2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

A Factor Graph Approach to Estimation and Model Predictive Control on Unmanned Aerial Vehicles

Duy-Nguyen Ta[1] , Marin Kobilarov[2] and Frank Dellaert[1]

Abstract— **In this paper, we present a factor graph framework to solve both estimation and deterministic optimal control problems, and apply it to an obstacle avoidance task on Unmanned Aerial Vehicles (UAVs). We show that factor graphs allow us to consistently use the same optimization method, system dynamics, uncertainty models and other internal and external parameters, which potentially improves the UAV performance as a whole. To this end, we extended the modeling capabilities of factor graphs to represent nonlinear dynamics using constraint factors. For inference, we reformulate Sequential Quadratic Programming as an optimization algorithm on a factor graph with nonlinear constraints. We demonstrate our framework on a simulated quadrotor in an obstacle avoidance application.**

I. Introduction

Although a unified framework for both estimation and control might be useful for autonomous systems, the standard practice in robotics community is to leverage the principle of separation [29] and treat the two problems separately, leading to vastly different models, formulations and optimization tools in each subfield. This practice is undesirable however, as in theory the separation principle does not hold in general for nonlinear systems; furthermore, it conceals the fact that both problems stem from a closely related principle of optimality [31], [35]. We argue that using the same representation and computational framework for both estimation and control problems is advantageous, as it allows us to exploit the inherent duality of estimation and control in theory [17], [32], and also allows the knowledge of the system dynamics, external disturbances, and uncertainty models to be shared consistently in both estimation and control processes. This will lead to significant improvements in coherence, stability and robustness of the system as a whole.

In this paper we propose factor graphs [23] as a unified representation framework and computational tool for both estimation and control problems, and apply it to an obstacle avoidance task on Unmanned Aerial Vehicles (UAVs). Over the years, factor graphs have established themselves as a useful tool for large-scale estimation problems [8], [16], [13]. Their success is largely due to their expressiveness ability to represent large-scale problems in graphs, and the availability of many efficient inference algorithms on graphical models, exploiting the locality, sparsity and tree-like structures of the graphs for efficient computation [6], [21]. The basic variable-elimination scheme for inference on factor graphs generalizes over many standard estimation algorithms such as the Extended Kalman Filter, dynamic programming on Hidden Markov Models, etc., leading to substantial improvements in state-of-the-art estimation systems, especially for large-scale SLAM and structure-from-motion [15], [26].

While factor graphs have been primarily used for estimation problems, they have remained largely underutilized to solve control problems, let alone *simultaneous* estimation and control. Although graphical models and inference methodologies have been applied to optimal control [19], [34], [33], [2], [28], the development of graphical models for control is still limited. To the best of our knowledge, none of the existing work exploits graphical model representation and inference techniques for deterministic optimal control problems. On the other hand, graphical model formulations of stochastic optimal control problems are either limited to linear cases [33], or impractical due to the lack of system dynamics in their final solutions [18].

In the paper we extend the capabilities of factor graphs beyond a standard estimation tool to a unified framework for both estimation and deterministic optimal control problems in the context of UAVs with nonlinear system dynamics and kinematics constraints. Beside the aforementioned advantages of a unified framework, the benefits of using factor graphs as a computational tool are manifolds. First, factor graph representations reveal sparsity structures of the problems, which can be exploited to speed up the inference process significantly. Second, while standard optimization methods operate on vector spaces, factor graph inference algorithms perform directly on variables' Lie-group manifolds [16], guaranteeing numerical stability and robustness. Furthermore, it opens future possibilities to employ large-scale and incremental inference techniques in estimation [7], [16] to improve the performance of the whole system.

In addition to a general formulation of the unified estimation-control framework (Section II), our main contributions are two key extensions of factor graphs which enable them to deal with nonlinear system dynamics constraints. First, to represent the nonlinear dynamics, we introduce into the graphs nonlinear constrained factors, which arise from integrating the differential dynamics equations on the state Lie-group manifolds (Section III). Second, we reformulate Sequential Quadratic Programming as an optimization algorithm on factor graphs with nonlinear constraints (Section IV). Our experiments in Section V demonstrate the potential of the unified framework on a simulated quadrotor in autonomous navigation and obstacle avoidance applications.

[1]The Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, USA {duynguyen, dellaert}@gatech.edu
[2]The Laboratory for Computational Sensing and Robotics, Johns Hopkins University, USA marin@jhu.edu

II. Estimation and Control in Factor Graphs

A. General Problem Formulation

At the current time t_c, a UAV has to solve two optimization problems for estimation and control. Let $x(t) \in \mathcal{G}$ be the UAV state at time t, $u(t) \in \mathcal{U}$ its control inputs, $w(t) \in \mathbb{R}^l$ dynamics noise process, $l_{1:K}$ external landmarks in the environment, and ρ_{int} a static variable related to all unknown or uncertain parameters of the system such as kinematic, dynamic and calibration terms which we wish to estimate and use to compute optimal control policy. At each point t_i in the time-horizon of interest $[t_0, t_f]$, the UAV might observe a new measurement z_{ij} of some landmarks l_j with noise v_{ij}. The complete process is as follows:

$$\dot{x}(t) = f(x(t), u(t), w(t), \rho_{int}, t) \quad \text{(dynamics)} \quad (1)$$

$$z_{ij} = h_{ij}(x(t_i), l_j) + v_{ij} \quad \text{(measurements)} \quad (2)$$

$$u(t) \in \mathcal{U},\ x(t) \in \mathcal{G} \quad \text{(constraints)} \quad (3)$$

where f and h_{ij} are nonlinear dynamics and measurement functions respectively.

For numerical purposes, we focus on a discrete time version of this process with time discretization points $\{t_0, t_1, \ldots, t_N\}$, discrete states $x_{0:N} = \{x_0, x_1, \ldots, x_N\}$ and controls $u_{0:N-1} = \{u_0, u_1, \ldots, u_{N-1}\}$, where $t_N = t_f$, $u_i \approx u(t_i)$, and

$$x_i \approx x(t_i) \quad (4)$$

which needs to be approximated by integrating (1) on the state manifold $\mathcal{G}$, as detailed later in Section III.

The estimation process minimizes a cost function J_{est} to obtain the optimal estimate of all landmarks l_j, the past trajectory $x_{0:c}$ during $[t_0, t_c]$, the internal parameters ρ_{int}, and the past process noises $w_{0:c}$. As standard in the literature [24], [30], [10], the cost function J_{est} is the negative log of the posterior $p(x_{0:c}, w_{0:c}, l_{1:K}, \rho_{int} | \{z_{ij}\}, u_{0:c-1})$, which provides the *maximum a-posteriori* (MAP) estimate. Under the common assumption that the states and parameters have a Gaussian prior, i.e. $x_0 \sim \mathcal{N}(\hat{x}_0, P_x^{-1})$ and $\rho_{int} \sim \mathcal{N}(\hat{\rho}_{int}, P_{\rho_{int}}^{-1})$, and that uncertainties are Gaussian, i.e. $w_i \sim \mathcal{N}(0, P_{w_i}^{-1})$ and $v_{ij} \sim \mathcal{N}(0, P_{v_{ij}}^{-1})$, J_{est} has the following form, subject to the dynamic constraint in (1):

$$\begin{aligned} J_{est} = &||x_0 - \hat{x}_0||^2_{P_x} + ||\rho_{int} - \hat{\rho}_{int}||^2_{P_{\rho_{int}}} \\ &+ ||w_i||^2_{P_{w_i}} + \sum_{ij} ||h_{ij}(x_i, l_j) - z_{ij}||^2_{P_{v_{ij}}} \end{aligned} \quad (5)$$

The control process minimizes another cost function J_{mpc} to compute the optimal control policy $u_{c:N-1}$ and the corresponding future trajectory $x_{c+1:N}$, also subject to the dynamic constraint in (1):

$$J_{mpc} = \varphi(x_N) + \sum_{i=c}^{N-1} L_i(x_i, u_i, l_{1:K}, \rho_{int}) \quad (6)$$

where $\varphi(x_N)$ and $L_i(\cdot)$ are the terminal cost and stage-wise cost functions respectively.

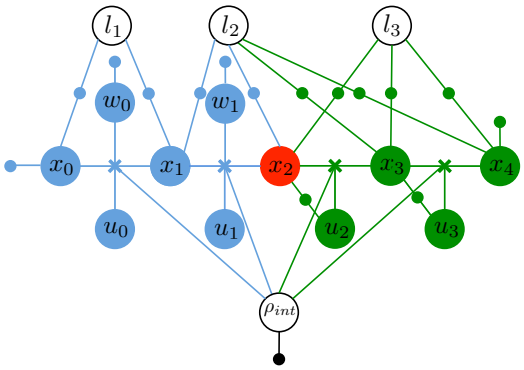

Fig. 1. The proposed factor graph framework for both estimation and control problems. The past estimation part is in blue, the future control part in green, and red color denotes the current state. Classic Gaussian motion models in estimation are now replaced with constrained dynamics factors.

B. Factor Graph Representation

We propose to use factor graphs [8] as a common framework to represent both estimation and control problems. Fig. 1 shows an example of our factor graph that includes both the past estimation part and the future control part.

The estimation part of the graph (blue color) includes unary factors on x_0, ρ_{int} and w_i, encoding the corresponding unary terms in (5), and binary factors between (x_i, l_j), encoding the measurement terms $||h_{ij}(x_i, l_j) - z_{ij}||^2_{P_{v_{ij}}}$. Due to the dynamics constraint in (1) that needs to be satisfied while minimizing J_{est}, we introduce new dynamics constrained factors, shown as crosses in Fig. 1, connecting the states x_i, x_{i+1} and control u_i to a common time-independent static parameter ρ_{int} and a time-dependent noise variable w_i.

Unlike traditional factor graphs for estimation [8], our graph has the general constrained dynamics factors, replaced for the traditional Gaussian motion models. These nonlinear constrained dynamics factors are detailed later in Section III. They expand the capability of factor graphs to represent general dynamics and kinematics, e.g. under-actuated dynamics with possibly multiplicative noise. With the dynamics model explicitly introduced into the graph, our estimates are expected to be better constrained and more accurate than the traditional estimation formulation. Furthermore, beside the usual state and landmark estimates, we can also obtain estimates for the internal parameters ρ_{int} and other time-dependent disturbances w_i in the environment.

In the control graph (green color), we leverage the current optimal estimates of landmarks and dynamics parameters to infer the optimal control subject to the deterministic dynamics constraints. Stage-wise cost functions $L_i(\cdot)$ to optimize in (6) are binary factors between corresponding states and landmarks, whereas the final cost function $\varphi(x_N)$ is a unary factor on the final state x_N. We optimize this graph to obtain one step model-predictive-control solution, together with the predicted future trajectory. After executing the first control and receiving new estimates from optimizing the estimation graph, we extend the time horizon one step further and repeat the whole process.

C. Inference as Optimization on Factor Graphs

Traditional techniques to solve nonlinear factor graphs without nonlinear constrained factors typically apply nonlinear optimization methods such as the Newton's method, or, for least-squares factors such as in (5), Gauss-Newton iterations or the Levenberg-Marquardt algorithm [27]. Similar to the extended Kalman filter [30], at each iteration these methods linearize the nonlinear factor graph, solve the linear graph using **variable elimination** algorithm [21], [6], [8] to obtain the δ-updates, i.e. δx_i or δl_j, of the corresponding variables, then update the original variables according to:

$$\begin{aligned} x_i &\leftarrow x_i + \delta x_i \\ \text{and } l_j &\leftarrow l_j + \delta l_j. \end{aligned} \tag{7}$$

However, with the existence of nonlinear constrained factors, these standard unconstrained optimization methods cannot be applied anymore. Hence, we have to use nonlinear constrained optimization methods on factor graphs, which will be detailed in Section IV.

Furthermore, when the domains of states and landmark variables are not vector spaces but Lie-group manifolds, e.g. the Special Orthogonal group of rotations $SO3$ or the Special Euclidean group of rigid-body poses $SE3$ commonly used in robotics, the subtract operators between two group elements in the Mahalanobis distances in (5) are undefined. We instead replace $||x - \hat{x}||^2_\Sigma$ with $||\text{logmap}(x^{-1}\hat{x})||^2_\Sigma$. The logmap operator maps the group-element $x^{-1}\hat{x}$, as the difference between x and $\hat{x}$, to its corresponding Lie-algebra, where the Mahalanobis distance $||\cdot||^2_\Sigma$ is well-defined. Similarly, we compute derivatives of functions on Lie-group manifolds [5] to linearize the motion and measurement models. The δ-updates are now elements in the corresponding Lie-algebras, and the update equations in (7) for vector spaces are generalized for Lie-group manifolds using the left-trivialization operator with the exponential map:

$$\begin{aligned} x_i &\leftarrow x_i \exp(\delta x_i), \\ \text{and } l_j &\leftarrow l_j \exp(\delta l_j). \end{aligned} \tag{8}$$

III. Nonlinear Constrained Factors for General System Dynamics and Kinematics

The traditional factor graph formulation for estimation is limited to fully-actuated dynamics motion models with additive noise, due to its Gaussian assumption. In this section, we expand its capabilities to represent more general dynamics and kinematics in (1), e.g. under-actuated dynamics with possibly multiplicative noise.

A. Dynamics and Integration on Lie-group Manifolds

We follow the formulation in [14] and represent the general dynamics and kinematics in (1) as a differential equation over the state Lie-group manifold:

$$\dot{x} = x\widehat{F(x,u)} \tag{9}$$

where $x \in \mathcal{G}$ is a state Lie-group element, $\dot{x} \in \mathrm{T}_x\mathcal{G}$ is a tangent vector at x, $u \in \mathbb{R}^m$ is the control input, $x(0) = x_0$ is the initial condition, and $F : \mathcal{G} \times \mathbb{R}^m \to \mathbb{R}^n$ defines the dynamics directly in the vector space $\mathbb{R}^n$ isomorphic to the Lie algebra $\mathfrak{g}$. The isomorphism between $\mathfrak{g}$ and $\mathbb{R}^n$ is identified by a bijective hat operator map $\widehat{\ }: \mathbb{R}^n \to \mathfrak{g}$, and its inverse "vee" map $\vee : \mathfrak{g} \to \mathbb{R}^n$.

To obtain the state discretization approximation in (4), we need to integrate the dynamics equation on the manifold $\mathcal{G}$. If $\mathcal{G}$ is a vector space, (9) reduces to $\dot{x} = F(x,u)$, which we can integrate forward with any suitable integration scheme. However, the main challenge is that classical integration methods on vector spaces do not work for manifolds.

We summarize here the main ideas of Runge-Kutta Munthe-Kaas (RK-MK) technique for integration on Lie-group manifolds [14]. We apply a change of variables as follows:

$$x(t) = x_0 \exp\widehat{\xi(t)} \tag{10}$$

where $\xi \in \mathbb{R}^n$ are the **exponential coordinates** of the first kind, and $\xi(t)$ is the trajectory in exponential coordinates corresponding to $x(t)$, which satisfies the differential equation

$$\dot{\xi} = f(\xi, u) \tag{11}$$

starting from $\xi(0) = 0$. Here $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ specifies the dynamics directly in exponential coordinates, where the integration to obtain $\xi(t)$ can be done easily with standard methods on vector spaces. After $\xi(t)$ is computed, $x(t)$ can be obtained trivially from (10).

The final missing piece is an expression for $f(\xi, u)$ from $F(x,u)$. To obtain that, we first take the derivative of (10) with respect to time:

$$\dot{x} = x_0 \frac{d}{dt} \exp\widehat{\xi(t)} \tag{12}$$

The time derivative $\frac{d}{dt}\exp\widehat{\xi(t)}$ is caused by an instantaneous change $\dot{\xi}$ of $\xi \in \mathfrak{g}$. But, as $\exp\widehat{\xi(t)}$ is an element of $\mathcal{G}$, $\frac{d}{dt}\exp\widehat{\xi(t)}$ is also a tangent vector in $\mathrm{T}_{\exp\widehat{\xi(t)}}\mathcal{G}$, which is associated with another Lie-algebra element $\widehat{\omega} \in \mathfrak{g}$ through the left-trivialization:

$$\frac{d}{dt}\exp\widehat{\xi(t)} = \exp\widehat{\xi(t)}\widehat{\omega}. \tag{13}$$

The relationship between ω and $\dot{\xi}$ is linear:

$$\omega = \text{dexp}_\xi \dot{\xi} = \text{dexp}_\xi f(\xi, u) \tag{14}$$

where $\text{dexp}_\xi(\cdot) : \mathfrak{g} \to \mathfrak{g}$ is the linear map mapping $\dot{\xi}$ to $\widehat{\omega}$ (through $\frac{d}{dt}\exp\widehat{\xi(t)}$!). Its inverse $\text{dexp}^{-1}_\xi(\cdot)$, mapping $\widehat{\omega}$ back to $\dot{\xi}$, can be computed from formula (4.5) in [12], pg. 84 and is also documented in [4], [14].

After some manipulation of (12), (13) and (14), we finally obtain the following formula, which can be used to integrate $\xi(t)$ from (11):

$$f(\xi, u) = \text{dexp}^{-1}_\xi F(x,u) = \text{dexp}^{-1}_\xi F(x_0 \exp(\hat{\xi}), u) \tag{15}$$

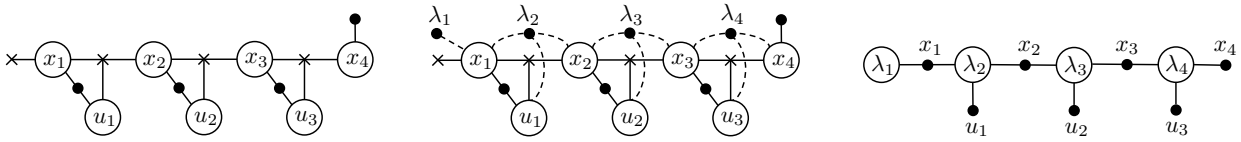

Fig. 2. A nonlinear constrained factor graph (left), and its SQP primal (center) and dual (right) linear graphs.

B. Dynamics Constrained Factors

We represent the general dynamics and kinematics (9) in factor graphs as constrained factors $F_i(x_i, u_i, x_{i+1})$ at every time step i as shown in Fig. 1. To derive that constraint, we integrate the system dynamics (9) from time t_i to t_{i+1} over the state manifold using the above RK-MK technique. According to (10), we have $x_{i+1} = x_i \exp(\widehat{\xi_i})$, hence,

$$\xi_i = \text{logv}(x_i^{-1} x_{i+1}) \tag{16}$$

where logv is the logmap operator composed with the $\vee$ map.

According to (11) and (15), ξ_i can be computed by integrating ξ in the following differential equation from 0 to $h = t_{i+1} - t_i$, with $\xi(0) = 0$:

$$\dot{\xi} \;=\; f(\xi, u_i) = \text{dexp}_{\xi}^{-1} F(x_i \exp(\hat{\xi}), u_i) \tag{17}$$

Our constrained factor $F_i(x_i, u_i, x_{i+1})$ depends on the specific scheme we use to integrate (17). To arrive at the constraint on x_i, u_i, and x_{i+1} only, we have to remove ξ_i from the picture. Essentially, we use (16) to replace every instance of $\xi_i = \xi(h)$ in the specific integration formula. For example, using the implicit trapezoidal scheme, the solution for $\xi_i = \xi(h)$ is given by:

$$\begin{aligned}
\xi_i = \xi(h) &= \xi(0) + \frac{h}{2}\left(f(\xi(0), u_i) + f(\xi(h), u_i)\right) \\
&= \frac{h}{2}\Big(\text{dexp}_{\xi(0)}^{-1} F(x_i \exp(\widehat{\xi(0)}), u_i) \\
&\qquad + \text{dexp}_{\xi(h)}^{-1} F(x_i \exp(\widehat{\xi(h)}), u_i)\Big) \\
&= \frac{h}{2}\Big(F(x_i, u_i) + \text{dexp}_{\xi_i}^{-1} F(x_{i+1}, u_i)\Big)
\end{aligned}$$

Using (16) to replace ξ_i, we have the following constraint

$$\begin{aligned}
\text{logv}(x_i^{-1} x_{i+1}) &= \frac{h}{2}\Big(F(x_i, u_i) \\
&\qquad + \text{dexp}^{-1}_{\text{logv}(x_i^{-1} x_{i+1})} F(x_{i+1}, u_i)\Big)
\end{aligned}$$

We simplify the constraint by applying an approximation: $\xi_i = \xi(h) \approx \xi(0) = 0$. With this approximation, we have $\text{dexp}_{\xi_i}^{-1} \approx I$ and arrive at the following constraint, which turn out to be similar to the implicit trapezoidal integration scheme on vector spaces:

$$x_{i+1} = x_i \exp \frac{h}{2}\left(\widehat{F(x_i, u_i)} + \widehat{F(x_{i+1}, u_i)}\right)$$

IV. Sequential Quadratic Programming for Nonlinear Constraints on Factor Graphs

As discussed in Section II-C, the standard Gauss-Newton (GN) and Levenberg-Marquardt (LM) methods cannot be used to optimize our graph due to the existence of nonlinear constrained factors. Hence, we choose to implement a factor graph version of Sequential Quadratic Programming (SQP), a well-known nonlinear programming method used in many optimal control packages [1]. A full description of SQP for nonlinear constrained factor graphs is beyond the scope of this paper. Here, we highlight some main points of our SQP implementation, which maintains the locality and sparsity properties of the graph for efficient computation.

Whereas GN and LM in factor graphs is straightforward, SQP is more involved with the introduction of *dual variables*. Let our general objective function be factorized as $J(\mathbf{x}) = \sum_i L_i(\mathbf{x}_i)$ and the constraints be $F_j(\mathbf{x}_j) = 0$, $j = 1..N_c$, where $\mathbf{x}_i, \mathbf{x}_j \subset \mathbf{x}$ are sets of variables in the cost functions L_i and the constraints F_j respectively. At k^{th} iteration, SQP solves the following quadratic program with linear constraints, derived from applying Newton method on the Karush–Kuhn–Tucker conditions of the Lagrangian function $\mathcal{L}(\mathbf{x}, \lambda_{1:N_c}) = \sum_i L_i(\mathbf{x}_i) + \sum_j \lambda_j F_j(\mathbf{x}_j)$ (see e.g. [27] for more details):

$$\begin{aligned}
\min_{\delta\mathbf{x}} \quad & \textstyle\sum_i \left(\frac{1}{2}\delta\mathbf{x}_i^T \nabla^2_{\mathbf{x}_i\mathbf{x}_i} L_i^{(k)} \delta\mathbf{x}_i + \delta\mathbf{x}_i^T \nabla_{\mathbf{x}_i} L_i^{(k)}\right) \\
& \qquad + \textstyle\sum_j \frac{1}{2}\lambda_j^{(k)} \delta\mathbf{x}_j^T \nabla^2_{\mathbf{x}_j\mathbf{x}_j} F_j^{(k)} \delta\mathbf{x}_j \\
\text{s.t.} \quad & \nabla_{\mathbf{x}_j} F_j^{(k)T} \delta\mathbf{x}_j + F_j^{(k)} = 0, \; \forall j = 1..N_c
\end{aligned} \tag{18}$$

where $\lambda_{1:N_c}$ are the *dual variables*.

We represent problem (18) in a *primal linear factor graph*. As usual, it contains linear factors encoding $\left(\frac{1}{2}\delta\mathbf{x}_i^T \nabla^2_{\mathbf{x}_i\mathbf{x}_i} L_i^{(k)} \delta\mathbf{x}_i + \delta\mathbf{x}_i^T \nabla_{\mathbf{x}_i} L_i^{(k)}\right)$ and $\nabla_{\mathbf{x}_j} F_j^{(k)T} \delta\mathbf{x}_j + F_j^{(k)} = 0$ in (18), which are the linearized versions of the original nonlinear factors $L_i(\mathbf{x}_i)$ and the nonlinear constraint factors $F_j(\mathbf{x}_j) = 0$ respectively. Besides, it also includes *new unconstrained factors* on constrained variables, $\frac{1}{2}\lambda_j^{(k)} \delta\mathbf{x}_j^T \nabla^2_{\mathbf{x}_j\mathbf{x}_j} F_j^{(k)} \delta\mathbf{x}_j$, which involves the current estimate $\lambda_j^{(k)}$ of the corresponding dual term. An example of our primal linear SQP graph for a simple control problem is shown in Fig. 2, where crosses denote constraints and dash lines connect constrained variables to the new factors. We solve this linear constrained graph using variable elimination with a special version of QR factorization which enforces linear constraints when eliminating constrained variables.

After solving for δX, we update the variables as usual following (8), and compute the next values $\lambda_{1:N_c}^{(k+1)}$ for the dual variables by solving the following linear system [27]:

$$\begin{aligned}
\sum_j \nabla_{\mathbf{x}} F_j^{(k)} \lambda_j^{(k+1)} &= \sum_i \left(\nabla^2_{\mathbf{xx}} L_i^{(k)} \delta\mathbf{x} + \nabla_{\mathbf{x}} L_i^{(k)}\right) \\
&\qquad + \sum_j \lambda_j^{(k)} \nabla^2_{\mathbf{xx}} F_j^{(k)} \delta\mathbf{x}.
\end{aligned} \tag{19}$$

While (19) has a global form involving all variables, we maintain the locality and sparsity by creating a new *dual linear factor graph*, which can be proved to be equivalent to (19). In our dual graph, each dual variable corresponds to a constrained factor in the primal linear graph, and each dual factor corresponds to a primal variable involving in the constraints. The Jacobian of a dual factor is the gradients of its corresponding constrained factors in the primal graph, $\nabla_{\mathbf{x}_j} F_j^{(k)}$, and its error term is computed from the gradients of all primal unconstrained factors of the corresponding constrained variables. An example of the dual graph for our control problem is shown in Fig. 2. We again solve this graph by variable elimination.

V. Experiments

A. Optimal Control for Obstacle Avoidance

We first apply our factor graph framework to solve an optimal control problem for obstacle avoidance on a simulated quadrotor. Fig. 3 shows the top view (left) and a 3D view (right) of our test environment, which includes a target point (the dark red circle), three wall obstacles (grey) and a cylinder obstacle (blue). The quadrotor (four red circles) starts at a certain point on the ground, and needs to get to the target after a predefined number of time steps while avoiding the obstacles. Each row in Fig. 3 shows an iteration of our SQP optimization process to compute the optimal control and trajectory, where the last row shows the final optimal solution and the quadrotor optimal state along the trajectory. As can be seen in the figure, the trajectory solution is improved at each step, i.e. the cost is decreasing at every step and the final state of the trajectory at the time horizon is closer to the goal, while not colliding to the obstacles.

The quadrotor state variable used in our experiments is $x = \{p, R, v\}$, where $p \in \mathbb{R}^3$ is the drone position, $R \in SO3$ its rotation, and $v \in \mathbb{R}^3$ its velocity in the world frame. For our outer loop controller, we use the drone body angular rate $\omega^b \in \mathbb{R}^3$ and total thrust $T \in \mathbb{R}$ as control inputs $u = \{\omega^b, T\}$, assuming that a fast inner loop controller to control w^b and T is available, as common in practice [3].

Our kinematics model of the quadrotor is as follows

$$\begin{aligned} \dot{p} &= v \\ \dot{R} &= R(w^b)^\vee \\ \dot{v} &= -\frac{T}{m} R e_3 + g \end{aligned}$$

where m is the drone's mass, g the gravity constant, and e_3 the unit vector $[0, 0, 1]^T$ such that $r_3 = Re_3$ is the third column of R, encoding up direction of the drone for the total thrust force in the world frame. Consequently, our Lie-algebra kinematics function $F(x, u)$, as required in (9), returns $[v, w^b, -\frac{T}{m} Re_3 + g]^T$ as its outputs.

For the final and stage-wise cost functions, we use potential-field types of functions for obstacle avoidance as common in the literature [9], [20]. More specifically, we use a cost function $L_{ig} = ||p_i - l_g||^2$ to encourage the drone at position p_i getting closer to the target at l_g. To enforce the drone at state x_i not to collide to an obstacle l_j, we use a cost function $L_{ij} = 1/d(x_i, l_j)$, where $d(x_i, l_j)$ computes the 2D ground-plane distance between the drone x_i and the object l_j. Furthermore, we also use unary control cost factors to minimize the control effort, and unary state cost factors to keep the drone always nearly leveled, and avoid undesirable states. To compute the Hessians of our factors for SQP, we use the finite difference method [1].

We vary the positions of the obstacles to study the robustness of our system. Our experiments show that the drone often manages to get to the target while avoiding the obstacles. However, for certain configuration of the environment and the cost function parameters, the drone might get stuck at a local minimum solution. Nevertheless, this is a well-known problem of using potential-field cost functions for obstacle avoidance in practice [22], [11].

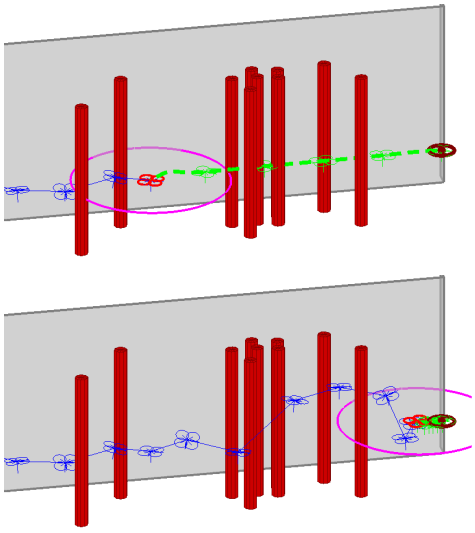

Fig. 4. 3D views of our combined estimation-control experiment. Blue: estimated trajectories. Green: optimal control trajectories. The wall to the right of the drone is omitted for clarity.

B. Combined Estimation and MPC for Obstacle Avoidance

We apply our unified framework for estimation and control problems to drive a simulated quad-rotor flying toward a target while avoiding obstacles at the same time. As shown in Fig. 5 and 4, our simulated quad-rotor is equipped with a bearing-range sensor (magenta circles) to detect obstacles in the environment, which includes two parallel walls and ten randomly generated vertical cylinders. We assume that the quadrotor can observe obstacles within a predefined range.

At each time step, after executing the first optimal control previously computed, the quadrotor estimates its state and the current map of the environment, then solves an MPC problem to get to the target while avoiding the known obstacles. Fig. 5 shows several steps of the entire process. The estimated past trajectories are shown in blue, while the optimal future trajectories are in green. As shown in the figure, the future trajectories adaptively change as the quadrotor moves and detects new objects obstructing its previously optimized path.

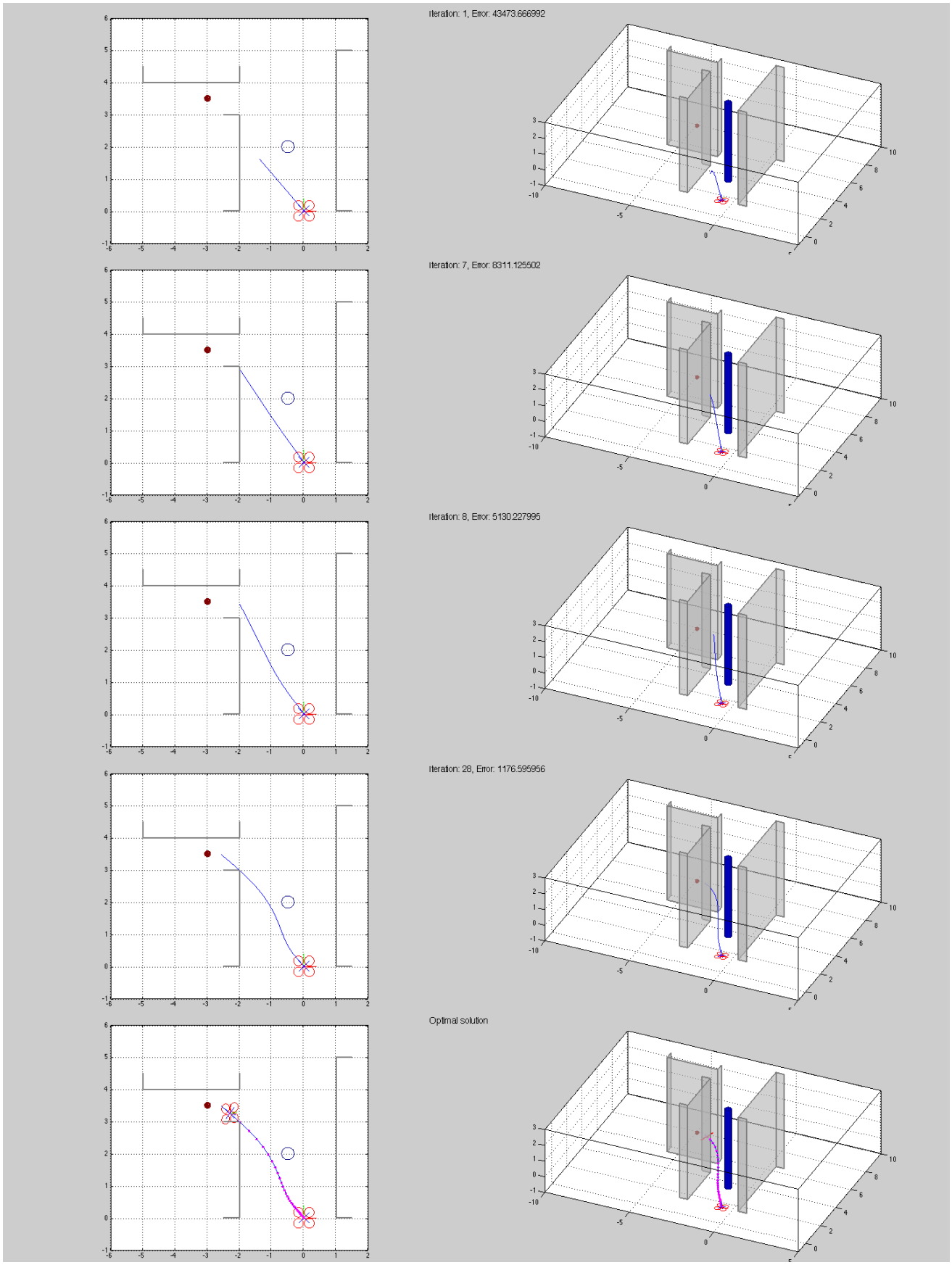

Fig. 3. Results of several SQP iterations on a factor graph to obtain the optimal control and trajectory for an obstacle avoidance problem

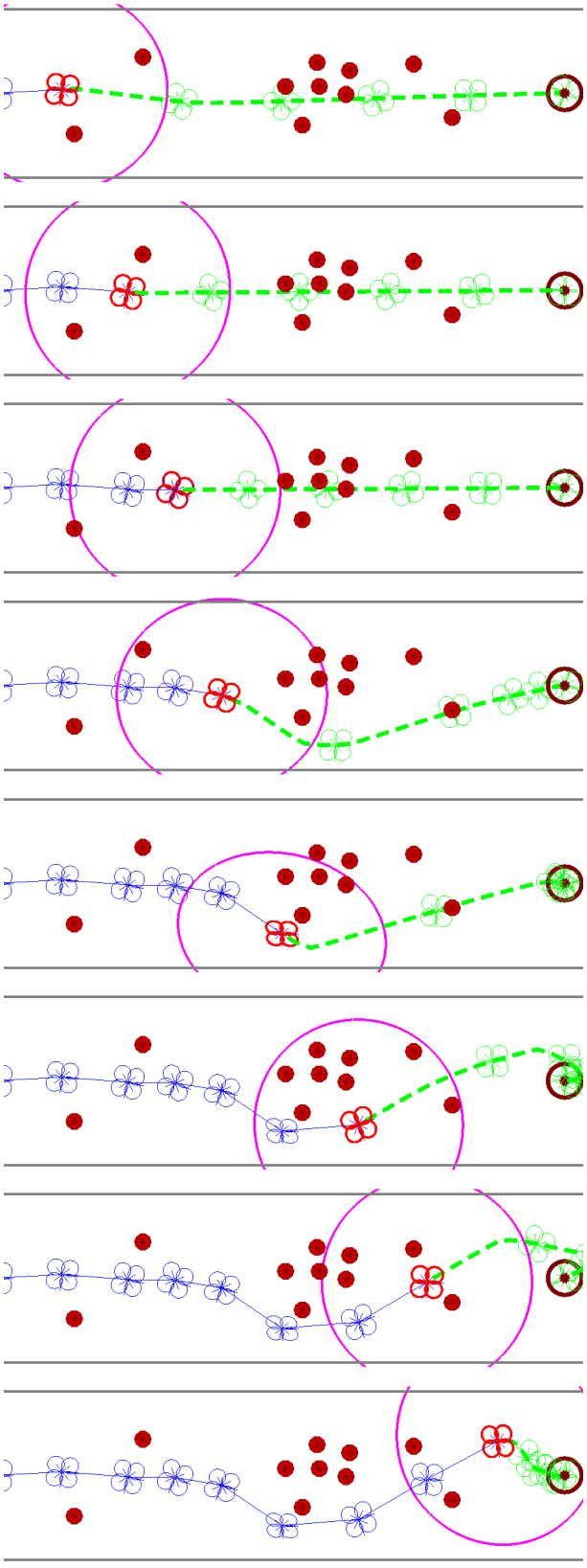

Fig. 5. Top views of our combined estimation-control experiment for obstacle avoidance on a quadrotor. Blue: estimated past trajectories. Green: future trajectories from optimal control. The future control trajectories are adaptively changed as the drone moves and observes new obstacles.

VI. Conclusion and Future Work

We have discussed a factor graph framework to solve both estimation and control problems, and applied it to solve an obstacle avoidance task on a quadrotor. Our framework potentially improves the system consistency, adaptiveness, robustness and agility in practice, because the same system dynamics, uncertainty models, internal dynamics parameters and external landmarks are utilized consistently between the two estimation and control processes. To meet that goal, we have extended the representation power of factor graphs to incorporate new system dynamics constrained factors by using RK-MK method to integrate over the state Lie-group manifolds. We have also developed factor-graph version of SQP, a common nonlinear constrained optimization method, to solve our new graphs with nonlinear constrained factors.

Our future work is to further extend the capabilities of factor graphs to deal with other types of constraints, e.g. inequality constraints and discrete constraints for deterministic optimal control problems. We also plan to generalize the framework for stochastic control problems, overcoming limitations of the state-of-the-art formulations in the field [33], [18]. Furthermore, the question of which practical benefits we can gain from combining estimation and control into the same framework is worth explored by itself to enhance our understanding about autonomous systems as a whole and further improve the system performance. An immediate benefit we expect to gain from explicitly modeling the dynamics in our new estimation framework is to obtain better constrained and more accurate estimates than traditional methods without dynamics modeling. Moreover, by formulating control problems in factor graphs, we are now ready to leverage advanced techniques in real-time [16] and large-scale estimation problems [25], [26], [15] to improve the performance of optimal control in complicated environments, e.g. with a large number of external landmarks for obstacle avoidance applications (Section V).

Acknowledgements

Duy-Nguyen Ta and Frank Dellaert are supported by an ARO MURI grant, award number W911NF-11-1-0046.

References

[1] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*, volume 19. Siam, 2010.

[2] Matthew Botvinick and Marc Toussaint. Planning as inference. *Trends in Cognitive Sciences*, 2012.

[3] P.J. Bristeau, F. Callou, D. Vissière, and N. Petit. The navigation and control technology inside the ar. drone micro uav. In *World Congress*, volume 18, pages 1477–1484, 2011.

[4] Elena Celledoni and Brynjulf Owren. Lie group methods for rigid body dynamics and time integration on manifolds. *Comput. meth. in Appl. Mech. and Eng.*, 19(3,4):421–438, 2003.

[5] Gregory S Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*, volume 2. Springer, 2011.

[6] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[7] F. Dellaert, J. Carlson, V. Ila, K. Ni, and C.E. Thorpe. Subgraph-preconditioned conjugate gradient for large scale slam. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[8] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–1203, Dec 2006.

[9] G. Droge and M. Egerstedt. Adaptive look-ahead for robotic navigation in unknown environments. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1134–1139. IEEE, 2011.

[10] H.F. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics & Automation Magazine*, Jun 2006.

[11] S.S. Ge and Y.J. Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, 13(3):207–222, 2002.

[12] E. Hairer, Ch. Lubich, and G. Wanner. *Geometric Numerical Integration*. Number 31 in Springer Series in Computational Mathematics. Springer-Verlag, 2006.

[13] V. Indelman, S. Wiliams, M. Kaess, and F. Dellaert. Information fusion in navigation systems via factor graph based incremental smoothing. *Robotics and Autonomous Systems*, 61(8):721–738, August 2013.

[14] A. Iserles, H. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna. Lie group methods. *Acta Numerica*, 9:215–365, 2000.

[15] Y.-D. Jian, D. Balcan, and F. Dellaert. Generalized subgraph preconditioners for large-scale bundle adjustment. In *Intl. Conf. on Computer Vision (ICCV)*, 2011.

[16] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31:217–236, Feb 2012.

[17] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering*, 82:35–45, 1960.

[18] Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

[19] Hilbert J Kappen, Wim Wiegerinck, and B van den Broek. A path integral approach to agent planning. *Autonomous Agents and Multi-Agent Systems*, 2007.

[20] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[21] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[22] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1398–1404. IEEE, 1991.

[23] F.R. Kschischang, B.J. Frey, and H-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2), February 2001.

[24] P. Maybeck. *Stochastic Models, Estimation and Control*, volume 1. Academic Press, New York, 1979.

[25] K. Ni, D. Steedly, and F. Dellaert. Out-of-core bundle adjustment for large-scale 3D reconstruction. In *Intl. Conf. on Computer Vision (ICCV)*, Rio de Janeiro, October 2007.

[26] Kai Ni and Frank Dellaert. Multi-level submap based slam using nested dissection. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[27] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, 1999.

[28] K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proc. of RSS*, 2012.

[29] R.F. Stengel. *Optimal control and estimation*. Dover Publications, 1994.

[30] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[31] E. Todorov. Optimality principles in sensorimotor control. *Nature neuroscience*, 7(9):907–915, 2004.

[32] E. Todorov. General duality between optimal control and estimation. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4286–4292. IEEE, 2008.

[33] M. Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1049–1056. ACM, 2009.

[34] Bart van den Broek, Wim Wiegerinck, and Bert Kappen. Graphical model inference in optimal control of stochastic multi-agent systems. *J. Artif. Intell. Res.(JAIR)*, 32:95–122, 2008.

[35] Paul J Werbos. Intelligence in the brain: A theory of how it works and how to build it. *Neural Networks*, 22(3):200–212, 2009.