Gaussian Variational Inference with Covariance Constraints Applied to Range-only Localization

Abhishek Goudar, Wenda Zhao, Timothy D. Barfoot, and Angela P. Schoellig

Abstract— **Accurate and reliable state estimation is becoming increasingly important as robots venture into the real world. Gaussian variational inference (GVI) is a promising alternative for nonlinear state estimation, which estimates a full probability density for the posterior instead of a point estimate as in maximum a posteriori (MAP)-based approaches. GVI works by optimizing for the parameters of a multivariate Gaussian (MVG) that best agree with the observed data. However, such an optimization procedure must ensure the parameter constraints of a MVG are satisfied; in particular, the inverse covariance matrix must be positive definite. In this work, we propose a tractable algorithm for performing state estimation using GVI that guarantees that the inverse covariance matrix remains positive definite and is well-conditioned throughout the optimization procedure. We evaluate our method extensively in both simulation and real-world experiments for range-only localization. Our results show GVI is consistent on this problem, while MAP is over-confident.**

I. INTRODUCTION

State estimation is a crucial component of any autonomous system. A robotic platform uses data from its sensors to estimate its state, which typically includes its location in the world and other physical parameters. This process is challenging since data reported by the sensors is generally noisy. Hence, it is important to not only estimate the state accurately, but also quantify the uncertainty in the estimated state. For instance, if the estimated position is off by some margin, any subsystem such as a planner or controller that uses this information should be aware of the uncertainty in the estimated position. Many planning algorithms use covariance as a tool for generating safe and efficient paths. As such, accurate estimation of uncertainty is important.

Probabilistic state estimation provides a systematic framework for managing uncertainty. In a probabilistic setting, the latest state $\mathbf{x}$ is treated as a random variable from a probability density function (PDF), $p(\mathbf{x})$. After sensor data $\mathbf{y}$ becomes available, Bayes' theorem can be used to infer the *posterior* PDF $p(\mathbf{x}|\mathbf{y})$. Application of Bayes' theorem involves calculation of the *partition function* $p(\mathbf{y}) = \int p(\mathbf{x}|\mathbf{y})p(\mathbf{x})$. The partition function is tractable only for a limited set of linear problems and linear approximations. For general nonlinear problems, evaluation of the partition function is generally intractable. Thus, different approximations are sought to calculate the posterior.

A popular choice for approximating the posterior is the *extended Kalman filter*, which performs a local linear approximation. However, this method can incur errors since the problem is linearized about an operating point only once. A popular approach to estimate the posterior is to use sampling-based methods such as Markov chain Monte Carlo (MCMC) [1]. While this approach yields accurate solutions, it requires high computation. An alternative approach involves calculating a point approximation to the posterior known as *maximum a posteriori*. In MAP, the uncertainty is approximated locally (known as the *Laplace* approximation), which can result in poor uncertainty estimates.

In this work, we use *variational inference* (VI) [2] for state estimation. In VI, the intractable posterior PDF $p(\mathbf{x}|\mathbf{y})$ is approximated with another PDF $q(\mathbf{x})$ that is generally tractable. VI turns the problem of estimating the true posterior into an optimization problem by finding another tractable PDF $q(\mathbf{x})$ that minimizes a particular distance, such as the Kullback Leibler (KL) divergence, to the true posterior distribution. In Gaussian variational inference (GVI), the distribution $q(\mathbf{x})$ is chosen from the parametric family of multivariate Gaussian distributions. This choice is motivated by the the *central limit theorem*. Some of the benefits of VI include: *(i)* unlike MCMC methods, VI lends itself to tractable methods for approximate Bayesian inference, *(ii)* VI estimates the entire PDF and not just the most probable state as in MAP, and *(iii)* VI lends itself to a systematic framework for learning different hyperparameters [3].

The optimization of variational parameters poses certain challenges. Firstly, the parameters of a multivariate Gaussian must satisfy constraints: the covariance matrix, which quantifies the uncertainty, must be positive definite. A second challenge is to ensure that the covariance matrix is well-conditioned. Thirdly, any inherent sparsity of the covariance matrix must be maintained during the course of optimization.

We propose an algorithm that guarantees these three covariance constraints. We also show that the estimated covariance with the proposed method is consistent. Finally, to make GVI tractable, we use the fact that in most robotics applications the joint likelihood function $p(\mathbf{x}, \mathbf{y})$ factors [4]. This results in expectations involving PDFs whose dimension is smaller compared to the full PDF. A factor graph with such a structure for *range-only localization* is shown in Figure 2.

In summary, the main contributions of this paper are *(i)* we present an algorithm for state estimation using tractable GVI that guarantees covariance constraints and provides a well-conditioned inverse covariance matrix, *(ii)* we show how efficient computation of expectations for joint likelihood functions that factor, as proposed in [4], can be incorporated into our method, and *(iii)* we perform evaluation of the

The authors are with the Institute for Aerospace Studies, University of Toronto, Canada, and affiliated with the Vector Institute for Artificial Intelligence in Toronto. E-mails: {firstname.lastname}@robotics.utias.utoronto.ca

proposed method by applying it to range-only localization in simulation and real-world experiments.

The paper is organized as follows. In Section II we review related work. We introduce notation in Section III and formulate the problem in Section IV. An introduction to GVI, its challenges, and the proposed solution are presented in Section V. Experimental evaluation is presented in Section VI followed by the conclusion in Section VII.

II. Related Work

In this section, we review the works that are the most relevant to our method. A detailed overview of the different estimation methods can be found in [5]–[7].

The application of variational inference to state estimation has gained traction recently. As alluded to in the introduction section, performing GVI is in general intractable. Various approximations have been proposed to address this. In [8], it is assumed that the posterior factors completely, giving rise to the *mean field approximation* [9]. This assumption ignores correlations between the states and can result in poor estimation [1]. More recently, a batch state estimation method called exactly sparse Gaussian variational inference (ESGVI), which calculates the full posterior by exploiting likelihood functions which factor, was proposed in [4]. In [4] the authors iteratively maximize a lower bound, known as the evidence lower bound (ELBO), which in turn minimizes the KL divergence. This is also the most relevant work to our method. The update equations proposed in [4] guarantee sparsity constraints. However, the covariance matrix is not guaranteed to be positive definite. The authors in [4] also present a Gauss-Newton (ESGVI-GN) style approach, which uses Jenson's inequality to ensure all desired covariance constraints are met. However, ESGVI-GN can result in parameter estimates that are conservative.

Variational inference has been widely used in the field of machine learning [10], [11]. As such, many methods have been proposed to minimize ELBO (or its variants) while ensuring parameter constraint satisfaction [12]–[14]. In [14], a method is proposed based on *natural gradient descent* [15] where the *Cholesky* factor of the covariance matrix is updated directly. In [12], parameters for the square root of the covariance matrix are optimized instead of directly optimizing for the covariance matrix. The matrix square root is then mapped to the covariance matrix using the *matrix exponential* map. This method is generalized to other structured covariance matrices (such as inverse-covariance matrix or low-rank covariance matrix) in [13]. We use this approach to make sure the convariance constraints are satisfied. Unlike [13], where the authors use a second-order approximation, we show that a first-order approximation can be used with our method, reducing the amount of computation required. Additionally, the previous works do not take advantage of likelihood functions that factor.

To the best of our knowledge, the application of GVI for tractable state estimation that guarantees all of our desired covariance constraints are satisfied and exploits likelihood functions that factor has not been proposed previously.

III. Notation

We represent scalar variables by unbold letters, vectors by bold lowercase letters, and matrices by bold uppercase letters. The set GL^N denotes the general linear group of all invertible $N \times N$ matrices. The sets $\mathcal{S}^N$, $\mathcal{S}^N_+$, and $\mathcal{S}^N_{++}$ denote the set of $N \times N$ symmetric, symmetric positive semidefinite, and symmetric positive definite matrices, respectively. The operator $\mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})]$ denotes the expected value of the function f under the distribution $p(\mathbf{x})$: $\int p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x}$. The determinant of a matrix $\mathbf{A}$ is denoted by $|\mathbf{A}|$. The gradient of a function $f(\mathbf{x})$ is given by $\nabla_{\mathbf{x}} f(\mathbf{x}) = \partial f(\mathbf{x})/\partial \mathbf{x}$ with $\nabla_{\mathbf{x}} f(\mathbf{x})|_{\mathbf{x}_l}$ denoting the value of the gradient evaluated at $\mathbf{x}_l$. Similarly, the Hessian is given by $\nabla^2_{\mathbf{xx}} f(\mathbf{x}) = \partial^2 f(\mathbf{x})/\partial \mathbf{x}^T \partial \mathbf{x}$. The letter $\mathbf{I}$ denotes the identity matrix whose dimensions are to be inferred from the context.

IV. Problem Formulation

We represent the general *latent* state by the N-dimensional vector, $\mathbf{x}$. In the context of state estimation, the latent space could denote the entire robot trajectory along with additional states such as position of landmarks and sensor-sensor extrinsic parameters. The set of observations is denoted by $\mathbf{y}$. We assume that the joint likelihood $p(\mathbf{x}, \mathbf{y})$ factors:

$$p(\mathbf{x}, \mathbf{y}) = \prod_{k=1}^{K} p_k(\mathbf{x}_k, \mathbf{y}_k), \tag{1}$$

where $\mathbf{x}_k$ and $\mathbf{y}_k$ are subsets of $\mathbf{x}$ and $\mathbf{y}$, and K is the total number of factors. The goal is to find a tractable approximation to the posterior $p(\mathbf{x}|\mathbf{y})$ using variational inference.

V. Gaussian Variational Inference

In variational inference, the goal is to select a probability density function, $q(\mathbf{x})$, over the latent variables that best approximates the true posterior density, $p(\mathbf{x}|\mathbf{y})$ [1]. A common measure of similarity between two probability densities is the Kullback-Leibler (KL) divergence [16]:

$$D_{\mathrm{KL}}(q||p) = -\int_{\mathbb{R}^N} q(\mathbf{x}) \ln\left(\frac{p(\mathbf{x}|\mathbf{y})}{q(\mathbf{x})}\right) d\mathbf{x}. \tag{2}$$

The problem of approximating the posterior density $p(\mathbf{x}|\mathbf{y})$ involves finding $q(\mathbf{x})$ that minimizes (2). In GVI, the density $q(\mathbf{x})$ is restricted to the family of multivariate Gaussian densities parameterized by the mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma} \in \mathcal{S}^N_{++}$, $q(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$:

$$q(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right). \tag{3}$$

As is convention in robotics applications, we parameterize $q(\mathbf{x})$ with the mean $\boldsymbol{\mu}$ and the *precision* or the *information matrix* $\boldsymbol{\Sigma}^{-1}$. The motivation is that the information matrix is generally sparse, while the covariance matrix is dense. Specifically, the information matrix is block-tridiagonal for batch trajectory estimation and is an 'arrowhead' matrix for SLAM problems [7]. Thus, the optimization parameters are $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}^{-1}\}$ and the parameter space is $\Omega_{\boldsymbol{\theta}} = \mathbb{R}^N \times \mathcal{S}^N_{++}$.

Using Bayes' theorem and the definition of entropy for a multivariate Gaussian, equation (2) can be rewritten as:

$$D_{\mathrm{KL}}(q||p) = \mathbb{E}_{q(\mathbf{x})}\left[-\ln p(\mathbf{x},\mathbf{y})\right] + \frac{1}{2}\ln|\boldsymbol{\Sigma}^{-1}| + \text{const},$$

where all the terms that are constant w.r.t. $q(\mathbf{x})$ are absorbed into the constant term. Thus, minimizing $D_{\mathrm{KL}}(q||p)$ is equivalent to finding the parameters $\boldsymbol{\theta}$ of $q(\mathbf{x})$ that solve the following optimization problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Omega_{\boldsymbol{\theta}}} \mathcal{L}(q), \tag{4}$$

where

$$\mathcal{L}(q) = \mathbb{E}_{q(\mathbf{x})}\left[\phi(\mathbf{x},\mathbf{y})\right] + \frac{\gamma}{2}\ln|\boldsymbol{\Sigma}^{-1}|, \tag{5}$$

and where $\phi(\mathbf{x},\mathbf{y}) = -\ln p(\mathbf{x},\mathbf{y})$ and $\gamma \geq 0$ is a hyperparameter. The first term in the cost function pushes the solution to match the observed data while the second term prevents the solution from being too (un)certain. The hyperparameter $\gamma \geq 0$ is used to balance between fitting the observed data and the solution being too (un)certain. The hyperparameter can be tuned depending on the application.

Optimization of (5) can be achieved using Newton-like methods, which minimize a local quadratic approximation to the cost function. Alternatively, a gradient-descent-based approach can be used. However, performing minimization without taking into account the *structure* of the parameter space can result in solutions that violate the set constraints. In particular, for the case of the multivariate Gaussian, it is necessary that the information matrix be in the set of positive definite matrices: $\boldsymbol{\Sigma}^{-1} \in \mathcal{S}_{++}^N$. Next we discuss two methods proposed in literature to minimize (5).

An iterative approach, based on Newton's method, is proposed in ESGVI [4]. It proceeds by minimizing a local quadratic approximation to the cost function (5) and updates the parameters $\boldsymbol{\theta}$ in an iterative manner:

$$\boldsymbol{\Sigma}_{i+1}^{-1} = \mathbb{E}_{q_i(\mathbf{x})}\left[\nabla^2_{\mathbf{xx}}\phi(\mathbf{x},\mathbf{y})\right], \tag{6}$$

$$\boldsymbol{\Sigma}_{i+1}^{-1}\delta\boldsymbol{\mu} = -\mathbb{E}_{q_i(\mathbf{x})}\left[\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right], \tag{7}$$

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \delta\boldsymbol{\mu}, \tag{8}$$

where i is the iteration index with $q_i(\mathbf{x})$ denoting the density obtained after the i^{th} iteration. A limitation of the above update scheme is that (6) does not ensure the information matrix $\boldsymbol{\Sigma}^{-1}$ remains positive definite. Specifically, the information matrix is set to the expected value of the Hessian of the cost function, which is not guaranteed to be positive definite for many problems, especially when the evaluation point is not close to a minimum of the cost function.

A popular choice for minimizing cost functions of the form (5) is *natural gradient descent* (NGD) [15]. In contrast to regular gradient descent, NGD exploits the manifold structure to achieve better convergence. Specifically, the set of all probability density functions parameterized by $\boldsymbol{\theta} \in \boldsymbol{\Omega}_{\boldsymbol{\theta}}$ is a manifold where each point $\boldsymbol{\theta}$ denotes a probability density function with KL divergence providing the manifold with a Riemannian structure [14]. The natural gradient $\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}$ of (5) is related to the regular gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ as: $\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(q) = F_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}$, where $F_{\boldsymbol{\theta}}^{-1}$ is the inverse of the Fisher information matrix (FIM). The connection between Gaussian variational inference and NGD has already been established in [4]. Note that the choice of parameterization $\boldsymbol{\theta}$ for $q(\mathbf{x})$ can simplify the computation of the FIM significantly [17]. The update equations for the mean and the information matrix using NGD can be written as [13], [14]:

$$\boldsymbol{\Sigma}_{i+1}^{-1} = (1-\beta\gamma)\boldsymbol{\Sigma}_i^{-1} + \beta\mathbb{E}_{q_i(\mathbf{x}))}\left[\nabla^2_{\mathbf{xx}}\phi(\mathbf{x},\mathbf{y})\right], \tag{9}$$

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i - \boldsymbol{\Sigma}_{i+1}\beta\mathbb{E}_{q_i(\mathbf{x})}\left[\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right], \tag{10}$$

where β is the step size and γ is the hyperparameter introduced earlier. Although NGD exploits the manifold structure of the space of probability density functions, the standard NGD *update* of (9) does not guarantee that the updated information matrix is positive definite.

A. NGD using Local Parameters

In order to ensure that the information matrix remains positive definite, we work with the matrix factor of the information matrix instead of the information matrix itself, where the matrix factor $\mathbf{A}$ of any matrix $\mathbf{S}$ is defined as $\mathbf{S} = \mathbf{A}\mathbf{A}^T$. We make use of the following two properties in the optimization process: *(i)* any matrix $\mathbf{S} \in \mathcal{S}_{++}^N$ can be decomposed as: $\mathbf{S} = \mathbf{A}\mathbf{A}^T$, where $\mathbf{A} \in \mathrm{GL}^N$, and *(ii)* for any real matrix $\mathbf{A}$, and $\epsilon > 0$, $\left(\mathbf{A}\mathbf{A}^T + \epsilon\mathbf{I}\right) \in \mathcal{S}_{++}^N$. The proofs can for these properties can be found [18].

An overview of the approach is as follows. We start by computing a matrix factor of the information matrix, $\boldsymbol{\Sigma}_i^{-1} = \mathbf{A}_i\mathbf{A}_i^T$. In each iteration of the optimization routine, we perform a NGD step on the matrix factor $\mathbf{A}_i$ to generate an updated estimated $\mathbf{A}_{i+1}$, which is used to update the information matrix as $\boldsymbol{\Sigma}_{i+1}^{-1} = \mathbf{A}_{i+1}\mathbf{A}_{i+1}^T + \epsilon\mathbf{I}$.

In this work, we chose the square root of the information matrix as the matrix factor: $\boldsymbol{\Sigma}^{-1} = \mathbf{A}\mathbf{A}^T$. To perform tractable NGD on the space of the matrix factors, we use the method of *structured NGD using local parameters* proposed in [13]. Using structured NGD, the update equations for the mean and the matrix factor are given by:

$$\mathbf{A}_{i+1} = \mathbf{A}_i\exp(\beta(\mathbf{A}_i^{-1}\nabla_{\boldsymbol{\Sigma}}\mathcal{L}(q_i)\mathbf{A}_i^{-T})), \tag{11}$$

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i - \boldsymbol{\Sigma}_{i+1}\beta\nabla_{\boldsymbol{\mu}}\mathcal{L}(q_i), \tag{12}$$

where

$$\exp(\beta\mathbf{M}) = \sum_{j=0}^{\infty}\frac{(\beta\mathbf{M})^j}{j!}, \tag{13}$$

is the matrix exponential. To make the computation further tractable, we consider a first-order approximation of the matrix exponential map:

$$h(\beta\mathbf{M}) \approx \mathbf{I} + \beta\mathbf{M}. \tag{14}$$

Using Stein's lemma [19], the gradient of (5) with respect to the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ can be written as

$$\nabla_{\boldsymbol{\mu}}\mathcal{L}(q) = \mathbb{E}_{q(\mathbf{x})}\left[\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right], \tag{15}$$

$$\nabla_{\boldsymbol{\Sigma}}\mathcal{L}(q) = \frac{1}{2}\mathbb{E}_{q(\mathbf{x})}\left[\nabla^2_{\mathbf{x},\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right] - \frac{\gamma}{2}\boldsymbol{\Sigma}^{-1}. \tag{16}$$

Inserting (16) in (11) and expanding (14) we get

$$\mathbf{A}_{i+1} = \left(1 - \frac{\beta\gamma}{2}\right)\mathbf{A}_i + \frac{\beta}{2}\mathbb{E}_{q_i(\mathbf{x})}\left[\nabla^2_{\mathbf{x},\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right]\mathbf{A}_i^{-T}.$$

The above expression relies on calculation of the matrix factor inverse, $\mathbf{A}_i^{-1}$, which will be dense. However, this can be avoided by a simple reformulation. Multiplying both sides of the above equation by $\mathbf{A}_i^T$ on the right and exploiting the symmetric nature of the Hessian and the information matrix we arrive at

$$\mathbf{A}_i\mathbf{A}_{i+1}^T = \underbrace{\left(1 - \frac{\beta\gamma}{2}\right)\mathbf{\Sigma}_i^{-1} + \frac{\beta}{2}\mathbb{E}_{q_i(\mathbf{x})}\left[\nabla^2_{\mathbf{x},\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right]}_{\mathbf{S}_i}. \tag{17}$$

In most robotics applications the Hessian and the information matrix are sparse. Thus, the updated matrix factor $\mathbf{A}_{i+1}$ can be calculated very efficiently as the solution to system of sparse linear equations: $\mathbf{A}_i\mathbf{A}_{i+1}^T = \mathbf{S}_i$. Additionally, if the initial information matrix, $\mathbf{\Sigma}_0^{-1}$, is sparse, then the update (17) will result in sparse matrix factors, $\mathbf{A}_i$ (see Figure 1).

The update equation for the mean is obtained by substituting (15) in (12):

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i - \beta\mathbf{\Sigma}_{i+1}\mathbb{E}_{q_i(\mathbf{x})}\left[\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right]. \tag{18}$$

Equations (17) and (18) are the proposed updates for the matrix factor and the mean, respectively. We call the proposed algorithm ESGVI-C to highlight the formal guarantees on covariance constraints. The complete algorithm is presented in Algorithm 1 about which we now make some remarks. In [13], the authors use a second-order approximation of the matrix exponential map, which ensures that $\mathbf{A}_{i+1}$ is invertible. However, the second-order approximation is computationally more expensive. Specifically, it incurs additional matrix multiplication operations of order $\mathcal{O}(N^3)$. The first-order approximation in (14) by itself does not guarantee $\mathbf{A}_{i+1}$ is invertible. In such a case the matrix factor can be updated after line 7 of Algorithm 1 by factoring $\mathbf{\Sigma}_{i+1}^{-1}$ as $\tilde{\mathbf{A}}_{i+1}\tilde{\mathbf{A}}_{i+1}^T = \mathbf{\Sigma}_{i+1}^{-1}$, and updating $\mathbf{A}_{i+1} \leftarrow \tilde{\mathbf{A}}_{i+1}$. In practice, we found that $\mathbf{A}_{i+1}$ was invertible after update (17). Additional computational savings can be obtained by using $\mathbf{A}_{i+1}$ to compute $\delta\boldsymbol{\mu}$ in line 8 of Algorithm 1. We now prove that the information matrix iterates obtained using Algorithm 1 remain positive definite.

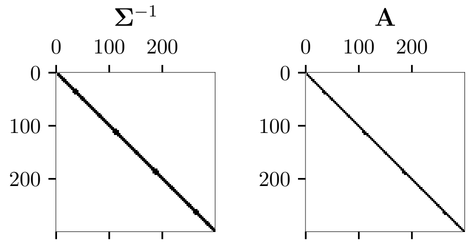


Fig. 1: Sparsity pattern of the information matrix $\mathbf{\Sigma}^{-1}$ and the matrix factor $\mathbf{A}$ corresponding to range-only localization setup involving 100 robot positions from a real-world experiment. The information matrix has block-tridiagonal structure with non-zero entries only along block lower diagonal, main diagonal, and upper diagonal.

Algorithm 1: ESGVI-C

```
Require: $\gamma, \beta, \epsilon, J$.
 1: Initialize $\boldsymbol{\mu}_0$ and $\mathbf{\Sigma}_0^{-1}$.
 2: $\mathbf{A}_0 \leftarrow (\mathbf{\Sigma}_0^{-1})^{\frac{1}{2}}$.
 3: while $i < J$ do
 4:     Compute $\mathbb{E}_{q_i(\mathbf{x})}[\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})]$.
 5:     Compute $\mathbb{E}_{q_i(\mathbf{x})}[\nabla^2_{\mathbf{x},\mathbf{x}}\phi(\mathbf{x},\mathbf{y})]$.
 6:     Compute $\mathbf{A}_{i+1}$.                                  ▷ (17)
 7:     $\mathbf{\Sigma}_{i+1}^{-1} \leftarrow \mathbf{A}_{i+1}\mathbf{A}_{i+1}^T + \epsilon\mathbf{I}$ .
 8:     Compute $\delta\boldsymbol{\mu}$: $\mathbf{\Sigma}_{i+1}^{-1}\delta\boldsymbol{\mu} = -\beta\mathbb{E}_{q_i(\mathbf{x})}[\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})]$.
 9:     $\boldsymbol{\mu}_{i+1} \leftarrow \boldsymbol{\mu}_i + \delta\boldsymbol{\mu}$.
10:     $i \leftarrow i + 1$.
11: end while
12: return $\boldsymbol{\mu}_J, \mathbf{\Sigma}_J^{-1}$.
```

Lemma 1: If $\mathbf{A}_i \in \mathrm{GL}^N$, then the information matrix $\mathbf{\Sigma}_{i+1}^{-1}$ computed using Algorithm 1 is positive definite.

Proof: For convenience, we assume $\beta = 1$. Let $\mathbf{M}_i = \mathbf{A}_i^{-1}\nabla_{\mathbf{\Sigma}}\mathcal{L}(q_i)\mathbf{A}_i^{-T}$. The precision matrix at the end of the $(i+1)$th iteration is given by line 7 in Algorithm 1:

$$\begin{aligned}
\mathbf{\Sigma}_{i+1}^{-1} &= \mathbf{A}_{i+1}\mathbf{A}_{i+1}^T + \epsilon\mathbf{I}, \\
&\overset{(11)}{=} \mathbf{A}_i h(\mathbf{M}_i)h(\mathbf{M}_i)^T\mathbf{A}_i^T + \epsilon\mathbf{I}, \\
&\overset{(14)}{\approx} \underbrace{\mathbf{A}_i(\mathbf{I}+\mathbf{M}_i)}_{\mathbf{B}_i}\underbrace{(\mathbf{I}+\mathbf{M}_i)^T\mathbf{A}_i^T}_{\mathbf{B}_i^T} + \epsilon\mathbf{I}, \\
&= (\mathbf{B}_i\mathbf{B}_i^T + \epsilon\mathbf{I}) \succ \mathbf{0} \text{ (positive definite).}
\end{aligned}$$

■

B. Connection to ESGVI

We now show that the update equations for ESGVI can be recovered from our proposed method. The ESGVI mean update step (8) is equal to (18) when $\beta = 1$, which corresponds to the Newton-like update method. For $\beta = 2$ and $\gamma = 1$, the matrix factor update is:

$$\mathbf{A}_i\mathbf{A}_{i+1}^T = \mathbb{E}_{q_i(\mathbf{x})}\left[\nabla^2_{\mathbf{x},\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right]. \tag{19}$$

Note that the update equation (6) is obtained by taking the gradient of the cost function w.r.t. $\mathbf{\Sigma}^{-1}$ and setting it to zero, which occurs at a stationary point of the cost function (5). Note that it is important for the cost function to be locally convex for equation (6) to be valid. In such a case, the matrix factor iterates converge to a stationary point: $\mathbf{A}_{i+1} \approx \mathbf{A}_i$:

$$\mathbf{A}_{i+1}\mathbf{A}_{i+1}^T = \mathbf{\Sigma}_{i+1}^{-1} = \mathbb{E}_{q_i(\mathbf{x})}\left[\nabla^2_{\mathbf{x},\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right],$$

which is identical to (6).

C. Calculation of Expectations

A central part of Algorithm 1 is the computation of the expectations, $\mathbb{E}_{q_i(\mathbf{x})}[\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})]$ and $\mathbb{E}_{q_i(\mathbf{x})}[\nabla^2_{\mathbf{x},\mathbf{x}}\phi(\mathbf{x},\mathbf{y})]$. A common approach is to approximate the expectations with a random sample $\tilde{\mathbf{x}} \sim q(\mathbf{x})$: $\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})|_{\tilde{\mathbf{x}}}$ and $\nabla^2_{\mathbf{x}\mathbf{x}}\phi(\mathbf{x},\mathbf{y})|_{\tilde{\mathbf{x}}}$, which gives a version of stochastic NGD [14]. However, this approach would require many samples to achieve a reasonably accurate approximation to the expectations. Alternatively, the expectations can be approximated at the mean $\boldsymbol{\mu}$ [13], $\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})|_{\boldsymbol{\mu}}$ and $\nabla^2_{\mathbf{xx}}\phi(\mathbf{x},\mathbf{y})|_{\boldsymbol{\mu}}$, giving MAP updates.

We use multidimensional *Gauss-Hermite quadrature* for calculating expectations in this work since it allows for efficient expectation computation for polynomials under Gaussian integrals [6]. The expected value of a function, $f(\mathbf{x})$, under a distribution, $q(\mathbf{x})$, is given by the weighted sum:

$$\mathbb{E}_{q(\mathbf{x})}[f(\mathbf{x})] = \int_{-\infty}^{\infty} f(\mathbf{x})q(\mathbf{x})d\mathbf{x} \approx \sum_{l=1}^{L} w_l f(\mathbf{x}_l),$$

where w_l are *weights*, $\mathbf{x}_l = \boldsymbol{\mu} + \sqrt{\boldsymbol{\Sigma}}\boldsymbol{\xi}_l$ are *sigmapoints*, $\boldsymbol{\xi}_l$ are *unit sigmapoints*, and L is total number of sigmapoints. Similarly, the expected value of the gradient and the Hessian can be calculated as

$$\mathbb{E}_{q_i(\mathbf{x})}\left[\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right] \approx \sum_{l=1}^{L} w_l \nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})|_{\mathbf{x}_l}, \tag{20}$$

$$\mathbb{E}_{q_i(\mathbf{x})}\left[\nabla^2_{\mathbf{x},\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right] \approx \sum_{l=1}^{L} w_l \nabla^2_{\mathbf{xx}}\phi(\mathbf{x},\mathbf{y})|_{\mathbf{x}_l}. \tag{21}$$

The accuracy of the numerical approximation depends on the *degree*, D, of the Gauss-Hermite quadrature. The number of weights and sigmapoints required for a D-degree Gauss-Hermite quadrature is $L = D^N$, where N is the dimension of $\mathbf{x}$. This can be infeasible even for moderately sized problems. Fortunately, as noted by [4], the number of sigmapoint evaluations can be drastically reduced by noting that many problems in robotics have a structure where the function $\phi(\mathbf{x},\mathbf{y})$ factors as

$$\phi(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{K} \psi_k(\mathbf{x}_k,\mathbf{y}_k), \tag{22}$$

where ψ_k is the k^{th} factor, which depends on a subset of the state $\mathbf{x}_k$ and a subset of the observations, $\mathbf{y}_k$. We incorporate an identical method in our approach. For completeness, we provide a brief overview here; a more detailed description can be found in [4]. With the factored function (22), the expectations (20) and (21) can be written as:

$$\begin{aligned}\mathbb{E}_{q_i(\mathbf{x})}\left[\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right] &= \sum_{k=1}^{K} \mathbf{P}_k^T \mathbb{E}_{q_i(\mathbf{x}_k)}\left[\nabla_{\mathbf{x}_k}\psi_k\right], \\ &\approx \sum_{k=1}^{K} \mathbf{P}_k^T \sum_{l=1}^{L_k} w_{k,l}\nabla_{\mathbf{x}_k}\psi_k|_{\mathbf{x}_{k,l}},\end{aligned} \tag{23}$$

$$\begin{aligned}\mathbb{E}_{q_i(\mathbf{x})}\left[\nabla^2_{\mathbf{x},\mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right] &= \sum_{k=1}^{K} \mathbf{P}_k^T \mathbb{E}_{q_i(\mathbf{x}_k)}\left[\nabla^2_{\mathbf{x}_k\mathbf{x}_k}\psi_k\right]\mathbf{P}_k, \\ &\approx \sum_{k=1}^{K} \mathbf{P}_k^T \left(\sum_{l=1}^{L_k} w_{k,l}\nabla^2_{\mathbf{x}_k\mathbf{x}_k}\psi_k|_{\mathbf{x}_{k,l}}\right)\mathbf{P}_k,\end{aligned} \tag{24}$$

where $\mathbf{P}_k$ is the projection matrix so that $\mathbf{x}_k = \mathbf{P}_k\mathbf{x}$. We have omitted the explicit dependency of ψ_k on $(\mathbf{x}_k,\mathbf{y}_k)$ to reduce clutter. The sigmapoints for the factors are given by $\mathbf{x}_{k,l} = \boldsymbol{\mu}_k + \sqrt{\boldsymbol{\Sigma}_{kk}}\boldsymbol{\xi}_{k,l}$, where $\boldsymbol{\Sigma}_{kk}$ corresponds to blocks of $\boldsymbol{\Sigma}$ related to $\mathbf{x}_k$. Note that the expectation is now over the factors and the number of sigmapoint evaluations drops to $L_k = D^{N_k}$ where N_k is the dimension of $\mathbf{x}_k$.

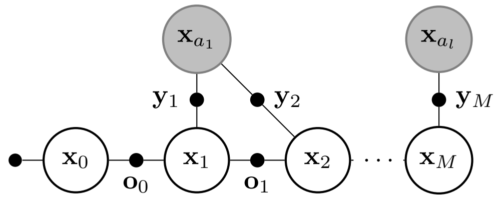


Fig. 2: Factor graph for a range-only (RO) localization setup. In RO localization, a robot equipped with a sensor, such as a wireless radio, estimates its position by measuring the distance to other wireless radios, known as *anchors*, installed in the environment. The trajectory consists of a set of nodes representing the robot position $\mathbf{x}_t$ at time t. The anchor positions, $\mathbf{x}_{a_\#}$, are known as indicated by the filled circles. Odometry measurements are denoted by factors $\mathbf{o}_t$ and the range measurements by factors y_t. Note that the measurements depend only on adjacent nodes. The joint probability density function (PDF) over the entire trajectory can be written as product of smaller PDFs over individual factors.

D. Hyperparameters

The value of the hyperparameter, γ, can be tuned by evaluating the algorithm against a particular metric. For instance, a common metric to measure the quality of estimation is the normalized estimation error squared (NEES):

$$\text{NEES} = \frac{1}{N}(\mathbf{x}_{\text{true}} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_{\text{true}} - \boldsymbol{\mu}), \tag{25}$$

where $\mathbf{x}_{\text{true}}$ is the true value of state, $\mathbf{x}$. If the estimated state is consistent, then the mean NEES value should be equal to the dimension of the state. For example, in the case of robot position estimation, the mean NEES value should be the dimension of the robot position. Hyperparameters, β and ϵ, can be calculated using a back-tracking approach.

VI. Experiments

In this section, we demonstrate the performance of the proposed method using *range-only* (RO) localization. RO localization is a challenging scenario, since the measurements are sparse and the measurement model is nonlinear.

A. Range-only Localization

In RO localization, a robot equipped with a sensor, typically a wireless radio, localizes by measuring the distance or range to multiple *anchors* installed in the environment. The range measurement at any time t is given by

$$r_{l,t} = \|\mathbf{x}_{a_l} - \mathbf{x}_t\|_2 + \eta_{r,t}, \tag{26}$$

where $\|\cdot\|_2$ is the ℓ_2-norm, $\mathbf{x}_t \in \mathbb{R}^3$ is the position of the robot at time t, $\mathbf{x}_{a_l} \in \mathbb{R}^3$ is the position of the a_l^{th} anchor, and $\eta_{r,t} \sim \mathcal{N}(0,\sigma_r^2)$ is the additive white Gaussian noise (AWGN) associated with range measurements. Since the range measurement (26) does not involve the orientation of the robot, we choose to represent the robot trajectory by the vector of robot positions. The state is a sequence of robot positions over time: $\mathbf{x} = [\mathbf{x}_0^T\mathbf{x}_1^T\mathbf{x}_2^T \ldots \mathbf{x}_M^T]^T$.

A single distance measurement cannot constrain the full 3D position at any given time. We assume that a source of odometry, which provides relative pose data $\mathbf{o}_{t-1} = \{\delta\mathbf{x}_{t-1}, \delta\mathbf{R}_{t-1}\}$, is available:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \delta\mathbf{R}_{t-1}\delta\mathbf{x}_{t-1} + \boldsymbol{\eta}_{o,t}, \tag{27}$$

where $\boldsymbol{\eta}_{o,t} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_o)$ is the AWGN, $\delta\mathbf{x}_{t-1}$ is relative position increment, and $\delta\mathbf{R}_{t-1}$ is the direction cosine matrix (DCM) representing the relative orientation increment from time $t-1$ to t as measured by the odometry sensor. The trajectory calculated using relative pose increments provides a suitable initialization for ESGVI-C. The set of observations is given by $\mathbf{y} = \{\mathbf{o}_0, \mathbf{o}_1, ..., \mathbf{o}_t, ..., r_{l,0}, r_{l,1}, ..., r_{l,t}, ...\}$. The factor graph for such a setup is shown in Figure 2. We see from the graph that the conditional independence of the different state variables gives rise to the following factorization of $\phi(\mathbf{x}, \mathbf{y})$:

$$\phi(\mathbf{x}, \mathbf{y}) = \sum_{t=0}^{M} \psi_{r,t} + \sum_{t=0}^{M} \psi_{o,t}, \tag{28}$$

with

$$\psi_{r,t} = \frac{1}{2\sigma_r^2}(r_{l,t} - ||\mathbf{x}_{a,l} - \mathbf{x}_t||_2)^2,$$

$$\psi_{o,t} = \frac{1}{2}\left(\delta\mathbf{x}_{t-1} - \delta\mathbf{R}_{t-1}^T(\mathbf{x}_t - \mathbf{x}_{t-1})\right)^T \boldsymbol{\Sigma}_\mathbf{o}{}^{-1} \times \left(\delta\mathbf{x}_{t-1} - \delta\mathbf{R}_{t-1}^T(\mathbf{x}_t - \mathbf{x}_{t-1})\right),$$

where we have omitted terms constant w.r.t. $\mathbf{x}$.

B. Setup

We evaluated the proposed method in both simulation and real-world experiments.

Simulation setup: Our simulation environment consists of the Gazebo simulator [20]. We use the Astec Firefly quadrotor from the Rotors Simulator [21] package as our robot. The quadrotor is equipped with a generic odometry sensor that provides relative pose measurements. The quadrotor is also equipped with an ultrawideband (UWB) radio, which provides range measurements by calculating the distance to UWB anchors. The sensor measurements are corrupted with Gaussian noise whose parameters are chosen to reflect the performance of real-world sensors.

Real-world setup: The real-world setup consists of a constellation of 6 UWB anchors, a motion-capture capture system, and a quadrotor. The quadrotor is equipped with a DW1000-based UWB radio and an Intel Realsense T265, which provides visual inertial odometry (VIO) measurements. The UWB radio is operated in two-way range (TWR) mode and calculates the distance to anchors by measuring the time-of-flight. The location of the anchors was measured using a Leica total station.

ESGVI-C hyperparameters: The hyperparameter, γ, was tuned using ground truth on a single trajectory with NEES as the metric. The same value of $\gamma = 0.72$ was used for all other experiments. A constant step size $\beta = 1$ was used in all the experiments. The value of ϵ was set to 1000 and reduced by a factor of 10 on each iteration. A more sophisticated backtracking method could be used to set the hyperparameters, β and ϵ. In all of the experiments, Gauss-Hermite integration of degree $D = 3$ was used.

C. Evaluation

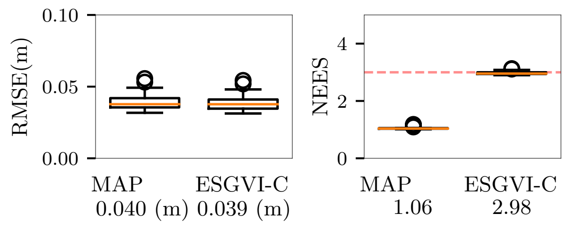
We evaluated the proposed method against maximum a posteriori (MAP) estimation and ESGVI, which represent the state-of-the-art in Gaussian state estimation.

1) Comparison against MAP: We use the Gauss-Newton optimizer for MAP estimation. For a detailed description of the MAP method, we refer the reader to [7]. The performance of the two approaches is compared using trajectory root-mean-square error (RMSE) and NEES metrics.

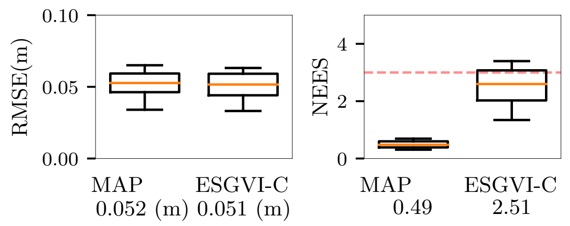
Simulation experiments: We performed several experiments where the quadrotor was commanded to execute multiple trajectories. In each case, the sensor data was used only for estimation and the ground truth data was used for control. The sensor data was recorded for offboard evaluation to compare different algorithms. In each case, the baseline MAP algorithm and the proposed ESGVI-C algorithm were run with the same parameters. The performance of the two methods from 20 experiments is shown in Figure 3a. The proposed approach performs similar to MAP estimation in terms of position RMSE. However, in terms of consistency of the reported covariance, the proposed approach outperforms MAP. Specifically, we see that the NEES value is close to the dimension of the robot position, which is the target for a consistent estimator. The improved consistency of ESGVI-C comes from the hyperparameter, $\gamma = 0.72$, which balances between being over-confident and fitting the observed data.

Real-world Experiments: We performed multiple experiments where the quadrotor was flown manually in different trajectories. All the sensor data, including ground truth data from the motion capture system, was recorded on the onboard computer. The values of the noise parameters, σ_r and $\boldsymbol{\Sigma}_o$, were obtained from datasheets. UWB range measurements are susceptible to reflections from objects in the environment. These reflections can result in large erroneous range measurements. We filtered outlier range measurements that exceeded the expected range measurement by more than 1 m. Results from 10 experiments are shown in Figure 3b. Similar to the simulation experiments, the proposed approach performs marginally better than MAP estimation in terms of position RMSE but reports better NEES values. As alluded to earlier, UWB range measurements are susceptible to reflections from objects in the environment. This gives rise to measurements that violate the AWGN assumption for $\eta_{r,t}$ and hence we see a lower average NEES value compared to simulation. The improved consistency of ESGVI-C comes from the hyperparameter, $\gamma = 0.72$, which balances between being over-confident and fitting the observed data. In the best case, ESGVI-C achieves a position RMSE of 3 cm.

2) Comparison against ESGVI: We observed that when the initial estimate for the robot trajectory was close to the true value, the performance of ESGVI was similar to our method. This is expected since near the local minimum, the update equations of ESGVI are similar to the proposed method (see Section V-B). However, away from the true value the Hessian is not guaranteed to be positive definite. To evaluate the robustness of our approach, we provide a qualitative Monte Carlo estimate of the basin of convergence.



(a) Simulation experiments



(b) Real-world experiments

Fig. 3: Performance of MAP estimation and the proposed ESGVI-C algorithm in (a) simulation and (b) real-world experiments. The proposed approach has similar performance as MAP in terms of position root-mean-square error (RMSE). In terms of the normalized estimation error squared (NEES), the proposed approach is better compared to MAP. The expected value of NEES is indicated by the red dashed line. The mean value for each algorithm is provided below the labels.

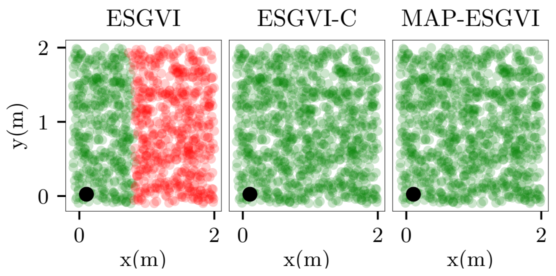


Fig. 4: Qualitative Monte Carlo estimate of basin of convergence for ESGVI, ESGVI-C and MAP-ESGVI. The axes represent the range of values from which x and y-coordinates of the initial state $\mathbf{x}_0$ are sampled. Each sample point is used generate an initial estimate for the trajectory using odometry data. Sample points for which the corresponding algorithms converge and diverge are represented by green circles and red circles, respectively. The black circle represents the ground truth (GT) value of $\mathbf{x}_0$. The figure clearly shows that ESGVI-C has a larger basin of convergence compared to ESGVI and comparable to that of MAP-ESGVI.

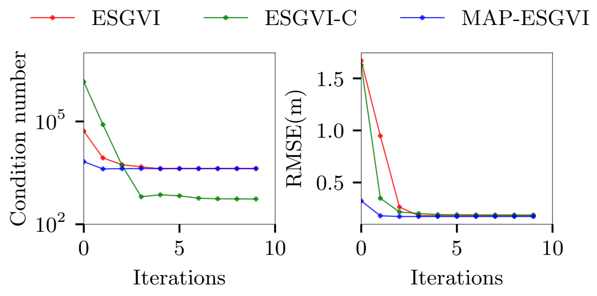


Fig. 5: Evolution of the condition number of the information matrix and the root mean square error (RMSE) for the proposed ESGVI-C method. In each experiment a different initial condition for which ESGVI fails to converge, is used. The condition number and RMSE of a successful run of ESGVI and MAP-ESGVI are plotted for comparison. The proposed ESGVI-C is able to accommodate poorer initial conditions and achieve better condition numbers with accuracy comparable to ESGVI.

We generated random initial conditions by uniformly sampling x and y-coordinates of the initial state, $\mathbf{x}_0$, around its true value. Since the initial trajectory is calculated using relative pose increments from odometry data, translating the initial state translates the entire initial trajectory, thus providing different initial estimates for the robot trajectory.

We evaluated the convergence of ESGVI and ESGVI-C on 1000 random initial trajectories. Qualitative Monte Carlo estimates of the basins of convergence for ESGVI and ESGVI-C are shown in Figure 4. We see that the proposed ESGVI-C method converges even with poor initial conditions, indicating a larger basin of convergence.

Lemma 1 provides formal guarantees that the information matrix remains positive definite. However, another critical aspect is that the information matrix must be well-conditioned i.e., the ratio of the largest to smallest eigenvalue of the information matrix must be not too large. The condition number of ESGVI-C from a real-world experiment with random initial condition, for which ESGVI fails to converge, is shown in Figure 5. We also calculated the condition number of ESGVI with an initial condition where it converges (red line in Figure 5). We see that ESGVI-C achieves better condition numbers and similar accuracy with poor initial condition.

3) Comparison against MAP-ESGVI: As demonstrated in the previous section, ESGVI has a smaller basin of convergence compared to ESGVI-C. MAP estimation, on the other hand, has a larger basin of convergence since it uses an approximation to the Hessian and hence can be used as a source to initialize ESGVI, which we refer to as MAP-ESGVI. Such an approach was followed in [4]. We calculated the basin of convergence for MAP-ESGVI using the same random initial trajectories generated previously. In each case, we ran MAP to convergence and then used the output of MAP to initialize ESGVI. A qualitative estimate of the basin of convergence for MAP-ESGVI is shown in Figure 4. We see that the MAP-ESGVI and ESGVI-C have identical basins of convergence. This is because MAP provides a good initial estimate for which the Hessian is generally convex.

We calculated the condition number and RMSE for MAP-ESGVI with the same initial condition used for ESGVI-C in the previous section. The results are shown in Figure 5. We see that ESGVI-C achieves reasonable condition numbers than MAP-ESGVI. However, MAP-ESGVI achieves similar accuracy as both ESGVI and ESGVI-C. This is because MAP provides a better initial condition for ESGVI in MAP-ESGVI (blue line in Figure 5) compared to the initial condition used for ESGVI-only case (red line in Figure 5).

We calculated the total computation time of ESGVI-C and MAP-ESGVI. The results are summarized in Table I. The higher computation time of ESGVI-C is largely due to the calculation of expectations of gradients and Hessians. Using MAP to initialize ESGVI-C results in similar computation

TABLE I: Total wall-clock time of different algorithms on real-world data.

Algorithm	Wall-clock time(s)
MAP-ESGVI	452
ESGVI-C	800

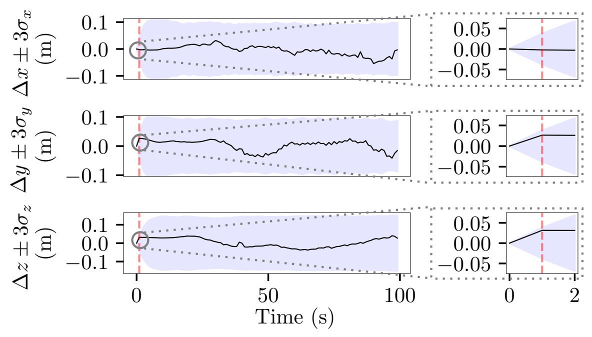


Fig. 6: Error plots for ESGVI-C from a real-world experiment with odometry dropout. The residual position errors, $\mathbf{x}_{\text{true}} - \boldsymbol{\mu}$, are shown along with the 3σ covariance bounds. Due to odometry dropout the Hessian is ill-conditioned at the start of the trajectory. The time at which odometry dropout ends is indicated with red vertical lines. The proposed method is still able to provide reliable estimates in such cases. Note that the errors are bounded within the estimated uncertainty. Magnified error plots corresponding to odometry dropout period are shown on the right.

time as MAP-ESGVI. Although MAP-ESGVI has a lower computation time, unlike the proposed method, there are no formal guarantees about covariance constraint satisfaction.

An advantage of ESGVI-C is demonstrated in cases where the Hessian is ill-conditioned. In our experiments, we observed that the VIO output would dropout sporadically at startup. In this case, consecutive range measurements are received without any odometry data to constrain the relevant states resulting in ill-conditioned Hessian. In this case, both ESGVI and MAP-ESGVI fail as we do not include regularization to better condition the problem. However, the proposed method is able to handle such scenarios. The results from a real-world experiment with odometry dropout at startup are shown in Figure 6. We see that ESGVI-C provides reasonable estimates in such cases as well. Additionally, the errors are bounded by the 3σ covariance envelopes, indicating consistent estimation.

VII. Conclusion and Future work

In this paper, we presented an algorithm for state estimation using Gaussian variational inference. We proved theoretically that the covariance constraints of the multivariate Gaussian probability density function are satisfied during optimization. Through simulation and real-world experiments, we demonstrated that our method produces consistent uncertainty estimates while achieving accuracy comparable to the state-of-the-art. We would still like to expand our experimental comparisons to see how ESGVI-C holds up against other methods of regularizing such estimation problems including the use of motion priors (e.g., constant-velocity [22])

Most sensors are susceptible to environmental factors that result in outlier measurements. We are looking at the inclusion of robust cost functions for reliable state estimation in adverse conditions. In this work, we assumed that the sensor noise parameters obtained from the datasheet were reliable. However, the sensor noise parameters can be influenced by the surrounding environment. A future direction aims at learning the sensor noise parameters online along the lines of parameter learning described by [4] and [23]. We are also looking at inclusion of neural network parameter learning for improved measurement models as demonstrated in [24].

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[2] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[3] Radford M Neal and Geoffrey E Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[4] Timothy D Barfoot, James R Forbes, and David J Yoon. Exactly sparse Gaussian variational inference with application to derivative-free batch nonlinear state estimation. *The International Journal of Robotics Research (IJRR)*, 39(13):1473–1502, 2020.

[5] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[6] Särkkä Simo. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[7] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, Toronto, 2021.

[8] Hongwei Wang, Hongbin Li, Jun Fang, and Heping Wang. Robust Gaussian Kalman filter with outlier detection. *IEEE Signal Processing Letters*, 25(8):1236–1240, 2018.

[9] Giorgio Parisi and Macht Jonathan. *Statistical Field Theory*. Springer-Verlag, 1989.

[10] M.J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London., 2003.

[11] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(12):1–305, 2008.

[12] Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, and Jürgen Schmidhuber. Exponential natural evolution strategies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 393–400, 2010.

[13] Wu Lin, Frank Nielsen, Khan Mohammad Emtiyaz, and Mark Schmidt. Tractable structured natural-gradient descent using local parameterizations. In *International Conference on Machine Learning*, pages 6680–6691. PMLR, 2021.

[14] Linda S. L. Tan. Explicit natural gradient updates for Cholesky factor in Gaussian variational approximation, 2022.

[15] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[16] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[17] Timothy D Barfoot. Multivariate Gaussian variational inference by natural gradient descent. *arXiv preprint arXiv:2001.10025*, 2020.

[18] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2 edition, 2012.

[19] Manfred Opper and Cédric Archambeau. The variational Gaussian approximation revisited. *Neural computation*, 21(3):786–792, 2009.

[20] N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004.

[21] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016.

[22] Tim D Barfoot, Chi Hay Tong, and Simo Särkkä. Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In *Robotics: Science and Systems*, volume 10, pages 1–10. Citeseer, 2014.

[23] Jeremy Nathan Wong, David Juny Yoon, Angela P. Schoellig, and Timothy D. Barfoot. Variational inference with parameter learning applied to vehicle trajectory estimation. *IEEE Robotics and Automation Letters*, 5(4):5291–5298, 2020.

[24] David J. Yoon, Haowei Zhang, Mona Gridseth, Hugues Thomas, and Timothy D. Barfoot. Unsupervised learning of lidar features for use ina probabilistic trajectory estimator. *IEEE Robotics and Automation Letters*, 6(2):2130–2138, 2021.