mobile robot equipped with a 3D LiDAR. Our method results in a 70∼100 times reduction in the communication rate compared to sending the raw pointcloud. We have provided a demonstration video[1] and open-sourced our code [2].**

I. Introduction

With the rapid advancement of LiDAR technology, we now can build maps with remarkably high resolution. For example, each full scan of an only 16-channel 3D LiDAR can give us 57600 points in the pointcloud that represents the surrounding obstacles. However, a price for using the high resolution LiDAR is the computation, storage, and communication costs when mapping the environments. While one might be able to upgrade the computation and storage by using a high performance computer system, the communication usually becomes a bottleneck due to the low communication bandwidth available. In practice, the low bandwidth communication is considered as a major challenge for many robotics applications such as occupancy mapping of underwater and subterranean environments (caves, tunnels, mines, etc), search-and-rescue missions in disaster scenarios with a degraded communication infrastructure, and planetary exploration missions [1]. The low bandwidth can prevent a robot from real-time sharing its sensor observations, and this can significantly degrade the system responsiveness if the robot needs to follow or interact with external control or supervision platforms. This work tackles the problem of sharing high-fidelity 3D pointcloud through a limited bandwidth communication channel.

The system we consider consists of a robot (the scout) equipped with a LiDAR and a communication apparatus, and deployed in a low-bandwidth environment. The scout sends the observations that it acquires to a base for building the occupancy map of the environment, see Fig. 1. Our approach exploits the Variational Sparse Gaussian Process (VSGP) [2] as a generative model to represent the pointcloud in a compact form. This lightweight representation is transmitted through low-bandwidth communication to the base where the original pointcloud is reconstructed. Extensive evaluations reveal that our approach results in a 70∼100 times reduction in the memory as well as the communication rate required to transmit pointcloud data. For example, Fig. 2a shows a scene of a simulated mine tunnel, where its raw pointcloud (shown in red, Fig. 2b) requires around 750 KB of memory. Our approach is able to represent the same observation using only 6 KB of memory and transmit through limited-bandwidth communication. On the receiver side of the communication channel, the compact representation is used to reconstruct the original pointcloud (reconstructed pointcloud shown in white, Fig. 2b). An occupancy map of the scene can be built using the reconstructed pointcloud, see Fig. 2c.

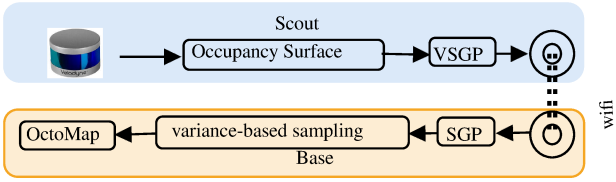
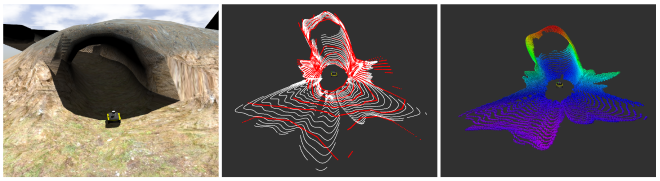


Fig. 1: System Overview.



(a) (b) (c)

Fig. 2: (a) Gazebo simulated mine tunnel; (b) Original pointcloud generated by a VLP16 LiDAR in red, and reconstructed pointcloud from the VSGP model in white; (d) Occupancy Map generated by OctoMap from the reconstructed pointcloud.

II. Related Work

Pointcloud compression algorithms have been investigated in recent years to cope with the demands to store and communicate the high-precision 3D points [3]. For example, the space partitioning trees approaches that exploit the 3D correlation between pointcloud points are widely used to compress the pointcloud data [4]–[9]. Recently, deep learning based approaches were also proposed to leverage data and learn or encode the pointcloud compression [10]–[14]. Different from these frameworks, the probabilistic approaches exploit the compactness of the distributions to compress 3D

[1]Mahmoud Ali and Lantao Liu are with the Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN 47408 USA, {alimaa, lantao}@iu.edu

[1]Video: https://youtu.be/BQZzXiCFGrM

[2]Code: https://github.com/mahmoud-a-ali/vsgp_pcl

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10161111

sensor observation. For instance, Gaussian Mixture Models (GMM) [15]–[17] have been proposed as a generative model to encode 3D occupancy map. The GMM approach encodes the 3D data as a mixture of Gaussian densities to represent the occupied and free spaces around the robot.

Gaussian Process (GP) has been proven to be an excellent framework to model spatial phenomena or features in a continuous domain [18]–[21]. Unfortunately, the standard GP has a cubic time complexity and this results in very limited scalability to large datasets. Methods for reducing the computing burdens of GPs have been previously investigated. For example, GP regressions can be done in a real-time fashion where the problem can be estimated locally with local data [22]. Sparse GPs (SGPs) [23]–[26] tackle the computational complexity of the normal GP through leveraging the Bayesian rule with a sequential construction of the most relevant subset of the data.

We propose a new probabilistic pointcloud compression approach which is based on the VSGP [2] and inspired by the GMM approach. While the GMM shares the accumulated sensory information as a set of accumulated Gaussian densities which are sampled and used as an occupancy map of the environment, in contrast, the proposed approach relies on sharing of immediate sensor observation to be reconstructed on the other side of the communication channel for further processing based on the required task (e.g. 3D mapping, object recognition, tracking, etc). This proposed VSGP-based approach offers a few advantages over the recent GMM approach: while the GMM approach uses two 3D GMMs to fit the occupied and free points [15]–[17], our approach uses only one 2D VSGP to fit all the occupancy surface, including both the occupied and free points. The primary reason that our approach uses one VSGP instead of two is that we are using the variance calculated by the VSGP at each sampled point during the reconstruction process to decide if it belongs to the occupied or the free space. Therefore, the proposed approach results in a more compact representation of the sensor observation, which requires less memory than the GMM approach and, as a consequence, leads to a lower communication rate.

III. Background

GP is a non-parametric model described by a mean function $m(\mathbf{x})$, and a co-variance function (kernel) $k(\mathbf{x},\mathbf{x}')$, where $\mathbf{x} \in \mathbb{R}^d$ is the GP input [27]:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k\left(\mathbf{x},\mathbf{x}'\right)\right). \tag{1}$$

By training the GP on a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=0}^{n-1}$ with n training inputs and their corresponding scalar outputs (observations) $\mathbf{y} = \{y_i\}_{i=0}^{n-1}$, the output y^* for any new query $\mathbf{x}^*$ can be estimated using the GP prediction equation:

$$p(y^*|\mathbf{y}) = \mathcal{N}(y^*|m_y(\mathbf{x}^*), k_y(\mathbf{x}^*,\mathbf{x}^*) + \sigma_n^2), \tag{2}$$

where $m_{\mathbf{y}}(\mathbf{x})$ and $k_{\mathbf{y}}(\mathbf{x},\mathbf{x}')$ are the posterior mean and co-variance functions, and σ_n^2 is the noise variance [2]. The GP prediction equation depends on the values of the hyperparameters (Θ, σ_n^2) where Θ is the kernel parameters.

Various approximation methods have been proposed in the literature to mitigate the computational complexity of a full GP, which is $\mathcal{O}(n^3)$ for n training data points. These approximation methods consider only m input points (*inducing points* X_m) to represent the entire training data [27], where their corresponding values of the underlying function $f(\mathbf{x})$ are called the *inducing variables* f_m. By replacing the entire data set with only the m inducing points, the computational complexity is reduced to $\mathcal{O}(nm^2)$. Titsias [2] proposed a variational learning framework to jointly estimate the kernel hyperparameters and the inducing points by approximating the true exact posterior of a GP $p(f|y)$ with a variational posterior distribution $q(f, f_m)$,

$$q(f, f_m) = p(f|f_m)\phi(f_m), \tag{3}$$

where $\phi(f_m)$ is the free variational Gaussian distribution, and $p(f|f_m)$ is the conditional GP prior. The Kullback-Leibler ($\mathbb{KL}$) divergence is used to describe the discrepancy between the approximated and the true posteriors. Minimizing the $\mathbb{KL}$ divergence between the approximated and the true posteriors $\mathbb{KL}[q(f, f_m)||p(f|y,\Theta)]$ is equivalent to maximizing the variational lower bound of the true log marginal likelihood:

$$\begin{aligned} F_V(X_m) &= \log\left[\mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \sigma^2 I + Q_{nn}\right)\right] - \frac{1}{2\sigma^2}\operatorname{Tr}(\widetilde{K}), \\ Q_{nn} &= K_{nm}K_{mm}^{-1}K_{mn}, \\ \widetilde{K} &= \operatorname{Cov}(\mathbf{f} \mid \mathbf{f}_m) = K_{nn} - K_{nm}K_{mm}^{-1}K_{mn}, \end{aligned} \tag{4}$$

where $F_V(X_m)$ is the variational objective function, $\operatorname{Tr}(\widetilde{K})$ is a regularization trace term, K_{nn} is the original $n \times n$ co-variance matrix, K_{mm} is $m \times m$ co-variance matrix on the inducing inputs, K_{nm} is $n \times m$ cross-covariance matrix between training and inducing points, and $K_{nm} = K_{mn}^T$. More details on VSGP can be found in Titsias's work [2].

IV. Methodology

The proposed approach exploits the VSGP as a generative model to encode 3D pointcloud. The VSGP is selected among different approximation approaches of GP due to the following reasons: i) The variational approximation distinguishes between the inducing points X_m (as a variational parameter) and the kernel hyperparameters (Θ, σ_n). ii) The regularization term $\operatorname{Tr}(\widetilde{K})$ in the variational objective function (Eq. (4)) regularizes the hyperparameters to avoid overfitting of the data. iii) The variational approximation offers a discrete optimization scheme for selecting the inducing inputs X_m from the original data[3].

A. VSGP as a generative model for the occupancy surface

Inspired by [15], we project the occupied points observed by a ranging sensor, e.g., LiDAR, onto a circular surface around the sensor origin with a predefined radius r_{oc}. This surface is called *occupancy surface*, see Fig. 3. In our approach, the sensor observation is defined in the spherical coordinate system, where any *observed point* is described by the tuple $(\theta_i, \alpha_i, r_i)$ which represents the azimuth, elevation, and radius values, respectively. Also, any pointcloud data can be converted from the cartesian coordinates (x_i, y_i, z_i)

[3]For more information about the inducing point selection, check [2]

to the spherical coordinates $(\theta_i, \alpha_i, r_i)$ using the following equations:

$$r_i = \sqrt{x_i^2 + y_i^2 + z_i^2}, \quad \theta_i = \tan^{-1}(y_i, x_i), \quad \alpha_i = \cos^{-1}(z_i / r_i). \tag{5}$$

All observed points that lie on or outside the circular occupancy surface (with a radius $r_i \geq r_{oc}$) are neglected and considered as free space. The rest of the points that are inside the circular surface (with a radius $r_i < r_{oc}$) are projected on the occupancy surface and called the *occupied points*. Therefore, the occupancy surface radius r_{oc} acts as the maximum range of the sensor. Each occupied point $\mathbf{x}_i$ on the surface is defined by two attributes: the azimuth and elevation angles $\mathbf{x}_i = (\theta_i, \alpha_i)$, and assigned an *occupancy value* $f(\mathbf{x}_i)$ that is a function of the point radius r_i. The probability of occupancy $f(\mathbf{x}_i)$ at each point on the occupancy surface is modeled by a VSGP:

$$f(\mathbf{x}) \sim \mathcal{VSGP}\left(m(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right). \tag{6}$$

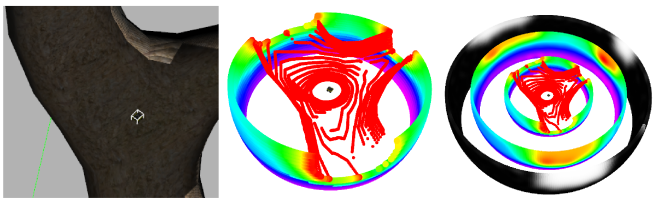
Considering noisy measurements, we add a white noise ε to the occupancy function $f(\mathbf{x})$, so the observed occupancy is described as $y_i = f(\mathbf{x}_i) + \varepsilon$ where ε follows a Gaussian distribution $\mathcal{N}(0, \sigma_n^2)$. The final model of the occupancy surface is a 2D VSGP where the input is the azimuth and elevation angles, $\mathbf{X} = \{(\theta, \alpha)\}_{i=1}^n$, and the corresponding output is the expected occupancy $\mathbf{y} = \{oc_i\}_{i=1}^n$. The three main components of the final VSGP are:

1) Zero-Mean Function $m(\mathbf{x})$ There are different formulas to describe the relationship between the occupancy of a point $f(\mathbf{x}_i)$ on the occupancy surface and its radius r_i [15]. For example, one candidate is $f(\mathbf{x}_i) = 1/r_i$ where r_i is bounded by the minimum and the maximum range of the sensor $r_{min} < r_i < r_{max} = r_{oc}$, where $r_{min} > 0$. Our approach relates the occupancy of a point $f(\mathbf{x}_i)$ to its radius r_i by the following equation $f(\mathbf{x}_i) = r_{oc} - r_i$. This mapping between the occupancy and the radius of a point is compatible with the previous assumption that the occupancy surface radius r_{oc} represents the maximum range of the sensor. Moreover, this mapping is encoded in our VSGP model as a zero-mean function $m(\mathbf{x}) = 0$ that sets the occupancy value of the non-observed points to zero. This mapping behavior mimics the mechanism of the LiDAR itself.

2) Rational Quadratic (RQ) Kernel The RQ kernel is selected because a GP prior with an RQ kernel is expected to have functions that vary across different length scales. This quality of the RQ kernel copes with the nature of the occupancy surface, specifically in unstructured environments where a range of diverse length scales is required, i.e.,

$$k_{\mathrm{RQ}}\left(\mathbf{x}, \mathbf{x}'\right) = \sigma^2 \left(1 + \frac{(\mathbf{x} - \mathbf{x}')^2}{2\alpha\ell^2}\right)^{-\alpha}, \tag{7}$$

where σ^2 is the signal variance, l is the length-scale, and α sets the relative weighting of large and small scale variations. The RQ kernel is more expressive in terms of modeling the occupancy surface than the most commonly used Squared Exponential (SE) kernel. This can be reasoned by the fact that the RQ kernel (when α and $l > 0$) is equivalent to a scale mixture of SE kernels with mixed characteristic length-scales [27]. In practice, we take into account the LiDAR's resolution along both the azimuth and elevation axes to initiate different length-scales along the azimuth and elevation axes, respectively.



(a) (b) (c)

Fig. 3: (a) Simulated scene of a robot in a tunnel (black); (b) The occupancy surface generated from the original pointcloud, where warmer colors reflect smaller $f(\mathbf{x}_i)$ values (less occupancy); (c) The inner surface represents the original occupancy surface (same as in b), and the middle surface represents the reconstructed occupancy surface using the VSGP model. The outer grey-coded surface represents the variance associated with each point on the reconstructed occupancy surface where brighter colors reflect high uncertainty. Raw pointcloud is shown in red in (b) and (c).

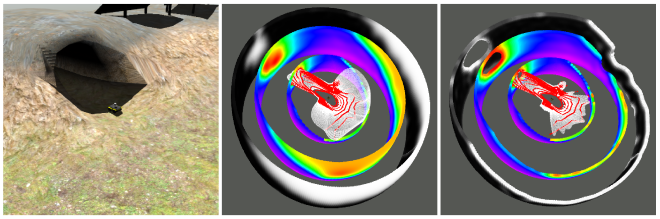
3) Inducing Points Selection The variational learning framework proposed in [2] jointly optimizes the variational parameters (inducing points) and the hyperparameters (Θ, σ) through a variational Expectation-Maximization (EM) algorithm. In general, the original discrete optimization framework [2] suggests having an incremental set of the inducing points, so that during the Expectation step (E-step) a point from the input data is added to the inducing points set to maximize the variational objective function F_V (4) and minimize the $\mathbb{KL}$ divergence between the true and approximated posteriors $\mathbb{KL}[q(f)||p(f|y,\Theta)]$. Then the hyperparameters are updated during the Maximization step (M-step).

The projection of the observed points on the circular surface leads to a limited input domain for the VSGP. In our case, the azimuth and the elevation axes are limited to $(-\pi$ to $\pi)$ and $(-15^\circ$ to $15^\circ)$, respectively. The limited input domain is used to initiate a *fixed number of inducing points* at evenly distributed locations on the *occupied part* of the occupancy surface. In this way, a different combination of the points is selected at each E-step to maximize the variational objective function F_V and minimize the $\mathbb{KL}$ divergence. Then the hyperparameters are updated during the M-step. The number of the inducing points m is chosen to compromise the computational and memory complexity on one side and the accuracy of the reconstructed pointcloud on the other side. More inducing points result in higher computations complexity $\mathcal{O}(nm^2)$, larger memory to store the encoded observation, and higher bandwidth to transfer it. However, more inducing points increase the accuracy of the reconstructed pointcloud. We chose $m = 500$ to keep the average deviation between the reconstructed pointcloud and the original pointcloud under 15 cm, see Section V-A2. After the training phase on the scout side is completed, the selected inducing points are combined together with the hyperparameters values of the VSGP and are transmitted from the scout to the base.

B. Variance-based sampling

On the base side, the inducing points and the values of the hyperparameters, which are received from the scout, are used to reconstruct the original occupancy surface. The reconstruction is done through a GP configured with the same kernel (RQ) and likelihood (Gaussian) as the VSGP on the scout side. The base GP is trained on the inducing points and has a computation complexity of $\mathcal{O}(m^3)$ where m is the number of the inducing points, so we refer it as a sparse GP (SGP) and refer the reconstructed occupancy surface as the *SGP occupancy surface*. A grid of query points $\mathbf{X}^* = \{(\theta_i, \alpha_i)\}_{i=1}^{K}$ with the same resolution of the LiDAR along the azimuth and the elevation axes is generated to reconstruct the original pointcloud from the SGP occupancy surface – we refer to the reconstructed pointcloud as the *SGP pointcloud*. If up-sampling of the pointcloud is required for any reason, a query grid with higher resolution can be used for the reconstruction process. The SGP occupancy model is used to predict the occupancy $f(\mathbf{x}_i)$ of each point $\mathbf{x}_i$ of the query grid $\mathbf{X}^*$. The occupancy is converted back to the spherical radius $r_i = r_{oc} - f(\mathbf{x}_i)$ to restore the 3D spherical coordinates of each point.

One advantage of the GP and its variants over other modeling techniques is the uncertainty (variance) associated with the predicted value at any query point. Considering the VSGP model of the occupancy surface on the scout side, the variance associated with the occupied points is low compared to the variance related to the free points. Selecting the inducing points X_m as a set from the original occupied points maintains low-variance values for the occupied part of the reconstructed SGP occupancy surface on the base side. Therefore, the variance value associated with any point on the reconstructed SGP occupancy surface is used to predict if that point belongs to the *occupied* or the *free* part of the occupancy surface, see Fig. 4. We use a variance threshold V_{th} as a judging criterion. In fact, the variance related to the occupancy surface is different from one observation to another, and it is affected by both the number of observed (occupied) points and their distribution over the occupancy surface. Therefore, we chose the variance threshold V_{th} as a variable that changes with the distribution of the variance over the occupied and free parts of the occupancy surface. V_{th} is defined as a linear combination of the variance mean v_m and standard deviation v_{std} over the surface, i.e., $V_{th} = K_m * v_m + K_{std} * v_{std}$ where K_m and K_{std} are tuning parameters. K_m and K_{std} are tuned by first setting $V_{th} = v_m$ ($K_m = 1$, $K_{std} = 0$), then we increase K_{std} and decrease K_m gradually till we get the values that give the highest accuracy for the reconstructed SGP pointcloud (considering a fixed number of inducing points). Our sampling-based approach is capable of discriminating between the free points that most likely belong to the free part of the SGP occupancy surface and the occupied points that belong to the the occupied part of the SGP occupancy surface. After removing *the free part* of the SGP occupancy surface, the Cartesian coordinates of the occupied points are calculated using the inverse form of Eq. (5) to restore the original point cloud, see Fig. 4c.



(a) (b) (c)

Fig. 4: Variance-based sampling. (a) Simulated scene of the entrance of a tunnel; (b) The original (inner), reconstructed (middle), and variance (outer) surfaces. It also shows the reconstructed pointcloud (in white) through reconstructing from all points (free and occupied) of the occupancy surface. (c) Reconstructed SGP pointcloud after removing all points that most likely belong to the free part of the occupancy surface. Raw pointcloud is shown in red in (b) and (c).

V. Experimental Design and Results

The proposed approach is implemented in Python3 on top of GPflow-v2 [28] under ROS framework [29]. Both real-time simulation and real-time demonstration were considered to evaluate the proposed approach. In both the simulation and the hardware experiments, a VLP-16 LiDAR was used with a maximum range of $10m$, a frequency of $4Hz$, and a resolution of $(0.1^\circ, 2^\circ)$ along the azimuth and the elevation axis, respectively. This configuration results in a maximum pointcloud size of 57600 points. The query grid, which is used to sample the SGP occupancy surface on the base side, has the same resolution as the VLP-16 LiDAR. A 3D occupancy grid map with a resolution of $5cm$ is generated from the reconstructed SGP pointcloud through Octomap [30].

We investigate the performance of our framework and compare it with the GMM approach [15]–[17]. While the GMM approach tackles the occupancy mapping problem as a whole, our approach focuses on compressing sensor observations through limited-bandwidth communication channels. To be able to compare the two approaches, we implemented the GMM approach in such a way that it is used to encode one sensor observation at a time instead of generating an entire occupancy map. We compared our approach with two versions of the GMM approach: i) A CPU-based implementation of GMM that follows the same guidelines of [15]. ii) An upgraded GPU-based implementation of GMM. We implemented the GPU-GMM to have a fair computation comparison with our VSGP approach which runs on GPU.

A. Simulation Experiments

1) Simulation Setup: The simulation setup consists of two machines that communicate to each other over WiFi: The first machine, where the scout and the environment are simulated, is a 64-GB RAM Intel® Core™ i7 PC (*NUC11*) equipped with 6-GB Geforce *RTX*2060 GPU. The second machine, which acts as the base, is a 32-GB RAM Intel® Core™ i7 Laptop (*Alienware*) equipped with 8-GB Geforce *RTX*2080 GPU. Both are connected using a 2.4 GHz WiFi router. The network flow is monitored using the

ifstat tool to evaluate the communication performance. The mine tunnel of the *cpr_inspection* environment, developed by *ClearPath* robotics, is used as our simulation environment. This environment is selected because it represents one of the targeted low-bandwidth subterranean environments. The mine tunnel part of the environment fits in a rectangular with an approximated area of $30 \times 65m^2$. The tunnel length is around $135m$, where the ground elevation and the tunnel height are different from one place to another. The ClearPath Jackal robot is used as the scout. The proposed approach was evaluated through 20 real-time simulation trials. In each trial, the robot starts at the beginning of the cave and follows a predefined path along the mine using way-point-based navigation.

2) Simulation Results We evaluate the performance of our approach based on the reduction in the memory and the communication rate required to transmit the sensor observations between the scout and the base. The VSGP representation requires only 1514 floating points (FP) to represent the entire pointcloud (3 FP for each inducing point (3x500) + 6 FP for robot pose + 6 FP for the hyperparameters). This value is less than the memory needed by the GMM approach which requires $\sim$ 2000 FP (10 FP for each component (10x200) distributed as 6 FP for covariance + 3 FP for mean + 1 FP for weight) [15]. We send the robot pose to the base because our approach encodes the observation relative to the robot body frame, while the GMM approach first transforms the observation from the robot body frame to a global frame using the robot's current pose, then sends the encoded Gaussians densities with respect to the global frame.

To quantify the accuracy of the reconstructed SGP pointcloud, we use the Root Mean Square Deviation (RMSD) between the radius predicted by our approach and the actual radius of each point on the occupancy surface.

$$\text{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} (r_i - \hat{r}_i)^2}{N}}, \tag{8}$$

where N is the size of the pointcloud, r_i is the actual radius at (θ_i, α_i), and $\hat{r}_i$ is the estimated radius value at the same point. Fig. 5a shows the mean and the standard deviation of the RMSD for each predicted point over 110 observations (each observation has around $10K$ to $50K$ points). Also, Fig. 5a implicitly reflects the memory required by VSGP and GMM to store one observation, as described before that the memory required to store one observation can be calculated by multiplying the number of inducing points (bottom x-axis) by 3 and multiplying the number of components (top x-axis) by 10. We match pairs of the VSGP and GMM models (in terms of the number of inducing points and components) based on the memory requirement and the accuracy of the reconstructed pointcloud (reflected by the RMSD) for each pair, see table I. For example, 500-inducing points VSGP results in an average RMSD value for each point of 9 cm with a standard deviation of 10 cm. This corresponds to an average RMSD of 11 cm with a standard deviation of 25 cm for a 200-components GMM.

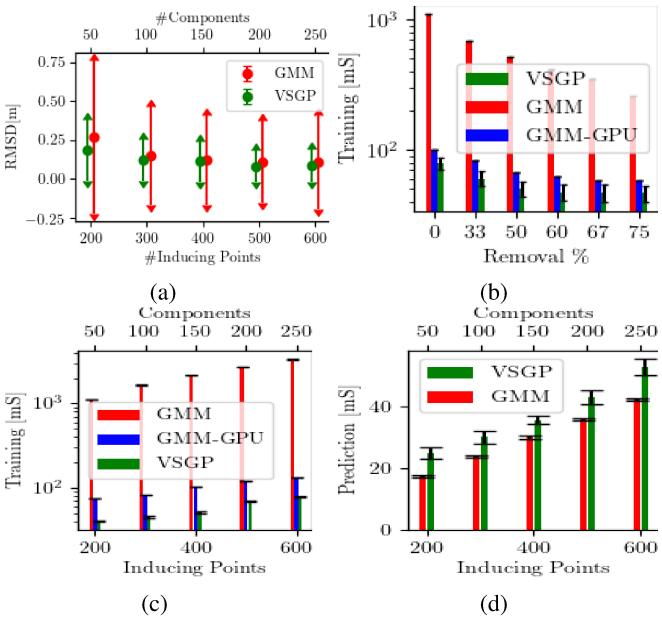


Fig. 5: Performance comparisons. (a) shows the RMSD between the reconstructed and the original pointcloud for VSGP(vs #inducing points) and GMM(vs #components); (b) illustrates the training time against the pointcloud size (considering 500-inducing points VSGP, and equivalently, 200-components GMM); (c) represents the training time versus the #VSGP-inducing points and #GMM-components; (d) shows the prediction time versus the #VSGP-inducing points and #GMM-components.

TABLE I: VSGP vs GMM(ind: inducing, cps: components)

VSGP			GMM		
# ind	Memory ~FPs	RMSD ~cm	# cps	Memory ~FPs	RMSD ~cm
200	600	20±22	50	500	27±50
300	900	14±15	100	1000	16±35
400	1200	12±14	150	1500	13±31
500	1500	9±10	200	2000	11±29
600	1800	9±10	250	2500	11±30

Now we analyze the results in Fig. 5. Fig. 5a shows the RMSD values associated with VSGP have a smaller standard deviation than the GMM's. It also shows that increasing the number of the VSGP-inducing points (bottom x-axis) or the number of the GMM-components (top x-axis) will result in smaller RMSD (higher accuracy).

An intensive evaluation of the training and the prediction phases is presented in Figs. 5b-5d. The reduction in the training time versus the reduction in the size of the raw pointcloud is presented in Fig. 5b, where 0% removal percent means a pointcloud size of 57.6K points. Fig. 5c shows the increase in training time versus the number of inducing points and the number of components. We compare the training time of the VSGP, the GMM-CPU (considering the default configuration of the GMM approach used in [15]), and the GMM-GPU implementation. The results show that our approach outperforms both the CPU and GPU implementation of the GMM approach in terms of training time. Fig. 5d presents the variation of the prediction time of the VSGP versus the number of the inducing points, where the

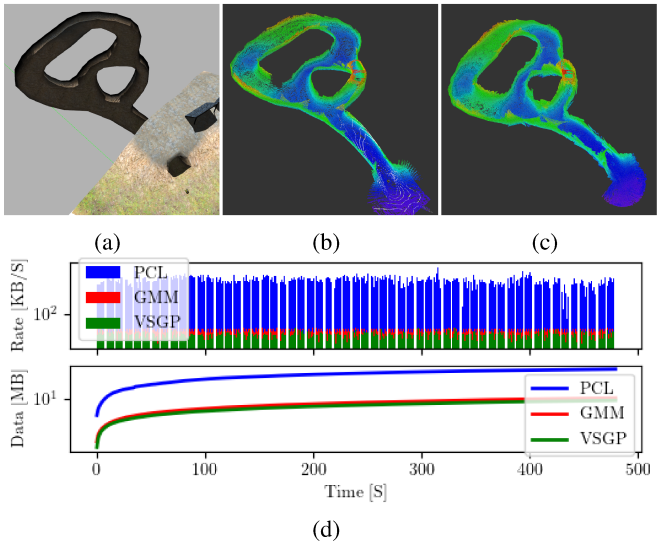


Fig. 6: (a) Simulated mine environment; (b) Octomap of the mine generated from the original pointcloud; (c) Octomap generated from the reconstructed SGP pointcloud; (d) Communication rate and accumulated data sent from the scout to the base for sending raw pointcloud PCL(1750KB/S, 840MB), GMM data(25.8KB/S, 12.4MB), and VSGP data(18.2KB/S, 8.7MB). The y-axis is plotted in log-scale.

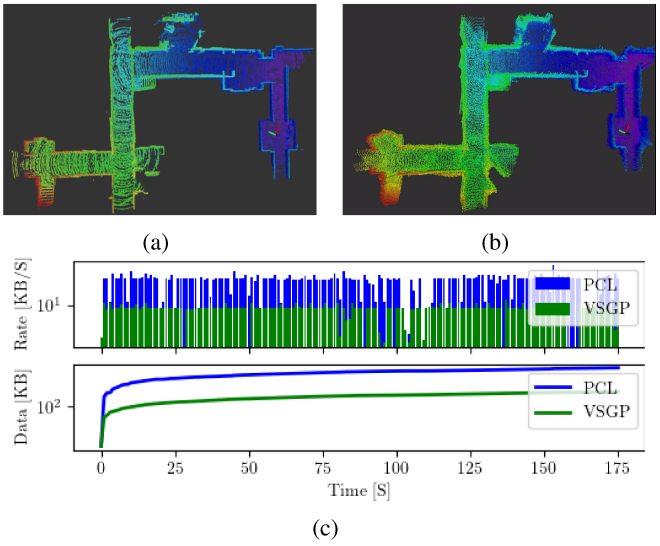


Fig. 7: Indoor demonstration. (a) Octomap of the laboratory building generated from the original pointcloud. (b) Octomap generated from the reconstructed SGP pointcloud. (c) Reduction in the communication rate and the accumulated data sent from the scout to the base, where log-scale is used for y-axis. PCL represents the raw pointcloud.

values shown in the figure represent the time required to predict the occupancy value associated with all the points of the grid query $\mathbf{x}$ (57600 points).

Fig. 5d indicates that for a matching pair of GMM and VSGP (Table I), GMM has a less sampling time than the paired VSGP. However, the pointcloud reconstruction process of the VSGP is more convenient than the GMM approach because a fundamental difference between sampling the VSGP and the GMM is that: when we sample from a GMM, we get a sample (from a distribution) with random values $(\theta_s, \alpha_s, r_s)$, so we *do not have control* over the location of the sample on the occupancy surface (θ_s, α_s). In contrast, for the VSGP approach, we predict the radius value r_s for a certain point on the occupancy surface defined by (θ_s, α_s). So, we *have control* over the point location on the occupancy surface. While constructing the 3D octomap of the tunnel environment using the scout-base scheme, the average communication rate was 1750 KB/S, 25.8 KB/S, and 18.2 KB/S for sending raw point clouds, GMM encoded data, and VSGP encoded data respectively, see Fig. 6d. The accumulated data sent through the network is reduced from 840 MB for sending raw pointcloud to 12.4 MB in case of GMM and 8.7 MB in case of VSGP. This indicates a compression ratio of ~ 96 $(840/8.7 \sim 1750/18.2)$.

B. Hardware Experiment

A Jackal mobile robot, equipped with a VLP-16 LiDAR and the *NUC11* PC, was used as the scout, while the *Alienware* laptop was used as the base. The demonstration was conducted in an indoor environment, where the VSGP-encoded pointcloud data was sent from the scout to the base to generate a 3D Octomap [30] of the building from the SGP reconstructed pointcloud in real-time, see Fig. 7. Fig. 7c shows the reduction in the communication rate for the hardware experiment. The communication rate dropped from around 560 KB for transmitting raw pointcloud to around 8 KB for transmitting the encoded VSGP (this ratio is equivalent to 70 times smaller rate). The communication rate of the hardware experiment is low compared to the simulation experiment because the LiDAR resolution was halved during the hardware experiment. The total amount of data transmitted at the end of each trial was around 100 MB for sending raw pointcloud and only around 1.4 MB for sending the VSGP encoded observation.

VI. Conclusion

In this paper, we introduce a lightweight representation for the 3D pointcloud using the VSGP. This representation allows high-fidelity observations to be efficiently stored and transmitted through limited-bandwidth communication channels. Based on the results of the simulation and hardware experiments, our approach results in around 70-100 times smaller size representation of the sensor observation. This compact representation can facilitate many of the robotics applications which are limited by the communication bandwidth such as subterranean and underwater exploration, search and rescue missions, and planetary exploration. In addition, our approach can also be beneficial in the context of multi-robot collaboration where a number of robots are required to share high-volume information (3D pointcloud) through low-bandwidth channels.

Acknowledgement

This work was supported by National Science Foundation with grant numbers 2006886 and 2047169.

REFERENCES

[1] V. H. Cid et al. Keeping communications flowing during large-scale disasters: leveraging amateur radio innovations for disaster medicine. *Disaster medicine and public health preparedness*, 12(2):257–264, 2018.

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

[3] Chao Cao, Marius Preda, and Titus Zaharia. 3d point cloud compression: A survey. In *The 24th International Conference on 3D Web Technology*, pages 1–9, 2019.

[4] Yu Feng, Shaoshan Liu, and Yuhao Zhu. Real-time spatio-temporal lidar point cloud compression. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 10766–10773. IEEE, 2020.

[5] Tim Golla and Reinhard Klein. Real-time point cloud compression. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5087–5092. IEEE, 2015.

[6] Sébastien Lasserre, David Flynn, and Shouxing Qu. Using neighbouring nodes for the compression of octrees representing the geometry of point clouds. In *Proceedings of the 10th ACM Multimedia Systems Conference*, pages 145–153, 2019.

[7] Dorina Thanou, Philip A Chou, and Pascal Frossard. Graph-based compression of dynamic 3d point cloud sequences. *IEEE Transactions on Image Processing*, 25(4):1765–1778, 2016.

[8] Yan Huang, Jingliang Peng, C-C Jay Kuo, and M Gopi. Octree-based progressive geometry coding of point clouds. In *PBG@ SIGGRAPH*, pages 103–110, 2006.

[9] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1313–1323, 2020.

[10] Maurice Quach, Jiahao Pang, Dong Tian, Giuseppe Valenzise, and Frédéric Dufaux. Survey on deep learning-based point cloud compression. *Frontiers in Signal Processing*, 2022.

[11] Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters*, 6(2):2060–2067, 2021.

[12] Wei Yan, Shan Liu, Thomas H Li, Zhu Li, Ge Li, et al. Deep autoencoder-based lossy geometry compression for point clouds. *arXiv preprint arXiv:1905.03691*, 2019.

[13] Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma. Sparse tensor-based multiscale representation for point cloud geometry compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[14] Yun He, Xinlin Ren, Danhang Tang, Yinda Zhang, Xiangyang Xue, and Yanwei Fu. Density-preserving deep point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2333–2342, 2022.

[15] W. Tabib et al. Real-time information-theoretic exploration with gaussian mixture model maps. In *Robotics: Science and Systems*, 2019.

[16] C. O'Meadhra et al. Variable resolution occupancy mapping using gaussian mixture models. *IEEE Robotics and Automation Letters*, 4(2):2015–2022, 2018.

[17] Task-Specific Manipulator Design. Communication-efficient planning and mapping for multi-robot exploration in large environments. *Journal Article*, 15(2):e1971, 2019.

[18] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[19] A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin. Efficient planning of informative paths for multiple robots. In *the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2204–2211, 2007.

[20] Ruofei Ouyang, Kian Hsiang Low, Jie Chen, and Patrick Jaillet. Multi-robot active sensing of non-stationary gaussian process-based environmental phenomena. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pages 573–580, 2014.

[21] Weizhe Chen, Roni Khardon, and Lantao Liu. Ak: Attentive kernel for information gathering. *arXiv preprint arXiv:2205.06426*, 2022.

[22] Duy Nguyen-tuong and Jan Peters. Local gaussian process regression for real time online model learning and control. In *In Advances in Neural Information Processing Systems 22 (NIPS*, 2008.

[23] Lehel Csató and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668, 2002.

[24] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18:1257, 2006.

[25] Matthias Seeger. Bayesian gaussian process models: Pac-bayesian generalisation error bounds and sparse approximations. Technical report, University of Edinburgh, 2003.

[26] Rishit Sheth, Yuyang Wang, and Roni Khardon. Sparse variational inference for generalized gp models. In *International Conference on Machine Learning*, pages 1302–1311. PMLR, 2015.

[27] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[28] A. Matthews et al. Gpflow: A gaussian process library using tensorflow. *J. Mach. Learn. Res.*, 18(40):1–6, 2017.

[29] 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.

[30] A. Hornung et al. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.