IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

Handling Constrained Optimization in Factor Graphs for Autonomous Navigation

Barbara Bazzana[1], Tiziano Guadagnino[1], and Giorgio Grisetti[1]

***Abstract*—Factor graphs are graphical models used to represent a wide variety of problems across robotics, such as Structure from Motion (SfM), Simultaneous Localization and Mapping (SLAM) and calibration. Typically, at their core, they have an optimization problem whose terms only depend on a small subset of variables. Factor graph solvers exploit the locality of problems to drastically reduce the computational time of the Iterative Least-Squares (ILS) methodology. Although extremely powerful, their application is usually limited to unconstrained problems. In this paper, we model constraints over variables within factor graphs by introducing a factor graph version of the Augmented Lagrangian (AL) method. We show the potential of our method by presenting a full navigation stack based on factor graphs. Differently from standard navigation stacks, we can model both optimal control for local planning and localization with factor graphs, and solve the two problems using the standard ILS methodology. We validate our approach in real-world autonomous navigation scenarios, comparing it with the de facto standard navigation stack implemented in ROS. Comparative experiments show that for the application at hand our system outperforms the standard nonlinear programming solver Interior-Point Optimizer (IPOPT) in runtime, while achieving similar solutions.**

***Index Terms*—Localization, Integrated Planning and Control, Optimization and Optimal Control**

I. Introduction

FACTOR graphs are a general graphical formalism to model several problems. The robotics community extensively used factor graphs to approach estimation problems, such as SfM, SLAM and calibration [8], [17], [18], [27]. On the one hand, they provide a graphical representation that exposes the general structure of the problem which, if exploited, allows the design of efficient algorithms. On the other hand, factor graphs allow to naturally couple problems sharing common variables by simply joining them. The solution to the joined factor graph will be then the optimum of the coupled problem. Well-known methodologies to solve large factor graphs use variants of Iterative Least Squares (ILS) [16], [19], [20] that do not support *constrained* optimization in its original formulation.

Albeit mapping and control are coupled in the reality, they are usually addressed separately due to their complexity.

Manuscript received: July, 30, 2022; Revised October, 1, 2022; Accepted November, 25, 2022.

This paper was recommended for publication by Editor S. Behnke upon evaluation of the Associate Editor and Reviewers' comments.

[1]Barbara Bazzana, Tiziano Guadagnino, Giorgio Grisetti are with the Department of Computer, Control, and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Rome, Italy{bazzana, guadagnino, grisetti}@diag.uniroma1.it

Digital Object Identifier (DOI): see top of this page.

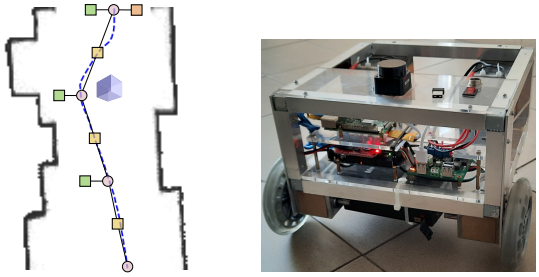


Fig. 1: Left: a trajectory executed by our real robot avoiding an unforeseen obstacle, a schematic factor graph is superposed to it where some robot states are represented (pink circles). Both the localization problem (orange square) and the optimal control are depicted. States are linked by the motion model (yellow squares) and subject to obstacle avoidance and trajectory tracking (green squares). Right: our custom made unicycle.

However, extending existing ILS solvers to constrained optimization provides a common framework where both the above-mentioned properties of factor graphs can be exploited. In particular, the two problems could consistently share common information such as system dynamics, external disturbances, and uncertainty models. Further, the sparsity pattern of both problems allows designing efficient algorithms to maintain high computational performance in joint optimization.

Leveraging on the results of [29], the contribution of this paper is an Augmented Lagrangian-version of factor graph optimization. It was applied to the development of a factor graph-based Model Predictive Control (MPC) framework for unicycle navigation. The optimal control problem we consider is that of generating a collision-free trajectory for a differential drive robot with actuation limits in a dynamic environment. To this extent, we introduce *constraint* factors and *obstacle avoidance* factors: the latter ones model the minimization of an artificial potential function [28] [3]. Our approach was validated both in simulation and on a custom-made robot, where all software runs on a Raspberry PI4. Fig. 1 shows a real-world trajectory and a simplistic representation of the related factor graph. In this paper, we focus on the development of the navigation module, assuming that the map is given. Localization is also addressed in the framework of ILS on factor graphs, in particular the likelihood of the current estimate is evaluated through the distance between the laser scan and the map.

We conducted comparative experiments with ROS navigation stack, comprising `amcl` for localization and `move_base`[1] with `teb_local_planner`[2] for planning. Experiments show that (i) our approach gives on-pair and

[1]http://wiki.ros.org/move_base

[2]http://wiki.ros.org/teb_local_planner

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

repeatable results (ii) the CPU consumption of our approach is lower. Further, comparative experiments with the standard nonlinear programming solver IPOPT show that (iii) when approaching constrained optimization problems on the models we analyzed, our generic solver outperforms IPOPT in runtime while producing comparable trajectories and final cost value. This gives a glimpse of the computational performances of factor graphs motivating their use also for very large problems.

II. Related Work

Factor graphs are a very powerful tool to model a great variety of unconstrained optimization problems, spacing from SLAM to SfM [18] [27]. In the recent literature, they have been applied to model optimal control problems as well [14], [21], [30]–[32]. To this extent, extensions of factor graph solvers have been proposed to handle constraints arising from optimal control modeling. Formalizing optimal control problems using factor graphs allows unifying the dual problems of estimation and control under the same representation. Thanks to this unification, common information and variables can be shared consistently in both estimation and control processes. Further, these two aspects can be addressed jointly taking advantage of efficient factor graph solvers [16], [19], [20].

To the best of our knowledge, Mukadam *et al.* [21] were the first to optimize a unique factor graph for both trajectory estimation and motion planning, where trajectories are modeled using continuous Gaussian Processes [13]. However, they were only relying on soft constraints. Later on, the problem of addressing constrained optimization received increasing attention, starting from Ta *et al.* [30]. They present a factor graph version of Sequential Quadratic Programming (SQP) to handle nonlinear equality constraints and apply their approach to the development of an MPC framework on unmanned aerial vehicles. Our work can be viewed as an extension of this method, where we model both equality and inequality constraints using factor graphs. In place of SQP, we rely on the AL method because, differently from SQP, no quadratic programming solver is required for the internal iterations when addressing inequalities.

Yang *et al.* [32] include control inputs in a factor graph to solve a Linear Quadratic Regulator problem subject to auxiliary equality constraints. During the variable elimination process, when constrained variables are eliminated, a specialized solver is used for solving the constrained sub-problem separately. Differently from them, our method can handle both equality and inequality constraints and does not require a specialized solver, as it uses a standard ILS algorithm.

To the best of our knowledge, only Xie *et al.* [31] introduce both equality and inequality constraints in factor graphs. In particular, they present a factor graph version of a barrier-based approach to constrained optimization similar to the well-known Interior-Point Method. Our work can be viewed as complementary to Xie *et al.*, in that we present an implementation of the AL method, as an alternative to the Interior-Point Method.

Sodhi *et. al* [29] proposes an approach to refine the estimate of past trajectories: to accomplish collision-free trajectories, states are subject to inequality constraints. To the best of our knowledge they have been the first to introduce the AL method in the context of incremental smoothing. We rely on the same concepts to develop a factor-graph version of this method. We propose a different application than Sodhi *et. al*: instead of state estimation, we address optimal control problems to develop a factor graph-based MPC controller that can be used as an elegant and compact local planner. In addition, we address the problem of obstacle avoidance by designing a factor that relies on a distance cost function inspired by scan registration techniques. The resulting factor graph is solved by our generic factor graph solver [16].

III. Factor graph optimization

In this paper, we model both localization and MPC using factor graphs, which are bipartite graphs with two kinds of nodes: variables and factors. Variables represent the state of our system, while factor nodes model measurements connecting the set of variables from which they depend.

More formally, let $\mathbf{x} = \mathbf{x}_{0:N}$ be the set of all variables which can span over arbitrary continuous domains, with $\mathbf{x}_k \in \mathbb{R}^n$. If the measurements $\mathbf{z} = \mathbf{z}_{0:N}$ are affected by Gaussian noise, we can represent the k^{th} factor as the tuple $\langle \mathbf{z}_k, \mathbf{\Omega}_k, \mathbf{h}_k(\cdot) \rangle$ comprising the mean $\mathbf{z}_k$, the information matrix $\mathbf{\Omega}_k$, and the prediction function $\mathbf{h}_k(\mathbf{x}^k)$, depending on a subset of variables $\mathbf{x}^k = \mathbf{x}_{k_0:k_K}$. In this setting the negative log-likelihood of the factor graph becomes:

$$F(\mathbf{x}) = \sum_{k=0}^{K} ||\mathbf{h}_k(\mathbf{x}^k) - \mathbf{z}_k||^2_{\mathbf{\Omega}_k} = \sum_{k=0}^{K} ||\mathbf{e}_k(\mathbf{x}^k)||^2_{\mathbf{\Omega}_k} \tag{1}$$

Solving a factor graph means finding the $\mathbf{x}$ which minimizes Eq. (1), i.e. the $\mathbf{x}$ which is maximally consistent with the measurements. We refer the reader to [11], [16] for more details on formulating robotics problems with factor graphs.

ILS minimizes Eq. (1) by using Gauss-Newton (GN) algorithm that iteratively refines the current solution $\hat{\mathbf{x}}$ by solving the quadratic approximation of Eq. (1). The first-order Taylor expansion of the errors $\mathbf{e}_k(\mathbf{x}^k)$ around $\hat{\mathbf{x}}^k$ given the perturbation $\mathbf{\Delta x}^k$ is computed as:

$$\mathbf{e}_k(\hat{\mathbf{x}}^k + \mathbf{\Delta x}^k) \simeq \overbrace{\mathbf{e}_k(\hat{\mathbf{x}}^k)}^{\hat{\mathbf{e}}_k} + \mathbf{J}_k \mathbf{\Delta x}^k \tag{2}$$

By substituting Eq. (2) in Eq. (1), we obtain a quadratic form that approximates the cost function around $\hat{\mathbf{x}}$:

$$F(\hat{\mathbf{x}} + \mathbf{\Delta x}) \simeq c + 2\mathbf{b}^T \mathbf{\Delta x} + \mathbf{\Delta x}^T \mathbf{H} \mathbf{\Delta x} \tag{3}$$

where

$$\mathbf{b} = \sum_{k=0}^{K} \mathbf{J}_k^T \mathbf{\Omega}_k \hat{\mathbf{e}}_k \qquad \mathbf{H} = \sum_{k=0}^{K} \mathbf{J}_k^T \mathbf{\Omega}_k \mathbf{J}_k \tag{4}$$

The minimum $\mathbf{\Delta x}$ of the quadratic form of Eq. (3) is the solution of the linear system $\mathbf{H\Delta x} = -\mathbf{b}$, while the next estimate is updated by applying the perturbation $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \mathbf{\Delta x}$.

Algorithm 1 Augmented Lagrangian-Iterative Least Squares

F total number of factors
$\langle \mathbf{z}_k, \mathbf{\Omega}_k, \mathbf{h}_k(\cdot) \rangle$ error factor
$\langle \boldsymbol{\lambda}_k, \rho_k, \mathbf{c}_k(\cdot) \rangle$ constraint factor
while !converged **do**
  $\mathbf{H} \leftarrow \mathbf{0}_{n \times n}$
  $\mathbf{b} \leftarrow \mathbf{0}_{n \times 1}$
  for $k = 0, ..., \mathrm{F}$ **do**
    if error factor **then**
      $\mathbf{b} \mathrel{+}= \mathbf{J}_k^T \mathbf{\Omega}_k \hat{\mathbf{e}}_k$
      $\mathbf{H} \mathrel{+}= \mathbf{J}_k^T \mathbf{\Omega}_k \mathbf{J}_k$
    else if constraint factor **then**
      $\mathbf{b} \mathrel{+}= \frac{\rho_k}{2} \mathbf{C}_k^T \hat{\mathbf{c}}_k + \frac{1}{2} \mathbf{C}_k^T \boldsymbol{\lambda}_k$
      $\mathbf{H} \mathrel{+}= \frac{\rho_k}{2} \mathbf{C}_k^T \mathbf{C}_k$
  $\mathbf{\Delta x} \leftarrow \mathrm{solve}(\mathbf{H \Delta x} = -\mathbf{b})$
  $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \mathbf{\Delta x}$
  for $k = 0, ..., \mathrm{F}$ **do**
    if constraint factor **then**
      update $\boldsymbol{\lambda}_k$ according to Eq. (8) or Eq. (10)

IV. Extending iterative Least Squares solvers for constrained optimization

Lagrange Multipliers [2] is perhaps the most known method to deal with constrained optimization. Although the method tackles nonlinear constraint functions as well, we will focus on the linear case to give a simple overview. Consider the problem:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\mathrm{argmin}} \overbrace{\sum_{k=0}^{K} ||\mathbf{e}_k(\mathbf{x}_k)||^2_{\mathbf{\Omega}_k}}^{F(\mathbf{x})}, \quad \text{s.t.}\, \mathbf{Cx} + \mathbf{c} = 0 \tag{5}$$

where $\mathbf{C} \in \mathbb{R}^{c \times n}$. The augmented Lagrangian of Eq. (5) is defined as:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^c; \rho^c) = F(\mathbf{x}) + (\boldsymbol{\lambda}^c)^T (\mathbf{Cx} + \mathbf{c}) + \frac{\rho^c}{2} ||\mathbf{Cx} + \mathbf{c}||_2^2. \tag{6}$$

Here, $\boldsymbol{\lambda}^c \in \mathbb{R}^c$ is a vector of Lagrange Multipliers and the term $\frac{\rho^c}{2} \| \cdot \|_2^2$, with $\rho^c \in \mathbb{R}$, is a penalty term. It can be shown [4] that the minimum of Eq. (5) can be found by iteratively computing:

$$\mathbf{x} \leftarrow \underset{\mathbf{x}}{\mathrm{argmin}}\, \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^c; \rho^c) \tag{7}$$

$$\boldsymbol{\lambda}^c \leftarrow \boldsymbol{\lambda}^c + \rho^c (\mathbf{Cx} + \mathbf{c}). \tag{8}$$

This method can be generalized to deal also with inequality constraints $\mathbf{Dx} + \mathbf{d} \leq 0$ with $\mathbf{D} \in \mathbb{R}^{d \times n}$ by adding an additional multiplier $\boldsymbol{\lambda}^d \in \mathbb{R}^d$ and an additional penalty term weighted by $\rho^d \in \mathbb{R}$. The new augmented Lagrangian then becomes:

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^c, \boldsymbol{\lambda}^d; \rho^c, \rho^d) = & F(\mathbf{x}) \\ & + (\boldsymbol{\lambda}^c)^T (\mathbf{Cx} + \mathbf{c}) + \frac{\rho^c}{2} ||\mathbf{Cx} + \mathbf{c}||_2^2 \\ & + (\boldsymbol{\lambda}^d)^T (\mathbf{Dx} + \mathbf{d}) + \frac{\rho^d}{2} ||\mathbf{Dx} + \mathbf{d}||_2^2 \end{aligned} \tag{9}$$

The current state estimate is the $\mathbf{x}$ which minimizes Eq. (9). Further, the update-law of the inequality multiplier $\boldsymbol{\lambda}^d$ is the following:

$$\boldsymbol{\lambda}^d \leftarrow \max(0, \boldsymbol{\lambda}^d + \rho^d (\mathbf{Dx} + \mathbf{d})) \tag{10}$$

The AL method belongs to the class of primal-dual methods which split the optimization in two steps: a primal step (7) and a dual step (8, 10). Implementing the dual update requires extending the *constrained variables* with the multipliers. An iteration proceeds by updating the constrained variable $\mathbf{x}$ while keeping the multipliers fixed through the primal step; once $\mathbf{x}$ is refined, the multipliers $\boldsymbol{\lambda}^c$ and $\boldsymbol{\lambda}^d$ are updated with $\mathbf{x}$ fixed through the dual step.

The primal step Eq. (7) is a minimization that can be solved through ILS since the multipliers are fixed. Let $\hat{\mathbf{x}}$ be the current solution, $\hat{\mathbf{c}} = \mathbf{C}\hat{\mathbf{x}} + \mathbf{c}$ and $\hat{\mathbf{d}} = \mathbf{D}\hat{\mathbf{x}} + \mathbf{d}$ be the constraint values computed at $\hat{\mathbf{x}}$. During the linearization step of ILS, the quadratic form approximating Eq. (9) around $\hat{\mathbf{x}}$ then becomes:

$$\mathcal{L}(\hat{\mathbf{x}} + \mathbf{\Delta x}, \boldsymbol{\lambda}^c, \boldsymbol{\lambda}^d; \rho^c, \rho^d) \simeq c + 2\mathbf{b}^T \mathbf{\Delta x} + \mathbf{\Delta x}^T \mathbf{H} \mathbf{\Delta x} \tag{11}$$

where

$$\mathbf{b} = \sum_{k=0}^{K} \mathbf{J}_k^T \mathbf{\Omega}_k \hat{\mathbf{e}}_k + \overbrace{\frac{\rho^c}{2} \mathbf{C}^T \hat{\mathbf{c}} + \frac{1}{2} \mathbf{C}^T \boldsymbol{\lambda}^c}^{\mathbf{b}^c} + \overbrace{\frac{\rho^d}{2} \mathbf{D}^T \hat{\mathbf{d}} + \frac{1}{2} \mathbf{D}^T \boldsymbol{\lambda}^d}^{\mathbf{b}^d} \tag{12}$$

$$\mathbf{H} = \sum_{k=0}^{K} \mathbf{J}_k^T \mathbf{\Omega}_k \mathbf{J}_k + \overbrace{\frac{\rho^c}{2} \mathbf{C}^T \mathbf{C}}^{\mathbf{H}^c} + \overbrace{\frac{\rho^d}{2} \mathbf{D}^T \mathbf{D}}^{\mathbf{H}^d}.$$

By comparing Eq. (12) and Eq. (4), it turns out that the primal update can be implemented in ILS solvers by introducing *constraint factors*. They are represented by tuples as $\langle \boldsymbol{\lambda}_k, \rho_k, \mathbf{c}_k(\cdot) \rangle$, and contribute to the $\mathbf{H}$ and $\mathbf{b}$ matrices respectively through $\mathbf{H}^c$ and $\mathbf{b}^c$, or $\mathbf{H}^d$ and $\mathbf{b}^d$ depending on the type of constraint. The proposed approach is summarized in Alg. 1. In our implementation, we use constant $\rho_k = 1$ but an adaptive scheme as in [29] can be introduced. Our derivation of constraint factors is made possible by the Lagrangian being a simple extension of the usual cost function. As a result, the primal update is similar to the usual update of ILS, which then triggers the update of the multipliers through the dual update.

V. Problem description

Our factor-graph based navigation framework consists of two modules: MPC controller for local planning Sec. V-A and scan-matched based localization Sec. V-B. Both problems are modeled through factor graphs and tackled by the generic ILS solver [16]. This confirms the generality of our approach.

A. MPC with obstacle avoidance for unicycle

MPC is a well-established technique for optimally controlling systems subject to constraints. At each time step, an MPC controller computes the best feasible trajectory which minimizes the assigned objective function, while avoiding

obstacles and satisfying the constraints. Only the first control input is applied to the system; at the next time step, a new trajectory embedding information coming from the most recent measurements is computed. More in detail, let $\mathbf{u} = \mathbf{u}_{0:N-1}$ be the sequence of controls, $\mathbf{x} = \mathbf{x}_{0:N}$ the resulting chain of states, $\mathbf{u}^{\text{ref}} = \mathbf{u}^{\text{ref}}_{0:N-1}$ and $\mathbf{x}^{\text{ref}} = \mathbf{x}^{\text{ref}}_{0:N}$ the reference values for controls and states respectively, a suitable objective is the Quadratic Regulator (QR) cost function $\Phi_{QR}(\mathbf{u}, \mathbf{x})$ [1]

$$\Phi_{QR}(\mathbf{u}, \mathbf{x}) = \sum_{n=0}^{N} \| \overbrace{\mathbf{x}_n - \mathbf{x}_n^{\text{ref}}}^{\mathbf{e}_n^x(\mathbf{x}_n)} \|^2_{\mathbf{\Omega}_n^x} + \sum_{n=0}^{N-1} \| \overbrace{\mathbf{u}_n - \mathbf{u}_n^{\text{ref}}}^{\mathbf{e}_n^u(\mathbf{u}_n)} \|^2_{\mathbf{\Omega}_n^u} \tag{13}$$

subject to the robot kinematics and the actuation limits. The QR cost function in Eq. (13) is an instance of the standard Least Squares problem in Eq. (1), which is addressed by general factor graph solvers, when the control chain becomes part of the state. As the task of MPC is to compute both the trajectory *and* the sequence of controls, in the language of factor graphs controls will become variable nodes as well. Hence, each term in the above equation can be modeled as the single-variable factor $\langle \mathbf{x}_n^{\text{ref}}, \mathbf{\Omega}_n^x, \mathbf{I} \rangle$, with $\mathbf{I}$ identity matrix. In contrast, obstacle avoidance and constrained optimization require more effort to be translated into the same formalism. Fig. 2 illustrates the factor graph formulation of an optimal control problem.

A feasible trajectory can be found by minimizing a cost function $g(\mathbf{x})$ that decreases with the distance from obstacles, being zero at safe locations. According to the theory of Artificial Potential Fields [28] [3], having denoted as k a positive real scalar, as μ a low-distance threshold, and as ρ a high-distance threshold, we designed $g(\mathbf{x})$ as:

$$g(\mathbf{x}) = \begin{cases} k\left(\frac{1}{\mu} - \frac{1}{\rho}\right) & \text{if } d(\mathbf{x}) < \mu \\ k\left(\frac{1}{d(\mathbf{x})} - \frac{1}{\rho}\right) & \text{if } \mu < d(\mathbf{x}) < \rho \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The above equation imposes a finite maximum value to $g(\mathbf{x})$ by clamping it to its maximum where $d(\mathbf{x}) < \mu$, and shapes a valley of zero-potential where $d(\mathbf{x}) > \rho$. Points in the region $d(\mathbf{x}) > \rho$ are safe and do not need to be pushed away by the optimization process. In our implementation of Eq. (14) we use $k = 0.075$, $\rho = 0.8\,\text{m}$, and $\mu = 0.05\,\text{m}$. Therefore we model the obstacle avoidance through a factor whose error is:

$$\mathbf{e}_n^o(\mathbf{x}_n) = g(\mathbf{x}_n). \tag{15}$$

It is well known that Artificial Potential Fields are affected by local minima. In our implementation, we have modified the gradient of the repulsive field generated by Eq. (14) to overcome such minima. Using the idea of vortex fields [10], we add to the repulsive gradient a vector which is tangent to the equipotential contours of the repulsive field, as it can be seen in Fig. 3. By applying the obstacle avoidance factor to each robot state, the vortex field acts on the whole trajectory rather than on a single state [10], to deflect it from unexpected obstacles.

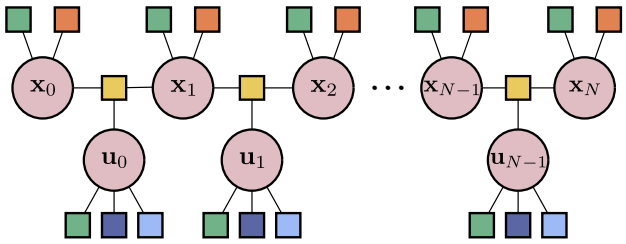


Fig. 2: Factor graph modeling an optimal control problem: robot poses $\mathbf{x}_{0:N}$ and controls $\mathbf{u}_{0:N\ \ 1}$ are variables, represented here by pink circles, factors instead are represented by squares. Colors should be interpreted as follows: green for factors penalizing deviations from reference values $\mathbf{x}^{\text{ref}}_{0:N}$ and $\mathbf{u}^{\text{ref}}_{0:N\ \ 1}$, yellow for motion model factors, blue for upper and lower constraint factors, orange for obstacle avoidance factors.

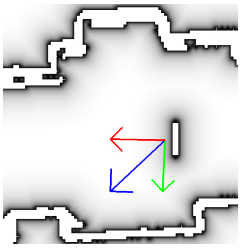


Fig. 3: The Artificial Potential Field is represented as the background in gray-scale. The gradient, blue arrow, is obtained by summing the repulsive gradient in red, and its normal in green. By doing this at all points in space we generate a vortex around the obstacle.

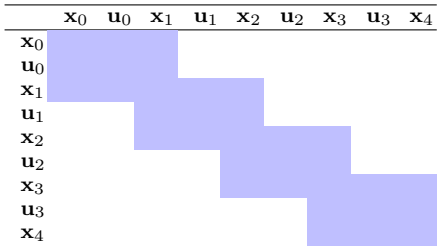


TABLE I: Sparsity pattern of the optimal control problem under variable ordering $\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, ..., \mathbf{x}_4$.

As suggested by Rösmann *et al.* [26], the kinematic model $\mathbf{f}(\cdot)$ mapping current state $\mathbf{x}_n$ and control $\mathbf{u}_n$ to next state $\mathbf{x}_{n+1}$ is incorporated into the objective function as a quadratic penalty for faster convergence of the algorithm. Such a penalty can be modeled by a factor with error function $\mathbf{e}_n^p(\mathbf{x}_n, \mathbf{u}_n, \mathbf{x}_{n+1}) = \mathbf{x}_{n+1} - \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n)$. The weighting matrix $\mathbf{\Omega}_n^p$ was tuned by trial and error to preserve the correctness of the result despite such approximation of the constraint. In an MPC fashion, at each time step a graph is constructed starting from the current localization estimate which becomes the first (fixed) variable $\mathbf{x}_0$. Our robot is a unicycle, controlled in translational and rotational velocities, denoted respectively with v_n and ω_n. Using Runge-Kutta integration [5] with T_s as integration interval and being $(x_n, y_n)^T$ the position and θ_n the orientation of the n^{th} pose $\mathbf{x}_n$, the kinematics is as follows:

$$x_{n+1} = x_n + v_n T_s \cos\left(\theta_n + \frac{\omega_n T_s}{2}\right) \tag{16}$$

$$y_{n+1} = y_n + v_n T_s \sin\left(\theta_n + \frac{\omega_n T_s}{2}\right) \tag{17}$$

$$\theta_{n+1} = \theta_n + \omega_n T_s \tag{18}$$

Applying Eq. (16) in the solution scheme of Sec. III might

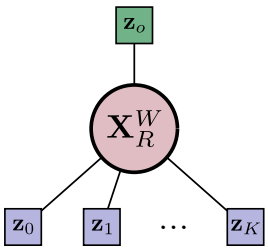


Fig. 4: Factor graph modeling the localization problem. The variable to be estimated is the pose of the robot in the world $\mathbf{X}_R^W$, and is repesented by the pink circle. Each measurement contributes with a factor: odometry is represented by the green square, exteroceptive measurements, e.g. laser endpoints, by the purple ones.

result in controls $\mathbf{u} = (v, \omega)_{0:N-1}^T$ that exceed the actuation limits. Hence we need to enforce *inequality* constraints over the variables $\mathbf{u}_n$, which are not addressed by regular ILS. In the next section, we introduce *constraint factors* and *constrained variables* to model both inequality and equality constraints.

Summarizing, the problem can be described through the set of equations

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{u}} \ \Phi_{MPC}(\mathbf{u},\mathbf{x}) \\ &\text{s.t. } \forall n, \ |v_n| \leq v_{\max}, |\omega_n| \leq \omega_{\max} \end{aligned} \tag{19}$$

where

$$\begin{aligned} \Phi_{MPC}(\mathbf{u},\mathbf{x}) = &\ \Phi_{QR}(\mathbf{u},\mathbf{x}) \\ &+ \sum_{n=0}^{N-1} \left[\|\mathbf{e}_n^p(\mathbf{x}_n, \mathbf{u}_n, \mathbf{x}_{n+1})\|^2_{\mathbf{\Omega}_n^p} + \|\mathbf{e}_n^o(\mathbf{x}_n)\|^2_{\mathbf{\Omega}_n^o} \right] \\ &+ \|\mathbf{e}_N^o(\mathbf{x}_N)\|^2_{\mathbf{\Omega}_N^o} \end{aligned} \tag{20}$$

Let $\mathbf{u}_{\max} = (v_{\max}, \omega_{\max})^T$, the Lagrangian function is

$$\begin{aligned} \mathcal{L}(\mathbf{u},\mathbf{x},\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2;\rho_1,\rho_2) = &\ \Phi_{MPC}(\mathbf{u},\mathbf{x}) \\ &+ \boldsymbol{\lambda}_1^T(\mathbf{u}-\mathbf{u}_{\max}) + \frac{\rho_1}{2}||\mathbf{u}-\mathbf{u}_{\max}||_2^2 \\ &+ \boldsymbol{\lambda}_2^T(-\mathbf{u}-\mathbf{u}_{\max}) + \frac{\rho_2}{2}||-\mathbf{u}-\mathbf{u}_{\max}||_2^2 \end{aligned} \tag{21}$$

By imposing the variable ordering $\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, ...$, we obtain the maximally sparse structure of the linear system shown in Tab. I for the case of $N = 4$.

B. Localization as registration on a distance matrix

Localization aims at estimating the most likely robot pose in the world frame $\mathbf{X}_R^W \in SE(3)$, given the robot measurements $\mathbf{z} = \mathbf{z}_{0:K}$ and the odometry prior $\mathbf{z}_O$. In particular, our robot moves on a plane $\mathbf{X}_R^W \in SE(2)$ and is equipped with a 2D lidar. The map is given and stored in memory as a 2D matrix $\mathbf{M}$ where each cell contains the distance $d(\mathbf{p}) : \mathbb{R}^2 \to \mathbb{R}$ of point $\mathbf{p}$ from the closest obstacle in the map. Factor graph in Fig. 4 summarizes an instance of such problem.

Let us indicate as $\mathbf{z}_k \in \mathbb{R}^2$ a generic laser endpoint expressed in the robot frame. A suitable localization factor associated to the k-th measurement would minimize the error function:

$$\mathbf{e}_k^l(\mathbf{X}_R^W) = d(\mathbf{X}_R^W \mathbf{z}_k) \tag{22}$$

where the distance from the laser endpoint to the closest obstacle can be retrieved in $\mathcal{O}(1)$ by querying the matrix $\mathbf{M}$ at the global coordinates of the laser endpoint $\mathbf{z}_k^W = \mathbf{X}_R^W \mathbf{z}_k$. Should the robot be perfectly localized in a static environment, all distances will be close to zero, and error in Eq. (22) would be zero as well. Summing the contribution of all laser endpoints in the laser scan, the localization graph has therefore the following cost function:

$$G(\mathbf{X}_R^W) = \sum_{k=0}^{K} \|\mathbf{e}_k^l(\mathbf{X}_R^W \mathbf{z}_k)\|_{\mathbf{\Omega}_k^l} \tag{23}$$

$$\mathbf{X}_R^{W*} = \underset{\mathbf{X}_R^W}{\operatorname{argmin}} \, G(\mathbf{X}_R^W) \tag{24}$$

In the implemented navigation stack, the same extended ILS solver addresses both MPC and localization as described above. Albeit we implemented the error function in Eq. (22) because of its compactness, it can be replaced with more elaborated state-of-the-art error functions [9], [22], [33] without changing the solver implementation, as done in [12].

VI. EXPERIMENTS

We tested our system both in simulation and on a real custom made unicycle robot[3] equipped with a 2D lidar (InnoMaker-LD06), running on a Raspberry PI4 board at our Department, whose map is shown in Fig. 5. Comparative experiments have been conducted in simulation with a similar robot configuration, on a laptop Intel(R) Core(TM) i7-10750H CPU running at 2.60GHz with 16GB of RAM.

We compare our method with the most recent version of the ROS navigation stack consisting of `amcl` for localization [23] and `move_base` with Timed Elastic Band (TEB) [25] local planner for control. The two navigation stacks consist of a global and a local planner. Both ROS and our global planner compute the obstacle-free path by Dijkstra's algorithm relying on the knowledge of the map. In our navigation stack, the map is given as a 2D occupancy grid map obtained by running a standard SLAM algorithm [15] beforehand. This path is then fed to the local planner that calculates the optimal control, and considers potential dynamic obstacles detected through the lidar. The robot estimates its position by using the factor graph-based localizer of Sec. V-B.

Further, we feed sample instances of our optimization problem to the well-known state-of-the-art general purpose nonlinear programming solver IPOPT [24]. Our experiments are designed to show the validity of our approach and support our key claims (i) our approach gives repeatable results, on-pair with ROS navigation stack at a (ii) lower CPU consumption, (iii) when approaching constrained optimization problems on the models we analyzed, our generic solver outperforms IPOPT in runtime while producing comparable trajectories and final cost value.

A. Comparison with ROS navigation stack

We report here the results obtained by running our navigation stack and the ROS one to travel across 8 goals 30 times

[3]https://www.marrtino.org

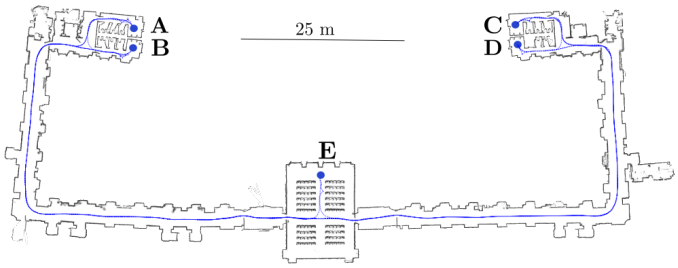


Fig. 5: Map of our Department.

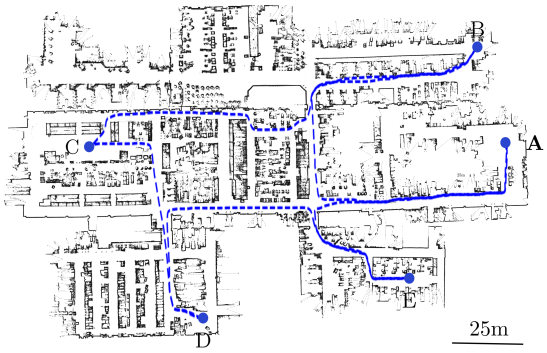


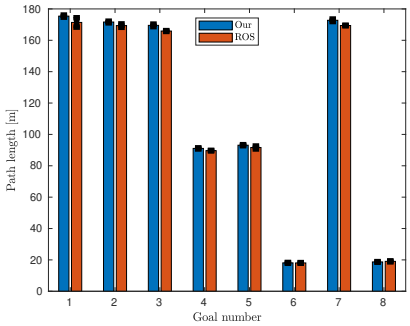
Fig. 6: Map of factory-like environment.

within the map of our Department shown in Fig. 5, and 5 goals 30 times within the map shown in Fig. 6. While map in Fig. 6 is better for overall evaluation as it has more corners, map in Fig. 5 alternates long straight corridors and very narrow passages as those in the upper left and right corners. For each path we store the ground truth poses published by ROS `stage` and the duration in seconds. The path length is computed by summing the distance between subsequent ground truth poses. Fig. 7 summarizes the obtained results, using bars for data, and error bars for standard deviation. Both approaches safely drove the robot during the whole 12 hours of simulation.

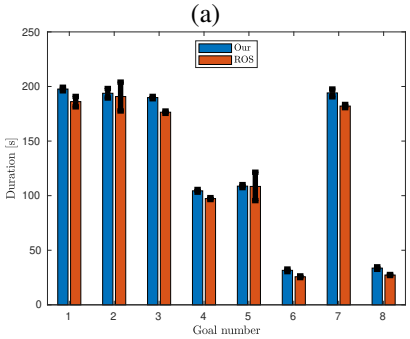
As it can be seen in Fig. 7(a)-(c), path length is comparable, being only slightly larger using our approach due to the global planner providing more conservative paths in narrow passages. The repeatability of our approach is also on-pair with `move_base`, giving an average standard deviation of 0.62%, compared to 0.63% of our competitor.

Concerning path duration, Fig. 7(d) shows our approach performs better than `move_base` in the map of Fig. 6. Instead, Fig. 7(b) shows that the duration is larger for our method on the map of Fig. 5, and depends on the path itself: in long paths, where narrow passages account only for a small portion of the whole path, ROS duration is on average 95% of ours, while on paths mostly consisting of narrow passages ROS duration is 81% of ours. As already mentioned, this is due to the global planner being more conservative in very narrow passages. However, our approach is on average more repeatable with a standard deviation of 1.57%, compared to 4.35% of our competitor.

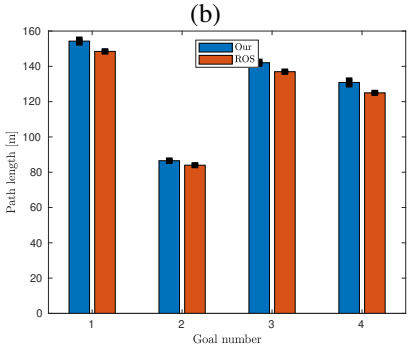
The comparable results shown in Fig. 7 come at a considerably lower CPU consumption: our factor graph-based local planner accounts for 3.8%, our global planner for 1.1%, and finally our localization module for 2.0%, giving a total



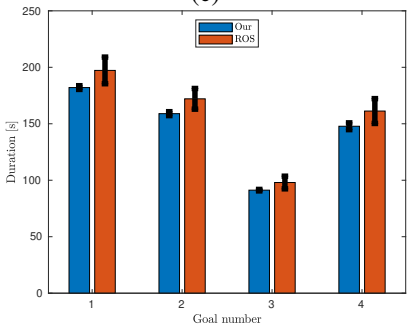
(a)



(b)



(c)



(d)

Fig. 7: Comparison with ROS navigation stack. Map of Fig. 5: (a) path length, (b) duration. Map of Fig. 6: (c) path length, (d) duration.

of 6.9%. Differently, `move_base` accounts for 13.5%, and `amcl` for additional 2.8%, giving a total of 16.3%. For efficient computation, TEB method addresses the optimal control problem as multi-objective optimization within factor graphs, and uses the popular solver by Kümmerle et al. [19]. TEB relies on polynomial penalty functions to enforce constraints, and augments the state by incorporating the time intervals between two consecutive poses. Since our time step is fixed, the number of variables is lower in our case which contributes in reducing CPU consumption. At the same time, our implementation of AL method is competitive with a

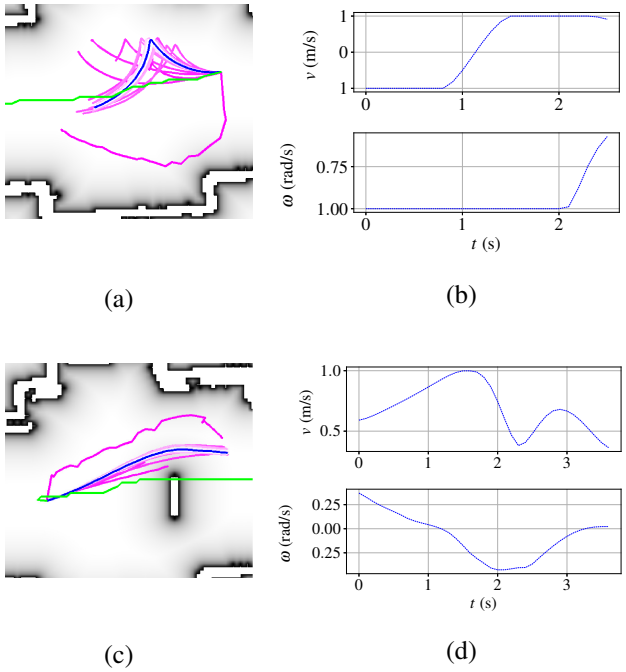


Fig. 8: Initial guess in green, iterations whitening pink, optimal path in blue: (a) backward maneuver from right to left; (b) control inputs; : (c) obstacle avoidance maneuver from left to right; (d) corresponding control inputs.

Solver	Percentage decrease [%]	Iterations	Time[s]
Ours	99.32	**118**	**0.011**
IPOPT	**99.36**	445	1.459
Ours	97.96	**125**	**0.013**
IPOPT	**98.11**	809	2.497
Ours	98.96	**158**	**0.013**
IPOPT	**99.03**	931	2.943
Ours	97.97	**189**	**0.031**
IPOPT	**97.98**	479	1.872
Ours	97.95	**366**	**0.055**
IPOPT	**97.97**	549	1.951
Ours	97.73	**294**	**0.043**
IPOPT	**97.75**	439	1.682

TABLE II: Comparison with IPOPT: convergence analysis.

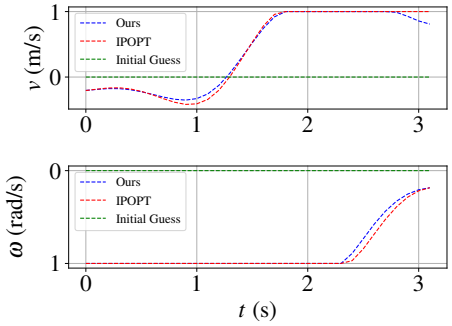


Fig. 9: Comparison with IPOPT: control inputs corresponding to the backward maneuver of last row of Tab. II.

penalty-based approach to constrained optimization.

Finally, for the sake of completeness, Fig. 8 reports the internals of an MPC iteration in whitening pink, with associated control input profiles.

B. Comparison with IPOPT

We evaluated the performance of our solver compared to the well-known solver IPOPT, by feeding the optimal control problem in Eq. (19) to it, with same weights, limits and initial guess. Four different goals were considered starting from a given initial pose, resulting in both forward and backward trajectories. Tab. II shows the results obtained, while Fig. 9 plots control inputs optimized with the two methods corresponding to the last row of Tab. II. As a termination criterion for our solver, we consider the norm of the perturbation vector and that of the constraint violations. Optimization is stopped when: $||\mathbf{\Delta x}||_2 < \epsilon_{\mathbf{x}}$, $||\mathbf{Cx}+\mathbf{c}||_\infty < \epsilon_{\mathbf{c}}$, $||max(\mathbf{Dx}+\mathbf{d}, \underline{0})||_\infty < \epsilon_{\mathbf{d}}$, where $\underline{0}$ is the vector of all zeros, and $\epsilon_{\mathbf{x}}, \epsilon_{\mathbf{c}}, \epsilon_{\mathbf{d}} = 1\text{e}{-4}$.

Our solver is around 2 orders of magnitude faster than IPOPT at converging. This makes our solver preferable for online onboard applications. Two aspects make this possible. The first is that, given the same objective functions and constraints, AL iterations are cheaper than those of Interior Point. Thanks to the split in primal and dual update, the dimension of the linear system we solve at each step is equal to the number of actual variables. Instead, in Interior Point the linear system is larger comprising also the set of auxiliary variables, as big as twice the number of inequality constraints plus once that of equality constraints. Further, a line-search procedure is implemented to choose the weight of the step. The second aspect motivating the difference with IPOPT is that the factor graph solver [16] [20] we use is highly optimized for sparse optimization. We can choose proper variable ordering so as to maximize the sparsity pattern, which results in easy inverse computations for linear system solving to compute the update step of ILS. The final cost of our solution is slightly higher, however both a quantitative comparison with the initial value reported in Tab. II and a graphical comparison based on Fig. 9 show that the two solutions are very similar. In summary, our evaluation suggests that our approach is considerably more efficient than standard non-linear programming solvers.

VII. Conclusions

We presented an approach to embedding constraints on variables as factors of a graph, based on the AL method. Having introduced an Iterative Least Squares solver for constrained optimization, we have contributed to extending the variety of problems across robotics that can be modeled using factor graphs.

In this paper, we focus on optimal control for MPC as an application field of constrained optimization to support the validity of our approach. In particular, we develop a full navigation stack based on factor graphs: both localization and MPC are modeled here through factors. Experiments show that our navigation stack compares favorably to the ROS navigation stack. Moreover, experiments conducted with IPOPT confirm that our solver gives the advantage of both an improvement in computational time and a unified formulation for estimation and control problems.

While our navigation stack is a working proof of concept, our method is general and can be easily used to model other classes of problems. As future work in the field of optimal control, different kinematic and dynamic models

can be considered. Active SLAM [6], [7] is an example application where both estimation and control are coupled. The two problems share the knowledge of the map and the robot position estimate and contribute to the goal of having a robot that autonomously builds an accurate map in the shortest possible time. In the near future we envision future applications of our findings in this domain.

REFERENCES

[1] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[2] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[3] Farid Bounini, Denis Gingras, Herve Pollart, and Dominique Gruyer. Modified artificial potential field method for online path planning applications. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 180–185, 2017.

[4] Stephen P. Boyd, Neal Parikh, Eric King wah Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3:1–122, 2011.

[5] J.C. Butcher. A history of runge-kutta methods. *Applied Numerical Mathematics*, 20(3):247–260, 1996.

[6] H. Carrillo and J.A. Castellanos. *Navigation Under Uncertainty Based on Active SLAM Concepts*, pages 209–235. Springer International Publishing, 2015.

[7] Yongbo Chen, Shoudong Huang, and Robert Fitch. Active SLAM for mobile robots with area coverage and obstacle avoidance. *IEEE/ASME Transactions on Mechatronics*, 25(3):1182–1192, 2020.

[8] M.D. Cicco, B.D. Corte, and G. Grisetti. Unsupervised Calibration of Wheeled Mobile Platforms. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[9] Lakshitha Dantanarayana, Ravindra Ranasinghe, and Gamini Dissanayake. C-LOG: A chamfer distance based method for localisation in occupancy grid-maps. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 376–381, 2013.

[10] Alessandro De Luca and Giuseppe Oriolo. Local incremental planning for nonholonomic mobile robots. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 104–110 vol.1, 1994.

[11] Frank Dellaert. Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):141–166, 2021.

[12] Luca Di Giammarino, Irvin Aloise, Cyrill Stachniss, and Giorgio Grisetti. Visual place recognition using lidar intensity information. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4382–4389, 2021.

[13] Jing Dong, Mustafa Mukadam, Byron Boots, and Frank Dellaert. Sparse gaussian processes on matrix lie groups: A unified framework for optimizing continuous-time trajectories. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6497–6504, 2018.

[14] Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion planning as probabilistic inference using gaussian processes and factor graphs. In *Robotics: Science and Systems XII*, 2016.

[15] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Trans. on Intelligent Transportation Systems Magazine*, 2:31–43, 2010.

[16] Giorgio Grisetti, Tiziano Guadagnino, Irvin Aloise, Mirco Colosi, Bartolomeo Della Corte, and Dominik Schlegel. Least squares optimization: from theory to practice. *Robotics*, 9(3):51, July 2020.

[17] Giorgio Grisetti, Rainer Kümmerle, and Kai Ni. Robust optimization of factor graphs by using condensed measurements. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 581–588, 2012.

[18] Viorela Ila, Lukas Polok, Marek Solony, and Pavel Svoboda. SLAM++-a highly efficient and temporally scalable incremental slam framework. *The International Journal of Robotics Research*, 36(2):210–230, 2017.

[19] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[20] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.

[21] Mustafa Mukadam, Jing Dong, Frank Dellaert, and Byron Boots. STEAP: simultaneous trajectory estimation and planning. *Autonomous Robots*, 43:415–434, 2019.

[22] Helen Oleynikova, Alexander Millane, Zachary Taylor, Enric Galceran, Juan I. Nieto, and Roland Y. Siegwart. Signed distance fields: A natural representation for both mapping and planning. 2016.

[23] Patrick Pfaff, Wolfram Burgard, and Dieter Fox. Robust monte-carlo localization using adaptive likelihood models. In Henrik I. Christensen, editor, *European Robotics Symposium*, volume 22 of *Springer Tracts in Advanced Robotics*, pages 181–194. Springer, 2006.

[24] Florian A. Potra and Stephen J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1):281–302, 2000.

[25] Christoph Rösmann, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram. Efficient trajectory optimization using a sparse model. In *2013 European Conference on Mobile Robots*, pages 138–143, 2013.

[26] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Kinodynamic trajectory optimization and control for car-like robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5681–5686, 2017.

[27] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[28] Joe Sfeir, Maarouf Saad, and Hamadou Saliah-Hassane. An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment. In *2011 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pages 208–213, 2011.

[29] Paloma Sodhi, Sanjiban Choudhury, Joshua G. Mangelson, and Michael Kaess. ICS: Incremental constrained smoothing for state estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 279–285, 2020.

[30] Duy-Nguyen Ta, Marin Kobilarov, and Frank Dellaert. A factor graph approach to estimation and model predictive control on unmanned aerial vehicles. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 181–188, 2014.

[31] Mandy Xie, Alejandro Escontrela, and Frank Dellaert. A factor-graph approach for optimization problems with dynamics constraints. *CoRR*, abs/2011.06194, 2020.

[32] Shuo Yang, Gerry Chen, Yetong Zhang, Howie Choset, and Frank Dellaert. Equality constrained linear optimal control with factor graphs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9717–9723, 2021.

[33] Mingming Zhang, Yiming Chen, and Mingyang Li. SDF-Loc: Signed distance field based 2d relocalization and map update in dynamic environments. In *2019 American Control Conference (ACC)*, pages 1997–2004, 2019.