Efficient Updates for Data Association with Mixtures of Gaussian Processes

Ki Myung Brian Lee[1], Wolfram Martens[2], Jayant Khatkar[1], Robert Fitch[1] and Ramgopal Mettu[3]

***Abstract*— Gaussian processes (GPs) enable a probabilistic approach to important estimation and classification tasks that arise in robotics applications. Meanwhile, most GP-based methods are often prohibitively slow, thereby posing a substantial barrier to practical applications. Existing "sparse" methods to speed up GPs seek to either make the model more sparse, or find ways to more efficiently manage a large covariance matrix. In this paper, we present an orthogonal approach that memoises (i.e. reuses) previous computations in GP inference. We demonstrate that a substantial speedup can be achieved by incorporating memoisation into applications in which GPs must be updated frequently. Moreover, we derive a novel online update scheme for sparse GPs that can be used in conjunction with our memoisation approach for a synergistic improvement in performance. Across three robotic vision applications, we demonstrate between 40-100% speed-up over the standard method for inference in GP mixtures.**

I. INTRODUCTION

A fundamental problem in robotics is to robustly associate noisy sensor measurements to the relevant environmental phenomena. Instances of such data association problems include object segmentation, where pixel or pointcloud measurements are segmented into different objects [1–3], and multi-target tracking, where measurements need to be attributed to a (possibly unknown) number of targets [4]. More generally, most active perception algorithms makes use of the data association step as a fundamental primitive. A promising approach is to model the measurements as a mixture of Gaussian processes; approaches in this direction include Gaussian process (GP) regression and Dirichlet process (DP) mixture models. GP [5] is already widely used in robotics applications because they enable powerful non-parametric Bayesian inference on both spatial and temporal data. Similarly, DP mixture models can describe problems where observed data needs to be associated to distinct latent components. Markov-Chain Monte Carlo (MCMC) methods such as Gibbs sampling are used to work with DP mixture models. DP mixture models over GP components have been used in robotics applications such as object segmentation [6], gas distribution mapping [7], and spatiotemporal topic modelling for active exploration [1].

This work is supported in part by an Australian Government Research Training Program (RTP) Scholarships, the University of Technology Sydney, and Tulane University.

[1]University of Technology Sydney, Ultimo, NSW 2006, Australia {brian.lee, jayant.khatkar}@student.uts.edu.au, rfitch@uts.edu.au

[2]Software and Prototypes for Automation (SPA), Siemens, Berlin, Germany wolfram.martens@siemens.com

[3]Department of Computer Science, Tulane University, New Orleans, LA 70118, USA rmettu@tulane.edu

A major challenge in the context of robotics is that MCMC methods for DP mixture models typically require frequent re-association and likelihood inference that are computationally expensive. When GPs are used as mixture components, these operations then require frequent inversion of a large covariance matrix; the cost of this operation scales quadratically with the number of data points. Thus, these approaches scale poorly to real-world applications, and there is a pressing need to improve computational efficiency in order to support online robotics applications such as active perception.

In this paper, we address this inefficiency through a novel *memoisation* framework that reuses results from previous computations for GP inference. Our core insight is that we can exploit the linear algebraic operations for online updates to not only share the results between different types of operations, but we can also reuse the results from prior inference computations. This approach is orthogonal to existing methods to speed up GPs, which generally seek to improve computational efficiency by considering a low-rank approximation of the covariance matrix.

We show that our memoisation framework can be used in both exact and approximate settings, and demonstrate substantial benefits for three applications. In the exact setting, memoisation provides a speedup ranging between 40-100%. We also develop a novel online update scheme for two existing sparse approximation methods to allow using our memoisation approach in conjunction. In the sparse setting, incorporating memoisation incurs a small overhead performance for moderate data sizes, but clear gains are evident for large data sizes.

II. RELATED WORK

Infinite GP mixture models are a powerful tool for solving general data association problems in a non-parametric setting. The standard approach is to assume a DP model [8] over GPs, and perform Gibbs sampling [9–11]. In the robotics community, [6] used Gibbs sampling to cluster pointcloud measurements into apple fruit detection. Such a probabilistic formulation allowed principled multi-robot active perception in [1]. Despite potential usefulness, GP inference on a large dataset can be prohibitively expensive for even a small number of evaluations, and scales quadratically with size of dataset. To address this bottleneck, low-rank approximation methods, or sparse methods have been extensively studied. The subset of regressors (SoR) [12], and deterministic training conditional (DTC) methods [13] are standard approaches based on a probabilistic formulation.

Our memoisation approach is orthogonal to these methods because it does not entail additional approximation. In fact, it is possible to use both in conjunction, provided there is a suitable online update scheme for the sparse approximation method used. A challenge is that most sparse approximation methods only consider batch inference. To this end, we derive an online update scheme for SoR and DTC methods, and show how our approach can be used in conjunction.

III. PRELIMINARIES

A. Gaussian Process Regression

We are interested in zero-mean scalar Gaussian processes (GPs) over $\mathbf{x} \in \mathbb{R}^d$,

$$f \sim \mathcal{GP}\left(0, k_f(\mathbf{x}, \mathbf{x}')\right), \tag{1}$$

with kernel function $k_f(\mathbf{x}, \mathbf{x}') = \text{cov}(f(\mathbf{x}), f(\mathbf{x}'))$.

Let $\mathcal{D} = \{d_1, \cdots, d_N\}$ be a set of data points with $d_i = \{\mathbf{x}_i, y_i\}$, where $y_i = f(\mathbf{x}_i) + \epsilon_i$ is an observation of the value of f at $\mathbf{x}_i$ with measurement noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Given $\mathcal{D}$, the conditional likelihood of measurement y^* at a query location $\mathbf{x}^*$ is given by [5]:

$$\log \mathcal{P}(y^* \mid \mathbf{x}^*, \mathcal{D}) = -\frac{1}{2}\Big(\frac{(y^* - \mu^*)^2}{\sigma^{*2}} + \log 2\pi\sigma^{*2}\Big), \tag{2}$$

with mean and variance:

$$\begin{aligned} \mu^* &= \mathbf{k}^{*T}(\mathbf{K} + \sigma^2\mathbf{I}_\text{N})^{-1}\mathbf{y}_\mathcal{D}, \\ \sigma^{*2} &= k^{**} - \mathbf{k}^{*T}(\mathbf{K} + \sigma^2\mathbf{I}_\text{N})^{-1}\mathbf{k}^*. \end{aligned} \tag{3}$$

Here, $\mathbf{y}_\mathcal{D} \in \mathbb{R}^{N\times 1}$ denotes the concatenation of the observations for all data points. The covariance terms, $\mathbf{K} \in \mathbb{R}^{N\times N}$, $\mathbf{k}^* \in \mathbb{R}^{N\times 1}$, and k^{**} are constructed as $\mathbf{K}_{mn} = k(\mathbf{x}_m,\ \mathbf{x}_n)$, $\mathbf{k}^*_n = k(x_n, \mathbf{x}^*)$, and $k^{**} = k(\mathbf{x}^*, \mathbf{x}^*)$ respectively.

B. Sparse Gaussian Process

We consider two variants of sparse approximations, namely Subset of Regressors (SoR) and Deterministic Training Conditional (DTC). In these approximations, we pick a subset from the measurements $\mathcal{D}$ to serve as the active points that induce the rest of the measurements. Throughout the paper, we assume that the data points are ordered such that the active points come first. Given M active points, the SoR approximation is to assume:

$$\hat{k}(\mathbf{x}, \mathbf{x}') \approx \mathbf{k}_\text{M}(\mathbf{x})^T\mathbf{K}_\text{MM}^{-1}\mathbf{k}_\text{M}(\mathbf{x}'), \tag{4}$$

where $\mathbf{K}_\text{MM}$ is the covariance matrix of the active points, and $(\mathbf{k}_\text{M}(\mathbf{x}))_i = k(\mathbf{x}, \mathbf{x}_i)$ for active points $\mathbf{x}_i$.

With (4), the inference equations become [5]:

$$\begin{aligned} \mu^*_\text{SR} &= \mathbf{k}^{*T}_M(\mathbf{K}^T_M\mathbf{K}_M + \sigma^2\mathbf{K}_\text{MM})^{-1}\mathbf{K}^T_M\mathbf{y}_\mathcal{D}, \\ \sigma^{*2}_\text{SR} &= \sigma^2\mathbf{k}^{*T}_M(\mathbf{K}^T_M\mathbf{K}_M + \sigma^2\mathbf{K}_\text{MM})^{-1}\mathbf{k}^*_M, \end{aligned} \tag{5}$$

where $\mathbf{k}^*_\text{M} = \mathbf{k}_\text{M}(\mathbf{x}^*)$, and $\mathbf{K}_\text{M}$ is the first M columns of the full covariance matrix.

DTC approximation has the same mean as SoR, but a 'correction term' is applied to the variance term:

$$\sigma^{*2}_\text{DTC} = k^{**} - \mathbf{k}^{*T}_M\mathbf{K}^{-1}_\text{MM}\mathbf{k} + \sigma^{*2}_{SoR}. \tag{6}$$

Algorithm 1 GP MIXTURE INFERENCE

1: **for** fixed number of iterations **do**
2: Sample d_i from $\mathcal{D}$
3: Remove d_i from current expert
4: **for** each expert e_j **do**
5: Compute GP likelihood $\mathcal{P}(d_i \mid \mathcal{D}_j)$
6: Compute expert prior $\mathcal{P}(e_j \mid \mathcal{E} \setminus e_j)$
7: Compute posterior using lines 5 and 6
8: Choose e^* based on the posterior
9: Add d_i to $\mathcal{D}^*$

It is important to note that both the DTC and SoR approximations require the inversion of an $M \times M$ matrix, instead of the usual $N \times N$ covariance matrix. Since $M \ll N$, this leads to a substantial speed-up.

C. Mixture of GP Experts

The aim of the data association problem is to correctly associate each data point $d_i \in \mathcal{D}$ to disjoint subsets $\mathcal{D}_j \subset \mathcal{D}$, where each subset $\mathcal{D}_j$ follows a GP 'expert' $e_j = \mathcal{GP}(0, k_j(\mathbf{x}, \mathbf{x}')) \in \mathcal{E}$. For simplicity, we will assume that the kernel function $k_j(\mathbf{x}, \mathbf{x}')$ for each expert e_j are drawn from a known set of kernel functions with fixed hyperparameters.

An important class of algorithms for GP mixture inference is sampling-based algorithms such as Gibbs sampling [9], or GP-INSAC [14]. These sampling-based algorithms typically follow the pattern shown in Algorithm 1. Each iteration begins with a random selection of a measurement pair to process (Line 2). The selected pair is first dissociated from its current expert (Line 3). Then, for each expert, we compute the measurement likelihood of the selected pair through GP inference (2), and a prior on the expert itself. Given the likelihood and prior, we compute the posterior, and select the new expert for the current measurement pair.

For example, in infinite mixture of GP experts [9], the expert prior is a DP, and the expert is selected by sampling from posterior. In GP-INSAC [14], the expert prior is a Bernoulli distribution over inlier and outlier events, and the expert with maximal posterior is chosen.

IV. INCREMENTAL UPDATES

Addition and removal of data points to a GP expert are necessary for GP mixture updates. In this section, we present how to efficiently update the inference results when data points are being added or removed. We review the case of exact inference, and derive a similar method for sparse approximate inference.

A. Inference

1) Exact Inference: A standard method for efficient incremental updates is to use the Cholesky factor of the covariance matrix. The Cholesky factor $\mathbf{L} = \text{chol}(\mathbf{K} + \sigma^2\mathbf{I}_{N\times N})$ is a lower triangular matrix such that $\mathbf{L}\mathbf{L}^T = \mathbf{K} + \sigma^2\mathbf{I}_{N\times N}$. Using $\mathbf{L}$, we re-write the inference equation (3) in a form that will become useful later. Define $\mathbf{p} = \mathbf{L}^{-1}\mathbf{k}^*, \mathbf{q} = \mathbf{L}^{-1}\mathbf{y}_\mathcal{D} \in \mathbb{R}^N$. Then, the inference equation can be re-written as:

$$\begin{aligned} \mu^* &= \mathbf{p}^T\mathbf{q}, \\ \sigma^{*2} &= k^{**} - \mathbf{p}^T\mathbf{p}. \end{aligned} \tag{7}$$

Therefore, inference is of order $\mathcal{O}(N)$ given $\mathbf{p}$ and $\mathbf{q}$.

2) Sparse Approximate Inference: Despite the compelling need for incremental sparse approximate GP inference in robotics applications, there is currently no literature on doing so to the best of our knowledge. Here, we give a formulation for updates in sparse approximations that is similar to the exact case. Assume, for the moment, that the set of active points are given, and comprise the first M data points. Divide the covariance matrix as:

$$\hat{\mathbf{K}} = \begin{bmatrix} \mathbf{K}_{\text{MM}} & \mathbf{K}_{\text{M(N-M)}} \\ \mathbf{K}_{\text{(N-M)M}} & \mathbf{K}_{\text{(N-M)M}}\mathbf{K}_{\text{MM}}^{-1}\mathbf{K}_{\text{M(N-M)}} \end{bmatrix}. \quad (8)$$

An analogue of the Cholesky decomposition in the sparse case is the incomplete Cholesky decomposition. The incomplete Cholesky decomposition of $\hat{\mathbf{K}}$ is computed as [15]:

$$\hat{\mathbf{L}} = \begin{bmatrix} \hat{\mathbf{L}}_{\text{MM}} \\ \hat{\mathbf{L}}_{\text{(N-M)M}} \end{bmatrix} = \begin{bmatrix} \text{chol}(\mathbf{K}_{\text{MM}}) \\ \mathbf{K}_{\text{(N-M)M}}\hat{\mathbf{L}}_{\text{MM}}^{-T} \end{bmatrix}. \quad (9)$$

In terms of the incomplete Cholesky factor, the inference equations (5) can be written as:

$$\begin{aligned} \mu_{\text{SR}}^* = \mu_{\text{DTC}}^* &= \mathbf{k}^{*T}\hat{\mathbf{L}}_{\text{MM}}^{-T}(\hat{\mathbf{L}}^T\hat{\mathbf{L}} + \sigma^2\mathbf{I}_{\text{M}})^{-1}\hat{\mathbf{L}}_{\text{MM}}^{-1}\hat{\mathbf{L}}^T\mathbf{y}_{\mathcal{D}}, \\ \sigma_{\text{SR}}^{*2} &= \sigma^2\mathbf{k}^{*T}\hat{\mathbf{L}}_{\text{MM}}^{-T}(\hat{\mathbf{L}}^T\hat{\mathbf{L}} + \sigma^2\mathbf{I}_{\text{M}})^{-1}\hat{\mathbf{L}}_{\text{MM}}^{-1}\mathbf{k}^*, \quad (10) \\ \sigma_{\text{DTC}}^{*2} &= k^{**} - \mathbf{k}^{*T}\hat{\mathbf{L}}_{\text{MM}}^{-T}\hat{\mathbf{L}}_{\text{MM}}^{-1}\mathbf{k}^* + \sigma_{\text{SR}}^{*2}, \end{aligned}$$

which is also referred to as the V-method in [16]. It is still difficult to update (10) incrementally as we did for exact inference, because of the inversion of $\mathbf{X} = (\hat{\mathbf{L}}^T\hat{\mathbf{L}} + \sigma^2\mathbf{I}_{\text{M}})$, and the multiplication of $\mathbf{y}_{\mathcal{D}}$ by $\hat{\mathbf{L}}_{\text{MM}}^{-1}\hat{\mathbf{L}}^T$.

In this work, we use the QR decomposition to compute the Cholesky factor of $\mathbf{X}$, which now allows for incremental updates. QR decomposition of a matrix $\mathbf{A} \in \mathbb{R}^{\text{P}\times\text{Q}}$ with $\text{P} \geq \text{Q}$ is given by $\mathbf{A} = \mathbf{QR}$ where $\mathbf{Q} \in \mathbb{R}^{\text{P}\times\text{P}}$ is a unitary matrix (i.e. $\mathbf{QQ}^T = \mathbf{I}_p$) and $\mathbf{R} \in \mathbb{R}^{\text{P}\times\text{Q}}$ is upper triangular.

Define $\tilde{\mathbf{L}} = \begin{bmatrix}\hat{\mathbf{L}}^T & \sigma\mathbf{I}_{\text{M}}\end{bmatrix}^T$, and consider $\tilde{\mathbf{L}} = \mathbf{QR}$. Then, we have $\mathbf{R}^T = \text{chol}(\mathbf{X})$, because $\mathbf{X} = \mathbf{R}^T\mathbf{R}$ and $\mathbf{R}$ is upper triangular. It can be shown that the $\hat{\mathbf{L}}_{\text{MM}}^{-1}\hat{\mathbf{L}}^T$ term also cancels, and the inference equations become:

$$\begin{aligned} \mu_{\text{SR}}^* = \mu_{\text{DTC}}^* &= \mathbf{k}^{*T}\hat{\mathbf{L}}_{\text{MM}}^{-T}\mathbf{R}^{-1}\mathbf{Q}^T\mathbf{y} \\ \sigma_{\text{SR}}^{*2} &= \sigma^2\mathbf{k}^{*T}\hat{\mathbf{L}}_{\text{MM}}^{-T}\mathbf{R}^{-1}\mathbf{R}^{-T}\hat{\mathbf{L}}_{\text{MM}}^{-1}\mathbf{k}^* \quad (11) \\ \sigma_{\text{DTC}}^{*2} &= k^{**} - \mathbf{k}^{*T}\hat{\mathbf{L}}_{\text{MM}}^{-T}\hat{\mathbf{L}}_{\text{MM}}^{-1}\mathbf{k}^* + \sigma_{\text{SR}}^{*2}. \end{aligned}$$

Similar to exact inference, define $\hat{\mathbf{p}} = \mathbf{L}_{\text{MM}}^{-1}\mathbf{k}_{\text{M}}^*$, $\hat{\mathbf{q}} = \mathbf{Q}^T\mathbf{y}_{\mathcal{D}}$, and $\hat{\mathbf{r}} = \mathbf{R}^{-1}\mathbf{p}$. Then, the inference equations become:

$$\begin{aligned} \mu_{SoR}^* = \mu_{DTC}^* &= \hat{\mathbf{r}}^T\hat{\mathbf{q}} \\ \sigma_{SoR}^{*2} &= \sigma^2\hat{\mathbf{r}}^T\hat{\mathbf{r}} \quad (12) \\ \sigma_{DTC}^{*2} &= k^{**} - \hat{\mathbf{p}}^T\hat{\mathbf{p}} + \sigma_{SoR}^{*2}. \end{aligned}$$

Therefore, performing sparse approximate inference is of order $\mathcal{O}(M)$ given $\hat{\mathbf{p}}$, $\hat{\mathbf{q}}$, and $\hat{\mathbf{r}}$.

B. Insertion of a Data Point

1) Exact Case: Consider the insertion of data point $d^+ = \{\mathbf{x}^+, y^+\}$. For simplicity, we only append data points to the end of the covariance matrix:

$$\mathbf{K}^+ = \begin{bmatrix} \mathbf{K} & \mathbf{k}_+ \\ \mathbf{k}_+^T & k_{++} \end{bmatrix}, \quad (13)$$

Then, the Cholesky factor can be updated as [17]:

$$\mathbf{L}^+ = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{l}_+ & l_{++} \end{bmatrix}, \quad (14)$$

where $\mathbf{l}_+ = (\mathbf{L}^{-1}\mathbf{k}_+)^T$, and $l_{++} = \sqrt{k_{++} - \mathbf{l}_+^T\mathbf{l}_+}$.

2) Sparse Case: For sparse inference, we need to update the incomplete Cholesky factor $\hat{\mathbf{L}}$ and the QR decomposition $\mathbf{Q}$ and $\mathbf{R}$. We begin with $\hat{\mathbf{L}}$. Let $\mathbf{k}_+ = [\mathbf{k}_{\text{M}}^T\ \mathbf{k}_{\text{(N-M)}}^T\ k_{++}]^T$ be the covariance vector to be added, and define $\hat{\mathbf{l}}_{\text{M}} = \hat{\mathbf{L}}_{\text{MM}}^{-1}\mathbf{k}_{\text{M}}$. A convenient criterion for deciding whether a new point should be added to the active set is to use:

$$\hat{l}_{++}^2 = k_{++} - \hat{\mathbf{l}}_M^T\hat{\mathbf{l}}_M. \quad (15)$$

The new point is added to the active set if $\hat{l}_{++}^2 > \sigma_T^2$, or the inactive set otherwise. From a probabilistic perspective, the criterion (15) is the predictive variance of the new point given the current active set. A new data point is considered inactive if it is almost certainly predicted by the existing active data points (i.e. low variance). Numerically, (15) ensures that the diagonals of $\hat{\mathbf{L}}$ are sufficiently greater than zero, which is essential for stable computation of future updates. While it is also possible to compute and use other criteria (e.g. [18]) through algebraic manipulations, we defer the use of other criteria to future work for simplicity.

If the new point is decided as inactive, the update is simply a row insertion $\hat{\mathbf{L}}^+ = [\hat{\mathbf{L}}^T\ \hat{\mathbf{l}}_M]^T$. If it is active, the matrix is updated as:

$$\hat{\mathbf{L}}^+ = \begin{bmatrix} \hat{\mathbf{L}} & 0 \\ \hat{\mathbf{l}}_{\text{M}}^T & \hat{l}_{++} \\ \hat{\mathbf{L}}_{\text{(N-M)M}} & \hat{\mathbf{l}}_{\text{(N-M)}} \end{bmatrix}, \quad (16)$$

where $\hat{\mathbf{l}}_{\text{(N-M)}} = \hat{l}_{++}^{-1}(\mathbf{k}_{\text{(N-M)}} - \hat{\mathbf{L}}_{\text{(N-M)M}}\hat{\mathbf{l}}_M)$.

Afterwards, the QR decomposition is updated. Note that, whether active or inactive, the updates on $\hat{\mathbf{L}}$ are row and/or column insertions. Equivalently, we need to compute the QR decomposition after inserting row and/or column to the augmented matrix $\tilde{\mathbf{L}}$. Fortunately, this can be computed efficiently with Givens rotations in $\mathcal{O}(M^2)$ time.

C. Deletion of a Data Point

1) Exact Case: Consider the removal of a data point d_j. Divide $\mathbf{L}$ into block matrices as:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{l}_{j*} & l_{jj} & \mathbf{0} \\ \mathbf{L}_{31} & \mathbf{l}_{*j} & \mathbf{L}_{33} \end{bmatrix}. \quad (17)$$

Let $\mathbf{K}_{(\backslash j,\backslash j)}$ be the covariance matrix after removing the jth row and column from $\mathbf{K}$. Then, we have:

$$\text{chol}(\mathbf{K}_{(\backslash j,\backslash j)}) = \mathbf{L}_{(\backslash j)} = \begin{bmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{L}_{31} & \mathbf{U} \end{bmatrix}, \quad (18)$$

where $\mathbf{U} = \text{cu}(\mathbf{L}_{33}, \mathbf{l}_{*j})$. $\text{cu}(\mathbf{L}, \mathbf{x})$ is the rank-1 update of $\mathbf{L}$ with $\mathbf{x}$, defined by $\text{cu}(\mathbf{L}, \mathbf{x}) = \text{chol}(\mathbf{LL}^T + \mathbf{xx}^T)$. Rank-1 updates can be performed in $\mathcal{O}((N-j)^2)$ time using Givens rotations.

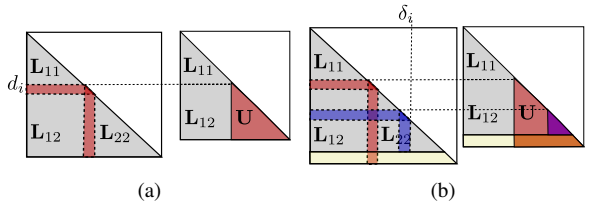


Fig. 1. An illustrative example for rank-1 update memoisation. Suppose d_i had been temporarily removed previously in (a). Between (a) and (b), one point was removed (blue), and another was added (yellow). If we attempt removal of d_i again in (b), the red part of the matrix remains unchanged. Best viewed in colour.

2) Sparse case: Again, in the sparse case, we need to update $\hat{\mathbf{L}}$, $\mathbf{Q}$ and $\mathbf{R}$. There are two possibilities. First, if the data point being removed is inactive, then we simply need to remove the corresponding row from the Cholesky factor. Then, the QR factors are updated incrementally, through a row deletion operation [19]. The row deletion operation is of order $\mathcal{O}(M^2)$.

If the data point is active, then a rank-1 update must be performed on the Cholesky factor. The process is the same as the usual rank-1 update, except that the Givens rotation need not go beyond column M. Therefore, rank-1 update is of complexity $\mathcal{O}((M-j)(N-j))$. Because the contents of the Cholesky factor has been modified, the QR decomposition must be recomputed in full, which is of order $\mathcal{O}(NM^2)$.

V. MEMOISATION OF ONLINE UPDATES

In this section, we present a memoisation approach that reuses previous results of incremental updates described in Sec. IV.

A. Cache Structure

Each GP expert is endowed with a cache to store previous results of online updates that we will re-use later. The content of the cache is as follows for the exact and sparse cases.

1) Exact case: For all data points $d_i \in \mathcal{D}_j$ currently associated to the expert, we store $\mathbf{L}_{(\setminus i)}$, the result of temporary removal for association testing, and $\mathbf{q}_{(\setminus i)} = \mathbf{L}_{(\setminus i)}^{-1} y_{\mathcal{D}\setminus\{d_i\}}$.

For all data points $\mathcal{D}$ (not necessarily in $\mathcal{D}_j$), we store $\mathbf{p} = \mathbf{L}^{-1}\mathbf{k}$. Most importantly, we store the *discrepancy index* δ_i for all data points in $\mathcal{D}$, which we define as the furthest left (i.e. numerically lowest) column index of the data points that have been removed from e_j since the last update. The predominant component of space complexity is the storage of $\mathbf{L}_{(\setminus i)}$, which is of order $\mathcal{O}(N^3)$.

2) Sparse Case: Similarly to the exact case, we store $\hat{\mathbf{L}}_{(\setminus i)}$ for all associated data points, and $\hat{\mathbf{p}} = \hat{\mathbf{L}}^{-1}\mathbf{k}_{\text{M}}$ for any data point. The discrepancy index δ_i is also stored, but we only consider the column indices of the *active* data points removed since the last update. The space complexity is of order $\mathcal{O}(M^2N)$ for storage of $\hat{\mathbf{L}}_{(\setminus i)}$, where M is the size of active set. With correct choice of threshold (15), $M \ll N$.

B. Temporary Removal

Let us first consider a simple example depicted in Fig. 1. We would like to temporarily remove d_i for association testing (Algorithm 1, Line 3). Suppose d_i has been considered for removal previously, so that the previous result of rank-1 update is already available in cache as $\mathbf{L}_{(\setminus i)}$ (the area marked red in Fig. 1a). Since then, one data point, say d_*, has been removed (marked blue in Fig. 1b, and another has been added (marked yellow in Fig. 1b).

Recall from Sec. IV-C that removal of a data point affects only the columns beyond, and insertion affects the rows below. Therefore, changes are limited to the columns beyond d_*, and the rows beyond the size of current cache. The previous result in cache is valid up to the column where removal occurred, and up to the row where the last update was computed. More generally, if multiple points have been removed since the last update, the rank-1 update result will remain the same up to the furthest left (i.e. lowest) column removed, which is precisely the discrepancy index δ_i we stored in the cache.

Therefore, if $\delta_i > d_i$, we can 'hot-start' the rank-1 update from column δ_i onwards. This way, the complexity of the rank-1 update is reduced to $\mathcal{O}((N-\delta_i)^2)$.

In the sparse case, the cache can help reduce complexity of the temporary removal of *active* points (Sec. IV-C). Because rank-1 update is identical in the sparse case, we can achieve a similar reduction in complexity through 'hot-start'. The complexity is then $\mathcal{O}((N-\delta_i)(M-\delta_i))$.

C. Likelihood Inference and Insertion

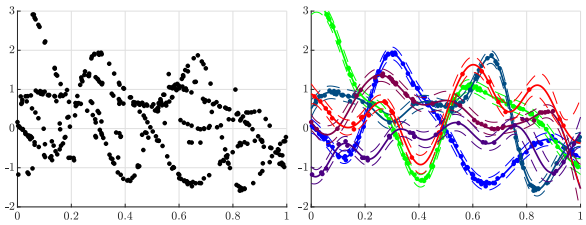
For exact inference, we need to update $\mathbf{p}$ and $\mathbf{q}$ in (3). Similar to the case of rank-1 updates, the changes in the Cholesky factor $\mathbf{L}$ are limited to the columns beyond the discrepancy index δ_i. Further, the changes to the covariance vector $\mathbf{k}$ or $\mathbf{y}_D$ is limited to the rows beyond δ_i. Because $\mathbf{L}$ is triangular, the solutions $\mathbf{p}$ and $\mathbf{q}$ also remain unchanged above δ_i. We can use back-substitution to update only the changed part of $\mathbf{p}$. The complexity of solving for $\mathbf{p}$ is reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N(N-\delta_i))$.

The same argument can be made for the update of $\hat{\mathbf{p}}$ in the sparse case, because $\hat{\mathbf{L}}_{\text{MM}}$ also remains unchanged in columns beyond δ_i. The complexity of solving for $\hat{\mathbf{p}}$ is reduced from $\mathcal{O}(M^2)$ to $\mathcal{O}(M(M-\delta_i))$. A difference is that $\hat{\mathbf{q}}$ and $\hat{\mathbf{r}}$ need to be updated after the QR row/column insertion operations, as all of $\mathbf{Q}$ and $\mathbf{R}$ change. Re-computing $\hat{\mathbf{q}}$ through matrix multiplication is of order $\mathcal{O}(NM)$, and solving for $\hat{\mathbf{r}}$ is of order $\mathcal{O}(M^2)$.

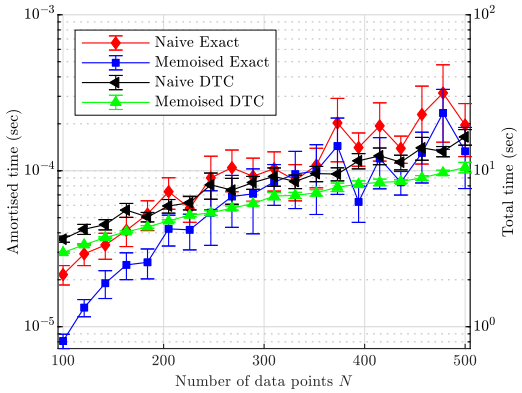
For both exact and sparse inference, insertion requires $\mathbf{L}^{-1}\mathbf{k} = \mathbf{p}$ and $\mathbf{L}_{\text{MM}}^{-1}\mathbf{k}_{\text{M}} = \hat{\mathbf{p}}$ respectively. In the context of mixture updates, inference is always performed immediately before insertion. Therefore, the cache entry for $\mathbf{p}$ and $\hat{\mathbf{p}}$ is always up to date, and we can immediately re-use the result.

VI. EMPIRICAL STUDIES

In this section, we consider three applications of our approach and demonstrate its benefits with experimental evaluations. We consider naive and memoised versions of exact and DTC inference, a total of four combinations. In the 'naive' version, the GP experts are updated incrementally, but do not re-use previous results. In all applications considered, we choose the threshold σ_T^2 used for sparse approximation to



(a) Raw data. (b) Association result.



(c) Computation time

Fig. 2. Application of Gibbs sampling for a synthetic data association problem. In (b), Different colours represent association to a different GP. In (c), amortised time is the average computation time per iteration, and total time is the total for 100,000 iterations.

be equal to the noise variance σ^2. In other words, if a data point is more uncertain than the assumed noise model, it is considered active. These combinations were implemented using the Armadillo library [20].

Overall, memoisation provides a clear benefit in performance over the standard exact approach to performing inference in GPs. We also show that incorporating memoisation into a sparse approximation provides a synergistic benefit when data size is large, but imposes a small overhead when data sizes are more moderate.

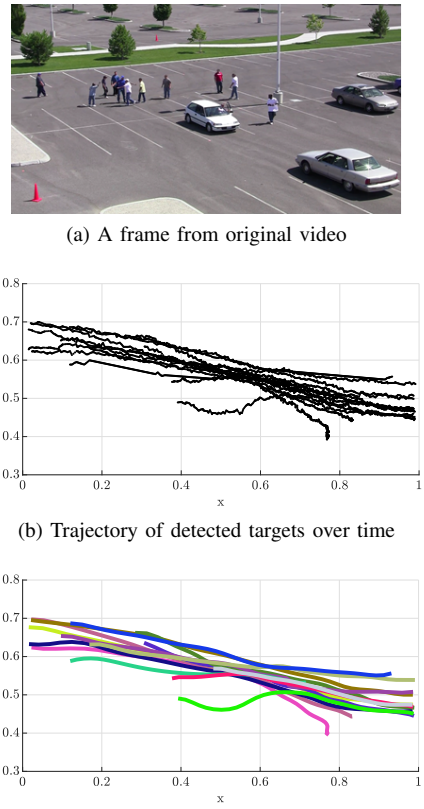
A. DP Mixture of 1D GPs

We first consider a synthetic dataset generated from a Dirichlet process (DP) mixture of GPs as a controlled benchmark for the behaviour of amortised computation time with dataset size. The DP prior is given by:

$$\begin{aligned} \mathcal{P}(e_j \mid \mathcal{E} \setminus \{e_j\}) &\propto \|\mathcal{D}_j\|, \\ \mathcal{P}(e \mid e \notin \mathcal{E}) &\propto \alpha, \end{aligned} \tag{19}$$

where $\alpha \geq 0$ controls the generation of new experts. In other words, a new sample is drawn from a particular GP expert with probability proportional to its current size, and a new GP expert are created with a non-zero probability. As such, it is possible to generate, or associate with *unknown* number of GP experts.

With the generated data, we used Gibbs sampling to solve the data association problem. Gibbs sampling is a special case of our template in Algorithm 1, where the expert prior is set as DP (19), and selection is done by sampling. One instance of the result is shown in Fig. 2, where it can be seen that the Gibbs sampler correctly associates and infers over the raw data.



(a) A frame from original video

(b) Trajectory of detected targets over time

(c) Result after Gibbs sampling

Fig. 3. Data association for multi-target tracking using the Parking Lot dataset [21]. Combining Gibbs sampling with GPs allows solving not only the data association problem, but also inference of missing measurements; in (b) and (c), x and y are image coordinates.

The results in Fig. 2c show that memoisation consistently provides around 50% speed-up in the sparse case, and around 60% in the exact case for larger dataset sizes. In terms of total computation time, the naive exact method requires around 20 seconds while memoisation requires 10 seconds. Interestingly, the computation time for the exact inference shows increasing deviation as the dataset size grows for both memoised and naive case. Detailed profiling revealed that the difference is due to memory allocation and manipulation. This is consistent with the lack of variation for the sparse cases, because the sparse inference requires much less memory. Regardless, memoisation consistently exhibits computational benefits even under deviations. Another interesting pattern is that sparse inference is initially slower than exact inference for smaller dataset sizes, but eventually becomes faster for larger dataset sizes. This is as expected, because for a smaller dataset size maintaining QR decomposition in addition to the Cholesky factor incurs substantial overhead.

B. Multi-Target Tracking

To examine the computational benefits in a practical scenario, we consider the example of data association and tracking for multiple targets, a compelling application of mixture of GPs. We consider the parking lot dataset in [21], which provides detections of pedestrians in a surveillance

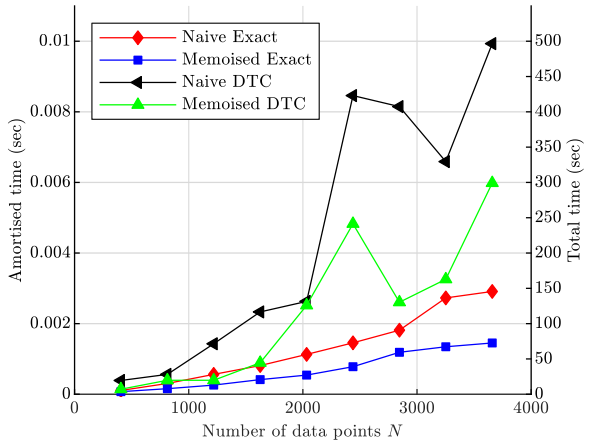


Fig. 4. Computation time of Gibbs sampling with varying dataset size for multi-target tracking. Amortised time is the average time per iteration, and total time is for all 50,000 iterations.

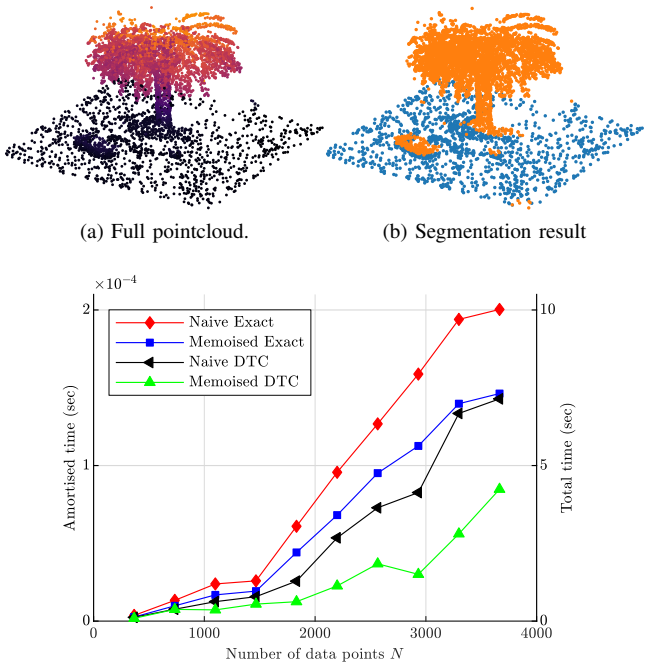
footage over time. An example frame from the video, and the raw detection results provided by the dataset are shown in Figs. 3a and 3b respectively.

We model the position of each pedestrian as a two dimensional GP over time, except with no correlation. This allows modelling each component as separate GPs, which is a straightforward extension from the benchmark example. As before, we pose a DP prior on the GP experts (i.e. individual pedestrians), and use Gibbs sampling to associate detections to the correct targets. The result of Gibbs sampling is shown in Fig. 3c. As can be seen, the Gibbs sampler can produce correct associations between detection and pedestrians, and, at the same time, generate a smooth trajectory through inference.

We examined the behaviour of amortised computation time of the Gibbs sampler with variation in dataset size through downsampling. As shown in Fig. 4, memoisation provides up to 100 % speed up (i.e. half the computation time) for full inference, and around 50% speed up for sparse inference. In terms of total computation time, memoisation offers up to 200 seconds reduction. It is worth noting that sparse inference scales worse for this application than the benchmark case, and is in fact slower than exact inference. This is because the measurements of each pedestrian are already sparse, and numerous active points are produced for sparse inference. In turn, these active points are temporarily removed during the mixture update iterations, incurring a large computational cost due to full re-computation of QR decomposition, which is not covered by memoisation.

C. Pointcloud Segmentation

To examine the behaviour in a large, dense dataset, we consider ground segmentation in pointcloud data. Similar to GP-INSAC [22], we pose the problem as outlier detection by modelling the height of the ground as a scalar GP over two dimensions, and treating the non-ground objects as outliers. Then, outlier detection can be solved as a special case of Algorithm 1 with a simple Bernoulli distribution for expert prior. We used a dataset containing a palm tree shown in Fig. 5a, collected using a Riegl 3D scanner. An instance of



(a) Full pointcloud. (b) Segmentation result

(c) Computation time

Fig. 5. Ground segmentation. In (a), the color represents height. In (b), blue is classified as ground, and orange is classified as outlier (i.e. an object). In (c), amortised time is the average time per iteration, and total time is for 50,000 iterations.

the result is shown in Fig. 5b. It can be seen that the ground is segmented correctly, with few erroneous classifications.

A comparison of amortised computation time is shown in Fig. 5c. It can be seen that memoisation offers speed-up of around 40% for exact inference, and around 70% for sparse inference. In this application, sparse inference outperforms exact inference, because pointcloud data is much denser than the image data in the multi-target tracking application. Further, it is worth noting that computation time of memoised sparse inference is less than half of the naive exact case, with 6 second reduction in computation time. This demonstrates that combining memoisation with existing sparse inference methods can lead to substantial computational benefits, enabling the use of GP mixture models in practical applications that are otherwise infeasible.

VII. CONCLUSION

In this paper, we have presented a memoisation approach to increasing the efficiency of inference using GPs. Our method achieves substantial speedup when combined with sampling methods such as MCMC, where we must repeatedly add and remove points from one or more GPs. The memoisation approach is orthogonal to the conventional sparse approximation methods, and, as we have demonstrated, can be combined with the conventional methods for even greater computational benefit.

In the future, we would like to combine our memoisation framework with more advanced variable reordering methods such as [15]. Further, we would like to scale our approach to bigger problem instances such as 3D multi-object segmentation [1, 6, 23].

REFERENCES

[1] F. Sukkar, G. Best, C. Yoo, and R. Fitch, "Multi-robot region-of-interest reconstruction with dec-mcts," in *International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 9101–9107.

[2] H. van Hoof, O. Kroemer, and J. Peters, "Probabilistic segmentation and targeted exploration of objects in cluttered environments," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1198–1209, 2014.

[3] L. L. Wong, L. P. Kaelbling, and T. Lozano-Pérez, "Data association for semantic world modeling from partial views," *Int. J. Robot. Res.*, vol. 34, no. 7, pp. 1064–1082, 2015.

[4] V. Indelman, E. Nelson, N. Michael, and F. Delaert, "Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization," in *Proc. of IEEE ICRA*, 2014, pp. 593–600.

[5] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[6] P. R. Soria, F. Sukkar, W. Martens, B. C. Arrue, and R. Fitch, "Multi-view probabilistic segmentation of pome fruit with a low-cost rgb-d camera," in *Iberian Robotics conference*. Springer, 2017, pp. 320–331.

[7] C. Stachniss, C. Plagemann, A. J. Lilienthal, and W. Burgard, "Gas distribution modeling using sparse gaussian process mixture models," in *International Conference on Robotics Science and Systems, Robotics: science and systems, 2008, Zürich, Switzerland, June 25-28, 2008*, vol. 4. MIT Press, 2008, pp. 310–317.

[8] R. M. Neal, "Markov chain sampling methods for Dirichlet process mixture models," *J. Comput. Graph. Stat.*, vol. 9, no. 2, pp. 249–265, 2000.

[9] C. E. Rasmussen and Z. Ghahramani, "Infinite mixtures of Gaussian process experts," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 881–888.

[10] J. Shi, R. Murray-Smith, and D. Titterington, "Bayesian regression and classification using mixtures of Gaussian processes," *Int. J. Adapt. Control. Signal. Process.*, vol. 17, no. 2, pp. 149–161, 2003.

[11] E. Meeds and S. Osindero, "An alternative infinite mixture of Gaussian process experts," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds. MIT Press, 2006, pp. 883–890.

[12] G. Wahba, *Spline models for observational data*. Siam, 1990, vol. 59.

[13] M. Seeger, C. Williams, and N. Lawrence, "Fast forward selection to speed up sparse gaussian process regression," in *9th Workshop on Artificial Intelligence and Statistics*, 2003.

[14] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3D LIDAR point clouds," in *Proc. of IEEE ICRA*, 2011, pp. 2798–2805.

[15] F. R. Bach and M. I. Jordan, "Predictive low-rank decomposition for kernel methods," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 33–40.

[16] L. Foster, A. Waagen, N. Aijaz, M. Hurley, A. Luis, J. Rinsky, C. Satyavolu, M. J. Way, P. Gazis, and A. Srivastava, "Stable and efficient gaussian process calculations," *Journal of Machine Learning Research*, vol. 10, no. Apr, pp. 857–882, 2009.

[17] M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings, "Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes," in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*. IEEE, 2008, pp. 109–120.

[18] A. J. Smola and P. L. Bartlett, "Sparse greedy Gaussian process regression," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2001, pp. 619–625.

[19] S. Hammarling and C. Lucas, "Updating the qr factorization and the least squares problem," in *Manchester University MIMS EPrints*, 2008.

[20] C. Sanderson and R. Curtin, "Armadillo: a template-based C++ library for linear algebra," *Journal of Open Source Software*, vol. 1, p. 26, 2016.

[21] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah, "Part-based multiple-person tracking with partial occlusion handling," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 1815–1821.

[22] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3d lidar point clouds," in *Proc. of IEEE ICRA*. IEEE, 2011, pp. 2798–2805.

[23] W. Martens, Y. Poffet, P. R. Soria, R. Fitch, and S. Sukkarieh, "Geometric priors for gaussian process implicit surfaces," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 373–380, April 2017.