Probabilistic Inference-Based Robot Motion Planning via Gaussian Belief Propagation

Salman Bari, Volker Gabler, and Dirk Wollherr

***Abstract*—Robot motion planning via probabilistic inference renders a unique viewpoint on the trajectory optimization problem, in which the joint distribution of motion objectives is represented as a factor graph. Thus, the objectives are solved by obtaining the Maximum a Posteriori of the factor graph. While this distinctly improves the computational efficiency by applying least square optimization, the approach is incapable of handling hard constraints directly. In this work, we put forth an alternate perspective and argue that a message passing framework, such as Belief Propagation, offers greater utility as a solution method for robot planning problems. We present the theoretical formulation of Gaussian Belief Propagation (GaBP) as a generic message passing framework that exploits the structure of the factor graph to solve multiple planning scenarios such as batch planning, incremental planning and re-planning. In addition, the GaBP algorithm has been extended to handle hard state constraints by adopting the Difference Map strategy. We benchmark our framework in a simulation environment. The results show that our algorithms outperform the state-of-the-art with respect to collision avoidance and constraint handling ability within our benchmark. We close this article with the outline of a real-world robotic application within industrial disassembly.**

***Index Terms*—Motion and path planning, probabilistic inference, constrained motion planning, factor graph, Gaussian belief propagation.**

I. Introduction

THE robot workspace influences the type of planning algorithm required to generate a feasible trajectory from the start position to the target position. If the workspace of the robot contains only static obstacles, a *batch planning* [1], [2] approach is used. In this case, the complete feasible trajectory form start to goal state is determined by a planner. Then, a lower-level controller is used to track the planned trajectory. This approach does not work in a dynamic environment. An *incremental planning* [3] strategy is required if moving obstacles are within reach of a robotic manipulator. Another possible scenario is *re-planning* [4], where a certain part of the trajectory has to be re-planned according to the partially changed planning conditions, e.g. change in goal position during robot motion. Re-planning requires only a partial update in the planned trajectory. These different planning scenarios have been mostly considered as separate planning problems in literature and specific algorithms have been developed for each case. Our study tackles the problem of multi-scenario robot planning and explores the message passing framework for finding Maximum a Posteriori (MAP) estimation to generate optimal trajectories.

Robot motion planning can be formulated as planning-as-inference (PAI), which orchestrates the planning problem on a graphical model and then utilizes the already existing and established algorithms for finding MAP [5], [6]. The PAI offers the utility of approximate inference algorithms to compute fast solution as compared to other optimization and sampling-based planning approaches [7]. The computation of the MAP in a factor graph using message passing approaches, such as Belief Propagation (BP) [8], involves local communication among nodes in the graph to determine marginal distributions of variable nodes. This characteristic of message passing algorithms can be harnessed to develop motion planning algorithms that can be employed in different planning scenarios.

In this letter, we explore the potential of GaBP as a message passing framework for solving robot motion planning problems. We also address one of the major limitations of Gaussian Process Motion Planning (GPMP2), its incapability to handle hard constraints. We begin by outlining the theoretical framework for solving batch planning problems. Then, a novel method to enforce hard state constraints in configuration space has been proposed. This is achieved by applying the Difference Map (DM) approach to satisfy constraints via *bound projections*. Moreover, conditional dependencies in the factor graph nodes have been highlighted, which results in a clear distinction between local and global graph reasoning. Then, the property of local convergence is exploited to formulate local sub-graphs. Based on these sub-graphs, GaBP has been extended to Incremental Gaussian Belief Propagation (iGaBP) to address incremental planning and re-planning problems.

We evaluate the proposed algorithm in an exemplary simulation setting using a 6-DoF COMAU robot manipulator. A comparative analysis with state-of-the-art probabilistic inference and optimization-based motion planning algorithms has been performed. The results show the superior performance of the proposed approach in simulated planning scenarios with respect to success rate and constraint handling. In addition, we showcase

Manuscript received 8 March 2023; accepted 19 June 2023. Date of publication 7 July 2023; date of current version 13 July 2023. This letter was recommended for publication by Associate Editor L. Rozo and Editor A. Faust upon evaluation of the reviewers' comments. This work was supported by the Project "HR-Recycler - Hybrid Human-Robot Recycling Plant for Electrical and Electronic Equipment" through Horizon 2020 Research and Innovation Programme under Grant 820742. *(Corresponding Author: Salman Bari.)*

The authors are with the Chair of Automatic Control Engineering (LSR), TUM School of Computation, Information and Technology, Technical University of Munich, 80333 Munich, Germany (e-mail: s.bari@tum.de; v.gabler@tum.de; dw@tum.de).

Digital Object Identifier 10.1109/LRA.2023.3293320

a real-world application example for an industrial Waste Electrical and Electronic Equipment (WEEE) disassembly setting and discuss the key findings along with limitations and future work.

II. Related Work

Planning-as-inference has been used to solve planning and sequential decision-making problems [9], [10]. Eventually, the same idea was adopted by the robotics community to solve trajectory optimization and related motion planning problems [5]. In this approach, a multi-variate description of the robot action or state is represented as a factor graph and MAP estimate is computed based on the desired planning objective [6]. Recently, the GPMP2 framework [7] has been proposed which represents the prior trajectory as a Gaussian Process (GP) generated from a linear time variant, stochastic differential equation (LTV-SDE), and finds collision-free trajectories with fast numerical least square optimization. MAP inference of probabilistic graphical models is equivalent to least square optimization solution as per inference-optimization duality [11]. In order to tackle re-planning scenario, GPMP2 proposes Bayes tree formulation to perform incremental trajectory updates. Recently, a stochastic interpretation of GPMP2 has also been proposed [12] that takes into account uncertainties by means of Gaussian variational inference. However, GPMP2 does not fully exploit the underlying structure of the factor graph as it considers the solution of the entire factor graph holistically by fusing all motion objectives. The fusion of motion objectives also limits the ability of this approach to deal with hard constraints.

The application of the message passing approach as a solution method for a motion planning problem has not yet been studied in detail. Early work on robot planning via probabilistic inference [5], [6] adopts a similar message passing approach but it does not provide detailed investigation of its utility. More recently, the GaBP-based inference approach [13] has been used to solve a multi-robot planning problem. The proposed algorithm is very similar to our work as it also employs GaBP but multiple planning scenarios are not investigated. We fill this research gap by complementing [13] and discussing the GaBP approach that exploits factor graph structure to generate trajectories in multiple planning scenarios, which differs from multimodel trajectory optimization [14] that generates multiple solutions based on various cost functions.

We rely on Danny Bickson's GaBP algorithm [15] to solve non-linear factor graph for batch planning. We leverage the insights from our earlier work [16], which suggested that optimizing the local nodes of the factor graph offers greater sensitivity to obstacle avoidance. However, it should be noted that the approach proposed in our previous work [16], which utilized the min-sum message passing algorithm and numerical optimization for local node belief computation, differs from the approach we present in this letter. We also propose the sub-graph formulation and incremental inference via GaBP which leads to a message passing approach capable of planning in various real-world scenarios.

In addition, we address hard constraints handling using GaBP. Typically, constraints can be modeled in *soft manner* as proposed in GPMP2 or the joint limits can be imposed by clamping such that each optimization update follows the limits as proposed in [17]. While recent work [18], [19] has targeted the constrained optimization solution of factor graphs, they still rely on traditional constrained optimization approaches similar to optimization-based planning algorithms [20]. In context of constraint satisfaction via message passing, Difference-map Belief Propagation (DMPB) [21] has been proposed for low-density parity-check codes in signal processing research based on DM and divide-and-concur methodology [22]. In this letter, we apply DM principles to propose bound projection to deal with joint-limits in robot planning.

III. Preliminaries and Background

Robot trajectory is represented as $\boldsymbol{\theta}(t) : t \rightarrow \mathbb{R}^D$, where D is the dimensionality of the state and $\boldsymbol{\theta}(t)$ is a continuous time function that maps time t to robot joint states $\boldsymbol{\theta}$ in configuration space. Following Mukadam et al. [7], robot trajectory is sampled from a continuous time GP,

$$\boldsymbol{\theta}(t) = [\boldsymbol{\theta}_0,\ ...,\ \boldsymbol{\theta}_N]^T \sim \mathcal{N}\left(\boldsymbol{\mu}(t), \boldsymbol{\mathcal{K}}\right), \tag{1}$$

that is parameterized by N states for a set of times $\boldsymbol{t} = t_0, \ldots, t_N$, where $\boldsymbol{\mu}(t)$ is the mean vector and $\boldsymbol{\mathcal{K}}(t, t')$ is the covariance function matrix. The robot workspace $\chi \subset \mathbb{R}^D$ is divided into obstacle-space $\chi_{\text{obs}} \subset \chi$ and free-space $\chi_{\text{free}} \subset \chi$. The robot planning problem is characterized as finding the robot trajectory $\boldsymbol{\theta}^*(t) \in \chi_{\text{free}}$. The state constraints are represented as $\boldsymbol{\theta}_{\text{min}}$ and $\boldsymbol{\theta}_{\text{max}}$.

A. Trajectory Optimization as Probabilistic Inference

Formally, we are interested in finding the optimal state sequence $\boldsymbol{\theta}^*_{0:N}$ that maximizes the conditional posterior $p(\boldsymbol{\theta}|\mathbf{e})$. The prior on the trajectory is defined as $p(\boldsymbol{\theta})$ that incorporates initial belief over $\boldsymbol{\theta}$. Whereas, $l(\boldsymbol{\theta}; \mathbf{e})$ is the likelihood of states $\boldsymbol{\theta}$ with respect to conditional events $\mathbf{e}$ on the trajectory such as collision avoidance and constraints. Given the prior and likelihood, the optimal trajectory $\boldsymbol{\theta}^*$ is computed by obtaining MAP estimation,

$$\begin{aligned} \boldsymbol{\theta}^*_{0:N} = \arg\max_{\boldsymbol{\theta}} \ & \underbrace{p\left(\boldsymbol{\theta}|\mathbf{e}\right)}_{p(\boldsymbol{\theta})l(\boldsymbol{\theta};\mathbf{e})}, \\ \text{s.t.} \ \ & \boldsymbol{\theta}^* \in \chi_{\text{free}}, \\ & \boldsymbol{\theta}^*_0 = \boldsymbol{\theta}_{\text{start}} \ \text{ and } \ \boldsymbol{\theta}^*_N = \boldsymbol{\theta}_{\text{goal}}, \\ & \boldsymbol{\theta}_{\text{min}} < \boldsymbol{\theta}^* < \boldsymbol{\theta}_{\text{max}}. \end{aligned} \tag{2}$$

The robot motion planning problem formulation in (2) consists of two components, prior and likelihood. Adopting the same methodology as in [7] to define the prior as a structured kernel generated from LTV-SDE, the GP prior of $\boldsymbol{\theta}$ results in

$$p\left(\boldsymbol{\theta}\right) \propto \exp\left\{-\frac{1}{2}\left\|\boldsymbol{\theta} - \boldsymbol{\mu}\right\|^2_{\boldsymbol{\mathcal{K}}}\right\}, \tag{3}$$

where $\|\boldsymbol{\theta}-\boldsymbol{\mu}\|^2_{\mathcal{K}}$ is the Mahalanobis distance. The kernel $\mathcal{K}$ induces smoothness and it can also be used to put soft constraints on start state $\boldsymbol{\theta}_{\text{start}}$ and goal state $\boldsymbol{\theta}_{\text{goal}}$. The likelihood function $l(\boldsymbol{\theta};\mathbf{e})$ which specifies the probability of avoiding obstacles is defined as

$$l\left(\boldsymbol{\theta};\mathbf{e}\right)=\exp\left\{-\frac{1}{2}\left\|\mathbf{h}\left(\boldsymbol{\theta}\right)\right\|^2_{\boldsymbol{\Sigma}_{\text{obs}}}\right\}, \tag{4}$$

where $\mathbf{h}(\boldsymbol{\theta})$ represents the vector-valued obstacle cost, and $\boldsymbol{\Sigma}_{\text{obs}}$ is a hyperparameter matrix.

B. Factor Graph Formulation

A factor graph $\mathcal{G}=(\boldsymbol{\Theta},\mathcal{F},\mathcal{E})$ is used to model a belief over continuous, multivariate random variables $\boldsymbol{\theta}$ with respect to conditional probability density functions (PDFs)

$$p\left(\boldsymbol{\theta}|\mathbf{e}\right)\propto\prod_{m=1}^{M}f_m\left(\boldsymbol{\Theta}_m\right), \tag{5}$$

where $f_m\in\mathcal{F}$ are factors that are attached to the corresponding variable nodes $\boldsymbol{\Theta}_m\in\boldsymbol{\Theta}$ via the edges $\mathcal{E}$ of the factor graph. The trajectory prior (3) is formulated as prior factors

$$p\left(\boldsymbol{\theta}\right)\propto f_0^p\left(\boldsymbol{\theta}_0\right)f_N^p\left(\boldsymbol{\theta}_N\right)\prod_{i=0}^{N-1}f_i^{\text{GP}}\left(\boldsymbol{\theta}_i,\boldsymbol{\theta}_{i+1}\right), \tag{6}$$

where $f_0^p(\boldsymbol{\theta}_0)$ and $f_N^p(\boldsymbol{\theta}_N)$ act as soft constraint factors that provide the functionality of $\boldsymbol{\theta}_0^*=\boldsymbol{\theta}_{\text{start}}$ and $\boldsymbol{\theta}_N^*=\boldsymbol{\theta}_{\text{goal}}$. The GP prior factor f_i^{GP} is defined as

$$f_i^{\text{GP}}\left(\boldsymbol{\theta}_i,\boldsymbol{\theta}_{i+1}\right)=\exp\left\{-\frac{1}{2}\left\|\boldsymbol{\Phi}(t_{i+1},t_i)\boldsymbol{\theta}_i-\boldsymbol{\theta}_{i+1}+\boldsymbol{\mu}_{i,i+1}\right\|^2_{\boldsymbol{\mathcal{Q}}_{i,i+1}}\right\}, \tag{7}$$

where $\boldsymbol{\Phi}(t_i,t_{i+1})$ is a state transition matrix and $\boldsymbol{\mathcal{Q}}_{i,i+1}$ represents power spectral density matrix (see [23] for details).

Similarly, the collision-free likelihood (4) is factorized via unary factors f_i^{obs}

$$l\left(\boldsymbol{\theta};\mathbf{e}\right)=\prod_{i=1}^{N-1}f_i^{\text{obs}}\left(\boldsymbol{\theta}_i\right). \tag{8}$$

Finding the most probable values for $\boldsymbol{\theta}$ from a factor graph is,

$$\boldsymbol{\theta}^*=\arg\max_{\boldsymbol{\theta}}\prod_{m=1}^{M}f_m\left(\boldsymbol{\Theta}_m\right). \tag{9}$$

This factor graph formulation as shown in Fig. 1 is a structured representation of the planning problem, where the system state is represented as number of state variables. An inference algorithm on the factor graph is used to compute the posterior distribution over all trajectories fulfilling the planning objectives imposed through attached factors.

IV. Gaussian Belief Propagation for Robot Planning

In this section, we outline the equations for GaBP algorithm to compute MAP in (9). We first present the theoretical formulation of our GaBP-based batch planning algorithm that can handle hard bounded-variable constraints. Then, iGaBP has been proposed to generate trajectories for incremental planning and re-planning scenarios.

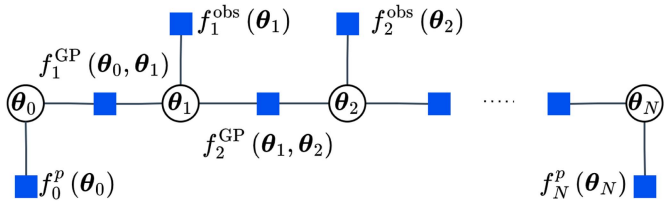

Fig. 1. A general factor graph architecture for robot motion planning problem representation.

A. GaBP as a Solution Method for Batch Planning

GaBP is a message passing framework [8] for computing the marginals of a joint distribution via local communication among nodes in a factor graph. The factor graph in (9) contains non-linear factors restricting the direct applicability of GaBP, which by design solves linearized factor graph. Therefore, we outline an iterative strategy for solving linearized factor graph. Following Bickson's linear GaBP [15] and using similar notations, we start with the assumption that the planning factor graph only contains unary and binary factors.

Assumption 1: A robot planning problem can be fully described by a pair-wise factor graph $\mathcal{G}$, which consists of two kinds of factors. Unary factors (also called self-potentials), $\phi_i:\boldsymbol{\Theta}\to\mathbb{R}\cup\{\infty\}$ that are attached to corresponding variable $\boldsymbol{\theta}_i$, and binary factors (also called edge potentials) $\psi_{ij}:\boldsymbol{\Theta}\times\boldsymbol{\Theta}\to\mathbb{R}\cup\{\infty\}$ connected to two consecutive variables $\boldsymbol{\theta}_i,\boldsymbol{\theta}_j$ where $\{ij\}\in\mathcal{E}$.

Message passing equations for the pair-wise factor graph can be divided into two types of messages. Messages passed from variables to factor nodes, denoted as $m_{\theta\mapsto f}$, and messages passed form factor nodes to variable nodes, denoted as $m_{f\mapsto\theta}$. Denoting the adjacent nodes of a factor node as $\boldsymbol{\Theta}_f$ and the adjacent nodes of a variable node as $\mathcal{F}_\theta$, the messages from an arbitrary factor f_k are iteratively communicated as

$$m_{f_k\to\theta}\left(\boldsymbol{\theta}_i\right)=\sum_{\boldsymbol{\Theta}_{f_k}\setminus\theta_i}f_k\left(\boldsymbol{\Theta}_{f_k}\right)+\prod_{\boldsymbol{\Theta}_{f_k}\setminus\theta_i}m_{\theta\to f_k}\left(\boldsymbol{\theta_j}\right), \tag{10}$$

where $\setminus$ denotes the set-theoretic difference and $\boldsymbol{\theta_j}$ is the variable node adjacent to $\boldsymbol{\theta_i}$. In case of pair-wise factor graphs $\boldsymbol{\theta}_j=\boldsymbol{\theta}_{i\pm1}$. The messages traversing from variable to factor node is

$$m_{\theta_j\to f_k}=\prod_{b\in\mathcal{F}_{\theta_j}\setminus f_k}m_{f_b\to\theta}\left(\boldsymbol{\theta}_j\right). \tag{11}$$

Given the messages from adjacent factor nodes, the marginal belief of a variable node is approximated via

$$b\left(\boldsymbol{\theta}_i\right)=\prod_{k\in\mathcal{F}_{\theta_i}}m_{f_k\to\theta}\left(\boldsymbol{\theta}_i\right). \tag{12}$$

Following [15], we use the canonical representation to derive the equations for messages among nodes and marginal means. As all the factors in the graph (9) have the quadratic energy

Algorithm 1: GaBP for Batch Planning.

Data: trajectory prior $\boldsymbol{\theta}$, factor graph $\mathcal{G} = \{\boldsymbol{\Theta}, \mathcal{F}, \mathcal{E}\}$, number of states N, convergence threshold ϵ
Result: optimized trajectory $\boldsymbol{\theta}^*$

while $\Delta\boldsymbol{\theta} \geq \epsilon$ **do**
  /* Linearize factor graph (Sec. IV-A) */
  $\boldsymbol{\eta}, \boldsymbol{\Lambda} \leftarrow \mathcal{G} = \{\boldsymbol{\Theta}, \mathcal{F}, \mathcal{E}\}$ see (17)
  for $ij \in \mathcal{E}$ **do**
    /* message Passing */
    $\boldsymbol{\Lambda}_{ij} \leftarrow -\boldsymbol{\Lambda}_{ij}^2 \boldsymbol{\Sigma}_{i/j}$
    $\delta\boldsymbol{\theta}_{ij} \leftarrow -\boldsymbol{\Sigma}_{ij}\boldsymbol{\Lambda}_{ij}\delta\boldsymbol{\theta}_{i/j}$
  /* Compute marginal mean */
  $\delta\boldsymbol{\theta}_i \leftarrow \boldsymbol{\Sigma}_{i/j}\left(\boldsymbol{\Lambda}_{ii}\delta\boldsymbol{\theta}_{ii} + \sum_{k\in\boldsymbol{\Theta}_{f_k}} \boldsymbol{\Lambda}_{ki}\delta\boldsymbol{\theta}_{ki}\right)$, $\forall i$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$
  /* Constraint violation check */
  if $\boldsymbol{\theta}_{\min} > \boldsymbol{\theta} > \boldsymbol{\theta}_{\max}$ **then**
    /* Difference map methodology */
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + [P_B(\boldsymbol{\theta}) - (\boldsymbol{\theta})]$ see (23)

return $\boldsymbol{\theta}^*$

function represented in exponential form, edge potentials ψ_{ij} and self potentials ϕ_i can be written in canonical form as

$$\psi_{ij}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \exp\left\{-\frac{1}{2}\boldsymbol{\theta}_i^\top \boldsymbol{\Lambda}_{ij} \boldsymbol{\theta}_j\right\}, \tag{13}$$

$$\phi_i(\boldsymbol{\theta}_i) = \exp\left\{-\frac{1}{2}\boldsymbol{\theta}_i^\top \boldsymbol{\Lambda}_{ii} \boldsymbol{\theta}_i + \boldsymbol{\eta}_i^\top \boldsymbol{\theta}_i\right\}, \tag{14}$$

where $\boldsymbol{\Lambda}$ is the data matrix also called precision matrix and $\boldsymbol{\eta}$ is the observation matrix. Similarly, the alternate representation of the factor graph is

$$f(\boldsymbol{\theta}) = \exp\left\{-\frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{\Lambda}\, \boldsymbol{\theta} + \boldsymbol{\eta}^\top \boldsymbol{\theta}\right\}, \tag{15}$$

where mean $\boldsymbol{\mu}$ and precision $\boldsymbol{\Lambda}$ are related by $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$.

As the obstacle factor function $\mathbf{h}(\boldsymbol{\theta})$ is non-linear, the energy is not quadratic in $\boldsymbol{\theta}$, meaning the factor is not Gaussian distributed. To approximate the Gaussian form of the factor graph, a first order Taylor series expansion is applied for non-linear factors $f(\boldsymbol{\theta})$ to find its approximate values $\boldsymbol{\theta}$ close to $\bar{\boldsymbol{\theta}}$

$$f(\boldsymbol{\theta}) \approx \mathbf{h}(\bar{\boldsymbol{\theta}}) + \mathbf{J}\underbrace{(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})}_{\delta\boldsymbol{\theta}}, \tag{16}$$

where $\mathbf{J}$ is the Jacobian which is a multi-variate partial derivative $\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}}$ at the linearized state $\bar{\boldsymbol{\theta}}$ with $\delta\boldsymbol{\theta} \triangleq (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})$ being the state update vector. Then, the linear form in terms of information and precision matrix is,

$$\begin{aligned} \boldsymbol{\eta} &= \mathbf{J}^\top \boldsymbol{\Lambda}\left[\mathbf{J}\bar{\boldsymbol{\theta}} + \boldsymbol{\mu} - \mathbf{h}(\bar{\boldsymbol{\theta}})\right], \\ \boldsymbol{\Lambda} &= \mathbf{J}^\top \boldsymbol{\Lambda}\, \mathbf{J}. \end{aligned} \tag{17}$$

Finding the MAP estimation in the linearized graph $\boldsymbol{\Lambda}\delta\boldsymbol{\theta} - \boldsymbol{\eta}$ translates to solving the vector-matrix linear equations [15] as,

$$\delta\boldsymbol{\theta}^* = \underset{\delta\boldsymbol{\theta}}{\arg\min} \|\boldsymbol{\Lambda}\delta\boldsymbol{\theta} - \boldsymbol{\eta}\|. \tag{18}$$

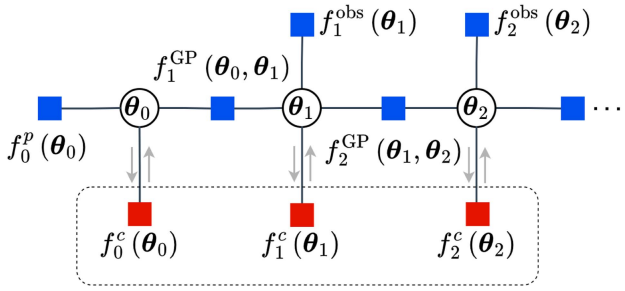

Fig. 2. Updated graph with constraint factors.

The structure of the matrix $\boldsymbol{\Lambda}$ is dictated by the graph topology. As per Bickson's methodology [15], the messages among nodes are proportional to normal distributions with precision as,

$$\begin{aligned} \boldsymbol{\Lambda}_{ij} &= -\boldsymbol{\Lambda}_{ij}^2 \boldsymbol{\Sigma}_{i/j}, \\ \delta\boldsymbol{\theta}_{ij} &= -\boldsymbol{\Sigma}_{ij}\boldsymbol{\Lambda}_{ij}\delta\boldsymbol{\theta}_{i/j}. \end{aligned} \tag{19}$$

The (19) represents the messages propagated in the GaBP algorithm. The marginals are computed as follows (cf. [15], Sec. 2.3),

$$\delta\boldsymbol{\theta}_i^* = \boldsymbol{\Sigma}_{i/j}\left(\underbrace{\boldsymbol{\Lambda}_{ii}\delta\boldsymbol{\theta}_{ii}}_{\phi_i(\boldsymbol{\theta}_i)} + \sum_{k\in\boldsymbol{\Theta}_{f_k}} \underbrace{\boldsymbol{\Lambda}_{ki}\delta\boldsymbol{\theta}_{ki}}_{m_{f_k\to\boldsymbol{\theta}_i}}\right), \quad \forall i. \tag{20}$$

Similarly, the precision can be calculated as

$$\boldsymbol{\Lambda}_i = \boldsymbol{\Lambda}_{ii} + \sum_{k\in N(i)} \boldsymbol{\Lambda}_{ki}, \ \forall i. \tag{21}$$

The solution for the non-linear factor graph is then updated according to the linear graph solution $\delta\boldsymbol{\theta}$ as

$$\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}^* \tag{22}$$

In order to impose hard constraints to the factor graph solution in (22), we introduce a *constraint graph* as shown in Fig. 2. We leverage the constraint satisfaction methodology from DM, which is an approach based on divide-and-concur projections [22] for constraint satisfaction that has been extended to message passing framework DMPB [21]. In this approach, for each constraint imposed on a variable, a replica of that variable is created in an updated constraint factor graph [21] and then DM-based divide-and-concur approach is used for constraint satisfaction. Instead of divide-and-concur projection as in [21], we propose the *bound projection* denoted by P_B. The messages from constraint factor nodes to variable nodes are updated according to

$$m_{f^c\mapsto\boldsymbol{\theta}} = \boldsymbol{\theta} + [P_B(\boldsymbol{\theta}) - (\boldsymbol{\theta})], \ \forall i, \tag{23}$$

where, the P_B represents the bound projection and defined as,

$$P_B(\boldsymbol{\theta}) = \begin{cases} \boldsymbol{\theta} & if\ \boldsymbol{\theta}_{\min} \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}_{\max} \\ \boldsymbol{\theta}_{\min} & if\ \boldsymbol{\theta} < \boldsymbol{\theta}_{\min} \\ \boldsymbol{\theta}_{\max} & if\ \boldsymbol{\theta} > \boldsymbol{\theta}_{\max} \end{cases}. \tag{24}$$

Algorithm 1 entails the details of solving batch planning problems including constraint handling via bound projection.

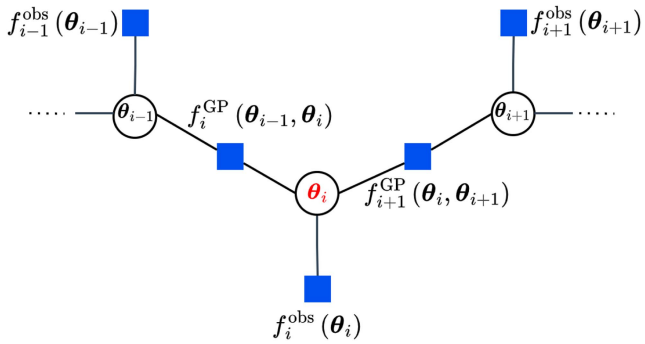

Fig. 3. Proposed sub-graph $\mathcal{G}_i^{\text{sub}}$ structure for incremental inference. The figure represents the inclusion of the factors required to update the solution of a single variable $\boldsymbol{\theta}_i$.

Note that the constraint factors are not Gaussian and these are not included in the message passing iterations of belief update. The major benefit of the proposed approach is that the DM methodology step of the GaBP algorithm is only activated in case of constraint violation making the overall approach more efficient. The algorithm proposed in this section solves the complete factor graph, however, an incremental strategy can also be devised (next Sec.) that can solve the factor graph incrementally.

B. Incremental Inference Via GaBP

In this section, we exploit the *local convergence* characteristic of GaBP for finding an incremental solution of the factor graph. Due to the factorized structure of the planning problem, global inference can be achieved by individually solving many interdependent local sub-problems [16]. We propose the formation of local *sub-graphs* and solving them incrementally for individual variable nodes that leads to global MAP estimation of the entire factor graph. The sub-graph for $\boldsymbol{\theta}_i$ is,

$$\mathcal{G}_i^{\text{sub}} = \prod_{m=i}^{i+1} f_m^{\text{GP}}(\boldsymbol{\theta}_{m-1}, \boldsymbol{\theta}_m) \prod_{m=i-1}^{i+1} f_m^{\text{obs}}(\boldsymbol{\theta}_m). \tag{25}$$

Fig. 3 shows the proposed sub-graph structure for finding local MAP estimation for $\boldsymbol{\theta}_i$. In order to obtain $\boldsymbol{\theta}_i^*$, the variable nodes $\boldsymbol{\Theta}^{sub} = \{\boldsymbol{\theta}_{i-1}, \boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}\}$ are initialized according to (3). Although the value of $\boldsymbol{\theta}_{i+1}$ is also partially optimized, we consider its value to be initialized from prior $p(\boldsymbol{\theta})$ as it helps in inducing smoothness as per GP prior. The solution for the sub-graph is obtained from GaBP as outlined in Algorithm 1. However, it is an independent node optimization. Solving individual sub-graphs incrementally leads to full graph optimization resulting in $\boldsymbol{\theta}^*$ (for detailed convergence analysis, cf. [16], Appendix A).

For incremental planning, the key idea is to optimize only the next variable node instead of the entire factor graph. As a result, only one graph node $\boldsymbol{\theta}_i$ is optimized at a time-step and the full optimized trajectory $\boldsymbol{\theta}^*$ is generated incrementally. Note that the proposed incremental inference approach iGaBP is different from GaBP; the batch optimization of the complete graph is replaced by an iterative local optimization of each node that has the capability to cater moving obstacles in a dynamic environment. iGaBP can also be used for re-planning scenario. A naive way to solve the re-planning problem in case of partial modifications in planning conditions is to solve the entire factor graph. However, it is an in-efficient approach for real-world robotic applications. iGaBP offers the functionality to generate trajectory conveniently by simply updating the graph settings to changes and solving only the local sub-graph.

V. Implementation Details

The GaBP-based message passing framework has been implemented on top of Georgia Tech Smoothing and Mapping (GTSAM) library similar to GPMP2. The implemented algorithm has two layers: the top layer is programmed in MATLAB and incorporates robot kinematic model, obstacles and trajectory characteristics, the lower-level GaBP solver is implemented in C++, augmenting the GTSAM library.

A. Prior and Likelihood

We use a constant-velocity prior similar to [7] with a Markovian state consisting of position and velocity in configuration space. The trajectory is generated from LTV-SDE [23]. Similar to recent state-of-the-art motion planning algorithms [2], [7], the robot collision body is represented by a set of spheres. The obstacle cost function is obtained by computing hinge loss

$$\mathbf{c}(d) = \begin{cases} -d + \varepsilon & \text{if } d \leq \varepsilon \\ 0 & \text{if } d > \varepsilon \end{cases}, \tag{26}$$

for the spheres. d represents the signed distance from the sphere to the closest obstacle surface in the workspace, and ε is the *safety distance*. For likelihood in (4), the $\boldsymbol{\Sigma}_{\text{obs}}$ is defined as

$$\boldsymbol{\Sigma}_{\text{obs}} = \sigma_{\text{obs}}^2 \mathbf{I}. \tag{27}$$

The trajectory prior is initialized as a constant velocity straight line with $N = 11$ states from the start state to the goal state. The term σ_{obs} puts weight on observing the obstacles. We set $\sigma_{\text{obs}} = 0.0015$ and $\varepsilon = 0.2$ for our manipulator planning tasks.

B. Motion Constraints

Motion constraint handling is very crucial in real-world motion planning tasks. There are two kinds of constraints that we encounter in real-world implementation of motion planning algorithms. We handle the equality constraints in a soft manner similar to [7] by imposing them directly into the trajectory prior. To prevent joint limit violations, we use the GaBP as outlined in Section IV-A.

VI. Evaluation

We benchmark the proposed algorithms on an exemplary motion planning task: finding collision-free trajectories in a simulated WEEE disassembly cell. WEEE disassembly is a process in the recycling procedure in which the components of the WEEE devices such as PC-towers and Microwave ovens are dismantled. For that purpose, the robot is mounted with end-effector tools to perform the disassembly of the electronic devices. A COMAU 6-DOF racer robot arm has been used for benchmarking the

TABLE I
RESULTS OBTAINED FOR 24 BATCH PLANNING PROBLEMS

	GaBP	GPMP2	GPMP2-inter	TrajOpt	GaBP (N=51)	TrajOpt (N=51)
Success (%)	**100**	87.5	95.8	**100**	95.8	**100**
Average time (ms)	9.78	**2.08**	8.03	12.69	48.94	17.71
Max. time (ms)	21.04	**3.25**	13.16	16.60	65.60	24.68

Bold values represent the best performing results.

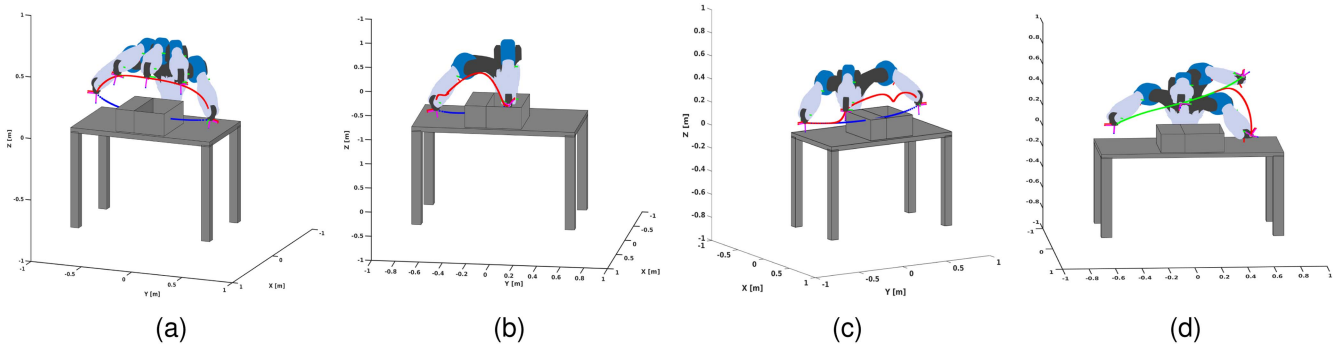

Fig. 4. Trajectory (in red) generated by GaBP and iGaBP for different planning scenarios. (a) Successful batch planning trajectory via GaBP. (b) Batch planning result in case when goal pose is inside the PC-Tower. (c) Incremental planning result when PC-Tower is placed on the table during robot motion at $t = 4$ time-step. (d) Trajectory re-planning via iGaBP in case of a changed goal pose during motion.

TABLE II
RESULTS OF 12 INCREMENTAL PLANNING PROBLEMS

	iGaBP
Success (%)	91.67
Average time (ms)	17.64
Max. time (ms)	21.19
Average time per iteration (ms)	1.96
Max. time per iteration (ms)	4.86

TABLE III
RESULTS OF 12 RE-PLANNING PROBLEMS

	iGaBP	iGPMP2
Success (%)	**100**	**100**
Average time (ms)	**1.05**	1.21
Max. time (ms)	2.08	**1.50**

proposed algorithms. All the benchmarks have been run on an x86_64 of 12th Gen Intel 24-Core i9-12900 K system.

Comparing different motion planning algorithms is a difficult task mainly due to the inconsistent mechanisms used for planning problem formulations, solution methodologies, and planning environments. In order to keep the comparison fair, we select the planning algorithms that use similar methodologies as the proposed approach and benchmark on two fundamental criteria: success rate and computation time. Comparative analysis of the proposed algorithms has been performed against GPMP2, GPMP2-inter and TrajOpt [24]. Planning conditions and parameter values have been kept identical for all cases. In the case of GPMP2-inter, GP interpolation has been performed by adding factors and the resulting output of the algorithm is an interpolated trajectory with the total states $N = 51$. Whereas for GaBP, trajectory interpolation is not part of the algorithm

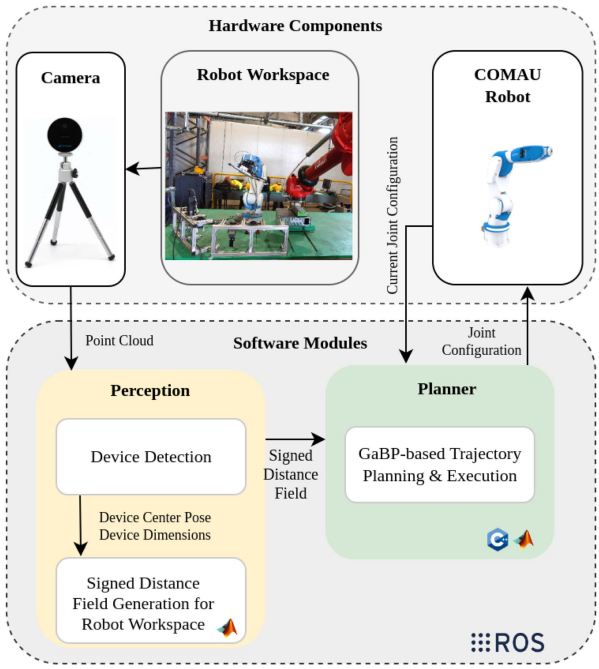

Fig. 5. Overall planner implementation architecture for WEEE Disassembly Cell.

and it has been performed as a separate step before trajectory execution for real-world experiments.

A. Batch Planning Benchmark

We evaluated the proposed algorithm for 24 unique batch planning problems for three different scenarios of PC-Tower disassembly that include, the trajectory going around it and

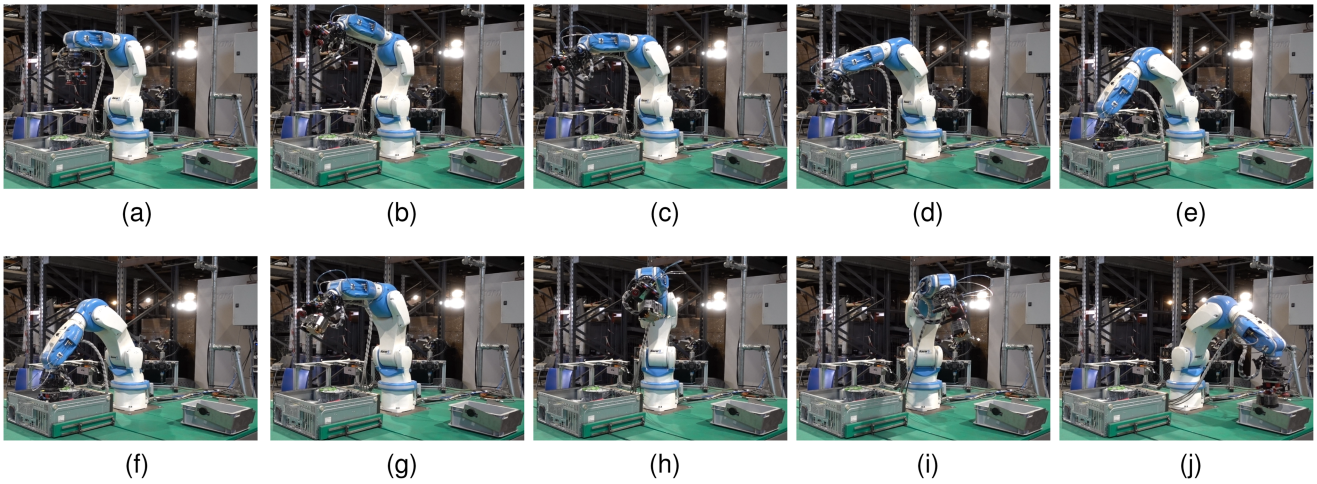

Fig. 6. Robot configurations during pick and place task for PC-Tower. (a)–(e) Robot attempting to pick the PC fan inside the PC-Tower, whereas (f)–(j) show the dropping of PC fan to designated goal pose.

TABLE IV
RESULTS OF 24 BATCH PLANNING PROBLEMS FOR WEEE DISASSEMBLY

	GaBP	GPMP2	GPMP2-inter
Success (%)	**93.33**	66.67	26.67
Average time (ms)	17.45	**2.15**	4.17
Max. time (ms)	33.69	**3.49**	8.01
Joint Limit violations	**0/24**	4/24	10/24

also going in and outside of the PC-tower. For each scenario, different start and end configurations have been tested. In order to make the planning problem more challenging, the start and end configurations have been kept very close to the body of PC-tower.

The results for batch planning are summarized in Table I. Overall, GPMP2 offers fast solutions but it comes at the behest of lower success rate. GPMP2-inter takes more time to converge but the success rate is better than GPMP2 as the obstacle factors have also been attached to the interpolated states. As the number of states considered increases, TrajOpt [24] demonstrates improved performance, but this improvement comes at the cost of increased computation time. GaBP offers better collision avoidance capabilities due to the local message communication among graph nodes. The role of unary obstacle factors and their associated cost is more critical than the binary factors, making GaBP more sensitive to obstacles. For a larger number of states, the scalability of GaBP is constrained by the increased communication among graph nodes, leading to longer computation times. Nevertheless, this limitation can be overcome by implementing parallel processing of messages.

B. Incremental Planning Benchmark

Fig. 4(c) shows the resulting trajectory within a dynamic environment, in which the PC-Tower is placed on the table at time-step $t = 4$. The average time per iteration from Table II indicates the computing efficiency of the proposed approach. We observed that the trajectory smoothness does get affected in case of individual node optimization via iGaBP. However, this is not a major concern as, even in real-world implementation, trajectory interpolation takes place after the planning phase which handles the overall smoothness.

C. Re-Planning Benchmark

We benchmark the re-planning results against Incremental Gaussian Process Motion Planning (iGPMP2). Fig. 4(d) shows the trajectory produced in the case when goal pose is changed during motion. The resulting re-planned trajectory is the updated trajectory given the new goal pose. iGaBP computes only the portion of the factor graph in which the planning conditions have been changed, making the re-planning faster as presented in Table III.

D. Performance Evaluation for WEEE Disassembly

The performance of the proposed algorithm has also been tested for WEEE disassembly in a real-world pilot setup. The software stack for WEEE disassembly motion planner is shown in Fig. 5. The robot is mounted on one edge of the disassembly table. We implemented the planner as a *ROS Action Server*. The client (*a high-level task planner*) can send the request to the *Planner action server* to plan and execute the trajectory. The trajectory is interpolated via piece-wise cubic B-splines before executing it on the robot. Fig. 6 shows the robot trajectory produced for the pick-and-place task of PC-Tower disassembly procedure.

The batch planning results in Table IV for the WEEE disassembly cell indicate the necessity of handling hard constraints in real-world implementations as the 6-DoF COMAU robot in combination with the end-effector toolchain is prone to run into joint limits. In GaBP, no joint limit violations were observed, while GPMP2 had 4 out of 24 planning tests with violated joint limits. The activation of constraint violation check

in Algorithm 1 causes the computation time for GaBP to increase, as more iterations are required for convergence. Putting the constraint factors on interpolated states can further improve the performance of GPMP2-inter. However, it is not expected to perform better than GPMP2.

VII. Limitations and Future Work

The proposed algorithm is limited in a sense that it can only handle hard inequality constraints. However, we did not find it limiting in our real-world experiments as joint limits are more crucial that are tackled via proposed GaBP algorithm. Furthermore, The graph structure matrix $\mathbf{\Lambda} \in \mathbb{R}^{\mathrm{m} \times n}$ is not square $(\mathrm{m} \neq n)$ in (17). We transform the matrices to $\mathbf{\Lambda}^{T}_{\mathrm{n}\times\mathrm{m}}\ \mathbf{\Lambda}_{\mathrm{m}\times\mathrm{n}} \mapsto \mathbf{\Lambda}_{\mathrm{n}\times\mathrm{n}}$ and $\mathbf{\Lambda}^{T}_{\mathrm{n}\times\mathrm{m}}\ \boldsymbol{\eta}_{\mathrm{m}\times 1} \mapsto \boldsymbol{\eta}_{\mathrm{n}\times 1}$ which needs to be computed prior to running messages which add extra computational load. The methodology proposed in (cf. [15], Sec. 7.1) can be adopted to overcome this in future work. Also, incremental planning and re-planning have been only tested in simulation for the WEEE disassembly cell due to the COMAU robot missing a real-time joint control interface for incremental execution.

For future work, we plan to investigate the distributed properties of GaBP to address its scalability limitations. The local sub-graph formulation and solution proposed in Section IV-B allows for parallel processing and different messaging schedules that can further increase the computing efficiency, possibly bringing it at par with GPMP2. Furthermore, GaBP can run continually in the background and compute the inference as new data arrives. We plan to explore further the similar kind of capabilities e.g. just-in-time computation of GaBP for robot planning.

VIII. Conclusion

The problem targeted by this work is a standard kinematic motion planning for a robot manipulator in the presence of obstacles and joint limit constraints. We argue the case of GaBP as a solution method for MAP estimation of probabilistic inference-based robot motion planning. We highlight that the GaBP framework offers the utility to cater batch planning, incremental planning and re-planning scenarios. Further, our investigation emphasizes the necessity of handling hard constraints. It has been observed that handling the joint limit constraints in soft manner reduces the success rate of planning algorithms in real-world robot settings. The investigation and experiments performed in this work also indicated that the GaBP has the right structural and computational properties to act as a generic framework to produce motion trajectories in various real-world planning scenarios. Its ability to handle hard constraints further strengthens its suitability for such applications. For future work, an exciting research direction is the examination of distributive properties of GaBP that could further improve its computing efficiency.

References

[1] M. Kalakrishnan, S. Chitta, E. A. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, Shanghai, China, 2011, pp. 4569–4574.

[2] M. Zucker et al., "CHOMP: Covariant hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, no. 9–10, pp. 1164–1193, 2013.

[3] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proc. 22nd Int. Conf. Automated Plan. Scheduling*, Atibaia, São Paulo, Brazil, 2012, pp. 207–215.

[4] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 354–363, Jun. 2005.

[5] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, Montreal, Quebec, Canada, 2009, pp. 1049–1056.

[6] M. Toussaint and C. Goerick, "A Bayesian view on motor control and planning," in *From Motor Learning to Interaction Learning in Robots* (Studies in Computational Intelligence Series), vol. 264, O. Sigaud and J. Peters, Eds. Berlin, Germany: Springer, 2010, pp. 227–252.

[7] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time Gaussian process motion planning via probabilistic inference," *Int. J. Robot. Res.*, vol. 37, no. 11, pp. 1319–1340, 2018.

[8] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," in *Proc. 13th Int. Conf. Neural Inf. Process. Syst.*, 2000, pp. 689–695.

[9] H. Attias, "Planning by probabilistic inference," in *Proc. 9th Int. Workshop Artif. Intell. Statist.*, 2003, pp. 9–16.

[10] M. Toussaint and A. J. Storkey, "Probabilistic inference for solving discrete and continuous state Markov decision processes," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 945–952.

[11] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1203, 2006.

[12] H. Yu and Y. Chen, "A Gaussian variational inference approach to motion planning," *IEEE Robot. Autom. Lett.*, vol. 8, no. 5, pp. 2518–2525, May 2023.

[13] A. Patwardhan, R. Murai, and A. J. Davison, "Distributing collaborative multi-robot planning with Gaussian belief propagation," *IEEE Robot. Automat. Lett.*, vol. 8, no. 2, pp. 552–559, Feb. 2023.

[14] T. Osa, "Multimodal trajectory optimization for motion planning," *Int. J. Robot. Res.*, vol. 39, no. 8, pp. 983–1001, 2020.

[15] D. Bickson, "Gaussian belief propagation: Theory and aplication," 2008, *arXiv:0811.2518*.

[16] S. Bari, V. Gabler, and D. Wollherr, "MS2MP: A min-sum message passing algorithm for motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, Xi'an, China, 2021. pp. 7887–7893.

[17] L. Petrovic, I. Markovic, and I. Petrovic, "Mixtures of Gaussian processes for robot motion planning using stochastic trajectory optimization," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 12, pp. 7378–7390, Dec. 2022.

[18] P. Sodhi, S. Choudhury, J. G. Mangelson, and M. Kaess, "ICS: Incremental constrained smoothing for state estimation," in *Proc. IEEE Int. Conf. Robot. Automat.*, Paris, France 2020, pp. 279–285.

[19] B. Bazzana, T. Guadagnino, and G. Grisetti, "Handling constrained optimization in factor graphs for autonomous navigation," *IEEE Robot. Autom. Lett.*, vol. 8, no. 1, pp. 432–439, Jan. 2023.

[20] B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. M. Lane, "Timed-elastic bands for manipulation motion planning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3513–3520, Oct. 2019.

[21] J. S. Yedidia, Y. Wang, and S. C. Draper, "Divide and concur and difference-map BP decoders for LDPC codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 786–802, Feb. 2011.

[22] S. Gravel and V. Elser, "Divide and concur: A general approach to constraint satisfaction," *Phys. Rev. E*, vol. 78, Sep. 2008, Art. no. 036706.

[23] T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch continuous-time trajectory estimation as exactly sparse Gaussian process regression," in *Proc. Robot.: Sci. Syst.*, vol. 10, 2014, pp. 1–10.

[24] J. Schulman et al., "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, 2014.