IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 December 2024, accepted 21 January 2025. Date of publication 00 xxxx 0000, date of current version 00 xxxx 0000.

Digital Object Identifier 10.1109/ACCESS.2025.3534993

Factor Graphs in Optimization-Based Robotic Control–A Tutorial and Review

ANAS ABDELKARIM[1,2], HOLGER VOOS[1,3], (Member, IEEE), AND DANIEL GÖRGES[2], (Member, IEEE)

[1]Interdisciplinary Center for Security, Reliability and Trust (SnT), University of Luxembourg, 1855 Luxembourg City, Luxembourg
[2]Department of Electrical and Computer Engineering, RPTU University of Kaiserslautern-Landau, 67663 Kaiserslautern, Germany
[3]Faculty of Science, Technology, and Medicine (FSTM), Department of Engineering, University of Luxembourg, 1359 Luxembourg City, Luxembourg

Corresponding author: Anas Abdelkarim (anas.abdelkarim@uni.lu)

This research was funded in whole, or in part, by the Luxembourg National Research Fund (FNR), MOCCA Project, ref. 17041397. For the purpose of open access, and in fulfilment of the obligations arising from the grant agreement, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

ABSTRACT Factor graphs, initially developed as probabilistic graphical models, have been widely employed for solving large-scale inference problems in robotics, particularly in tasks such as pose estimation, Structure from Motion (SfM), or Simultaneous Localization and Mapping (SLAM). Their capability to efficiently model uncertainty and the locality of sensor data has made them crucial for robotic perception and situational awareness. Recently, factor graphs have evolved beyond their probabilistic origins and are also being applied to deterministic optimization problems, such as robotic planning and control. This paper first aims to provide a comprehensive tutorial on factor graphs and the formulation and solution of the related optimization problems within the context of robotics perception. In addition, we undertake a thorough review of approaches that extend factor graphs—traditionally solved by unconstrained optimization—to optimal control tasks, emphasizing how they handle the constraints intrinsic to control problems. Finally, we analyze the potential of factor graphs for the seamless integration of robotic situational awareness, planning, and control, which remains one of the most critical challenges in achieving fully autonomous robot operations in complex environments.

INDEX TERMS Factor graphs, constrained factor graphs, robotic control, slam, situational awareness, robotic perception, graph optimization, model predictive control, least squares optimization.

I. INTRODUCTION

Factor graphs are powerful probabilistic graphical models that represent complex systems with uncertainty [1], [2], and offer a flexible and intuitive means of modeling probability distributions [3]. Uncertainty in the field of mobile robots can have various sources, including inconsistent actuation, noisy sensors, and unpredictable surroundings. To address these uncertainties, probabilistic robotics [4] has emerged as a successful paradigm. This approach utilizes probabilistic graphical models, such as factor graphs, as mathematical tools to model and handle uncertainty effectively. Herein, factor graphs have so far mainly been used with great success in various robotics perception tasks, including pose estimation and tracking, simultaneous localization and mapping (SLAM) [5], bundle adjustment [3], Structure from Motion (SfM) [6], and also higher-level situational awareness [7], [8]. In these applications, the tasks are formulated as probabilistic inference problems, solved by computing a Maximum A Posteriori (MAP) estimate using optimization techniques [9]. Specifically, solving such inference problems with factor graphs involves least-squares optimization, where graph optimization serves as the back-end, and the processing of sensor data integrated into the factor graph as the front-end [10].

Significant advancements over the past two decades have made factor graph-based approaches highly efficient for large-scale, real-time perception tasks [1]. The efficiency arises from the locality of the sensor data and the resulting sparsity in optimization problems common in robotic perception tasks [2], leading to the development of well-established solvers capable of handling sparse optimization problems in a very efficient way [1], [3], [11].

The associate editor coordinating the review of this manuscript and approving it for publication was Yangmin Li.

Beyond perception, the tasks of planning and control are essential for efficient operations in robotic applications. However, perception, planning, and control have traditionally been executed within different frameworks, leading to challenges in integration and efficiency. For instance, optimization-based planning and control often assume deterministic system models, while perception tasks are inherently probabilistic. This discrepancy results in different implementation requirements: perception tasks often rely on factor graph solvers designed specifically for robotics applications, whereas optimal control tasks frequently use general-purpose optimization solvers like IPOPT [12]. Consequently, interfacing between the perception layer and the control layer is required, introducing additional complexity and implementation effort.

Factor graphs hold immense potential as a unified framework for perception, planning, and control tasks. Extending their application to modeling and solving optimal control problems offers several key advantages. First, it simplifies implementation by using the same libraries across tasks, reducing the reliance on separate tools. Moreover, the outcomes of perception tasks, such as the robot pose, can be directly leveraged for control purposes. Additionally, factor graph libraries come equipped with high-performance optimization algorithms that can be effectively utilized in control tasks. By integrating these tasks into a single framework, factor graphs not only streamline implementation but also enhance the scalability and performance of robotic systems, simplifying software architectures and improving overall system efficiency.

However, unifying these frameworks also presents significant challenges. Factor graphs are typically designed for unconstrained optimization, primarily using least-squares methods, whereas optimal control inherently involves constraints. Handling these constraints is a primary obstacle to fully leveraging factor graphs for optimal control tasks. Recent research has highlighted this challenge and proposed solutions to integrate constraints into factor graph-based control, as we will later detail in this paper.

Although there are numerous reviews on factor graphs, such as [1], [13], [14], [15], and [16], they focus primarily on perception tasks. To the best of our knowledge, no review paper comprehensively covers the use of factor graphs for optimal control so far. This article aims to fill this gap by providing a detailed review of the literature that integrates optimal control with factor graph optimization, discussing the techniques and algorithms proposed to address constraint handling. It is important to note that factor graphs can be solved using various approaches other than optimization techniques, such as sampling methods [17] or message-passing algorithms [18]. Consequently, control tasks have also been expressed and solved using factor graphs through sampling methods or message-passing algorithms [19]. However, since robot perception tasks are fundamentally based on graph optimization, this article focuses on research that employs graph optimization in optimal control problems.

Notably, the current applications of factor graph-based optimization for control in robotics, as discussed in the literature reviewed in this article, predominantly address deterministic optimal control problems, as we will show later in this article. For example, dynamic models in these studies are typically assumed to be deterministic. Nevertheless, this focus on deterministic models does not limit the potential of using factor graphs for probabilistic optimization problems. Under certain assumptions, it is possible to approximate probabilistic optimal control problems efficiently as deterministic ones [20]. Furthermore, the success of leveraging efficient factor graph solvers for deterministic optimization problems paves the way for exploring the direct application of factor graphs to probabilistic optimal control, as both are fundamentally rooted in probabilistic theory.

Given the ongoing research and potential of leveraging factor graphs for optimal control, this article also aims to serve as an entry point to the topic. We address four key questions:

1) What are factor graphs from a robotics perception perspective, and how do the related tailored optimization algorithms work?
2) How can factor graphs be utilized for optimal control in robotics?
3) What is the current state-of-the-art in this regard?
4) What are potential future research directions to advance this field?

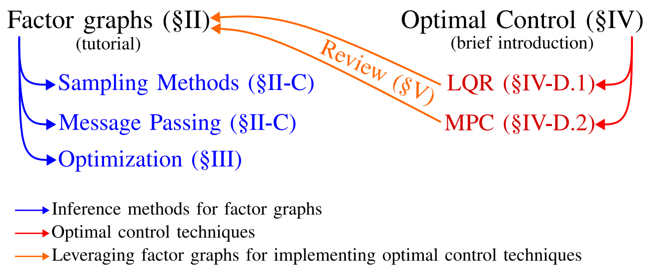


FIGURE 1. Roadmap for this article.

To enhance navigation through this paper, we provide a roadmap (Figure 1), with a detailed structure as follows: In section II, we briefly introduce factor graphs and their main solving approaches. Then, in section III, we provide a tutorial on graph optimization algorithms as the basis for state-of-the-art factor graph libraries employed in robotic perception, covering the different approaches and related aspects. In section IV, we give a concise introduction to optimal control and its key techniques. After that, section V presents a comprehensive review of existing literature that utilizes factor graphs for optimal control, followed by a discussion of future research perspectives in section VI. Finally, section VII concludes this article.

II. FACTOR GRAPHS IN A NUTSHELL

A. INTRODUCTION TO FACTOR GRAPHS

Consider a function $J(X)$ of a set of variables $X = \{x_1, \ldots, x_n\}$, that factors into a product of r local functions or "factors" $\phi_j(X_j)$ of a subset X_j of X:

$$J(X) = \prod_{j=1}^{r} \phi_j(X_j) = \phi_1(X_1) \cdot \phi_2(X_2) \cdots \phi_r(X_r) \quad (1)$$

A factor graph expresses the structure of this factorization. The factor graph $G = \{X, \mathcal{F}, \mathcal{E}\}$ is characterized as an undirected bipartite graph, consisting of two sets of nodes: a set of variable nodes denoted by X and a set of factor nodes denoted by $\mathcal{F}$. The set $\mathcal{E}$ involves the edges connecting variable nodes and factor nodes.

Variable nodes may represent various entities, such as a robot's pose at different time steps or landmarks. In factor graph-based control tasks, input commands and system states can be represented as variable nodes. Factor nodes, in turn, are functions of the variables connected to them.

An example of a factor graph for 6 variable nodes and 7 factor nodes is illustrated in Figure 2, where, $X = \{x_1, \ldots, x_n\}$, $\mathcal{F} = \{\phi_1, \ldots, \phi_r\}$, and $\mathcal{E} = \{\varepsilon_{ij} \mid i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, r\}\}$, with $n = 6$ and $r = 7$. Consequently, the factor graph represents the factorization of

$$\begin{aligned} J(X) &= \prod_{j=1}^{r} \phi_j(X_j) \\ &= \phi_1(x_1) \cdot \phi_2(x_2, x_3, x_6) \cdot \phi_3(x_1, x_2) \\ &\quad \cdot \phi_4(x_2, x_3) \cdot \phi_5(x_1, x_4) \cdot \phi_6(x_2, x_4) \cdot \phi_7(x_3, x_4) \quad (2) \end{aligned}$$

Moreover, the factors can be classified into different types based on the number of variables they represent. These types include unary factors (e.g., ϕ_1), binary factors (e.g., ϕ_3 to ϕ_7), ternary factors (e.g., ϕ_2), and so on.

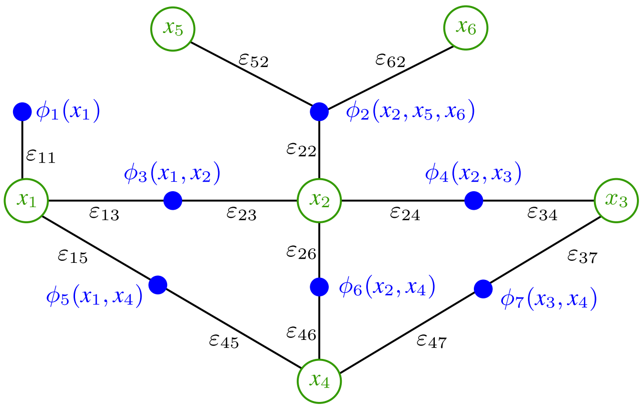


FIGURE 2. An illustrative example of a factor graph with 6 variable nodes depicted as green circles and 7 factor nodes represented as blue dots. Edges between nodes are visualized in black.

B. FACTOR GRAPHS: A UNIFYING FRAMEWORK FOR MODELING UNCERTAINTY

Graphical models, including factor graphs, provide a framework for representing uncertainty and performing probabilistic inference. Factor graphs are extensively used in probabilistic graphical modeling and one of their key advantages is their ability to serve as a unifying framework for both Bayesian networks and Markov random fields [21], which are two widely employed models in probabilistic reasoning.

Factor graphs have widespread applications in various fields, including error-correction coding theory, detection and estimation, wireless networking, artificial intelligence, control systems, and robotics. In robotics, they play a crucial role in tasks such as simultaneous localization and mapping (SLAM), structure from motion (SfM), bundle adjustment for 3D scene reconstruction, or more general situation awareness models such as S-Graphs [22].

C. FACTOR GRAPHS: A FOUNDATION FOR PROBABILISTIC INFERENCE

Statistical inference on probabilistic graphs (including factor graphs) involves several tasks or queries that aim at understanding the relationships between random variables based on observed data (evidence) [23]. Here are some common **inference tasks**:

1) Marginalization: This involves computing the probability distribution of a subset of variables in the graph, independent of the remaining variables. It essentially "marginalizes out" the influence of the remaining variables, allowing a focus on a specific set.
2) Conditional Inference: This task computes the conditional probability distribution of one set of variables given another set, considering observed evidence. It helps to understand how the values of one set of variables influence another set, given the available evidence.
3) Maximum A Posteriori (MAP) Inference: This determines the most probable values of all variables in the graph, given observed evidence. It essentially finds the maximum likelihood of variables that explain the given data.

In order to perform queries or tasks on factor graphs, various **inference techniques** can be used, including:

1) Sampling Methods
Sampling-based inference techniques are used to approximate the posterior distribution. These methods rely on generating a large number of random samples from the underlying probability distribution. Well-known examples include Markov chain Monte Carlo (MCMC) or Gibbs sampling [17], [24].
While powerful, sampling-based methods can be computationally demanding. The accuracy of the approximation depends on the number of generated samples, which often requires a significant amount of computational resources. Additionally, assessing convergence, which ensures that the samples accurately reflect the true distribution, can be challenging, especially for complex or high-dimensional problems like SLAM [25]. These limitations make sampling-based

methods less suitable for real-time inference tasks such as those encountered in SLAM.

2) Message Passing Methods
Message passing algorithms (MPAs) are a powerful approach to perform inference on factor graphs. These algorithms work by iteratively passing messages between nodes. Each node updates its belief about a variable based on information received from its connected neighbors in the graph [26], [27]. This distributed approach enables simultaneous computation of marginal functions for all variables by processing messages across all edges of the factor graph.
A fundamental MPA, called sum-product (also known as belief propagation or BP), is widely used for marginalization tasks. It estimates the probability distribution of each variable by considering incoming messages from connected factors. Max-product, a variant of BP, computes the maximum a posteriori (MAP) estimate, which corresponds to the most likely configuration of variables [28], [29].
While the standard BP offers an efficient algorithm, it only guarantees convergence for tree-structured graphs (without loops). For graphs with loops (loopy graphs), BP provides an approximate solution and convergence is not guaranteed. To address this limitation, researchers have proposed several extensions of BP, including generalized BP [30], [31], tree weighted BP [32], and affinity propagation [33]. These variants aim to improve the accuracy and convergence properties of BP for loopy graphs, as detailed in the review by Santana et al. [27]
It is important to note that MPAs have also been successfully applied to control problems (see [19], [34], [35], [36], [37], [38], [39], [40]). However, this paper focuses on control problems that exploit a state-of-the-art framework for SLAM and robot situational awareness, namely the graph optimization approach described below.

3) Graph Optimization Approach
In SLAM and robotic situational awareness, the inference problem on factor graphs is often formulated as an optimization problem and solved using numerical optimization algorithms [1]. This approach, known as graph optimization, is the primary method employed by state-of-the-art SLAM libraries, including:

- GTSAM [41]
- g2o [3], [42]
- Ceres Solver [11]
- SE-Sync [43]
- SLAM++ [44]
- SRRG2 [9]

A comparison of these libraries can be found in Juric et al. [45]. Since graph optimization is the primary approach used in SLAM and situational awareness tasks, we provide a tutorial on this approach in the following section and then present an overview of control techniques that utilize graph optimization.

III. FACTOR GRAPH OPTIMIZATION (TUTORIAL)

Graph optimization is a broad and well-studied subject. This tutorial provides a concise overview of the most important concepts in graph optimization, as this is essential to understand their application to optimal control problems in the following. To clarify the structure of this tutorial, a flowchart is shown in Figure 3. We begin by considering Maximum A Posteriori (MAP) inference as an optimization problem in § III-A. Then, we explain how this problem can be reformulated as a least squares optimization in § III-B. Depending on whether the factors are linear or can be easily linearized with a good linearization point, we can use direct methods to solve the resulting optimization problem, which are discussed in § III-C. For more complex cases, iterative solvers are used, as presented in the context of modern SLAM in §III-D. Finally, regardless of the solver type used, a linear solver is required, so we cover various aspects in this regard in § III-E.

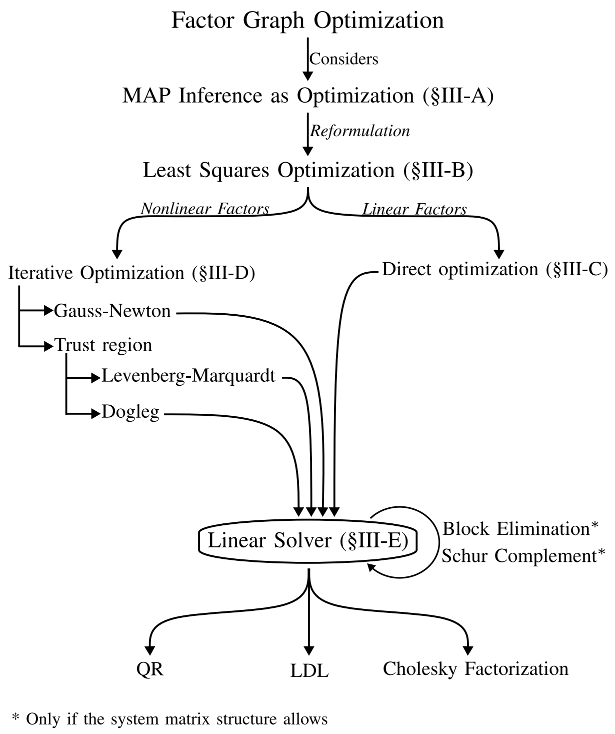

Factor Graph Optimization
Considers
MAP Inference as Optimization (§III-A)
Reformulation
Least Squares Optimization (§III-B)
Nonlinear Factors
Linear Factors
Iterative Optimization (§III-D)
Direct optimization (§III-C)
Gauss-Newton
Trust region
Levenberg-Marquardt
Dogleg
Linear Solver (§III-E)
Block Elimination*
Schur Complement*
QR
LDL
Cholesky Factorization
* Only if the system matrix structure allows

FIGURE 3. Optimization approach for factor graphs: A Flowchart.

A. MAP INFERENCE AS OPTIMIZATION PROBLEM

This section explores how Maximum A Posteriori (MAP) inference in factor graphs can be formulated as an optimization problem. From a probabilistic standpoint, the factor graph represents the factorization of a global joint (unnormalized) probability density function as

$$\begin{aligned} J_1(X) &= \prod_{j=1}^{r} \phi_j(X_j) \\ &= \phi_1(X_1)\cdot\phi_2(X_2)\cdots\phi_r(X_r), \end{aligned} \tag{3}$$

where r is the number of factors in the graph. Let's assume that each factor defines an (unnormalized) Gaussian probability density function as

$$\begin{aligned}\phi_j\left(X_j\right) &= c_j \exp\left(-\frac{1}{2} e_j\left(X_j\right)^{\mathrm{T}} \Omega_j\, e_j\left(X_j\right)\right)\\ &= c_j \exp\left(-\frac{1}{2}\left\|e_j\left(X_j\right)\right\|^2_{\Omega_j}\right)\end{aligned} \tag{4}$$

where exp () denotes the exponential function, c_j is a positive constant, $.^{\mathrm{T}}$ represents the transpose operator and e_j is a function of the set of variables X_j linked to that factor.

In addition, the information matrix Ω_j is positive definite and symmetric. This matrix is often defined as the inverse of the covariance matrix Σ_j of the corresponding factor. Therefore, the higher the numerical value of the information matrix, the more reliable that factor is and thus the more it contributes to determining the values of X. For instance, in a SLAM problem, such a factor ϕ_j could represent Gaussian priors or likelihood factors derived from measurements corrupted by zero-mean, normally distributed noise [1].

Since MAP inference for factor graphs implies maximizing the joint probability distribution represented by the graph, it can be mathematically expressed as

$$X^{\mathrm{MAP}} = \underset{X}{\operatorname{argmax}} \prod_{j=1}^{r} \phi_j(X_j) \tag{5}$$

As scaling the optimization problem with positive constants does not affect the solution of the optimization problem [46], we can simplify the MAP inference as

$$X^{\mathrm{MAP}} = \underset{X}{\operatorname{argmax}} \prod_{j=1}^{r} \exp\left(-\frac{1}{2}\left\|e_j\left(X_j\right)\right\|^2_{\Omega_j}\right) \tag{6}$$

As it becomes obvious, assuming that the factors represent Gaussian uncertainty models, MAP inference in factor graphs finally involves solving an unconstrained optimization problem. In the next section, we will explain how this optimization problem can be further reformulated as a least squares optimization problem.

B. MAP INFERENCE AS LEAST SQUARES OPTIMIZATION

The MAP optimization problem in (6) can be conveniently converted into a least squares problem using the following steps:

1) Logarithmic transformation: We apply the logarithm function (where it does not affect the solution) to transform the product of exponentials into a sum of the least squares.
2) Maximization to Minimization: To switch from maximizing a cost function to minimizing it, we multiply the entire expression by -2.

These steps can be presented mathematically as

$$\begin{aligned}X^{\mathrm{MAP}} &= \underset{X}{\operatorname{argmax}}\ \log\left(\exp\left(-\frac{1}{2}\left\|e_1(X_1)\right\|^2_{\Omega_1}\right)\right.\\ &\qquad \left.\cdots\exp\left(-\frac{1}{2}\left\|e_r(X_r)\right\|^2_{\Omega_r}\right)\right)\\ &= \underset{X}{\operatorname{argmax}}\ -\left[\frac{1}{2}\left\|e_1(X_1)\right\|^2_{\Omega_1} + \cdots + \frac{1}{2}\left\|e_r(X_r)\right\|^2_{\Omega_r}\right]\\ &= \underset{X}{\operatorname{argmin}}\ \underbrace{\sum_{j=1}^{r}\left\|e_j(X_j)\right\|^2_{\Omega_j}}_{J_2(X)}\end{aligned} \tag{7}$$

This reformulation leads to an unconstrained least squares problem, a well-studied area in optimization with numerous established methods [47], [48]. However, for practical applications in SLAM and situational awareness, specific optimization techniques have been derived and implemented in modern SLAM libraries such as GTSAM, g2o and Ceres Solver. These libraries typically leverage advanced optimization techniques such as the Gauss-Newton (GN) or Levenberg–Marquardt (LM) method to perform the back end of the SLAM task at high accuracy.

Example 1: Localization Using Factor Graphs

Consider a robot localization problem where we aim to estimate the pose of a moving robot, denoted as x_k at the time step k. The robot detects a landmark, represented by l, which is an identifiable and stable feature within the environment, crucial for accurate localization. Landmarks can be either natural or artificial. In this example, the landmark is observed at both time steps $k-1$, and k. In addition, we denote $\omega_.$ as a random variable following a normal distribution $\mathcal{N}$, where $\Sigma_.$ is the covariance matrix. In addition, the semicolon ";" is used to distinguish between variable nodes and parameters in the model.

For this problem, we assume the following:

1) **Process Model**: The robot's motion is described by a process model:

$$x_k = f_p(x_{k-1}; u_{k-1}) + \omega_p, \quad \omega_p \sim \mathcal{N}(0, \Sigma_p) \tag{8}$$

Here, u_{k-1} represents the control input at the previous time step, which is given and thus considered as a parameter.

2) **Measurement Model:** The robot is equipped with a sensor that can measure the position of the landmark when detected, modeled as:

$$z_k = h(x_k, l) + \omega_m, \quad \omega_m \sim \mathcal{N}(0, \Sigma_m) \tag{9}$$

The input z_k is also given by the sensor and thus considered as parameter.

3) **Initial Position Measurement**: The initial position of the robot is measured directly by a sensor, modeled as:

$$\mu_{x0} = x_0 + \omega_i, \quad \omega_i \sim \mathcal{N}(0, \Sigma_i) \tag{10}$$

The functions e_j in (6) in this case represent the differences between the predicted state from the process model and the actual state or between the predicted measurement and the observed values and will therefore be called "error functions" in the following. These error functions, along with their corresponding information matrices, are defined as follows:

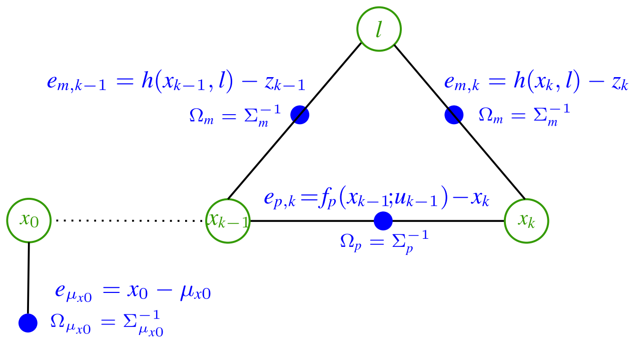
$$\begin{aligned} e_{p,k}(x_{k-1}, x_k; u_{k-1}) &= f_p(x_{k-1}; u_{k-1}) - x_k, \quad \Omega_p = \Sigma_p^{-1}, \\ e_{m,k}(x_k, l; z_k) &= h(x_k, l) - z_k, \quad \Omega_m = \Sigma_m^{-1}, \\ e_{\mu_{x0}}(x_0; \mu_{x0}) &= x_0 - \mu_{x0}, \quad \Omega_{\mu_{x0}} = \Sigma_{\mu_{x0}}^{-1}, \end{aligned} \tag{11}$$

The set of variables defining the variable nodes is $X = \{x_0, x_1, \ldots, x_k, l\}$ and the corresponding factor graph is shown in Figure 4. The optimization problem formulated by this factor graph can be expressed as:

$$X^{\text{MAP}} = \underset{X}{\operatorname{argmin}}\, J_{ex1},$$

where

$$J_{ex1} = \|e_{\mu_{x0}}\|^2_{\Omega_{\mu_{x0}}} + \sum_{i=k-1}^{k} \|e_{m,i}\|^2_{\Omega_m} + \sum_{j=1}^{k} \|e_{p,j}\|^2_{\Omega_p} \tag{12}$$



l
$e_{m,k-1} = h(x_{k-1}, l) - z_{k-1}$
$\Omega_m = \Sigma_m^{-1}$
$e_{m,k} = h(x_k, l) - z_k$
$\Omega_m = \Sigma_m^{-1}$
$e_{p,k} = f_p(x_{k-1}; u_{k-1}) - x_k$
$\Omega_p = \Sigma_p^{-1}$
x_0
x_{k-1}
x_k
$e_{\mu_{x0}} = x_0 - \mu_{x0}$
$\Omega_{\mu_{x0}} = \Sigma_{\mu_{x0}}^{-1}$

FIGURE 4. An illustrative example of a factor graph representing Example 1. The factors are depicted as blue circles, where each factor $\phi_j = c_j \exp\left(-\frac{1}{2}\|e_j\|^2_{\Omega_j}\right)$ is defined with the function e_j and the information matrix Ω_j.

The next section examines methods for solving this optimization problem, distinguishing between direct and iterative approaches.

C. DIRECT OPTIMIZATION

Direct optimization methods become applicable when the error function is linear or can be effectively linearized around a suitable point. Here, we consider a scenario involving nonlinear error functions with an appropriate linearization, denoted as $\widetilde{X}$. For the sake of clarity, we represent the error function as a function of all variables, $e_j(X)$, which can be linearized using the first-order Taylor series as

$$\begin{aligned} \|e_j(X)\|^2_{\Omega_j} &= \|e_j(\widetilde{X} + \Delta X)\|^2_{\Omega_j} \\ &\approx \|e_j(\widetilde{X}) + \mathcal{J}_{e_j}(\widetilde{X})\Delta X\|^2_{\Omega_j} = \|\mathrm{A}_j \Delta X - \mathrm{b}_j\|^2_2, \end{aligned} \tag{13}$$

where $\mathrm{A}_j = \Omega_j^{\frac{1}{2}} \mathcal{J}_{e_j}(\widetilde{X})$, $\mathrm{b}_j = -\Omega_j^{\frac{1}{2}} e_j(\widetilde{X})$, and $\mathcal{J}_{e_j}(\widetilde{X})$ denotes the Jacobian matrix of the error function $e_j(X)$ evaluated at $\widetilde{X}$, *i.e.*, $\mathcal{J}_{e_j}(\widetilde{X}) = \frac{\partial e_j(X)}{\partial X}\big|_{X=\widetilde{X}}$.

This linearization allows us to approximate the solution of the overall MAP problem (7) as

$$X^{\text{MAP}} \approx \widetilde{X} + \Delta X_d, \tag{14}$$

where the update step ΔX_{d} minimizes the approximated error terms. It can be written as

$$\Delta X_{\mathrm{d}} = \underset{\Delta X_{\mathrm{d}}}{\operatorname{argmin}} \sum_{j=1}^{r} \|\mathrm{A}_j \Delta X_{\mathrm{d}} - \mathrm{b}_j\|^2_2 = \underset{\Delta X_{\mathrm{d}}}{\operatorname{argmin}} \|\mathrm{A}\Delta X_{\mathrm{d}} - \mathrm{b}\|^2_2, \tag{15}$$

where A and b are composed of all A_j and b_j in the form of a "vertical stacking" as shown in (19). Note that the update step ΔX_{d} belongs to the class of second-order optimization methods.

To find the optimal update step, the stationary point condition is applied: *The gradient of the cost function must be zero at the minimum point*. Applying this condition to the optimization problem in (15) leads to the linear system

$$\nabla_{\Delta X} \|\mathrm{A}\Delta X_{\mathrm{d}} - \mathrm{b}\|^2_2 = 0 \Rightarrow \mathrm{A}^{\mathrm{T}}\mathrm{A}\Delta X_{\mathrm{d}} - \mathrm{A}^{\mathrm{T}}\mathrm{b} = 0, \tag{16}$$

where $\mathrm{A}^{\mathrm{T}}\mathrm{A}$ is called the information matrix of the system. The linear system ($\mathrm{A}^{\mathrm{T}}\mathrm{A}\Delta X_{\mathrm{d}} = \mathrm{A}^{\mathrm{T}}\mathrm{b}$) can be efficiently solved using various methods, such as Cholesky and QR factorization, as detailed in § III-E. In addition, GTSAM can use LDL factorization to efficiently solve the system ($\mathrm{A}\Delta X_{\mathrm{d}} = \mathrm{b}$) without computing the information matrix of the system.

In summary, a direct solver handles the MAP problem by

1) linearizing the error terms and compose the overall matrix A and vector b
2) computing the update step ΔX_{d} by solving either $\mathrm{A}^{\mathrm{T}}\mathrm{A}\Delta X_{\mathrm{d}} = \mathrm{A}^{\mathrm{T}}\mathrm{b}$ or $\mathrm{A}\Delta X_{\mathrm{d}} = \mathrm{b}$,
3) updating the variables by adding ΔX_{d} to the linearization point.

D. ITERATIVE OPTIMIZATION

If the error function is highly nonlinear, as is typically the case in SLAM problems, finding an appropriate linearization point for direct optimization might become challenging. In this case, iterative optimization algorithms are the preferred approach for solving these nonlinear problems. These algorithms fall under the umbrella of unconstrained optimization.

The basic idea of iterative optimization is that the algorithm starts with an initial guess, ideally close to the optimal point, and then calculates the update step in the direction of the optimal point until it meets the stopping criteria, which can be based on the change in the error function, a maximum number of iterations, or a combination of both.

Algorithms can be broadly classified into two categories: gradient-based and gradient-free algorithms. Gradient-based algorithms use first, second, or possibly higher-order derivatives to update the solution efficiently. Examples include methods like Newton's Method and Gradient Descent [49], [50], [51]. On the other hand, gradient-free algorithms do not rely on derivatives to make updates. Examples of gradient-free methods include the Nelder-Mead algorithm [52], genetic algorithms [53], and simulated annealing [54]. However, gradient-free methods are generally slower than gradient-based approaches [55] and are therefore typically not considered suitable for solving MAP problems. As a result, we focus on gradient-based methods with second-order derivatives in this article.

Second-order optimization methods leverage the second derivative (the Hessian matrix) of the cost function to compute the update step, enabling faster convergence. Although the computation of the Hessian matrix can be challenging, it can be easily approximated in the context of factor graphs by considering a linearized version of the error terms, similar to direct solvers. Therefore, let us consider an approximation of the cost function of MAP problems as

$$J(X^i) = \sum_{j=1}^{r} \left\| e_j(X^i + \Delta X^i_{\text{so}}) \right\|^2_{\Omega_j}$$
$$\approx \underbrace{\sum_{j=1}^{r} \left\| \underbrace{\Omega_j^{\frac{1}{2}} \mathcal{J}_{e_j}(X^i)}_{\mathrm{A}^i_j} \Delta X^i_{\text{so}} + \underbrace{\Omega_j^{\frac{1}{2}} e_j(X^i)}_{-\mathrm{b}^i_j} \right\|^2_2}_{\widetilde{J}_2(\Delta X_{\text{so}})}. \tag{17}$$

For second-order optimization, two general steps are followed to derive the update step. First, the cost function is approximated using the second-order Taylor series. Then, the stationary point condition is applied to find the update step. In the literature, the update step appears in two forms due to different procedures how the approximated costs are handled. For the sake of distinction, we call these procedures the stacking and the accumulating method:

- Stacking Method:
  The GTSAM library and related works such as [56], [57], and [58] consider this approach. In the stacking method, the approximated cost function is represented as

$$\widetilde{J}_2(\Delta X^i_{\text{so}}) = \sum_{j=1}^{r} \left\| \mathrm{A}^i_j \Delta X^i_{\text{so}} - \mathrm{b}^i_j \right\|^2_2 = \left\| \mathrm{A}^i \Delta X^i_{\text{so}} - \mathrm{b}^i \right\|^2_2 \tag{18}$$

  where A^i and b^i result from a vertical stacking of A_j and b_j, namely

$$\mathrm{A}^i = \begin{bmatrix} \mathrm{A}^i_1 \\ \vdots \\ \mathrm{A}^i_r \end{bmatrix}, \quad \text{and} \quad \mathrm{b}^i = \begin{bmatrix} \mathrm{b}^i_1 \\ \vdots \\ \mathrm{b}^i_r \end{bmatrix}. \tag{19}$$

  Hence, applying the stationary conditions yields the linear system for the update step as

$$\mathrm{A}^{i\mathrm{T}} \mathrm{A}^i \Delta X^i_{\text{so}} = \mathrm{A}^{i\mathrm{T}} \mathrm{b}^i \tag{20}$$

  This form allows the determination of the update step either by solving $\mathrm{A}^{i\mathrm{T}} \mathrm{A}^i \Delta X^i_{\text{so}} = \mathrm{A}^{i\mathrm{T}} \mathrm{b}^i$ or equivalently by solving $\mathrm{A}^i \Delta X^i_{\text{so}} = \mathrm{b}^i$. In the GTSAM library and related works, the variable elimination method is widely applied to solve the system $\mathrm{A}^i \Delta X^i_{\text{so}} = \mathrm{b}^i$.
- Accumulating Method:
  The g2o and SRRG2 libraries, and other related works such as [59], [60], use an expanded form for the approximated cost

$$\widetilde{J}_2(\Delta X^i_{\text{so}})$$
$$= \sum_{j=1}^{r} \Big(\Delta X^{i\ \mathrm{T}}_{\text{so}} \overbrace{\mathcal{J}_{e_j}(X^i)^{\mathrm{T}} \Omega_j \mathcal{J}_{e_j}(X^i)}^{\mathrm{H}^i_j} \Delta X^i_{\text{so}}$$
$$+ 2 \underbrace{\Big(\mathcal{J}_{e_j}(X^i)^{\mathrm{T}} \Omega_j e_j(X^i) \Big)^{\mathrm{T}}}_{-\mathfrak{b}^i_j} \Delta X^i_{\text{so}} + e_j(X^i)^{\mathrm{T}} \Omega_j e_j(X^i) \Big) \tag{21}$$

  By applying the stationary point condition, the update step can be described as:

$$\underbrace{\sum_{j=1}^{r} \mathrm{H}^i_j}_{\mathrm{H}^i_{\text{so}}} \Delta X^i_{\text{so}} = \underbrace{\sum_{j=1}^{r} \mathfrak{b}^i_j}_{\mathfrak{b}^i_{\text{so}}}, \tag{22}$$

  where we can see that the approximated Hessian and gradient are accumulated. This linear system can be solved efficiently using factorization algorithms like QR, Cholesky, and LDL factorization.

After exploring different representations of the update step in the second-order optimization, various algorithms have been developed to efficiently find the optimal point. These include methods like Gauss-Newton, which is usually faster than other methods but requires a full-rank H^i_{so}. In contrast, Trust Region methods, such as Levenberg-Marquardt [61], [62] and Dogleg [63], [64], offer greater stability of conversion at the cost of typically requiring more iterations to converge.

E. LINEAR SOLVERS

As discussed earlier, solving linear systems of equations is crucial for many of the previously described optimization problems as applied in robotic perception tasks such as, *e.g.*, in SLAM. Here, we will discuss several aspects of linear system solvers in this context.

1) COEFFICIENT MATRIX AND SOLVER TYPES

The type of solver best suited for a specific robotic perception problem depends on the properties of the coefficient matrix.

For a linear system defined in general as

$$CX = d, \tag{23}$$

C is called the coefficient matrix or system matrix and d is called the residual vector. Different properties of the coefficient matrix play important roles in choosing the algorithm and type of solver. Primary properties include:

- Sparsity: The system is called sparse if many of the entries in the coefficient matrix are zero, while it is called dense in the opposite case. Therefore, the solving algorithm treats a sparse matrix differently from a dense matrix in terms of focused entries, storage, memory demand and computational process. The sparse matrices are more common in robotic perception problems.
- Dimensions: The coefficient matrix can be square or rectangular—either short (more columns than rows) or tall (more rows than columns). Some solvers require specific dimensions, such as the Cholesky factorization, which requires the coefficient matrix to be square.
- Structure: If the coefficient matrix has a certain structure, such as easily invertible blocks, techniques like the Schur complement [65], [66] or block elimination [67], [68] can be employed to reduce the size of the system and simplify the solution process.

The solver can be divided into two main types:

- Iterative solvers: These are preferred for very large, sparse systems due to their lower memory requirements. However, their performance is problem-dependent, and the number of iterations needed to converge is generally unpredictable. In addition, preconditioning is often required. The preconditioning matrix is a customized matrix transformation that transforms the original linear system into one that produces the same solution but with better convergence speed [69].
- Direct solvers: return an exact solution by decomposing (factorizing) the coefficient matrix into simpler submatrices, leading to a simpler solution process [9]. Because of their efficiency, direct solvers are the preferred choice for graph optimization problems. We'll explore further details in the following section.

2) DIRECT SOLVERS FOR GRAPH OPTIMIZATION

Factor graph libraries typically utilize direct solvers customized for sparse matrices. Here, we explore some common factorization methods:

- Cholesky factorization:
  This requires the coefficient matrix to be symmetric and positive definite [70]. In this method, the coefficient matrix is decomposed into the product of a lower triangular matrix and its conjugate transpose, $C = LL^*$. The system is then solved in two steps: solving $Ly = d$ by forward substitution and then solving $L^*X = y$ by backward substitution.
- LDL factorization:
  LDL factorization is a variant of Cholesky factorization that relaxes some of its constraints. Unlike Cholesky factorization, LDL factorization does not require the matrix to be positive definite but only symmetric. LDL decomposes the matrix into $C = LDL^*$, where D is a diagonal matrix [71].
- QR-factorization:
  This provides a more accurate and numerically stable alternative to Cholesky factorization [1], particularly suited for cases where the matrix does not meet the requirements of being positive definite or square. In QR factorization, the coefficient matrix is decomposed into an orthogonal matrix (Q) and an upper triangular matrix (R):
  $C = QR$ (for square coefficient matrices)
  $C = Q\begin{bmatrix} R \\ \mathbf{0} \end{bmatrix}$(for tall coefficient matrix).
  Since the inverse of an orthogonal matrix is its transpose, solving the linear system via QR factorization can be done using back substitution of the linear system $RX = Q^{\mathrm{T}}d$ or $\begin{bmatrix} R \\ \mathbf{0} \end{bmatrix} X = Q^{\mathrm{T}}d$.
  Common methods for computing QR factorization include the Gram-Schmidt process, Householder reflections, and Givens rotations (details [68], [72, §5.1]). Moreover, QR factorization has been utilized in the Variable Elimination algorithms within the GTSAM Library to solve linear systems efficiently [1], [73].

3) EFFICIENT LINEAR SOLVER IMPLEMENTATIONS

Factor graph libraries leverage linear solver libraries that provide optimized algorithms for efficiently solving linear systems, especially those with large sparse structures common in optimization problems. Some popular libraries include:

- CHOLMOD [74]
- CSparse [75]
- SuiteSparseQR [76]
- Eigen [77]
- LAPACK [78]

For a more comprehensive list, we refer to [79, §13].

4) VARIABLE ELIMINATION IN GTSAM LIBRARY

Let us revisit the linear system corresponding to SLAM under the GTSAM library:

$$\mathrm{A}^i \Delta X^i = \mathrm{b}^i, \tag{24}$$

where A^i is a sparse-block matrix. Each column of the matrix A^i corresponds to a specific variable in the SLAM problem, while each row represents a single factor in the SLAM problem. The elements within a row essentially form the Jacobian of the factor with respect to the variables.

The library exploits the sparsity of this matrix through a variable elimination process. This section provides a high-level overview of the variable elimination method. Refer to [1, Chapter 3] for a more detailed explanation.

Assume the vector X is ordered in a way that minimizes fill-in during the elimination process. The elimination proceeds sequentially, eliminating variables one by one from the first element to the last in the ordered vector X. Here's a breakdown of each step:

1) Identify the submatrix $\left[\underline{\mathrm{A}}^{i}_{x_j} |\ \underline{\mathrm{b}}^{i}\right]$ corresponding to the variable to be eliminated. This submatrix is extracted from A^{i} and includes the variable to be eliminated and the variables connected to it (excluding the previously eliminated variables). The rows of the submatrix correspond to the factors connected to that variable.
2) Perform QR factorization on $\left[\underline{\mathrm{A}}^{i}_{x_j} |\ \underline{\mathrm{b}}^{i}\right]$.
3) Update the linear system $\left[\mathrm{A}^{i} |\ \mathrm{b}^{i}\right]$ by removing the rows corresponding to the factors considered in the first step and appending the factorized matrix from the previous step. Note that the Q matrix from the QR factorization performed in the last step is dropped because its norm is 1, and thus it does not alter the values of the norms involved.

After all variables are eliminated, the updated matrix will be a square upper triangular matrix, and the corresponding linear system can be solved using backward substitution. Additionally, the graphical representation of the new linear system forms a chordal Bayes net, which can be used to generate a Bayes tree, allowing incremental updates to the factor graphs [73].

5) HANDLING NEW MEASUREMENTS AND UPDATING THE LINEAR SYSTEM

In the GTSAM library, three approaches are used to handle the new measurements to update the linear system.

1) Square Root SAM ($\sqrt{\text{SAM}}$) [80]: This method updates the information matrix when new measurements arrive and then factorizes it completely. While this is computationally expensive, it provides the most accurate solution.
2) Incremental Smoothing and Mapping (iSAM) [81], [82]: When a new measurement arrives, the corresponding $\left[\mathrm{A}_j | b_j\right]$ are added to the previously factorized matrix and the Givens rotations are employed to zero out the new row. Periodic variable reordering is performed to avoid unnecessary fill-in in the factor matrix. This is combined with re-linearization and batch factorization to improve the estimation in the case of nonlinear measurement functions.
3) iSAM2 [56], [83]: utilizes a Bayes tree constructed from the chordal Bayes net obtained through variable elimination (as mentioned earlier). This allows efficient handling of new measurements:
    a) Isolating the impact: When a new measurement arrives, iSAM2 identifies the smallest subtree within the Bayes tree that is directly affected by the new information. This minimizes the computational cost of processing the update.
    b) Incorporating the new data: A temporary factor graph is created within the isolated subtree, including the new measurement as a factor. Then, variable reordering and elimination are applied to this factor graph to create a modified subtree.
    c) Reintegrating the update: The modified subtree, now reflecting the incorporated measurement, is seamlessly merged back into the original Bayes tree.

    Moreover, iSAM2 efficiently handles nonlinear factors through a technique called fluid linearization: Herein, only the subtree corresponding to the marked variables is isolated and treated with re-linearization, re-ordering, and an elimination process to generate the modified subtree. The variables are marked if the step update is higher than a predefined threshold. This approach minimizes the computational overhead needed to handle nonlinear factors, making iSAM2 efficient for real-time nonlinear applications.

In other libraries, *e.g.,* g2o, the update step uses accumulating method (*e.g.,* $\sum_{j=1}^{r} \mathrm{H}_j \Delta X = \sum_{j=1}^{r} \mathfrak{b}_j$) as presented earlier. Upon receiving a new measurement, the corresponding H_j matrix and the $\mathfrak{b}_j$ vector are calculated and accumulated into the overall coefficient matrix and residual vector before solving the updated linear system for the variable updates.

IV. CONTROL TASKS AND FACTOR GRAPHS

As described, factor graphs have been successfully employed in solving various robotic perception tasks, such as SLAM or the generation of situational awareness, using a graph optimization approach. However, their applications extend beyond these domains—they can also be used for optimal control. This section provides an overview of control theory and explores the integration of optimal control into factor graphs. In addition, it presents prominent techniques like Linear Quadratic Regulation (LQR) and Model Predictive Control (MPC) that have already been integrated in some works with factor graphs for solving robotic control tasks.

A. OVERVIEW OF CONTROL THEORY

Control engineering focuses on modeling system dynamics and designing control policies to ensure that these systems operate as desired, mainly in a closed-loop feedback structure. Its applications span various industries, including robotics [84], [85], autonomous vehicles [86], [87], aerospace [88], [89], [90], [91], and chemical processes [92], [93], [94].

Control theory offers a diverse toolbox of approaches categorized by several aspects, such as system type (linear vs. nonlinear systems [95], [96]), domain (frequency vs. time domain [97]), control strategy (open-loop vs. closed-loop [98]), and control design philosophy (classical vs. optimal [99]). While classical control methods like Proportional-Integral-Derivative (PID) control might be able to achieve stability and desired performance for linear time-invariant systems [100], [101], they are not applicable for non-linear and/or time-varying systems and are furthermore not always the most efficient or cost-effective solution, especially for complex systems with multiple constraints [102]. This is where optimal control comes in. It uses optimization techniques to determine the best control actions, taking into account both an objective function and various constraints, including system dynamics, actuator limits, safety bounds, and others [103], [104].

B. CLASSIFICATION OF OPTIMAL CONTROL METHODS

Optimal control methods can be classified based on the techniques used to solve them. For instance, Pontryagin's maximum principle utilizes a Hamiltonian function to derive necessary conditions for optimality, ultimately determining the optimal control law [105], [106]. In contrast, dynamic programming breaks the problem into smaller, manageable steps, solving each recursively [104], [107]. Moreover, nonlinear programming [108] is another technique, where the dynamic control problem is reformulated into a static optimization problem. Then nonlinear programming is applied, which can directly optimize control inputs while considering system dynamics and constraints, often yielding computationally efficient solutions [109].

Optimal control methods can also be categorized by how system dynamics are discretized. Methods such as direct collocation [110], [111], shooting methods [112], and pseudo-spectral methods [113] are commonly used to solve discretized control problems.

C. INTEGRATION OF OPTIMAL CONTROL INTO FACTOR GRAPHS

As discussed in the previous section, probabilistic robotic perception tasks, formulated as MAP inference using factor graphs, can finally be expressed as least-squares optimization problems. This problem is then solved numerically by iteratively updating the system through solving a linear system of equations, constructed depending on the specific optimization algorithm employed. Similarly, optimal control problems (*e.g.,* MPC) can also be formulated such that ultimately a system of linear equations needs to be solved to update the solution of the optimization problem. However, the construction of the linear system and the treatment of constraints vary based on the optimization algorithms used, which remains an active area of research.

Given that both robotic perception and control tasks rely on optimization and solving linear systems of equations, integrating these tasks within a unified and factor graph-based framework seems feasible and advantageous. This unified approach would simplify data flow, reduce the need for separate optimization solvers, and enhance overall system efficiency. An example of this integration, inspired by [114], is illustrated in Figure 5. The framework unifies perception (estimation) and control tasks of a robot within a single factor graph.

Let us assume that the robot is currently in state x_k. In the estimation part, which considers the states in the past, factor nodes represent the probabilistic measurement and process models, which are used to compute the current state of the robot, as discussed in Example 1. On the other hand, the control parts consider future states $\{x_{k+1}, x_{k+2}, \cdots\}$ over a certain horizon that the robot intends to reach using suitable control actions $\{u_k, u_{k+1}, \cdots\}$. **This segment of the factor graph incorporates constrained dynamics factors (depicted in yellow in Figure 5), which are typically treated as deterministic models in the literature, in contrast to the probabilistic process models used in estimation.** Additionally, cost factors represent the objective function of the optimal control problem. Finally, constraint factors (depicted in green in Figure 5) define the relationship between predicted states and environmental landmarks. These constraints may take the form of inequality constraints, such as obstacle avoidance, where the robot's distance from an obstacle must exceed a predefined threshold, or equality constraints, such as reaching a specific target point. The treatment of constraint factors in control varies, and the review section examines different approaches to address these constraints.

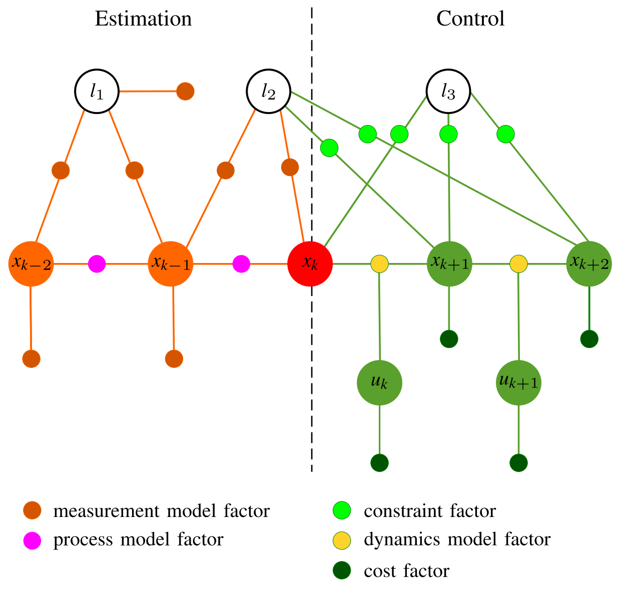


FIGURE 5. Unified framework for estimation and control tasks using a factor graph. The estimation part is on the left of the dashed line, while the control part is on the right.

D. PROMINENT OPTIMAL CONTROL TECHNIQUES INTEGRATED WITH FACTOR GRAPHS

Two prominent optimal control techniques that have already been integrated with factor graphs in the literature are Linear Quadratic Regulation (LQR) and Model Predictive Control (MPC).

1) LINEAR QUADRATIC REGULATION (LQR)

LQR is a state-feedback control strategy designed to compute the optimal control for a linear system, minimizing a quadratic cost function. The finite-horizon, discrete-time LQR formulation in general is as follows [99, Chapter 2]. Assume that the system is initially in the known state x_0 and consider a control horizon of N time steps. The related optimization problem can then be formulated as

$$\min_{u_k} x_N{}^{\mathrm{T}} \mathrm{Q}_N x_N + \sum_{k=1}^{k=N-1} x_N{}^{\mathrm{T}} \mathrm{Q}_k x_N + \sum_{k=0}^{k=N-1} u_k{}^{\mathrm{T}} \mathrm{R}_k u_k \tag{25a}$$

$$\text{s.t. } x_{k+1} = A_k x_k + B_k u_k, \qquad k = 0, \ldots, N-1 \tag{25b}$$

Here, A_k and B_k are the system and input matrices, respectively, and Q_N, Q_k, and R_k are symmetric and positive semidefinite weighting matrices. The LQR strategy is defined as $u_k = -K_k x_k$, where the optimal gain is obtained from dynamic programming:

$$K_k = (R_k + B_k{}^T P_{k+1} B_k)^{-1} B_k{}^T P_{k+1} A_k \tag{26a}$$

$$P_k = Q_k + A_k{}^T P_{k+1} A_k - K_k{}^T B_k{}^T P_{k+1} A_k, \; P_N = Q_N \tag{26b}$$

where P_k is the symmetric and positive semidefinite solution of the Riccati differential equation.

a: LQR IMPLEMENTATION USING FACTOR GRAPHS

Factor graphs have been successfully employed to implement Linear Quadratic Regulators (LQR) [115], [116], as we will detail in the following section. In the factor graph representation of LQR, the variable nodes correspond to the system states and control inputs.

Regarding factor nodes, there are two primary types in this setup: (I) those representing the cost function terms, and (II) those encoding equality constraints, which typically capture system dynamics or enforce relationships between variables. As noted, each factor node is characterized by an error function and an information matrix.

The cost function for LQR is typically expressed as:

$$J_{\mathrm{LQR}}(x, u) = \sum_{k=1}^{N} (x_k - c_k)^T Q_k (x_k - c_k) + \sum_{k=0}^{N-1} u_k^T R_k u_k, \tag{27}$$

where the corresponding error functions of the factor nodes representing the LQR cost function are $(x_k - c_k)$ for the states and u_k for the control inputs. The corresponding information matrices are Q_k and R_k, associated with the states and control inputs, respectively. The constant c_k typically represents a desired state and can be set to zero when no specific target is given. Regarding equality constraint factor nodes, the error function is directly defined as the constraint for example $x_{k+1} - (A_k x_k + B_k u_k)$, and the information matrix Q_{eq} is assigned a large value to ensure that the constraint is strictly enforced. In theory, this matrix should be set to infinity to rigorously impose the constraint, but for numerical stability, a sufficiently large value is used in practice. The factor graph, which reflects the LQR problem with equality constraints $h_k(x, u) = x_{k+1} - (A_k x_k + B_k u_k)$, solves the following cost function:

$$J_{\mathrm{FG}}(x, u) = \sum_{k=1}^{N} \|x_k - c_k\|_{Q_k}^2 + \sum_{k=0}^{N-1} \|u_k\|_{R_k}^2 + \sum_{k=0}^{N-1} \|h_k(x, u)\|_{Q_{eq}}^2$$

Solving the factor graph for this cost function yields the values of the control inputs and states. The control gain K_k for the control strategy $u_k = -K_k x_k$ is obtained when the variable elimination method is applied [117]. Notably, K_k is inherently embedded within the QR factorization process.

Figure 6 illustrates the factor graph for an LQR problem with N states and $N-1$ control inputs. The system dynamics are modeled by the state equation

$$x_{k+1} = A_k x_k + B_k u_k$$

An equality constraint is applied on the final state x_N to ensure that it satisfies the terminal condition $G_t x_N + g_t = 0$, where $G_.$ is a matrix and $g_.$ is a vector, both of appropriate dimensions. This constraint could represent a desired target for the final state in practical applications. Furthermore, additional equality constraints, such as e_{x_1,u_1}, can be easily incorporated between variables as needed.

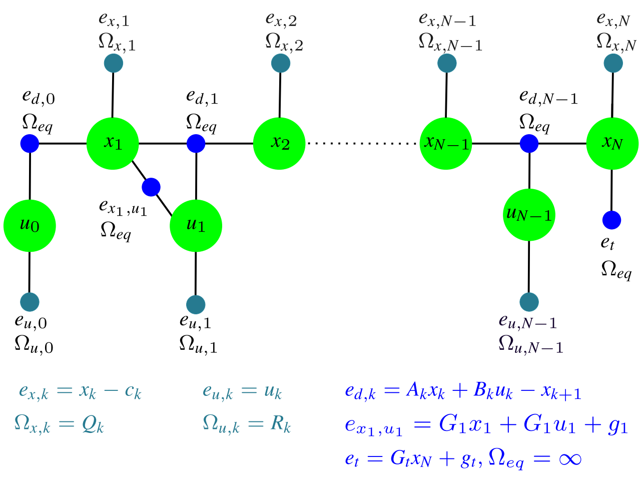


FIGURE 6. Factor graph representation of LQR with linear dynamics and equality constraints.

2) MODEL PREDICTIVE CONTROL (MPC)

MPC, also known as receding horizon control, is a well-established feedback control strategy with high performance [118], [119], [120], [121] and inherent robustness against disturbances and uncertainties [122].

MPC systematically handles state and input constraints, making it a versatile solution applicable to control various categories of systems, including Multiple-Input Multiple-Output (MIMO), nonlinear [123], [124], stochastic [20], [125], [126], [127], time-delayed, hybrid [128], [129], and distributed systems [130]. Moreover, MPC finds applications beyond control including path planning [131], trajectory generation [132], [133], [134], and the principle of optimizing over a receding time horizon can also be applied for receding horizon estimation [135], [136].

MPC can generally be divided into two types: explicit [137], [138] and implicit [139]. Explicit MPC precomputes control laws for all possible states and stores them for online retrieval, offering runtime efficiency but being limited to low-dimensional state spaces. In contrast, implicit MPC solves an optimization problem at each step, making it flexible and suitable for the integration with factor graphs. Its working principle is depicted in Figure 7. MPC utilizes the system dynamics to predict the behavior of the system over a prediction horizon N. It solves an optimization problem that minimizes a cost function subject to the system dynamics, input, and state constraints. The solution provides a sequence of input commands, with the corresponding prediction of system states over the horizon. Only the first input is applied to the system, while the other future inputs are discarded. At the next sampling instant, the optimization problem is solved again based on the updated system information, correcting any deviations from the previous prediction. This process is repeated at each sampling instant, which directly helps to compensate for disturbances and uncertainties.

Different toolkits are available to implement MPC, such as CasADi [140], AMPC [129], [141], and other options listed in [142]. These toolkits serve as interfaces to the optimization solvers, simplifying the implementation process. Additionally, MPC can be implemented directly using optimization solvers. However, this requires adherence to the specific syntax and formulation methods of each solver, making it cumbersome and time-consuming to switch from one solver to another. As a connection between Figure 7 and Figure 5, MPC starts from the current state node x_k, depicted in red. The green nodes represent the future predictions of the MPC, while the orange nodes indicate the past states associated with the estimator.

The optimization problems of MPC can be expressed in a general form as

$$\min_{Xc} J_c(Xc) \tag{28a}$$

$$\text{s.t. } h_j(Xc) = 0, \quad j = 1, \cdots, l \tag{28b}$$

$$g_j(Xc) \leq 0, \quad j = 1, \cdots, q \tag{28c}$$

where $J_c(Xc)$ is the cost and Xc involves all system states $\{x_{k+1}, x_{k+2}, \cdots\}$ and all control inputs $\{u_k, u_{k+1}, \cdots\}$. The cost function is typically a quadratic function, similar to (25a). In addition, a set of equality (28b) and inequality (28c) constraints are defined using the function h_j and g_j.

To solve the MPC problem, various constrained optimization approaches can be used. We present here the Augmented Lagrangian (AL) method [143], [144] because it has been also used in the factor graph framework.

AL tackles constrained optimization including both equality and inequality constraints by transforming it into an unconstrained problem using a Lagrangian function with

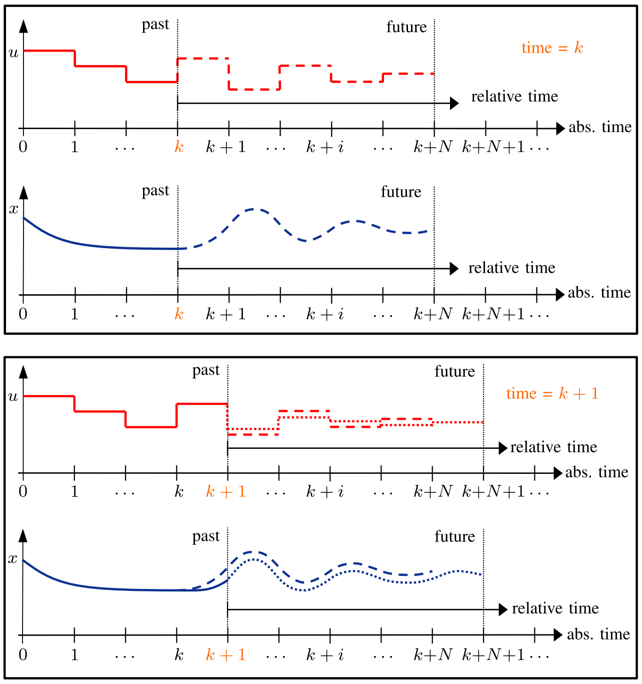


FIGURE 7. **Working principle of MPC.**

penalty terms

$$\begin{aligned} &L(X_c, \lambda, \gamma; \rho) \\ &= J_c(X_c) + \sum_{j=1}^{l} \left(\gamma_{h_j} h_j(X_c) + \rho_{h_j} \left\|h_j(X_c)\right\|_2^2\right) \\ &\quad + \sum_{j=1}^{q} \left(\lambda_{g_j} g_j(X_c) + \rho_{g_j} \left\|\max(0, g_j(X_c))\right\|_2^2\right) \end{aligned} \tag{29}$$

where $\lambda_{g_j} \geq 0$ and γ_{h_j} are the Lagrange multipliers associated with the inequality and equality constraints, respectively, and $\rho_. > 0$ is a penalty parameter (with $\rho_. = 0$, we get the Lagrangian function). The semicolon separates the variables from the parameters. The AL method iteratively solves two optimization problems: the primal and dual problem ($\max_{\lambda,\gamma} \min_{X_c} L(X_c, \lambda, \gamma; \rho)$):

- Primal Optimization Problem:
  Minimize the Augmented Lagrangian function with respect to X_c, treating the Lagrange multipliers as constants ($\min_{X_c} L(X_c; \lambda, \gamma, \rho)$). The penalty terms drive the variables to satisfy the constraints without requiring excessively high penalty parameters due to the presence of the Lagrange multiplier [48, Example 17.4]. This helps to prevent ill-conditioning.
- Dual Optimization Problem:
  Maximize the Augmented Lagrangian function with respect to the Lagrange multipliers, treating X_c as constant ($\max_{\lambda,\gamma} L(\lambda, \gamma; X_c, \rho)$). The Lagrange multipliers are updated using the gradient ascent method, with ρ as the

step size, which yields the following updating policy:

$$\gamma_{h_j}^{i+1} = \gamma_{h_j}^{i} + \rho_{h_j} h_j(X_c) \tag{30a}$$
$$\lambda_{g_j}^{i+1} = \max(0, \lambda_{g_j}^{i} + \rho_{g_j} g_j(X_c)) \tag{30b}$$

a: MPC IMPLEMENTATION USING FACTOR GRAPHS

Implementing Model Predictive Control (MPC) using factor graphs is more complex than Linear Quadratic Regulators (LQR) because it requires integrating constrained optimization algorithms within the factor graph solver. As demonstrated in the tutorial, the factor graph solver constructs a system of linear equations to encode the cost terms. Fortunately, also constraint optimization as applied in MPC primarily involves the solution of a system of linear equations, in addition to other steps. Different algorithms may yield different linear systems depending on how they handle constraints.

When building the graph for MPC, it is essential to distinguish between three types of factor nodes: equality, inequality, and cost factor nodes. Consider the following optimization problem:

$$\min_{Xc} \sum_{j=1}^{r} \left\| e_j(X_c) \right\|_{\Omega_j}^2 \tag{31a}$$
$$\text{s.t. } g_j(Xc) \leq 0, \qquad j = 1, \cdots, q \tag{31b}$$
$$h_j(Xc) = 0 \qquad j = 1, \cdots, l \tag{31c}$$

In this setup, the cost factor node represents a cost function term, such as $\left\| e_j(X_c) \right\|_{\Omega_j}^2$, which is defined by an error function and information matrix, similar to an unconstrained factor graph. The inequality factor node encodes inequality constraints, for example, $g_i(Xc)$, while the equality factor node encodes equality constraints, such as $h_i(Xc)$.

In the back-end, the constrained optimization algorithm builds a linear system while considering the influence of each type of factor node. For example, let $\mathcal{J}_f$ represent the Jacobian of the function. The Augmented Lagrangian (AL) algorithm [60] constructs the linear system — defined in (22) — as follows

- Cost Factor Node:
  $\mathrm{H}_{\mathrm{so}} = \mathrm{H}_{\mathrm{so}} + \mathcal{J}_{e_j}(X_c)^{\mathrm{T}} \Omega_j \mathcal{J}_{e_j}(X_c)$
  $\mathfrak{b}_{\mathrm{so}} = \mathfrak{b}_{\mathrm{so}} - \mathcal{J}_{e_j}(X_c)^{\mathrm{T}} \Omega_j e_j(X_c)$
- Equality Factor Node:
  $\mathrm{H}_{\mathrm{so}} = \mathrm{H}_{\mathrm{so}} + \mathcal{J}_{h_j}(X_c)^{\mathrm{T}} \rho_{h_j} \mathcal{J}_{h_j}(X_c)$
  $\mathfrak{b}_{\mathrm{so}} = \mathfrak{b}_{\mathrm{so}} - \mathcal{J}_{h_j}(X_c)^{\mathrm{T}} \left[\rho_{h_j} h_j(X_c) + \gamma_{h_j}\right]$
- Inequality Factor Node:
  $\mathrm{H}_{\mathrm{so}} = \mathrm{H}_{\mathrm{so}} + \mathcal{J}_{\hat{g}_j}(X_c)^{\mathrm{T}} \rho_{\hat{g}_j} \mathcal{J}_{\hat{g}_j}(X_c)$
  $\mathfrak{b}_{\mathrm{so}} = \mathfrak{b}_{\mathrm{so}} - \mathcal{J}_{\hat{g}_j}(X_c)^{\mathrm{T}} \left[\rho_{g_j} \hat{g}_j(X_c) + \lambda_{g_j}\right]$,
  where $\hat{g}_j = \max(0, g_j(X_c))$

After solving the linear system, other steps such as updating the Lagrange multipliers and checking the stopping criteria must be considered. These steps are handled by the optimization algorithm in the backend.

As an example, consider an MPC problem with dynamic equality constraints and box constraints on the state and input variables:

$$\min_{Xc} \sum_{k=1}^{N} (x_k - c_k)^T Q_k (x_k - c_k) + \sum_{k=0}^{N-1} u_k^T R_k u_k \tag{32a}$$
$$\text{s.t. } h_j(Xc) = f_m(x_k, u_k) - x_{k+1} = 0 \tag{32b}$$
$$\underline{u}_k \leq u_k \leq \bar{u}_k \tag{32c}$$
$$\underline{x}_{k+1} \leq x_{k+1} \leq \bar{x}_{k+1}, \tag{32d}$$

where X_c represents the vector containing all system states and control inputs, while $f_m(.)$ is the dynamics model, with $k \in 0, 1, \ldots, N-1$. The factor graph representation of this optimization problem is shown in Figure 8.

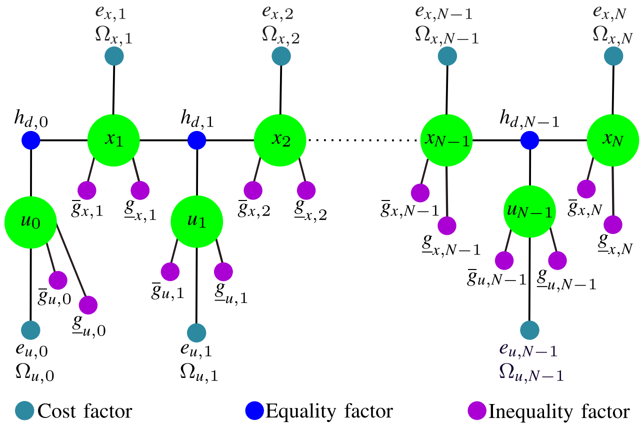


FIGURE 8. Factor graph representation of MPC with dynamics and box constraints. The terms $\bar{g}_{r,k} = r_k - \bar{r}_k$ and $\underline{g}_{r,k} = \underline{r}_k - r_k$ represent respectively the upper and lower bounds constraints for the control inputs or states.

Both LQR and MPC, when combined with factor graphs, can lead to powerful solutions to various control problems. In the next section, we will examine the works that have already used factor graphs to implement these techniques.

V. REVIEW OF OPTIMAL CONTROL LEVERAGING FACTOR GRAPHS

Both optimal control and factor graph optimization are formulated as optimization problems. However, there is a key difference: factor graphs are typically designed as unconstrained problems, whereas optimal control problems often include constraints arising from system dynamics (*e.g.,* LQR) or limitations of the physical system (*e.g.,* equality and inequality constraints in MPC).

Therefore, the main challenge in using factor graphs for optimal control lies in handling these equality and inequality constraints. This section provides an extensive review of works that employ factor graphs for optimal control implementation, detailing the types of constraints considered, the base frameworks utilized, and the specific applications addressed (see Table 1).

A. EQUALITY-CONSTRAINED CONTROL

One significant approach to handling equality constraints in factor graph-based realization of control problems is converting these constraints into soft constraints using

TABLE 1. Key works in optimal control using factor graphs (AL: Augmented Lagrangian, EM: Elimination Method, FPGA: Field Programmable Gate Array, SQP: Sequential Quadratic Programming).

References	Cnst.	Base FW.	Application
[115]	Dyn.	GTSAM: EM	General framework for **LQR-based** factor graphs
[145], [146]	Dyn.	GTSAM	Flow control of wireless mesh networks using **LQR**
[117]	Dyn. Eq.	GTSAM: EM	**LQR-based** control of autonomous vehicle for follow a preplanned trajectory
[147] [148]	Dyn. Eq.	iSAM2	**LQR-based** simultaneous tactile estimation and control of extrinsic contact to a grasped object
[149]–[152]	Dyn. Eq.	GTSAM	**iLQR-based** trajectory generation for a cable-controlled graffiti robot
[116]	Dyn.	FPGA	**LQR-based** control of autonomous vehicle for follow a preplanned trajectory
[114]	Dyn. Eq.	SQP	Simultaneous estimation and **MPC-based** control of unmanned aerial vehicles
[153]	Dyn. Eq. Ineq.	GTSAM	**MPC -based** control and motion planning for cart pole, a Kuka Arm, a quadrupedal robot
[154]	Dyn. Eq. Ineq.	GTSAM	Simultaneous estimation and **MPC-based** control of autonomous robotic spacecraft systems
[155], [156]	Dyn. Eq. Ineq.	GTSAM	Simultaneous estimation and MPC-based control of Unmanned Aerial Vehicles
[55]	Dyn. Eq.	GTSAM	**MPC-based** planning and control of multirobot systems
[157]–[160]	Dyn. Eq. Ineq.	GTSAM & iSAM2	Motion planning
[57], [161]	Dyn. Eq. Ineq.	iSAM & AL	State estimation
[59]	Dyn. Eq. Ineq.	SRRG2 & AL	Localization and **MPC-based** control of unicycle robot
[58]	Dyn. Eq. Ineq.	iSAM2 & AL	2D navigation Planar pushing 3D manipulation planning
[60]	Dyn. Eq. Ineq.	SRRG2 & AL	Pose estimation Rotation synchronization **MPC-based** control of a pseudo-omnidirectional platform

penalty terms. This method has been employed in the work of Chen and Zhang [115], who presented a general framework for LQR-based factor graphs within the GTSAM library. They defined new factors for each difference equation with zero covariance, effectively imposing a high weight on the cost function to guarantee the satisfaction of the equality constraints. The elimination algorithm within the GTSAM framework has been employed with the elimination order $\{x^N, u^{N-1}, x^{N-1}, \ldots, u^0, x^0\}$. This approach showed that solving LQR using factor graphs leads to an equivalent solution to solving the Ricatti equation. Darnley [145], [146] further explored the application of factor graphs in the context of flow control for wireless mesh networks. They compared the performance of factor graphs against other methods, such as different implementations of dynamic programming in C++ and least squares implemented in the Eigen library. Their results showed that factor graphs outperformed these methods, particularly in terms of runtime, which exhibited linear growth with the increasing state space size.

Yang et al. [117] extended the work of Chen et al. by advancing the Linear Quadratic Regulator (LQR) framework to include equality constraints, incorporating states and control inputs across different time steps. Their approach employs the elimination method from the GTSAM library while preserving the same variable ordering as presented by Chen et al. This technique has been validated using optimization problems for generating optimal trajectories for a single-leg hopping robot. To further evaluate the performance of the factor graph solver across varying scales, they formulated the following optimization problem:

$$\min_{Xc} x_N^T Q_N x_N + \sum_{k=0}^{N-1} \left(x_k^T Q_k x_k + u_k^T R_k u_k \right) \tag{33a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_t, \quad k = 0, \ldots, k-1 \tag{33b}$$

$$x_0 = 0 \tag{33c}$$

$$x_{N/2} + u_{N/2} + g_{N/2} = 0 \tag{33d}$$

$$x_{N/2} + g_N = 0, \tag{33e}$$

where $dt = 0.01$, $A = I_{n\times n} + I_{n\times n}dt$, $B = I_{n\times n}dt$, $Q_k = 0.001 I_{n\times n}$, $R_k = 0.001 I_{m\times m}$, and $Q_N = 500 I_{n\times n}$. The vectors $g_{N/2}$ and g_N have dimensions of n rows. In this context, the number of system states, n, and control inputs, m, are assumed to be equal. A comparative analysis was conducted against the state-of-the-art solver proposed by Laine and Tomlin [162], alongside factor graph (FG) solvers at a horizon length of $N = 100$. The results, presented in Table 2, demonstrate the runtime performance for various state and control dimensions.

TABLE 2. Comparison of run times for different state and control dimensions. Results are based on the study by Yang et al. [117].

n, m	10	20	30	40	50	60
[162] (ms)	**3.74**	14.5	44.1	83.5	152.3	247.7
FG (ms)	3.81	**11.8**	**27.1**	**51.2**	**99.0**	**170.2**

As depicted in Table 2, the factor graph approach consistently exhibits competitive performance as both state (n) and control (m) dimensions increase. The superior performance is attributed to the efficiency of the linear solver employed in GTSAM, where QR factorization outperforms the Singular Value Decomposition (SVD) used by the alternative solver. This leads to faster computations, particularly for larger state and control dimensions.

The factor graph methodology has also been adopted in other research. For instance, Kim et al. [147], [148] applied it in the context of tactile estimation and control of interaction forces between a grasped object and its environment using tactile sensors. Herein, LQR control has been deployed for the control task and factor graphs using iSAM2 were employed to simultaneously perform estimation and control tasks. The results demonstrate successful force control using factor graph optimization, taking advantage of the integrated tactile estimator-controller architecture.

Chen et al. [149], [150], [151], [152] applied iterative LQR (iLQR) within a factor graph framework for trajectory generation in a cable-controlled graffiti robot. They implemented the iLQR algorithm to calculate offline trajectories, involving reference states, inputs, and feedback gains. In terms of the hardware design for factor graph-based LQR, Hao et al. [116] introduced a factor graph accelerator for LQR control in autonomous vehicles using FPGA technology. This approach leveraged parallel computation to significantly speed up the LQR algorithm, demonstrating the potential of hardware acceleration in enhancing the performance of factor graph-based control systems.

In the realm of Model Predictive Control (MPC), Ta et al. [114] utilized Sequential Quadratic Programming (SQP) to solve optimization problems with nonlinear equality-constrained factor graphs. Their approach is to solve the factor graph by defining and solving sequentially a quadratic program (QP) derived from applying Newton's method on the Karush–Kuhn–Tucker conditions of the Lagrangian function. The constrained factor graph has been utilized to solve the simultaneous estimation and MPC-based flight control with obstacle avoidance for a quadrotor. King-Smith et al. [153] added inequality constraints to their MPC formulation, namely a box constraint for the joint angle, to cope with the physical limitations of the tested system. They handled the inequality constraints as soft constraints by leveraging the loss hinge function, which penalizes the angles that fall outside the acceptable bound. The approach has been validated in controlling different dynamics, including a simple cart pole, a 7-DoF Kuka arm, and a 12-DoF quadrupedal robot. Similar approaches can also be found in the work of King et al. [154] on estimation and MPC-based control of autonomous robotic spacecraft systems and the work of Yang et al. [155], [156] on MPC-based planning and control of multirobot systems. Both leveraged the GTSAM framework and treated the inequality constraints using the loss hinge function as well.

In addition, motion planning also benefited from the application of factor graphs, as shown by Dong et al. [157] and Mukadam et al. [158], [159], [160]. They utilized factor graphs within the GTSAM and iSAM2 frameworks for motion planning under dynamic, equality, and inequality constraints. This approach treated constraints as soft constraints using penalty terms and a loss hinge function, which were weighted carefully to ensure the constraints were satisfied without causing ill-conditioning in the optimization problem.

Finally, a recent work by Jaafar et al. [55] utilized the flexibility and scalability of factor graphs to implement MPC-based planning and control of multi-robot systems. Their approach highlighted the ease of adding or removing robots and new constraints, making it a robust solution for large-scale multi-robot scenarios.

B. INEQULAITY-CONSTRAINED CONTROL

While converting constraints to soft constraints using penalty terms is a common approach, it does not guarantee that the constraints will always be satisfied. Determining the appropriate weight for the penalty term that forces the variables to satisfy the constraints can be challenging, where a high weight can lead to numerical issues. To address this limitation, Sodhi et al. [57] proposed the Incremental Constrained Smoothing (ICS) framework, in which an Augmented Lagrangian (AL) is used to handle equality and inequality constraints combined with the incremental Smoothing and Mapping (iSAM) framework. This allows them to efficiently handle hard constraints with the capability to incrementally update the matrix factorization upon receiving new measurements or constraints. Their framework was evaluated on simulated datasets involving robotic manipulation tasks, where contact constraints were enforced to ensure the manipulated object remained in contact with the robotic gripper. These constraints were implemented as equality constraints to maintain a physically consistent interaction between the object and the gripper. Additionally, the framework was tested on both simulated and real-world robotic navigation datasets, which required the use of inequality constraints to ensure collision avoidance during navigation. Their results demonstrate the framework's capability to precisely satisfy these constraints, leading to improved estimation accuracy and performance compared to an incremental unconstrained optimization approach like iSAM. Inspired by the work proposed by Bazzana et al. [59] applied the Augmented Lagrangian method combined with the SRRG2 factor graph solver for localization and MPC-based local planning for a unicycle robot.

The main limitation in the ICS framework, however, is the assumption of a fixed linearization point throughout the optimization process, which makes it unsuitable for highly nonlinear problems. To address this limitation, Qadri et al. [58] proposed Incremental Constrained Optimization (InCOpt) using the iSAM2 framework, which efficiently handles nonlinear factors with fluid linearization techniques combined with AL for efficient constraint handling. InCOpt performs alternating upward and downward passes (equivalent to forward and back-substitution) on the Bayes tree to solve for both primal and dual variables until convergence. This was evaluated in simulations for 2D robotic navigation involving inequality constraints to keep the robot within boundaries, 2D planar pushing involving equality constraints to ensure that the object remains in contact with the probe, and 3D arm planning involving collision avoidance constraints.

The simulation results demonstrated that InCOpt not only improves accuracy but also reduces runtime compared to ICS.

Finally, leveraging the results from the above works, a recent study by Bazzana et al. [60] presented an extended version of the Augmented Lagrangian (AL) method combined with the SRRG2 factor graph solver. This approach was tested on various applications, including pose estimation, rotation synchronization, and real-world Model Predictive Control (MPC) of a pseudo-omnidirectional platform, showing significantly reduced runtime compared to the IPOPT solver. More specifically, the comparison involved IPOPT and two variants of the Augmented Lagrangian implemented using factor graph-based methods. The difference between the two AL variants lies in how they present the inequality constraints. The first variant, AL: $\mathbf{g}^+$, converts the inequality constraints into equality using slack variables, while the second variant uses the operator $\max(0,\mathbf{g})$ to ensure that the inequality constraints are satisfied. The results demonstrated an average runtime reduction by a factor of approximately seven for the factor graph-based approaches compared to IPOPT, as illustrated in Figure 9.

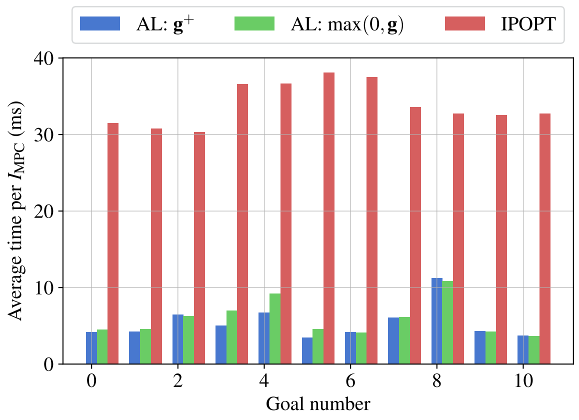


FIGURE 9. Comparison of average runtime per MPC optimization between the Augmented Lagrangian with two variants and IPOPT, with a horizon length of $N = 3$ and a goal number representing the number of target locations set for the robot to reach. Results are based on the study by Bazzana et al. [60].

VI. DISCUSSION AND PERSPECTIVE RESEARCH DIRECTIONS

Factor graphs have demonstrated impressive capabilities in handling large-scale and real-time robotic perception tasks, making them versatile tools for control tasks as well. While factor graphs excel at unconstrained optimization, optimal control in general requires the handling of equality as well as inequality constraints. The existing literature offers some approaches to tackle this challenge, with varying degrees of effectiveness. One method involves converting the equality and inequality constraints (leveraging a loss hinge function for inequality constraints) into soft constraints using weighted penalty functions. However, this approach has limitations: low weights may not be sufficient to satisfy the constraints, while high weights can lead to ill-conditioning. Additionally, relying on a hinge loss function may fail to meet inequality constraints. This function penalizes variables that violate constraints without knowing their actual values, so an initial guess is used to apply the penalty. If this guess is inaccurate, it can lead to unnecessary penalties or missed penalties, resulting in solutions that violate the constraints.

A more promising approach is the Augmented Lagrangian method. Most implementations use the Gauss-Newton method to solve the primal problem, while [60] added a constant damping term to the linear system to address ill-conditioning. However, further investigations are necessary in order to improve the efficiency and convergence of the Augmented Lagrangian method while effectively countering ill-conditioning. Additionally, in [58], two main limitations are highlighted: the system may become underdetermined in some cases and the threshold used to mark variables for linearization is set manually. Research into automatic threshold selection and automatic detection of causes leading to underdetermined systems could enhance robustness and performance. Finally, the works in [59], [60] show higher levels of the cost function compared to alternative solvers like IPOPT, indicating the need for further refinement and improvement.

As a general overview of the available literature, we observe that research on this topic is relatively recent and predominantly based on the GTSAM library. It primarily addresses deterministic system models for MPC and includes a limited number of applications. In contrast, the literature on perception and control, when considered separately, is extensive. This indicates significant potential for this research area, especially given the anticipated benefits of integrating perception and control.

Based on the discussion above, the following key directions for future exploration can be identified::

- Optimization Methods: Develop robust and generalized algorithms that efficiently handle nonlinear optimization problems with both equality and inequality constraints. This will enable broader applicability beyond current limitations.
- Applications: Extend research to safety-critical control systems in robotics and autonomous vehicles, as well as explore applications beyond, such as control of dynamic systems in various fields.
- Implementation Frameworks: While existing work primarily utilizes the GTSAM library, incorporating other state-of-the-art libraries like g2o and Ceres Solver can facilitate the integration of robotic perception tasks implemented with these libraries into control tasks.
- System Models and Probabilistic Framework: While factor graphs are powerful tools for probabilistic modeling, most current approaches applying factor graphs to optimal control primarily focus on deterministic models of system dynamics. A promising research direction is to investigate how stochastic MPC [125], [163], [164], [165] can be effectively integrated within the factor

graph framework to account for uncertainty in system dynamics.

By addressing these challenges and exploring these research directions, we may unlock the full potential of integrating optimal control, and MPC in particular, with factor graphs, leading to more robust and efficient control systems across various domains.

VII. CONCLUSION

This article introduces factor graphs as a unifying framework for various robotic perception tasks, such as SLAM and situational awareness, as well as control tasks such as robot motion planning and model predictive control (MPC). We highlighted the significant benefits obtained from this framework, including the potential for seamless integration of robotic situational awareness, planning, and control. Furthermore, we introduced the basic concept of factor graphs, provided a tutorial on graph optimization, and presented the methods utilized in this domain. Addressing the challenge of efficiently handling constraints remains a central issue mainly for factor graph-based optimal control and consequently for a unifying framework. Our extensive literature review provides an overview of how various approaches manage these constraints within factor graphs. This field remains promising for research, with substantial potential for future advancements. We propose several research directions, such as developing more robust and generalized algorithms for nonlinear and inequality-constrained optimization problems. Additionally, integrating stochastic model predictive control (MPC) with factor graphs as a probabilistic framework offers exciting opportunities for further research. In summary, the integration of MPC-based control with graph optimization is an evolving field. Further research is needed to fully realize its potential in diverse applications in robotics and beyond, such as in dynamic control systems and other engineering domains.

REFERENCES

[1] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Found. Trends Robot.*, vol. 6, nos. 1–2, pp. 1–139, 2017.

[2] F. Dellaert, "Factor graphs: Exploiting structure in robotics," *Annu. Rev. Control, Robot., Auto. Syst.*, vol. 4, no. 1, pp. 141–166, May 2021.

[3] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3607–3613.

[4] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[5] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, Winter 2010.

[6] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep. GT-RIM-CP&R-2012-002, 2012, vol. 2, p. 4.

[7] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "S-Graphs+: Real-time localization and mapping leveraging hierarchical representations," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 4927–4934, Aug. 2023.

[8] A. Tourani, H. Bavle, D. I. Avsar, J. L. Sanchez-Lopez, R. Munoz-Salinas, and H. Voos, "Vision-based situational graphs exploiting fiducial markers for the integration of semantic entities," *Robotics*, vol. 13, p. 106, 2024.

[9] G. Grisetti, T. Guadagnino, I. Aloise, M. Colosi, B. D. Corte, and D. Schlegel, "Least squares optimization: From theory to practice," *Robotics*, vol. 9, no. 3, p. 51, Jul. 2020.

[10] B. Alsadik and S. Karam, "The simultaneous localization and mapping (SLAM)—An overview," *J. Appl. Sci. Technol. Trends*, vol. 2, no. 2, pp. 147–158, Nov. 2021.

[11] S. Agarwal and K. Mierle. (Oct. 2023). *Ceres Solver*. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[12] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[13] X. Wu, B. Xiao, C. Wu, Y. Guo, and L. Li, "Factor graph based navigation and positioning for control system design: A review," *Chin. J. Aeronaut.*, vol. 35, no. 5, pp. 25–39, May 2022.

[14] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[15] A. Tourani, H. Bavle, J. L. Sanchez-Lopez, and H. Voos, "Visual SLAM: What are the current trends and what to expect?" *Sensors*, vol. 22, no. 23, p. 9297, Nov. 2022.

[16] H. Bavle, J. L. Sanchez-Lopez, C. Cimarelli, A. Tourani, and H. Voos, "From SLAM to situational awareness: Challenges and survey," *Sensors*, vol. 23, no. 10, p. 4849, May 2023.

[17] R. M. Neal, "Probabilistic inference using Markov chain Monte Carlo methods," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. CRGT-TR-93-1, 1993.

[18] P. Alevizos, "Factor graphs: Theory and applications," Ph.D. dissertation, Dept. Electron. Comput. Eng., Tech. Univ. Crete, Chania, Greece, 2012.

[19] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," 2018, *arXiv:1805.00909*.

[20] T. Brüdigam, "(Stochastic) model predictive control—A simulation example," 2021, *arXiv:2101.12020*.

[21] M. Wick, A. McCallum, and G. Miklau, "Scalable probabilistic databases with factor graphs and MCMC," 2010, *arXiv:1005.1934*.

[22] M. Fernandez-Cortizas, H. Bavle, D. Perez-Saura, J. Luis Sanchez-Lopez, P. Campoy, and H. Voos, "Multi S-graphs: An efficient distributed semantic-relational collaborative SLAM," 2024, *arXiv:2401.05152*.

[23] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: MIT Press, 2009.

[24] C. Zhang and C. Ré, "Towards high-throughput Gibbs sampling at scale: A study across storage managers," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2013, pp. 397–408.

[25] J. Winn and C. Bishop, "Variational message passing," *J. Mach. Learn. Res.*, vol. 6, no. 23, pp. 661–694, Dec. 2005.

[26] B. J. Frey, F. R. Kschischang, and H. A. Loeliger, "Factor graphs and algorithms," in *Proc. Annu. Allerton Conf. Commun. Control Comput.*, Jan. 2008, pp. 666–680.

[27] R. Santana, A. Mendiburu, and J. A. Lozano, "A review of message passing algorithms in estimation of distribution algorithms," *Natural Comput.*, vol. 15, no. 1, pp. 165–180, Mar. 2016.

[28] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[29] J. Coughlan, "A tutorial introduction to belief propagation," Kettlewell Eye Res. Inst., San Francisco, CA, USA, Tech. Rep., 2009.

[30] K. Tanaka, H. Shouno, M. Okada, and D. M. Titterington, "Accuracy of the Bethe approximation for hyperparameter estimation in probabilistic image processing," *J. Phys. A, Math. Gen.*, vol. 37, no. 36, pp. 8675–8695, Sep. 2004.

[31] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2282–2312, Jul. 2005.

[32] K. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," 2013, *arXiv:1301.6725*.

[33] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.

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

[35] H. J. Kappen, V. Gómez, and M. Opper, "Optimal control as a graphical model inference problem," *Mach. Learn.*, vol. 87, no. 2, pp. 159–182, May 2012.

[36] I. Sugiarto and J. Conradt, "A model-based approach to robot kinematics and control using discrete factor graphs with belief propagation," *Robot. Auto. Syst.*, vol. 91, pp. 234–246, May 2017.

[37] C. Hoffmann and P. Rostalski, "Linear optimal control on factor graphs—A message passing perspective," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6314–6319, Jul. 2017.

[38] C. Herzog né Hoffmann, E. Petersen, and P. Rostalski, "Iterative approximate nonlinear inference via Gaussian message passing on factor graphs," *IEEE Control Syst. Lett.*, vol. 3, no. 4, pp. 978–983, Oct. 2019.

[39] J. Watson, H. Abdulsamad, and J. Peters, "Stochastic optimal control as approximate input inference," in *Proc. Conf. Robot Learn.*, Jan. 2019, pp. 697–716.

[40] C. Herzog Né Hoffmann, F. Vollmer, J. Gruner, and P. Rostalski, "Teaching estimation and control via probabilistic graphical models—An intuitive and problem-based approach," *IFAC-PapersOnLine*, vol. 55, no. 17, pp. 206–211, 2022.

[41] F. Dellaert. (2022). *Borglab/GTSAM*. Accessed: Jun. 13, 2024. [Online]. Available: https://github.com/borglab/gtsam

[42] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. (2023). *Rainerkuemmerle, G2O*. Accessed: Jun. 13, 2024. [Online]. Available: https://github.com/RainerKuemmerle/g2o

[43] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard, "SE-sync: A certifiably correct algorithm for synchronization over the special Euclidean group," *Int. J. Robot. Res.*, vol. 38, nos. 2–3, pp. 95–125, Mar. 2019.

[44] V. Ila, L. Polok, M. Solony, and P. Svoboda, "SLAM++—A highly efficient and temporally scalable incremental SLAM framework," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 210–230, Feb. 2017.

[45] A. Juric, F. Kendeš, I. Markovic, and I. Petrovic, "A comparison of graph optimization approaches for pose estimation in SLAM," in *Proc. 44th Int. Conv. Inf., Commun. Electron. Technol. (MIPRO)*, Sep. 2021, pp. 1113–1118.

[46] A. Abdelkarim, "Development of numerical solvers for online optimization with application to MPC-based energy-optimal adaptive cruise control," Master thesis, Dept. Elect. Comput. Eng., Technische Universität Kaiserslautern, Kaiserslautern, Germany, 2020.

[47] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[48] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, USA: Springer, 2006.

[49] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[50] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*.

[51] S. H. Haji and A. M. Abdulazeez, "Comparison of optimization techniques based on gradient descent algorithm: A review," *PalArch's J. Archaeol. Egypt/Egyptol.*, vol. 18, no. 4, pp. 2715–2743, 2021.

[52] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, no. 4, pp. 308–313, Jan. 1965.

[53] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[54] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[55] H. A. Jaafar, C.-H. Kao, and S. Saeedi, "MR.CAP: Multi-robot joint control and planning for object transport," *IEEE Control Syst. Lett.*, vol. 8, pp. 139–144, 2024.

[56] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "ISAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 216–235, Feb. 2012.

[57] P. Sodhi, S. Choudhury, J. G. Mangelson, and M. Kaess, "ICS: Incremental constrained smoothing for state estimation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 279–285.

[58] M. Qadri, P. Sodhi, J. G. Mangelson, F. Dellaert, and M. Kaess, "InCOpt: Incremental constrained optimization using the Bayes tree," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 6381–6388.

[59] 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.

[60] B. Bazzana, H. Andreasson, and G. Grisetti, "How-to augmented Lagrangian on factor graphs," *IEEE Robot. Autom. Lett.*, vol. 9, no. 3, pp. 2806–2813, Mar. 2024.

[61] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quart. Appl. Math.*, vol. 2, no. 2, pp. 164–168, Jul. 1944.

[62] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, Jun. 1963.

[63] M. J. Powell, "A new algorithm for unconstrained optimization," in *Nonlinear Programming*. Amsterdam, The Netherlands: Elsevier, 1970, pp. 31–65.

[64] R. H. Byrd, R. B. Schnabel, and G. A. Shultz, "Approximate solution of the trust region problem by minimization over two-dimensional subspaces," *Math. Program.*, vols. 40–40, nos. 1–3, pp. 247–263, Jan. 1988.

[65] T. Ando, "Generalized Schur complements," *Linear Algebra Its Appl.*, vol. 27, pp. 173–186, Oct. 1979.

[66] F. Zhang, *The Schur Complement and Its Applications*, vol. 4. Cham, Switzerland: Springer, 2006.

[67] G. H. Golub, "Numerical methods for solving linear least squares problems," *Appl. Math.*, vol. 10, no. 3, pp. 213–216, 1965.

[68] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD, USA: JHU Press, 2013.

[69] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Philadelphia, PA, USA: SIAM, 2003.

[70] D. Dereniowski and M. Kubale, "Cholesky factorization of matrices in parallel and ranking of graphs," in *Proc. 5th Int. Conf. Parallel Process. Appl. Math.*, Czestochowa, Poland. Cham, Switzerland: Springer, Jan. 2004, pp. 985–992.

[71] N. J. Higham, "Cholesky factorization," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 1, no. 2, pp. 251–254, Jun. 2009.

[72] W. Gander, "Algorithms for the QR decomposition," *Res. Rep*, vol. 80, no. 2, pp. 1251–1268, 1980.

[73] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, "The Bayes tree: An algorithmic foundation for probabilistic robot mapping," in *Proc. 9th Int. Workshop Algorithmic Found. Robot.* Cham, Switzerland: Springer, Jan. 2010, pp. 157–173.

[74] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate," *ACM Trans. Math. Softw.*, vol. 35, no. 3, pp. 1–14, Oct. 2008.

[75] T. A. Davis, *Direct Methods for Sparse Linear Systems*. Philadelphia, PA, USA: SIAM, 2006.

[76] T. A. Davis, "Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1–22, Nov. 2011.

[77] G. Guennebaud and B. Jacob. (2010). *Eigen*. [Online]. Available: http://eigen.tuxfamily.org

[78] J. Demmel, "LAPACK: A portable linear algebra library for high-performance computers," *Concurrency, Pract. Exper.*, vol. 3, no. 6, pp. 655–666, Dec. 1991.

[79] T. A. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar, "A survey of direct methods for sparse linear systems," *Acta Numerica*, vol. 25, pp. 383–566, May 2016.

[80] 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, Dec. 2006.

[81] M. Kaess, A. Ranganathan, and F. Dellaert, "Fast incremental square root information smoothing," in *Proc. IJCAI*, Jan. 2007, pp. 2129–2134.

[82] M. Kaess, A. Ranganathan, and F. Dellaert, "ISAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[83] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "ISAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3281–3288.

[84] C. C. de Wit, B. Siciliano, and G. Bastin, *Theory of Robot Control*. Cham, Switzerland: Springer, 2012.

[85] S. B. Niku, *Introduction to Robotics: Analysis, Control, Applications*. Hoboken, NJ, USA: Wiley, 2020.

[86] B. Hernández and E. Giraldo, "A review of path planning and control for autonomous robots," in *Proc. IEEE 2nd Colombian Conf. Robot. Autom. (CCRA)*, Nov. 2018, pp. 1–6.

[87] A. Abdelkarim, Y. Jia, and D. Görges, "An accelerated interior-point method for convex optimization leveraging backtracking mitigation," in *Proc. 49th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2023, pp. 1–6.

[88] A. R. Mehrabian, C. Lucas, and J. Roshanian, "Aerospace launch vehicle control: An intelligent adaptive approach," *Aerosp. Sci. Technol.*, vol. 10, no. 2, pp. 149–155, Mar. 2006.

[89] U. Eren, A. Prach, B. B. Koçer, S. V. Raković, E. Kayacan, and B. Açıkmeşe, "Model predictive control in aerospace systems: Current state and opportunities," *J. Guid., Control, Dyn.*, vol. 40, no. 7, pp. 1541–1566, Jul. 2017.

[90] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. L. P. Chen, "Review of advanced guidance and control algorithms for space/aerospace vehicles," *Prog. Aerosp. Sci.*, vol. 122, Apr. 2021, Art. no. 100696.

[91] A. Mahfouz, G. Gaias, D. M. K. K. Venkateswara Rao, and H. Voos, "Autonomous optimal absolute orbit keeping through formation flying techniques," *Aerospace*, vol. 10, no. 11, p. 959, Nov. 2023.

[92] G. Stephanopoulos, *Chemical Process Control*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall, 1984.

[93] B. W. Bequette, "Nonlinear control of chemical processes: A review," *Ind. Eng. Chem. Res.*, vol. 30, no. 7, pp. 1391–1413, Jul. 1991.

[94] R. S. Gesser, R. Sartori, T. P. Damo, C. M. Vettorazzo, L. B. Becker, D. M. Lima, M. L. de Lima, L. D. Ribeiro, M. C. M. M. Campos, and J. E. Normey-Rico, "Advanced control applied to a gas compression system of an offshore platform: From modeling to related system infrastructure," *J. Petroleum Sci. Eng.*, vol. 208, Jan. 2022, Art. no. 109428.

[95] H. L. Trentelman, A. A. Stoorvogel, and M. Hautus, "Control theory for linear systems," *Appl. Mech. Rev.*, vol. 55, no. 5, p. B87, Jan. 2002.

[96] H. K. Khalil, *Control of Nonlinear Systems*. New York, NY, USA: Prentice-Hall, 2002.

[97] O. Katsuhiko, *Modern Control Engineering*. Miami, FL, USA: Félix Varela, 2009.

[98] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. Hoboken, NJ, USA: Wiley, 2005.

[99] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. Hoboken, NJ, USA: Wiley, 2012.

[100] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar, "A review of PID control, tuning methods and applications," *Int. J. Dyn. Control*, vol. 9, no. 2, pp. 818–827, Jun. 2021.

[101] H. R. Nohooji, A. Zaraki, and H. Voos, "Actor–critic learning based PID control for robotic manipulators," *Appl. Soft Comput.*, vol. 151, Dec. 2023, Art. no. 111153.

[102] O. A. Somefun, K. Akingbade, and F. Dahunsi, "The dilemma of PID tuning," *Annu. Rev. Control*, vol. 52, pp. 65–74, Jan. 2021.

[103] D. E. Kirk, *Optimal Control Theory: An Introduction*. Courier Corporation, 2004.

[104] D. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 4. Belmont, MA, USA: Athena Scientific, 2012.

[105] R. E. Kopp, "Pontryagin maximum principle," in *Mathematics in Science and Engineering*, vol. 5. Amsterdam, The Netherlands: Elsevier, 1962, pp. 255–279.

[106] A. D. Lewis, "The maximum principle of pontryagin in control and in optimal control," Dept. Math. Statist., Queen's Univ., Kingston, Canada, 2006.

[107] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[108] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Philadelphia, PA, USA: SIAM, 2010.

[109] D. P. Bertsekas, "Nonlinear programming," *J. Oper. Res. Soc.*, vol. 48, no. 3, p. 334, 1997.

[110] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *J. Guid., Control, Dyn.*, vol. 10, no. 4, pp. 338–342, Jul. 1987.

[111] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Rev.*, vol. 59, no. 4, pp. 849–904, Jan. 2017.

[112] M. Gerdts, *Optimal Control of ODEs and DAEs*. Walter de Gruyter GmbH & Co KG, 2023.

[113] R. Tedrake. (2023). *Underactuated Robotics*. [Online]. Available: https://underactuated.csail.mit.edu

[114] D.-N. Ta, M. Kobilarov, and F. Dellaert, "A factor graph approach to estimation and model predictive control on unmanned aerial vehicles," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, May 2014, pp. 181–188.

[115] G. Chen and F. Zhang, and Y. Dellaert. (2019). *LQR Control Using Factor Graphs*. Accessed: Jun. 15, 2024. [Online]. Available: https://gtsam.org/2019/11/07/lqr-control.html

[116] Y. Hao, B. Yu, Q. Liu, and S.-S. Liu, "FGLQR: Factor graph accelerator of LQR control for autonomous machines," 2023, *arXiv:2308.02768*.

[117] S. Yang, G. Chen, Y. Zhang, H. Choset, and F. Dellaert, "Equality constrained linear optimal control with factor graphs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 9717–9723.

[118] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[119] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, Dec. 2014.

[120] J. A. Rossiter, *Model-Based Predictive Control: A Practical Approach*. Boca Raton, FL, USA: CRC Press, 2017.

[121] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, vol. 2. Madison, WI, USA: Nob Hill Publishing, 2017.

[122] S. Yu, M. Reble, H. Chen, and F. Allgöwer, "Inherent robustness properties of quasi-infinite horizon nonlinear model predictive control," *Automatica*, vol. 50, no. 9, pp. 2269–2280, Sep. 2014.

[123] F. Allgöwer, R. Findeisen, and Z. K. Nagy, "Nonlinear model predictive control: From theory to application," *J.-Chin. Inst. Chem. Eng.*, vol. 35, no. 3, pp. 299–315, May 2004.

[124] L. Grüne, *Nonlinear Model Predictive Control*. Cham, Switzerland: Springer, 2017.

[125] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Syst. Mag.*, vol. 36, no. 6, pp. 30–44, Dec. 2016.

[126] A. Mesbah, "Stochastic model predictive control with active uncertainty learning: A survey on dual control," *Annu. Rev. Control*, vol. 45, pp. 107–117, Jan. 2018.

[127] T. A. N. Heirung, J. A. Paulson, J. O'Leary, and A. Mesbah, "Stochastic model predictive control—How does it work?" *Comput. Chem. Eng.*, vol. 114, pp. 158–170, Jun. 2018.

[128] E. F. Camacho, D. R. Ramirez, D. Limon, D. Muñoz de la Peña, and T. Alamo, "Model predictive control techniques for hybrid systems," *Annu. Rev. Control*, vol. 34, no. 1, pp. 21–31, Apr. 2010.

[129] A. Abdelkarim and P. Zhang, "Optimal scheduling of preventive maintenance for safety instrumented systems based on mixed-integer programming," in *Proc. 7th Int. Symp. Model-Based Saf. Assessment*, Lisbon, Portugal. Cham, Switzerland: Springer, Jan. 2020, pp. 83–96.

[130] P. D. Christofides, R. Scattolini, D. Muñoz de la Peña, and J. Liu, "Distributed model predictive control: A tutorial review and future research directions," *Comput. Chem. Eng.*, vol. 51, pp. 21–41, Apr. 2013.

[131] Z. Ajanović, M. Stolz, and M. Horn, "Energy-efficient driving in dynamic environment: Globally optimal MPC-like motion planning framework," in *Advanced Microsystems for Automotive Applications 2017: Smart Systems Transforming the Automobile*. Cham, Switzerland: Springer, 2018, pp. 111–122.

[132] G. F. Usieto, "Trajectory generation for autonomous highway driving using model predictive control," Master's thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 2017.

[133] D. M. K. K. Venkateswara Rao, H. Habibi, J. L. Sanchez-Lopez, and H. Voos, "An integrated real-time UAV trajectory optimization and potential field approach for dynamic collision avoidance," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2023, pp. 79–86.

[134] P. Kremer, H. Rahimi Nohooji, and H. Voos, "Constrained trajectory optimization and force control for UAVs with universal jamming grippers," *Sci. Rep.*, vol. 14, no. 1, p. 11968, May 2024.

[135] E. L. Haseltine and J. B. Rawlings, "Critical evaluation of extended Kalman filtering and moving-horizon estimation," *Ind. Eng. Chem. Res.*, vol. 44, no. 8, pp. 2451–2460, Apr. 2005.

[136] D. A. Allan and J. B. Rawlings, "Moving horizon estimation," in *Handbook of Model Predictive Control*, 2019, pp. 99–124.

[137] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, Jan. 2002.

[138] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear Model Predictive Control: Towards New Challenging Applications*, 2009, pp. 345–369.

[139] A. Bemporad, "Model predictive control design: New trends and tools," in *Proc. 45th IEEE Conf. Decis. Control*, Sep. 2006, pp. 6678–6683.

[140] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019.

[141] A. Abdelkarim, Y. Jia, and D. Gorges, "Optimization of vehicle-to-grid profiles for peak shaving in microgrids considering battery health," in *Proc. 49th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2023, pp. 1–6.
[142] D. T. Agi, K. D. Jones, M. J. Watson, H. G. Lynch, M. Dougher, X. Chen, M. N. Carlozo, and A. W. Dowling, "Computational toolkits for model-based design and optimization," *Current Opinion Chem. Eng.*, vol. 43, Mar. 2024, Art. no. 100994.
[143] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.
[144] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. New York, NY, USA: Academic, 2014.
[145] R. Darnley, "Flow control of wireless mesh networks using LQR and factor graphs," Master thesis, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2021.
[146] R. Darnley and M. Travers, "Flow control of wireless mesh networks using LQR and factor graphs," in *Proc. Amer. Control Conf. (ACC)*, Mar. 2022, pp. 2295–2302.
[147] S. Kim, D. K. Jha, D. Romeres, P. Patre, and A. Rodriguez, "Simultaneous tactile estimation and control of extrinsic contact," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 12563–12569.
[148] S. Kim, A. Bronars, P. Patre, and A. Rodriguez, "TEXterity—Tactile extrinsic deXterity: Simultaneous tactile estimation and control for extrinsic dexterity," 2024, *arXiv:2403.00049*.
[149] G. Chen, S. Baek, J.-D. Florez, W. Qian, S.-W. Leigh, S. Hutchinson, and F. Dellaert, "GTGraffiti: Spray painting graffiti art from human painting motions with a cable driven parallel robot," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 4065–4072.
[150] G. Chen, S. Hutchinson, and F. Dellaert, "Locally optimal estimation and control of cable driven parallel robots using time varying linear quadratic Gaussian control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 7367–7374.
[151] G. Chen, H. Muriki, A. Sharkey, C. Pradalier, Y. Chen, and F. Dellaert, "A hybrid cable-driven robot for non-destructive leafy plant monitoring and mass estimation using structure from motion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 11809–11816.
[152] G. Chen, T. Al-Haddad, F. Dellaert, and S. Hutchinson, "Architectural-scale artistic brush painting with a hybrid cable robot," 2024, *arXiv:2403.12214*.
[153] M. Xie, A. Escontrela, and F. Dellaert, "A factor-graph approach for optimization problems with dynamics constraints," 2020, *arXiv:2011.06194*.
[154] M. King–Smith, P. Tsiotras, and F. Dellaert, "Simultaneous control and trajectory estimation for collision avoidance of autonomous robotic spacecraft systems," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 257–264.
[155] P. Yang and W. Wen, "Tightly joining positioning and control for trustworthy unmanned aerial vehicles based on factor graph optimization in urban transportation," in *Proc. IEEE 26th Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2023, pp. 3589–3596.
[156] P. Yang, W. Wen, S. Bai, and L.-T. Hsu, "Tightly joined positioning and control model for unmanned aerial vehicles based on factor graph optimization," 2024, *arXiv:2404.14724*.
[157] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using Gaussian processes and factor graphs," in *Proc. Robot., Sci. Syst.*, 2016, vol. 12, no. 4.
[158] 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, Sep. 2018.
[159] M. Mukadam, J. Dong, F. Dellaert, and B. Boots, "STEAP: Simultaneous trajectory estimation and planning," *Auto. Robots*, vol. 43, no. 2, pp. 415–434, Feb. 2019.
[160] M. Mukadam, "Structured learning and inference for robot motion generation," Ph.D. dissertation, Georgia Inst. Technol., Atlanta, GA, USA, 2019.
[161] P. Sodhi, "Learning and inference in factor graphs with applications to tactile perception," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2022.
[162] F. Laine and C. Tomlin, "Efficient computation of feedback control for equality-constrained LQR," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6748–6754.
[163] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1433–1440.
[164] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3758–3765, Oct. 2019.
[165] F. Cursi, W. Bai, E. M. Yeatman, and P. Kormushev, "Task accuracy enhancement for a surgical macro-micro manipulator with probabilistic neural networks and uncertainty minimization," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 1, pp. 241–256, Jan. 2022.

ANAS ABDELKARIM received the B.S. degree in biomedical engineering from The Hashemite University, Jordan, and the M.S. degree in automation and control systems from the Technical University of Kaiserslautern, Germany, in 2020. He is currently pursuing the joint Ph.D. degree with the Automation and Robotics Research Group (ARG), Interdisciplinary Center for Security, Reliability and Trust (SnT), University of Luxembourg, headed by Prof. Holger Voos, and the Electromobility Department, TU KL, headed by Prof. Daniel Görges. His research interests include control systems, model predictive control, autonomous systems, and optimization.

HOLGER VOOS (Member, IEEE) received the Dipl.-Ing. degree in electrical engineering from Saarland University and the Dr.-Ing. degree in automatic control from the Technical University of Kaiserslautern, Germany. From 2000 to 2004, he was with Bodenseewerk Gerätetechnik GmbH, Germany, as a Systems Engineer, and the Project Manager in aerospace and robotics research and development. From 2004 to 2010, he was a Professor and the Head of the Mobile Robotics Laboratory, University of Applied Sciences Ravensburg-Weingarten, Germany. Since 2010, he has been a Full Professor with the University of Luxembourg and the Head of the Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability and Trust (SnT). From 2020 to 2024, he was the Founding Course Director of the New Interdisciplinary Space Master (ISM) Program, Faculty of Science, Technology and Medicine (FSTM). He is the author or co-author of more than 300 publications, comprising books, book chapters, journal and conference papers. His research interests include situational awareness and motion planning and control for autonomous vehicles and robots, and distributed and networked control and automation.

DANIEL GÖRGES (Member, IEEE) received the Dipl.-Ing. and Dr.-Ing. degrees from the Department of Electrical and Computer Engineering, University of Kaiserslautern, Kaiserslautern, Germany, in 2005 and 2011, respectively. He has held positions as a Researcher and a Junior Professor with the University of Kaiserslautern and the Research Group Leader of German Research Center for Artificial Intelligence (DFKI). Since 2021, he has been a Professor and the Head of the Institute of Electromobility, Department of Electrical and Computer Engineering, RPTU University of Kaiserslautern-Landau. His research interests include methods for model predictive control, distributed control, and learning control and their application in vehicular systems, transportation systems, mechatronic systems, robotics, and power systems.