

# Locally Optimal Estimation and Control of Cable Driven Parallel Robots using Time Varying Linear Quadratic Gaussian (LQG) Control

Gerry Chen, Seth Hutchinson, and Frank Dellaert

**Abstract**—We present a locally optimal tracking controller for Cable Driven Parallel Robot (CDPR) control based on a time-varying Linear Quadratic Gaussian (LQG) controller. In contrast to many methods which use fixed feedback gains, our time-varying controller computes the optimal gains depending on the location in the workspace and the future trajectory. Meanwhile, we rely heavily on offline computation to reduce the burden of online implementation and feasibility checking. Following the growing popularity of probabilistic graphical models for optimal control, we use factor graphs as a tool to formulate our controller for their efficiency, intuitiveness, and modularity. The topology of a factor graph encodes the relevant structural properties of equations in a way that facilitates insight and efficient computation using sparse linear algebra solvers. We first use factor graph optimization to compute a nominal trajectory, then linearize the graph and apply variable elimination to compute the locally optimal, time varying linear feedback gains. Next, we leverage the factor graph formulation to compute the locally optimal, time-varying Kalman Filter gains, and finally combine the locally optimal linear control and estimation laws to form a time-varying LQG controller. We compare the tracking accuracy of our time-varying LQG controller to a state-of-the-art dual-space feedforward controller on a 2.9m x 2.3m, 4-cable planar robot and demonstrate improved tracking accuracies of  $0.8^\circ$  and 11.6 mm root mean square error in rotation and translation respectively.

## I. INTRODUCTION

Given that cable robots have many desirable qualities, developing cable driven parallel robot (CDPR) tracking controllers that are accurate, robust, and easy to implement remains an important problem. CDPRs consist of a set of cables, through which forces are transmitted, connected to (usually) fixed winches on one end and to the floating end effector on the other. Due to the usage of cables to transmit forces, CDPR naturally scale well to large sizes [1]–[3], fast speeds/accelerations [4]–[6], and heavy loads [7]. Simultaneously, the inherent flexibility of cables makes achieving stiffness and accuracy challenging thereby limiting the utility of CDPR. Finally, despite the remarkable progress by a vibrant community, CDPR remain largely limited to academic settings, with the notable exception of *suspended* cable robots [8], [9] which pose a distinct set of challenges due to their not being redundantly actuated. We argue that the contrasting success of suspended cable robots compared with CDPR evidences that the complexity and difficulty in implementing CDPR tracking controllers is at least a large factor in curbing their adoption.

This work was supported by the National Science Foundation under Grant No. 2008302

Institute for Robotics and Intelligent Machines, College of Computing, Georgia Institute of Technology, Atlanta, GA, {gchen328, seth, fd27}@gatech.edu

Tracking control for CDPRs has been well studied since at least the late 1980's and has progressed from simple joint-space methods to advanced nonlinear, adaptive, and robust methods. The earliest examples of CDPR tracking control commanded position-controlled servos to the cable lengths computed by inverse kinematics, but these require tedious, critical calibration and are sensitive to modelling errors [11]. More advanced model-based techniques include operational-space PID, feedback linearization, sliding-mode control (SMC), and linear quadratic (LQR and LQG), and MPC, among others [12]. Linear approaches (e.g. PID) with feedforward terms to handle most of the nonlinearity are perhaps the most common [13]–[16], but the use of static gains can make them sensitive to operating point leading to non-uniform performance across the workspace and lack of foresight when approaching control limits [17]. Feedback linearization approaches have seen success in *suspended* cable robot designs but require special considerations for redundantly actuated CDPR [17]–[19]. Feedback linearization may also face difficulties modeling control limits. SMC [20] is perhaps most popular nonlinear approach for robust CDPR control, but is prone to chattering [21] due to control discontinuities. Although reducing chattering issues have been studied by a number of works [22]–[26], chattering does not appear to be completely resolved and these methods rely on adaptive algorithms to fallback onto linear controllers which, again, may not be workspace-aware. LQR and LQG have been primarily applied as tuning techniques for other controllers (e.g. tuning SMC [27] or feedback linearization gains [18]) and therefore inherit their limitations. Furthermore, applying LQR and LQG to systems that have been “pre-linearized” by other approaches tend to use fixed gains obtained using the infinite horizon LQR solution [27], [28]. Linear [29] and Nonlinear MPC [30], [31] are highly capable in providing consistent and robust performance near or even outside workspace boundaries [17], but MPC poses implementation challenges and must often make convexifying approximations to guarantee online convergence [17].

In this work, we propose a controller which is simple to implement but exhibits consistently accurate tracking performance across the workspace. We achieve this using factor graphs offline to solve for an optimal trajectory, locally optimal linear feedback gains, and a locally optimal linear estimator. Finally, we combine the feedback and estimator gains into an online controller to demonstrate the result on a real robot. Our contributions include the following:

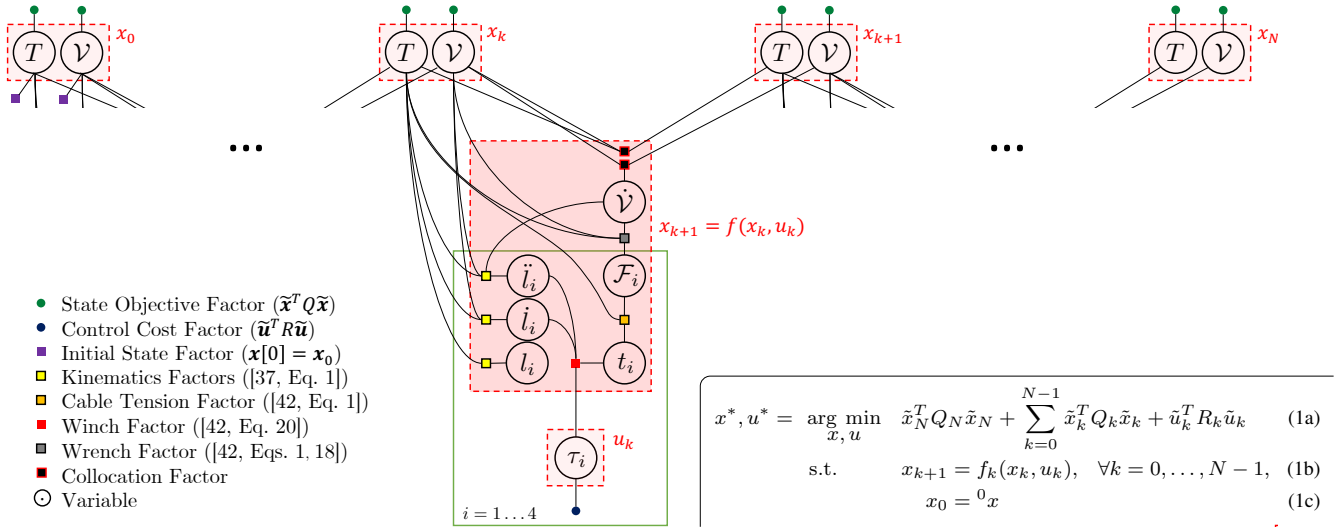


Fig. 1. Factor graph describing the full cable robot kinematics, dynamics, initial state, and state/control objectives, together with the optimization problem description (lower right). The kinematics and dynamics, contained in  $x_{k+1} = f_k(x_k, u_k)$ , are defined according to standard cable robot literature. Example references are given in the legend.  $T$  denotes the pose of the end effector;  $V$  denotes the twist of the end effector;  $\dot{V}$  denotes the twist acceleration of the end effector;  $l_i, \dot{l}_i, \ddot{l}_i, t_i$  denote the length, velocity, acceleration, and tension of cable  $i$  respectively;  $\tau_i$  denotes the torque commanded for motor  $i$ ; and  $F_i$  denotes the wrench applied to the end effector by cable  $i$ . Time indices are omitted for notational clarity. Together,  $T, V$  make up the state  $x$  and the torques  $\tau_0, \dots, \tau_3$  make up the control  $u$ .

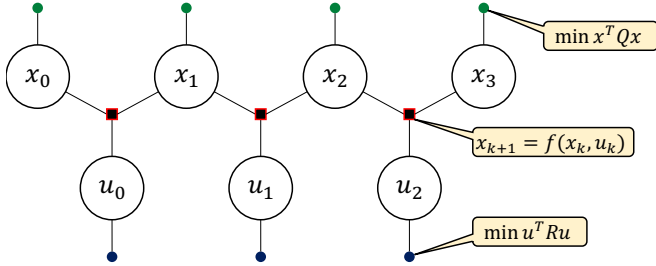


Fig. 2. Prototypical example of a factor graph describing the LQR problem with 4 timesteps, adapted with permission from [10]. Large circles are variables and small black circles/squares are factors.  $x_k$  represent state variables,  $u_k$  represent control variables, black squares represent dynamics constraint factors, and black circles represent optimal control objectives.

- 1) describe a factor graph representing the kinematics/dynamics, objectives, and/or stochastic elements of the CDRP tracking control problem;
- 2) solve the factor graph describing the optimal control problem to compute an optimal nominal trajectory;
- 3) show how a locally optimal linear time-varying estimator and controller can both be computed offline using factor graph elimination; and
- 4) combine the nominal trajectory, estimator, and controller to demonstrate our method on a real robot.

## II. APPROACH

We frame our approach around factor graphs because we believe them to be useful abstractions for describing trajectory optimization and controls topics. Simultaneously, factor graphs are highly efficient, visually insightful, and modular, allowing efficient incorporation of nearly any sensor, control objective, and/or constraint with minimal code modification

[32]. Probabilistic graphical models, including factor graphs, have seen great success in state estimation and perception [33] and are becoming increasingly popular as tools for optimal control [10], [32], [34]. We will attempt to describe the most pertinent concepts about factor graphs as we go, but additional introductions to factor graphs can be found in [33], [35].

In short, factor graphs are a graphical way to describe optimization problems where variables (unknowns to solve for) and factors (optimization objectives or constraints) are connected by an edge if the variable is involved in the objective/constraint. Fig. 2 shows a prototypical example of a factor graph describing the LQR problem.

Our algorithm consists of three offline stages followed by an online controller:

- 1) Compute a **nominal trajectory** that tracks the reference trajectory by creating and “solving” a factor graph. This is equivalent to using the iterative Linear Quadratic Regulator algorithm [36] for solving trajectory generation problems.
- 2) pre-compute **time-varying LQR gain matrices** by *linearizing* the graph around the nominal trajectory and using *variable elimination* on the linearized graph. Linearizing the graph is equivalent to applying a first-order Taylor expansion to convert the constrained *non-linear* least squares problem into a constrained *linear* least squares problem (linear time-varying LQR problem), and variable elimination is equivalent to using backward-induction to solve the Bellman equation.
- 3) Pre-compute **time-varying Kalman Filter gain matrices** by creating a graph with stochastic factors instead of cost/constraint factors, *linearizing* the graph around

the nominal trajectory, and marginalizing at each time step. The stochastic factors represent noise in the dynamics and/or measurements, and marginalizing at each time step is the Markov assumption that the Kalman Filter makes.

Finally, we combine the Kalman Filter and LQR gain matrices to form an online controller in the form of a time-varying LQG controller.

### A. Trajectory Generation

We seek to track the reference trajectory while also maintaining realizable control inputs. Least-squares objectives are used to minimize both the tracking error,  $\tilde{x} := x - d^x$ , and the control input objective,  $\tilde{u} := u - u_{mid}$ , where  $d^x$  is the desired trajectory and  $u_{mid} := \frac{1}{2}(u_{min} + u_{max})$  as described by [37] as an approximation for maximizing the margin to control limits. Constraints are given by the system dynamics. Our trajectory generation problem can then be expressed as in Fig. 1, which graphically depicts the trajectory generation problem as both a factor graph and the equivalent constrained nonlinear least squares problem. Notice that Fig. 1 contains equation (1).

We use the software library Georgia Tech Smoothing and Mapping (GTSAM) [38] to create and “solve” our factor graph, where “solve” is used to mean computing the solution to the equivalent constrained least squares problem. We use the Levenberg-Marquardt optimizer with variable elimination for the inner-loop linear solver (default solver in GTSAM). We will also make a note that, mathematically, Levenberg-Marquardt with variable elimination is identical to the iterative Linear Quadratic Regulator (iLQR) described by [36], although we run this optimization offline.

The computed solution to the factor graph is the nominal, feedforward trajectory and denoted  $x^*, u^*$ .

### B. Control

We use a time varying linear-quadratic regular (LQR) for our controller which is computed by linearizing the system around  $x^*, u^*$  and applying variable elimination.

1) *Linearizing the graph*: The linearization process is abstracted away and handled by GTSAM, but mathematically it can be described as applying a first-order Taylor expansion around  $x^*, u^*$  to convert the general optimal control problem (1) (which is transcribed as a constrained nonlinear least squares optimization) into the LQR problem (3) (which is transcribed as a constrained linear least squares problem). Upon linearizing, we will define  $\delta x_k := x_k - x_k^*$  and  $\delta u_k := u_k - u_k^*$ .

The result of linearizing is a graph with the same structure as in Fig. 1 except with all the factors representing either linear constraints or quadratic objectives. Mathematically, the linearized graph now represents a *linear* least squares problem (with linear equality constraints) which can be efficiently and exactly solved using variable elimination (which is equivalent to standard sparse linear algebra algorithms).

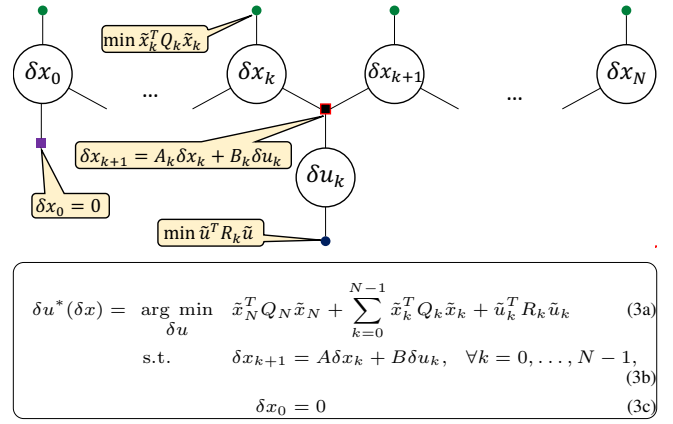


Fig. 3. After linearizing Fig. 1 and applying variable elimination to eliminate all the variables that are neither state nor control, we obtain this LQR graph, which is equivalent to (3). Green and navy blue factors represent state and control costs respectively, red-outlined-black factors represent dynamics constraints, and the purple factor represents the initial state constraint.

2) *Variable Elimination*: It is a well known result of linear-quadratic control [39] that the solution  $\delta u^*(\delta x)$  can be expressed as the linear control law:

$$\delta u_k^*(x_k) = K_k \delta x_k. \quad (2)$$

Notice that, since  $x^*, u^*$  satisfy (1), they are also guaranteed to satisfy (3). Thus when  $\delta x_k = 0$ ,  $\delta u_k^*(0) = 0$  so the affine term is zero.

Deriving  $K_k$  can be done in the standard LQR fashion, but we will summarize how it can also be easily done using factor graphs.

To obtain the LQR gains  $K_k$ , we apply the *variable elimination algorithm* on the linear graph in Fig. 3. The variable elimination algorithm is nothing more than an algorithmic method of solving for a variable and substituting the solution expression back into the system of equations. Although the factor graph machinery handles elimination for us, we can briefly summarize its underlying computation as performing the block QR factorization on a subsystem:

$$\begin{bmatrix} M_{11} & M_{12} & b_1 \\ M_{21} & M_{22} & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = Q \begin{bmatrix} R_{11} & S_{12} & d_1 \\ 0 & A_y & d_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

where  $M, b$  describes a linear least squares problem,  $x$  are the variables we seek to eliminate (no relation to  $x_k$ ),  $y$  consists of remaining variables (formally,  $y$  is the *separator*: set of variables not in  $x$  that share a factor with any of the variables in  $x$ ), and  $Q, R, S, A, d$  are results of the factorization. The top row denotes the solution  $x = R_{11}^{-1}(d_1 - S_{12}y)$  and the bottom row denotes the remaining system after substituting the solution for  $x$  back in:  $A_y y = d_2$ . A comprehensive descriptions of the variable elimination algorithm can be found in [33] for interested readers.

We first use GTSAM to eliminate all variables which are neither state nor control (“intermediate variables”): cable length, cable velocity, cable acceleration, tension, wrench, and twist acceleration. Reiterating, this is equivalent to solving for these variables first and substituting them back

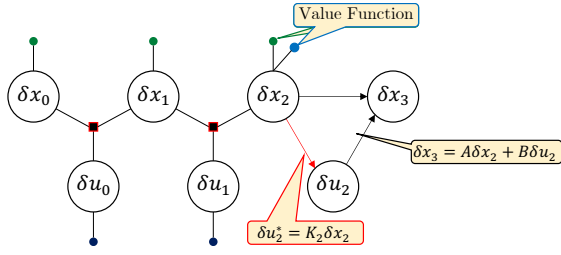


Fig. 4. Variable elimination is used to compute the optimal time-varying feedback control law starting at the final timestep and eliminating back to the beginning. This diagram depicts eliminating two variables:  $\delta x_3, \delta u_3$ . The blue factor is the cost that gets propagated upon “substituting” the optimal value back into the system, which makes up the value function.

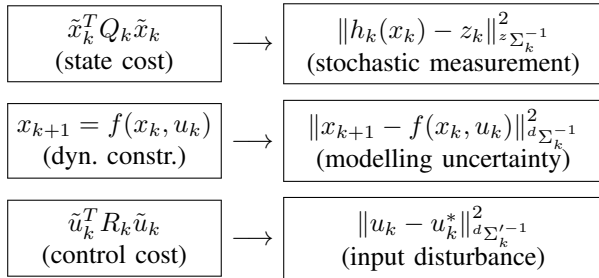
into the (3). After eliminating these variables, the resulting graph/system will have the form of Fig. 3 which can be equivalently described by the linear least squares problem (3). Also note that the computer has performed all the algebraic manipulation for us. Introducing e.g. additional parameter variables, periodicity constraint factors, state-dependent control limits are all possible with minimal designer effort.

Finally, to obtain the LQR gains  $K_k$ , we eliminate the states and controls one at a time in the order  $\delta x_N, \delta u_{N-1}, \delta x_{N-1}, \dots, \delta u_0, \delta x_0$ . It was proved in [10], [32] that this variable elimination process produces the finite horizon discrete LQR solution. As a brief intuition for the proof, this elimination follows the standard logic (backward induction on the Bellman equation) for deriving the finite horizon discrete algebraic ricatti equation: given the value function for  $x_{k+1}$ , solve for the  $u_k$  which minimizes the value function then substitute  $u_k^*$  back into the value function to obtain the new value function for  $x_k$ . Fig. 4 visually illustrates the elimination process.

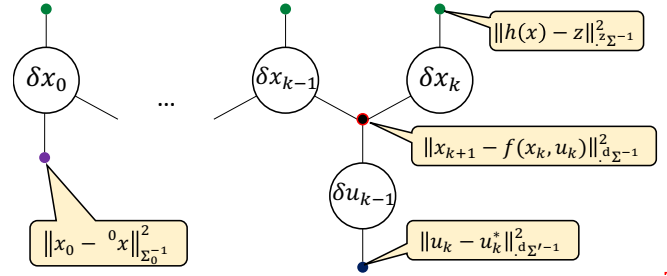
### C. Estimation

We can express our estimation problem by converting the factor graph in Fig. 1 to replace controls factors by estimation factors, followed by linearization and marginalization.

1) *Estimation Factor Graph*: To convert from the optimal control factor graph to the estimation factor graph, we make the following factor replacements:



where  $\|\square\|_{\Sigma^{-1}} := \square^T \Sigma^{-1} \square$  denotes the Mahalanobis norm;  $z_k := [l_k, \mathbf{l}_k]$  denotes the measurement vector at timestep  $k$ ;  $u_k^*$  and  $u_k$  denote the commanded and realized control torques vectors respectively;  $h(x_k)$  denotes the inverse kinematics to compute the cable length given the state; and



$$\hat{x} = \arg \min_{\delta x} \|\delta x_0\|_{\Sigma_0}^2 + \sum_{k=0}^{N-1} \left( \|H_k \delta x_k - z_k\|_{z_{\Sigma_k}^{-1}}^2 + \|\delta u_k - \delta u_k^*\|_{d_{\Sigma'}^{-1}}^2 + \|\delta x_{k+1} - (A_k \delta x_k + B_k \delta u_k)\|_{d_{\Sigma_{k+1}}^{-1}}^2 \right) \quad (7a)$$

$$\delta x_{k+1} = A_k \delta x_k + B_k (\delta u_k + v'_k) + v_k, \quad \begin{aligned} v'_k &\sim \mathcal{N}(0, d_{\Sigma'_k}) \\ v_k &\sim \mathcal{N}(0, d_{\Sigma_k}) \end{aligned} \quad (8a)$$

$$\delta z_k = H_k \delta x_k + w_k, \quad w_k \sim \mathcal{N}(0, z_{\Sigma_k}) \quad (8b)$$

Fig. 5. Nonlinear estimation factor graph (top) and the linearized estimation problem expressed as a minimization problem (middle) and system (bottom). After linearizing, we introduce the variables  $\delta z_k := z - z^*$  and  $z^* := h(x_k^*)$ , and  $H_k := \frac{\partial h_k}{\partial x_k} \big|_{x_k^*}$ .

$z_{\Sigma}, d_{\Sigma}, d_{\Sigma'}$  denote the measurement, dynamics, and input covariances respectively.

The factor graph now represents the system:

$$x_{k+1} = f(x_k, u_k + v'_k) + v_k, \quad v'_k \sim \mathcal{N}(0, d_{\Sigma'_k}) \quad (4)$$

$$v_k \sim \mathcal{N}(0, d_{\Sigma_k}) \quad (5)$$

$$z_k = h(x_k) + w_k, \quad w_k \sim \mathcal{N}(0, z_{\Sigma_k}) \quad (6)$$

2) *Linearization*: Next, we linearize and eliminate the intermediate variables the same way as in Section II-B to obtain the factor graph shown in Fig. 5 which is equivalent to the system (8) in Fig. 5.

After linearizing,  $v'_k$  can be combined with  $v_k$  by applying the transformation:

$$\delta x_{k+1} = A_k \delta x_k + B_k (\delta u_k + v'_k) \quad (9)$$

$$= A_k \delta x_k + B_k \delta u_k + B_k v'_k \quad (10)$$

where  $v'_k \sim \mathcal{N}(0, d_{\Sigma'})$  so  $v_k \sim \mathcal{N}(0, (B_k^T d_{\Sigma'}^{-1} B_k)^{-1})$  and  $d_{\Sigma_k} = (B_k^T d_{\Sigma'}^{-1} B_k)^{-1}$ . This is useful because most disturbances in our CDPR setup come in the form of parasitic torques/tensions in the cable as opposed to external wrenches acting directly on the end effector. Meanwhile, removing  $v'_k$  simplifies the factor graph and the notation.

3) *Marginalization to Extract Kalman Gains*: We now seek a Kalman Filter to estimate  $\hat{x}_k = E[\delta x_k]$ . Specifically, we would like to pre-compute the Kalman Gains for this filter to avoid needing to compute them online. Although, to the best of our knowledge, factor graphs cannot directly extract the Kalman Gains the way we were able to extract the LQR gains, they can nevertheless get us most of the way and provide intuition for the final expression.

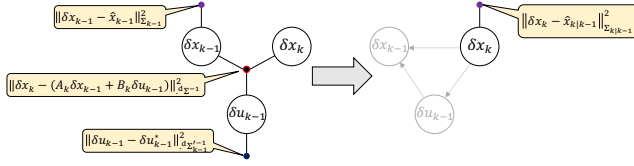


Fig. 6. Kalman Filter “Predict” step eliminates  $\delta x_{k-1}, \delta u_{k-1}$  to obtain  $\Sigma_k|_{k-1}$ .

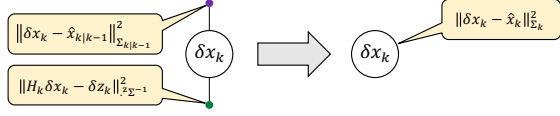


Fig. 7. Kalman Filter “Update” step eliminates  $\delta x_k$  to obtain  $\Sigma_k$ .

First, we can use marginalization to compute all the *a priori* and *a posteriori* covariances. In these cases, marginalization can be computed by simply eliminating every variable other than the one we seek the marginal of. Fig. 6 and 7 show how variable elimination is used to compute the new covariances after predicting and updating respectively. This computation is handled by GTSAM offline and  $\Sigma_{1|0}, \Sigma_1, \dots, \Sigma_{N|N-1}, \Sigma_N$  are the only results we need for the next step.

We now obtain expressions to compute  $\hat{x}$  given the covariance matrices we just computed. The equation to predict  $\hat{x}_{k|k-1}$  is straightforward since all noises are zero-mean:

$$\hat{x}_{k|k-1} = A_k \hat{x}_{k-1} + B_k \delta u_{k-1}. \quad (11)$$

The equation to update  $\hat{x}_k$  is less obvious, but more intuitive by observing Fig. 7: the optimal estimate for  $\delta x_k$  in Fig. 7 is the one which makes the best compromise between  $\|\delta x_k - \hat{x}_{k|k-1}\|_{\Sigma_{k|k-1}}^2$  and  $\|H_k \delta x_k - \delta z_k\|_{\Sigma_k}^2$ . We expect the optimal choice of  $\delta x_k$  to be a weighted mean of the two factors, and indeed with a bit of algebra it can be proven that it is:

$$\begin{aligned} \|\delta x_k - \hat{x}_k\|_{\Sigma_k}^2 &= \|\delta x_k - \hat{x}_{k|k-1}\|_{\Sigma_{k|k-1}}^2 \\ &\quad + \|H_k \delta x_k - \delta z_k\|_{\Sigma_k}^2 \\ \delta x_k^T \Sigma_k^{-1} \hat{x}_k &= \delta x_k^T \Sigma_{k|k-1}^{-1} \hat{x}_{k|k-1} + \delta x_k^T H_k^T \Sigma_k^{-1} \delta z_k \\ \Sigma_k^{-1} \hat{x}_k &= \Sigma_{k|k-1}^{-1} \hat{x}_{k|k-1} + H_k^T \Sigma_k^{-1} \delta z_k \\ \hat{x}_k &= \Sigma_k \left( \Sigma_{k|k-1}^{-1} \hat{x}_{k|k-1} + H_k^T \Sigma_k^{-1} \delta z_k \right) \end{aligned} \quad (12)$$

Finally, the predict and update steps can be combined to form a single state estimator update equation:

$$\hat{x}_k = {}^x K_k \hat{x}_{k-1} + {}^u K_k \delta u_{k-1} + {}^z K_k \delta z_k \quad (13)$$

where  ${}^x K_k := \Sigma_k \Sigma_{k|k-1}^{-1/2} A_k$ ,  ${}^u K_k := \Sigma_k \Sigma_{k|k-1}^{-1/2} B_k$ , and  ${}^z K_k := \Sigma_k H_k^T \Sigma_k^{-1/2}$  are pre-computed offline.

#### D. Combined Optimal Feedback Controller

Finally, we can use the standard LQG result: the stochastic optimal control policy is to feed the optimal estimator into



Fig. 8. Image of the CDPR we use for experimental results.

the optimal controller, for systems with only state- and control- independent noise. From (13), we obtain an optimal estimate  $\hat{x}_k$  for  $\delta x_k$  so LQG states that we may use  $\hat{x}_k$  in place of  $\delta x$  in (2):

$$\delta u_k^* = K_k \hat{x}_k \quad (14)$$

$$\hat{x}_k = {}^x K_k \hat{x}_{k-1} + {}^u K_k \delta u_{k-1} + {}^z K_k \delta z_k \quad (15)$$

Making the assumption that our controller always follows (14), we can also substitute  $\delta u_k^*$  in place of  $\delta u_k$ :

$$\hat{x}_k = {}^x K'_k \hat{x}_{k-1} + {}^z K_k \delta z_k \quad (16)$$

where  ${}^x K'_k := {}^x K_k + {}^u K_k K_k$ .

And finally, for completeness, we can substitute our  $\delta$  expressions to obtain our final time varying LQG controller:

$$\hat{x}_k = {}^x K'_k \hat{x}_{k-1} + {}^z K_k \delta z_k + k_k \quad (17)$$

$$u_k^* = K_k \hat{x}_k + u^* \quad (18)$$

where  $k_k := -{}^z K_k z_k^*$ .

The only computations that need to be run online are (18) and (17), which require only 3 matrix multiplications and 3 vector additions per update. Meanwhile,  $\mathcal{K}_k := K_k, {}^x K_k, {}^z K_k, {}^u K_k, u^*, k_k$  are pre-computed offline.

1) *Interpolation*: It may be desirable to run the controller at a faster rate than the set of precomputed  $\mathcal{K}$ , since  $\mathcal{K}$  may be only slowly changing while we typically want a controller update rate of at least several hundred Hz. We can devise a simple “zero-th” order interpolation:

$$\hat{x}(k dt + \Delta t) = {}^x K'_k \hat{x}_{k-1} + {}^z K_k z(k dt + \Delta t) + k_k \quad (19)$$

Although we found in our experiments that this interpolation was sufficient, more intelligent interpolation schemes may be investigated.

### III. EXPERIMENTS

We validate our approach experimentally by evaluating the trajectory tracking performance of our controller compared to a baseline controller implementation. We use the planar SE(2), 4-cable redundantly actuated robot shown in Fig. 8. We will give a brief description of the baseline controller then compare the tracking accuracies near the center of the workspace using a simple trajectory.



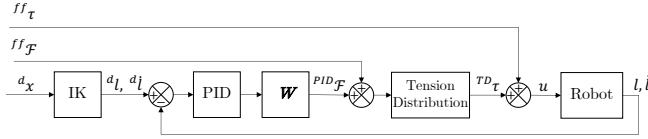


Fig. 9. Baseline controller, adapted from [40].

TABLE I  
GEOMETRIC, INERTIAL, AND FRICTION PARAMETERS

Frame	0	(2.815, 0.000)
Mounting	1	(2.845, 2.239)
Location (m)	2	(0.033, 2.225)
	3	(0.000, 0.000)
End Effector	0	(+0.063, -0.060)
Mounting	1	(+0.063, +0.060)
Location (m)	2	(-0.063, +0.060)
	3	(-0.063, -0.060)
End effector Inertia	$\mathcal{G}$	$\text{diag} \begin{pmatrix} 7.79 \text{ kg} \cdot \text{mm}^2 \\ 0.727 \text{ kg} \\ 0.727 \text{ kg} \end{pmatrix}$
Winch inertia		19.6 kg · mm <sup>2</sup>
Viscous friction	$F_v$	0.002 N · m · s
Static friction	$F_s$	0.12 N · m
tanh parameter	$\mu$	0.19 s/rad

### A. Baseline Controller

We model our baseline controller on the dual-space feed-forward controller in [40] as a standard controller used and demonstrated to be successful in tracking control of (redundantly actuated) CDRP. As compared to SMC-based and MPC-based controllers, PID-based controllers appear to be more widely used in redundantly actuated cable robots, of which [40] appears to be state of the art.

The baseline controller applies a PID controller in the joint-space, converts the correction to the operational-space to add inertial feedforward terms, applies a tension distribution algorithm to convert back into joint torques, and finally adds friction feedforward terms to generate the commanded torque. The controller is summarized in Fig. 9.

For the tension distribution algorithm, we apply some minor modifications since we have  $n + 1$  cables (4-cables for a 3DoF planar robot) whereas the tension distribution algorithm given in [40] is for  $n + 2$  cables. The  $n + 1$  cable case is significantly easier to solve than the  $n + 2$  cable case, and it suffices to exhaustively check each of the 8 control limits (4 upper-bound, 4 lower-bound) to find the 1-dimensional polytope (line segment) of feasible configurations. We then use [40, IV.A]: 2-Norm Optimal Solution. We were unable to successfully control the robot with the given objective  $\|u\|_2^2$ , but based on [37], using  $\|u - u_{mid}\|_2^2$  instead produced good performance.

### B. Robot and Controller Parameters

The robot and controller parameters are given in Tables I and II, respectively. Details on the measurement, calibration, and tuning of values are provided in the Appendix.

TABLE II  
LQG AND BASELINE CONTROLLER PARAMETERS

(a) LQG Parameters		(b) Baseline Controller Gains	
$\Sigma_0$	$\text{diag}([5.7^\circ \ 0.1 \text{ m} \ 0.1 \text{ m} \ 0 \ 0 \ 0])$	$K_p$	$3 \times 10^3 \text{ N/m}$
$z \Sigma$	$\text{diag}([(0.0018 \text{ m}) \times 4 \ (0.04 \text{ m/s}) \times 4])$	$K_i$	$5 \times 10^3 \text{ N/m} \cdot \text{s}$
$d \Sigma'$	$(0.059 \text{ N} \cdot \text{m}) I_{4 \times 4}$	$K_d$	$1 \times 10^1 \text{ N} \cdot \text{s/m}$
$Q$	$\text{diag}([10^2 \ 10^4 \ 10^4 \ 0 \ 0 \ 0])$		
$R$	$I_{6 \times 6}$		

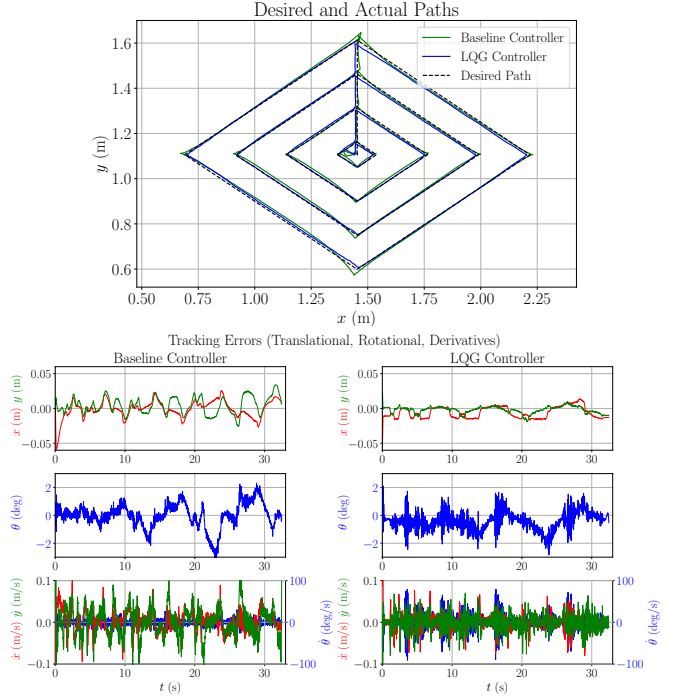


Fig. 10. Tracking accuracy comparison between baseline (bottom left) and our proposed controller (bottom right) for a test trajectory (top).

### C. Tracking Accuracy

To compare the tracking performances of the two controllers on a real robot, we use a trajectory composed of concentric diamonds in the center 1.5x1m area of the workspace with a maximum speed and acceleration of 0.5m/s and 1m/s<sup>2</sup> respectively. We use motion capture to measure the end effector poses and assess the tracking accuracy.

Fig. 10 and Table III compare the results of our controller against the baseline controller.

TABLE III  
TRACKING ACCURACIES NEAR WORKSPACE CENTER  
(ROOT MEAN SQUARE DEVIATION FROM  $d_x$ )

Dimension	Baseline	LQG
$\theta$ (deg)	0.9	0.8
$x$ (mm)	12.4	10.3
$y$ (mm)	12.8	5.4
$\dot{\theta}$ (deg/s)	15.2	26.1
$\dot{x}$ (mm/s)	44.3	36.1
$\dot{y}$ (mm/s)	44.3	33.4

#### IV. DISCUSSION

The results in Section III demonstrate that our LQG controller can perform favorably compared to the baseline state of the art PID-based controller.

Furthermore, we claim that, subjectively, the parameter tuning experience is much easier for our LQG controller since most parameters (noise parameters) are measured and the LQR gain matrices  $Q$  and  $R$  are more intuitive and forgiving than tuning gain matrices directly – indeed, LQR is often referred to as a technique for tuning gains [18], [27]. Although it is difficult to quantify how well we were able to tune our baseline gains, this can also be interpreted as evidence that tuning gain matrices is non-negligibly difficult.

Because our approach relies heavily on offline computation, we are also able to shift much of the work onto the computer. Performing the computation offline also simplifies workspace analysis since our nominal trajectory is pre-computed and can therefore be checked for feasibility before the trajectory is run.

Additionally, we believe that our proposed time-varying LQG controller is easier to implement since the embedded controller code consists of only 3 matrix multiplications and 3 vector additions per update. The rest of the computation is in a high-level programming language without time constraints.

Finally, the modularity afforded by the factor graph framework, which is highly general, allows designers to add a wide variety of additional constraints, sensors, and control objectives with minimal modification. Adding general constraints is described in [32], adding sensors is a common usage of GTSAM for sensor fusion, and additional control objectives such as signed distance field obstacles factors have been demonstrated in [41], [42].

#### V. CONCLUSIONS AND FUTURE WORK

In this work, we demonstrated how factor graphs can be used as a powerful tool to compute the optimal trajectory given quadratic state and cost objectives, compute the time varying Kalman Filter Gains, and compute the time varying LQR gains for optimal control of a redundantly actuated CDPR. We also showed how the time varying Kalman Filter and LQR are combined into an online time varying LQG controller. Finally, we successfully demonstrated our controller on a 4-cable planar CDPR and showed that it performs better across various regions in the workspace as compared to a baseline state-of-the-art dual-space feedforward controller. Qualitatively, we believe that the factor graph approach is easier to understand and implement, and that our optimal control formulation is easier to tune than the baseline controller.

Future works include spatial SE(3) CDPR control; parameter bias drift modeling; online parameter estimation and replanning using Bayes Trees for incremental solving (estimation using *smoothing* instead of *filtering*, and model predictive control respectively); additional sensor and control modes (e.g. tension feedback); and additional non-ideality modeling such as cable stretch and sag.

#### APPENDIX

##### PARAMETER CALIBRATION AND TUNING

The geometric, inertial, and friction parameters of the robot as listed in Table I. The geometry of the cable mounting points on both the fixed frame and movable end effector were directly measured, while the winch radii were estimated by a calibration procedure (described later). The translational inertia was measured directly with a scale and the rotational inertia estimated using a CAD model. The friction parameters were estimated by measuring winch no-load acceleration-velocity curves. Friction due to pulleys was assumed to be negligible compared to motor and winch frictions. Both controllers use the  $F_s \tanh(\mu \dot{l})$  model for static friction and we choose  $\mu = 0.19 \text{ s/rad}$  so that the static friction is modeled to reach  $0.9F_s$  by roughly  $0.1 \text{ m/s}$  ( $\tanh(0.19 \text{ s/rad} \cdot 7.87 \text{ rad/s}) \approx 0.9$ ).

Calibration of winch radii is done by regressing the inverse kinematic solution against the logged motor encoder values. This is done by recording the end effector pose and motor angles for an example trajectory. Then, inverse kinematics is used to compute the ground truth length of the cable as the distance between the cable mounting location on the winch and the pulley location. Finally, a linear regression is used to estimate winch radius and fine-tune the zero-position of the motor.

To tune the parameters for our LQG controller, we measured or estimated the covariances  $^z\Sigma$ ,  $^d\Sigma$ ,  $\Sigma_0$  and gave an intuitive guess for the LQR  $Q$  and  $R$  weighting matrices. We measured  $^z\Sigma$  and  $^d\Sigma$  by comparing the expected vs measured cable lengths, cable velocities, and end effector accelerations/motor torques from the same data as described in the previous paragraph. We assigned  $\Sigma_0$  to be  $0.1 \text{ m}$  as the initial placement error. We assigned  $R = I_{6 \times 6}$  and  $Q = \text{diag}([10^2 \ 10^4 \ 10^4 \ 0 \ 0 \ 0])$  which, intuitively, scales  $1 \text{ Nm}$  of torque to be comparable to  $0.57^\circ$  and  $0.1 \text{ mm}$  ( $10^{-2} \text{ rad}$ ,  $10^{-4} \text{ m}$ ) of rotation and translation error respectively while ignoring velocity tracking error.

To tune the gains  $K_p$ ,  $K_i$ ,  $K_d$  for our baseline dual-space feedforward controller, we applied manual methods. We increased  $K_p$  to a reasonable value without oscillations or shaking, then increased  $K_d$  until oscillations occurred, increased  $K_i$  until oscillations occurred, and finally fine-tuned the three. Although we attempted using well-known strategies such as Ziegler-Nichols or halving the ultimate gain, we found that those were gains were unusable.

#### REFERENCES

- [1] R. Nan, D. Li, C. Jin, Q. Wang, L. Zhu, W. Zhu, H. Zhang, Y. Yue, and L. Qian, "The five-hundred-meter aperture spherical radio telescope (fast) project," *International Journal of Modern Physics D*, vol. 20, no. 06, pp. 989–1024, 2011.
- [2] S. Bandyopadhyay, "Lunar crater radio telescope (LCRT) on the far-side of the moon," [https://www.nasa.gov/directorates/spacetechniac/2020\\_Phase\\_I\\_Phase\\_II/lunar\\_crater\\_radio\\_telescope/](https://www.nasa.gov/directorates/spacetechniac/2020_Phase_I_Phase_II/lunar_crater_radio_telescope/), April 2020.
- [3] P. Miermeister, M. Lächele, R. Boss, C. Masone, C. Schenk, J. Tesch, M. Kerger, H. Teufel, A. Pott, and H. H. Bühlhoff, "The cablerobot simulator large scale motion platform based on cable robot technology," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3024–3029.

- [4] G. Chen, S. Baek, J.-D. Florez, W. Qian, S. won Leigh, S. Hutchinson, and F. Dellaert, "Extended version of gtgraffiti: Spray painting graffiti art from human painting motions with a cable driven parallel robot," 2021.
- [5] K. Müller, C. Reichert, and T. Bruckmann, "Analysis of a real-time capable cable force computation method," in *Cable-Driven Parallel Robots*, A. Pott and T. Bruckmann, Eds. Cham: Springer International Publishing, 2015, pp. 227–238.
- [6] A. Rajnoha, "CableEndy juggler - cable-driven parallel robot (B&R Automation)," <https://youtu.be/7HNAL8ZKdyM>, 10 2017.
- [7] F. Shahmiri and R. Gentry, "A survey of cable-suspended parallel robots and their applications in architecture and construction," *Blucher Design Proceedings*, vol. 3, no. 1, pp. 914 – 920, 2016.
- [8] L. Cone, "Skycam: An aerial robotic camera system," 1985.
- [9] X. Tang and Z. Shao, "Trajectory generation and tracking control of a multi-level hybrid support manipulator in fast," *Mechatronics*, vol. 23, no. 8, pp. 1113–1122, 2013.
- [10] G. Chen and Y. Zhang, "LQR control using factor graphs," <https://gtsam.org/2019/11/07/lqr-control.html>, Nov 2019, note: Superseding ICRA 2021 paper pending review.
- [11] P. D. Campbell, P. L. Swaim, and C. J. Thompson, "Charlotte™ robot technology for space and terrestrial applications," *SAE Transactions*, vol. 104, pp. 641–648, 1995. [Online]. Available: <http://www.jstor.org/stable/44611968>
- [12] M. A. Khosravi and H. D. Taghirad, "Dynamic modeling and control of parallel robots with elastic cables: Singular perturbation approach," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 694–704, 2014.
- [13] J. Lamaury and M. Gouttefarde, "Control of a large redundantly actuated cable-suspended parallel robot," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 4659–4664.
- [14] M. Gouttefarde, J. Lamaury, C. Reichert, and T. Bruckmann, "A versatile tension distribution algorithm for  $n$ -dof parallel robots driven by  $n + 2$  cables," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1444–1457, Dec 2015.
- [15] W. Shang, B. Zhang, B. Zhang, F. Zhang, and S. Cong, "Synchronization control in the cable space for cable-driven parallel robots," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4544–4554, 2019.
- [16] W. Shang, B. Zhang, S. Cong, and Y. Lou, "Dual-space adaptive synchronization control of redundantly-actuated cable-driven parallel robots," *Mechanism and Machine Theory*, vol. 152, p. 103954, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0094114X20301750>
- [17] J. Cavalcanti santos, "Model predictive tracking control of cable-driven parallel robots : From concept to real-time validation," Ph.D. dissertation, 2020, 2020MONT017. [Online]. Available: <http://www.theses.fr/2020MONT017/document>
- [18] M. H. Korayem, M. Yousefzadeh, and B. Beyranvand, "Dynamics and control of a 6-dof cable-driven parallel robot with visco-elastic cables in presence of measurement noise," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 1, pp. 73–95, 2017. [Online]. Available: <https://doi.org/10.1007/s10846-017-0546-1>
- [19] M. Korayem, H. Tourajizadeh, and M. Bamdad, "Dynamic load carrying capacity of flexible cable suspended robot: Robust feedback linearization control approach," *Journal of Intelligent and Robotic Systems*, vol. 60, pp. 341–363, 12 2010.
- [20] J.-J. E. SLOTTINE, "Sliding controller design for non-linear systems," *International Journal of Control*, vol. 40, no. 2, pp. 421–434, 1984. [Online]. Available: <https://doi.org/10.1080/00207178408933284>
- [21] J. Niu, Q. Yang, G. Chen, and R. Song, "Nonlinear disturbance observer based sliding mode control of a cable-driven rehabilitation robot," in *2017 International Conference on Rehabilitation Robotics (ICORR)*, 2017, pp. 664–669.
- [22] C. Schenk, H. H. Bühlhoff, and C. Masone, "Robust adaptive sliding mode control of a redundant cable driven parallel robot," in *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)*, 2015, pp. 427–434.
- [23] C. Schenk, C. Masone, A. Pott, and H. H. Bühlhoff, "Application of a differentiator-based adaptive super-twisting controller for a redundant cable-driven parallel robot," in *Cable-Driven Parallel Robots*, C. Goselin, P. Cardou, T. Bruckmann, and A. Pott, Eds. Cham: Springer International Publishing, 2018, pp. 254–267.
- [24] *Design and Application of Chattering-Free Sliding Mode Controller to Cable-Driven Parallel Robot Manipulator: Theory and Experiment*, ser. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. Volume 2: 34th Annual Mechanisms and Robotics Conference, Parts A and B, 08 2010. [Online]. Available: <https://doi.org/10.1115/DETC2010-29153>
- [25] Y. Wang, J. Chen, F. Yan, K. Zhu, and B. Chen, "Adaptive super-twisting fractional-order nonsingular terminal sliding mode control of cable-driven manipulators," *ISA Transactions*, vol. 86, pp. 163–180, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0019057818304440>
- [26] E. Picard, E. Tahoumi, F. Plestan, S. Caro, and F. Claveau, "A new control scheme of cable-driven parallel robot balancing between sliding mode and linear feedback," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9936–9943, 2020, 21st IFAC World Congress. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2405896320334716>
- [27] M. H. Korayem, H. Tourajizadeh, M. Jalali, and E. Omid, "Optimal path planning of spatial cable robot using optimal sliding mode control," *International Journal of Advanced Robotic Systems*, vol. 9, no. 5, p. 168, 2012. [Online]. Available: <https://doi.org/10.5772/52892>
- [28] C. Lambert, M. Nahon, and D. Chalmers, "Implementation of an aero-stat positioning system with cable control," *IEEE/ASME Transactions on Mechatronics*, vol. 12, no. 1, pp. 32–40, 2007.
- [29] J. C. Santos, A. Chemori, and M. Gouttefarde, "Model predictive control of large-dimension cable-driven parallel robots," in *Cable-Driven Parallel Robots*, A. Pott and T. Bruckmann, Eds. Cham: Springer International Publishing, 2019, pp. 221–232.
- [30] C. Song and D. Lau, "Workspace-based model predictive control for cable-driven robots," *IEEE Transactions on Robotics*, pp. 1–20, 2022.
- [31] M. Katliar, J. Fischer, G. Frison, M. Diehl, H. Teufel, and H. H. Bühlhoff, "Nonlinear model predictive control of a cable-robot-based motion simulator," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9833–9839, 2017, 20th IFAC World Congress. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2405896317313678>
- [32] S. Yang, G. Chen, Y. Zhang, H. Choset, and F. Dellaert, "Equality constrained linear optimal control with factor graphs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9717–9723.
- [33] F. Dellaert, M. Kaess *et al.*, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.
- [34] F. Dellaert, "Factor graphs: Exploiting structure in robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 141–166, 2021. [Online]. Available: <https://doi.org/10.1146/annurev-control-061520-010504>
- [35] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.
- [36] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO., INSTICC.* SciTePress, 2004, pp. 222–229.
- [37] A. Pott, T. Bruckmann, and L. Mikelsons, "Closed-form force distribution for parallel wire robots," in *Computational Kinematics*, A. Kecskeméthy and A. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 25–34.
- [38] F. D. *et al.* (2022) Georgia Tech smoothing and mapping (GTSAM). <https://github.com/borglab/gtsam>.
- [39] E. Todorov, "Optimal control theory," in *Bayesian Brain: Probabilistic Approaches to Neural Coding*. MIT Press, 2006, pp. 269–298. [Online]. Available: <https://courses.cs.washington.edu/courses/cse528/09sp/optimality.chapter.pdf>
- [40] M. Gouttefarde, J. Lamaury, C. Reichert, and T. Bruckmann, "A versatile tension distribution algorithm for  $n$ -dof parallel robots driven by  $n + 2$  cables," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1444–1457, 2015.
- [41] M. Mukadam, J. Dong, F. Dellaert, and B. Boots, "STEAP: simultaneous trajectory estimation and planning," *Autonomous Robots*, vol. 43, no. 2, pp. 415–434, 2019. [Online]. Available: <https://doi.org/10.1007/s10514-018-9770-1>
- [42] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time Gaussian process motion planning via probabilistic inference," vol. 37, no. 11, 2018, pp. 1319–1340.