# Batch and Incremental Kinodynamic Motion Planning using Dynamic Factor Graphs

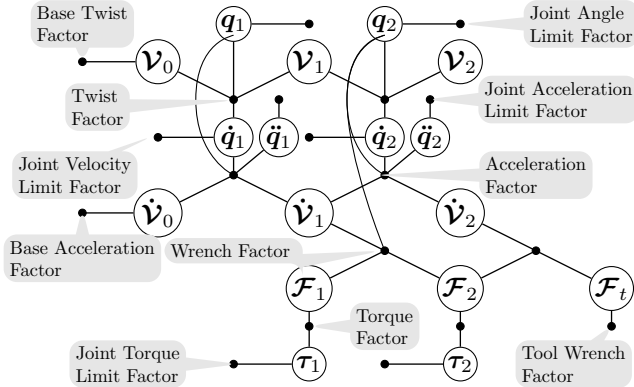Author Names Omitted for Anonymous Review. Paper-ID [1202]



Fig. 1: The dynamic factor graph (DFG) for a RR manipulator, where black dots represent factors, and circles represent variables. Figure adapted from [48].
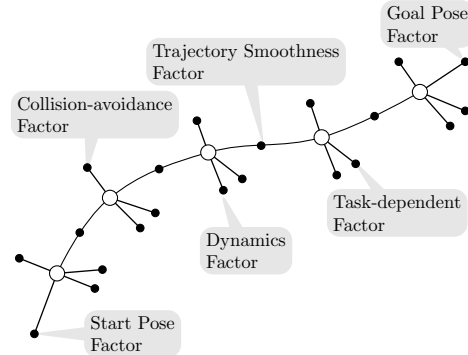


Fig. 2: Kinodynamic motion planning factor graph. To simplify the notation, we use a dynamics factor to represent all the dynamic constraints at each time slice in the factor graph.

*Abstract*—**This paper presents a kinodynamic motion planner that is able to produce energy efficient motions by taking the full robot dynamics into account, and making use of gravity, inertia, and momentum to reduce the effort. Given a specific goal state for the robot, we use factor graphs and numerical optimization to solve for an optimal trajectory, which meets not only the requirements of collision avoidance, but also all kinematic and dynamic constraints, such as velocity, acceleration and torque limits. By exploiting the sparsity in factor graphs, we can solve a kinodynamic motion planning problem efficiently, on par with existing optical control methods, and use incremental elimination techniques to achieve an order of magnitude faster replanning.**

## I. INTRODUCTION

Kinodynamic motion planning is an important and active research area in robotics [12, 13, 31, 21, 30, 38, 42]. Purely kinematic motion planning focuses on finding a trajectory through a robot's configuration space that satisfies multiple criteria such as collision-avoidance, obeying joint limits, and honoring smoothness constraints. However, to produce energy-efficient motions and satisfy dynamic constraints such as force and torque limits, we need *kinodynamic* motion planners that take the full robot dynamics into account. In addition, many tasks involving large payloads, fast motions, or both, require inherently dynamic behaviors that cannot be achieved by reasoning about kinematics alone.

In this paper we consider the kinodynamic motion planning as an optimal control (OC) problem, using factor graphs as a convenient and intuitive representation that also provides distinct computational advantages. In using OC-techniques we are inspired by the recent work by Zhao *et al.* [50] for manipulators, and many other OC-style approaches from the dynamic walking literature [28, 41]. In using a factor graph representation we follow the lead of GPMP2 [36], which solves the *kinematic* motion planning problem efficiently using a sparse nonlinear solver [9, 10], and performs fast replanning by taking advantage of an incremental solver [25].

To incorporate dynamics, we represent kinodynamic constraints with a *dynamic factor graph* (DFG) as introduced by Xie and Dellaert [48]. Fig. 1 shows the DFG of a RR manipulator, where the equations of motion are represented as factors, and variables involved in the equations are represented as nodes in the graph. We then construct an optimal control problem by adding factors representing all the usual trajectory optimization objectives, as well as factors enforcing the dynamics between successive time-slices, as shown in Figure 2.

By combining optimal control and factor graphs we obtain an intuitive, fast kinodynamic motion planner that is on par with the fastest OC methods, and adds efficient, incremental kinodynamic replanning. The use of GTSAM [9, 10] as an optimizer ensures that we are exploiting the benefits of both sparsity and automatic differentiation, which was shown to be crucial to achieving state of the art performance by Zhao *et al.* [50]. Fast re-planning is useful in many contexts, e.g., model-predictive control (MPC). And, as far as we know, no existing kinodynamic motion planning methods can perform fast replanning beyond warm-starting as used in MPC.

## II. RELATED WORK

Kinodynamic motion planning has a rich history. The term was coined in the papers by Donald *et al.* [12, 13], and the well known Rapidly-exploring Random Trees (RRTs) were in fact developed with kinodynamic constraints in mind from the very start [32, 31], as were the Expansive Space Tree algorithms by Hsu *et al.* [20, 21]. In many cases, however, sampling-based methods such as RRTs and Probabilistic Road Maps (PRM) [27, 2] are used for purely kinematic motion planning. Satisfying both kinematic *and* dynamic constraints simultaneously makes the kinodynamic motion planning problem more challenging. Instead of planning in the configuration space., we now need to plan in the state space, whose dimension is twice that of the configuration space.

More importantly, kinodynamic sampling-based methods need both good *distance metrics* and efficient *steering methods* [7, 1]. A distance metric is needed in RRT-style methods to choose which part of the tree to expand, and simple metrics like the Euclidean distance do not incorporate any knowledge of the system dynamics [38]. The other issue is the lack of computationally efficient steering methods to connect two states to produce a segment between them in a way that satisfies all differential constraints [29]. LQR-RRT* [38] addresses both of these issues by linearizing the system dynamics and using the LQR cost function as the distance metric, similar to the earlier work by Glassman and Tedrake [16]. Webb *et al.* [46] improve on this by connecting any pair of states exactly via a fixed-final-state-free-final-time OC formulation for systems with controllable linear dynamics. Unfortunately, this still requires a linearized version of the dynamics system. More recently, the AO-RRT method proposed by Hauser *et al.* [19] and the SST and SST* methods presented by Li *et al.* [49] are able to solve kinodynamics motion planning problems for systems with a few degrees of freedom.

Optimization-based motion planning methods have recently gained in popularity as they complement sampling-based methods [34, 40, 42]. Hence, while sampling methods sometimes produce inconsistent and non-smooth solutions [37], they are often used to provide a feasible path, after which trajectory optimization is used to smooth out the trajectory. Examples of optimization-based methods include CHOMP [34], STOMP [26], TrajOpt [22] [23], GPMP [35], and GPMP2 [36]. However, when applied to kinodynamic motion planning, the non-convex nature of the objective function, caused by dynamic constraints and constraints representing the obstacle-free region [3], makes these prone to converge to local minima unless properly initialized. To deal with the difficulty of finding optimal solutions that satisfy all the constraints simultaneously, Sintov *et al.* [43] generate random points within the allowable regions of the free parameters to find a set of feasible solutions, and use a gradient descent approach to refine the solution to a nearby local optimum. However, this approach is computationally demanding.

The kinodynamic motion planning problem can also be tackled with numerical optimal control methods [3], which

enable easy incorporation of constraints and straightforward definition of objective functions. Based on optimal control techniques, Zhao *et al.* [50] present a motion planning approach, which uses pseudospectral method for trajectory discretization and interior point method with automatic differentiation for optimization. They solved the motion planning problem of a 6-axis robot with dynamic constraints, and it is shown to be much faster than existing works such as [6, 11]. A key element to achieving state of the art performance is the use of automatic differentiation over numerical or symbolic differentiation. Optimal control methods have also been applied to dynamic walking [24, 47] and humanoid robots motion planning [28, 41].

Finally, there exists an alternative way to deal with kinodynamic motion planning, which is to decouple the problem into a geometric part followed by re-timing. For instance, Hauser *et al.* [17] use a dynamic interpolation method followed by a convex time-scaling optimization problem. Pham *et al.* [39] solve kinodynamic constraints with Admissible Velocity Propagation (AVP) based on Time-Optimal Path Parameterization (TOPP) [4], which can be combined with sampling-based method such as RRT. However, this method may lead to sub-optimal solutions which sometimes may even be dynamically unfeasible [5, 50].

## III. REVIEW: MOTION PLANNING WITH FACTOR GRAPHS

In this part, we review GPMP2 [36], a purely kinematic motion planning method that optimizes for a collision-free *and* smooth trajectory subject to a set of constraints. GPMP2 uses factor graphs to represent the corresponding objective functions, by making use of an incremental Bayes tree solver [25], which supports fast replanning under changing conditions and/or updated goal states.

### A. Fixed Start and Goal Poses

Since many applications demand a manipulator to move the end-effector from a start pose to a goal pose, both of the pose constraints are added to the motion planning problem. The cost of start and goal pose constraints can be expressed as a function of the state vectors $x_s$ and $x_g$:

$$h_1(x_s) = f(x_s) - p_s \tag{1}$$
$$h_2(x_g) = f(x_g) - p_g \tag{2}$$

where $f()$ is the forward kinematics which maps any configuration to a workspace (this definition applies to the rest of the paper), $p_s$ is the desired start pose and $p_g$ is the desired goal pose of the end effector.

### B. Collision-avoidance

To efficiently check collision for an arbitrarily shaped robot body, GPMP2 uses the method presented in CHOMP [34], in which a robot's body is simplified to be a set of spheres, and the distance of the body to any point in the workspace is given by the distance to the center of the sphere minus its radius. The obstacles are represented by precomputed Signed Distance Field (SDF) matrices. The distance is positive if a

point is outside of the obstacle, zero if it is on the surface, and negative when it lies inside the obstacle.

Hence, the obstacle cost function $h_3(x_i)$ is obtained by computing the signed distances for the robot in state $x_i$:

$$h_3(x_i) = c_{sdf}(f(x_i)) \tag{3}$$

where $f()$ is the forward kinematics, and $c_{sdf}()$ is defined as the hinge loss:

$$c_{sdf}(z) = \begin{cases} -d(z) + \varepsilon & if \ d(z) \leq \varepsilon \\ 0 & o/w \end{cases} \tag{4}$$

where $d(z)$ is the signed distance from any point z in the workspace to the closest obstacle surface, which is obtained from a pre-computed SDF. Here we add a threshold $\varepsilon$ to prevent the robot from getting too close to obstacles.

### C. Task-dependent Kinematic Constraints

In addition to the requirements of goal accomplishment, collision avoidance, and smoothness, the planner will sometimes have to satisfy user-specified constraints such as keeping the end-effector at a certain pose along the trajectory. For example, the cost of an end effector pose constraint can be expressed as a function of the state vector $x_i$:

$$h_4(x_i) = f(x_i) - p_e \tag{5}$$

where $f()$ is the forward kinematics, and $p_e$ is the desired end effector pose of the robot.

### D. Trajectory Smoothness

Finally, [36] add a Gaussian process (GP) smoothness prior with cost function

$$h_5(x_{i-1}, x_i) = x_i - \Phi(t_i, t_{i-1})x_{i-1} \tag{6}$$

and covariance matrix

$$\Sigma_5 = \begin{bmatrix} \frac{1}{3}\Delta t_i^3 \mathbf{Q}_C & \frac{1}{2}\Delta t_i^2 \mathbf{Q}_C \\ \frac{1}{2}\Delta t_i^2 \mathbf{Q}_C & \Delta t_i \mathbf{Q}_C \end{bmatrix}, \tag{7}$$

where $Q_c$ is the power-spectral density matrix associated with the GP, and the state transition matrix

$$\Phi(t_i, t_{i-1}) = \begin{bmatrix} 1 & \Delta t_i \\ 0 & 1 \end{bmatrix} \tag{8}$$

is associated with a constant velocity assumption between times $t_{i-1}$ and $t_i$, and $\Delta t_i = t_i - t_{i-1}$.

### E. Optimization with Factor Graphs

Given the cost functions associated with the constraints mentioned above, the trajectory optimization problem is formulated as follows:

$$x^* = \underset{x}{\operatorname{argmin}}\{\|h_1(x_s)\|_{\Sigma_1}^2 + \|h_2(x_g)\|_{\Sigma_2}^2 +$$
$$\sum_i (\|h_3(x_i)\|_{\Sigma_3}^2 + \|h_4(x_i)\|_{\Sigma_4}^2 + \|h_5(x_{i-1}, x_i)\|_{\Sigma_5}^2)\}. \tag{9}$$

where the $\Sigma$ terms specify the covariance for each cost function in the optimization. A smaller $\Sigma$ value imposes larger penalty for the constraint. Hence, the optimization result will
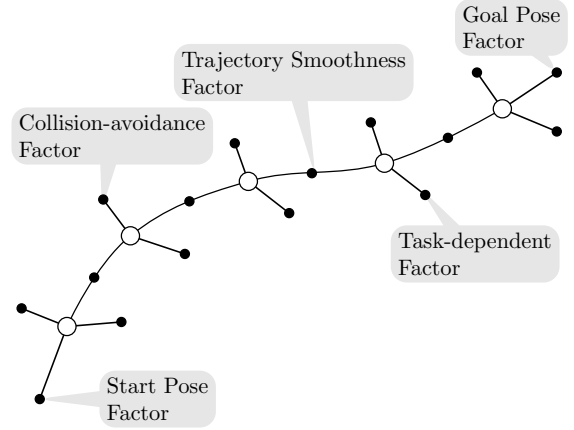


Fig. 3: Motion planning factor graph, adapted from [36].

tend to satisfy such a constraint at a higher priority. Eq. 9 can be represented as the factor graph shown in Fig. 3, and solved numerically by using iterative approaches such as Gaussian-Newton or Levenberg-Marquardt. In GPMP2, sparsity is exploited through use of the GTSAM solver [9, 10].

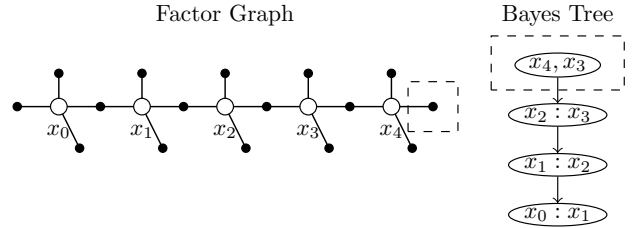### F. Fast Replanning using the Bayes Tree



Fig. 4: An example of motion planning factor graph and the corresponding Bayes tree, adapted from [36]. Here we use the goal change of the planning problem as an example to illustrate the idea of incremental replanning with the Bayes tree, where the change of factors in the graph marked with the dashed box only affects part of the Bayes tree as shown in the dashed box.

Replanning problems are very common in real-world tasks, for instance, when the goal state for the robot has been changed, or obstacles in the environment have been moved around. If the majority of the problem is not changed, then it will not be necessary to solve the entire problem again.

GPMP2 maintains the solution as a Bayes tree [25], which is a directed data structure generated from incremental elimination on the underlying factor graph, and incrementally updates this representation in response to changing conditions. Fig. 4 uses a goal change in a planning problem as an example to illustrate the idea of incremental replanning with the Bayes tree, where the change of factors in the graph only affects part of the Bayes tree as shown in the dashed box.

## IV. EXTENSION TO KINODYNAMIC MOTION PLANNING

When fulfilling both kinematic and dynamic constraints simultaneously, the kinodynamic planning problem becomes more difficult. However, we can formulate the planning problem using factor graphs with the addition of the dynamic constraints, and solve the problem efficiently with numerical optimization tools such as GTSAM, which is able to explore the sparsity in factor graphs, and use heuristics such as minimum-degree MD [15] or nested dissection (ND) [14] to generate elimination orders which reduce the computation complexity in solving the optimization problem [10].

### A. Manipulator kinodynamic Constraints

The manipulator kinodynamic constraints include both equality and inequality constraints. For equality constraints, we closely follow [33, 48]. The following equations of motion express the equality constraints between link $j$ and link $j-1$ imposed by joint $j$, assuming rotational joints.

$$\mathcal{V}_j = [Ad_{T_{j,j-1}(q_j)}]\mathcal{V}_{j-1} - \mathcal{A}_j \dot{q}_{j-1} \tag{10}$$

$$\dot{\mathcal{V}}_j = [Ad_{T_{j,j-1}(q_j)}]\dot{\mathcal{V}}_{j-1} - \mathcal{A}_j \ddot{q}_{j-1} - [ad_{\mathcal{V}_j}]\mathcal{A}_j \dot{q}_{j-1} \tag{11}$$

$$Ad^T_{T_{j+1,j}(q_{j+1})}\mathcal{F}_{j+1} = \mathcal{F}_j + \mathcal{G}_j \dot{\mathcal{V}}_j - [ad_{\mathcal{V}_j}]^T \mathcal{G}_j \mathcal{V}_j \tag{12}$$

$$\mathcal{F}_j^T \mathcal{A}_j = \tau_j \tag{13}$$

where $\mathcal{A}_j$ (expressed in link $j$ coordinate frame) is the screw axis for joint $j$, and $Ad_{T_{j,j-1}(q_j)}$ is the adjoint map of the transform $T_{j,j-1}$ between the two adjacent links. The relationship between twist $\mathcal{V}_j$ and twist $\mathcal{V}_{j-1}$ is described in Equation (10) where $\dot{q}_{j-1}$ represents the angular velocity of joint $j-1$. Equation (11) shows the constraint between acceleration $\dot{\mathcal{V}}_{j-1}$ of link $j-1$ and acceleration $\dot{\mathcal{V}}_j$ of link $j$. Equation (12) describes the balance between the wrench $\mathcal{F}_j$ and the wrench $\mathcal{F}_{j+1}$ applied to link $j$. The torque applied at joint j is expressed as the projection of wrench $\mathcal{F}_j$ on the screw axis $\mathcal{A}_j$ corresponding to joint $j$.

By representing all the variables in the equations of motion at time $t_i$ with state vector $x_i$, we write the cost function corresponding to the above equality constraints in a compact form as the following:

$$h_6(x_i) = LHS(EoM(x_i)) - RHS(EoM(x_i)) \tag{14}$$

where $EoM()$ is used to represent the equations of motion from (10) to (13), and LHS *and* RHS stand for "Left Hand Side" and "Right Hand Side" of the equations respectively.

Inequality constraints include joint limits such as limits on joint angles, angular velocities, angular accelerations, and torques applied at each joint. We use a hinge loss function to represent the i-th inequality constraint $f(x_i) \le e_i$, and the cost function $h_7(x_i)$ under the current state can be written as

$$h_7(x_i) = c_{limit}(f(x_i)) \tag{15}$$

where $c_{limit}()$ is defined as the hinge loss:

$$c_{limit}(z) = \begin{cases} a(z_l - z + \varepsilon) & if \ z - z_l \le \varepsilon \\ a(z - z_u + \varepsilon) & if \ z_u - z \le \varepsilon \\ 0 & o/w \end{cases} \tag{16}$$

where $z_l$ is the lower limit, $z_u$ is the upper limit, $\varepsilon$ is a constant threshold to prevent exceeding the limit, and $a$ is a constant ratio which determines how fast the error grows as the value approaches the limit. If the value is not within the threshold, then there will be no cost. In this way, the limit violations are prevented during the optimization.

The DFG of an RR manipulator shown in Fig. 1 is used to illustrate the dynamic constraints in kinodynamic motion planning. This DFG associated with the motion planning considers the robot dynamics, and ensures that all dynamic constraints are satisfied when solving for a motion plan.

### B. Task-dependent Dynamic Constraints

In addition, the planner sometimes will have to optimize a user-specified cost function such as a minimum torque constraint, for which the cost function can be expressed as

$$h_8(x_i) = \sum_j^N \tau_j(x_i) \tag{17}$$

where $\tau_j(x_i)$ is the torque at joint j expressed as a function of state vector $x_i$, which can be calculated through inverse dynamics. We can directly retrieve all the torque values from the DFG.

### C. Trajectory Smoothness

Here we use a continuous-time configuration space trajectory with a constant acceleration instead of a constant velocity, where the state transition matrix $\Phi(t_i, t_{i-1})$ in equation (6) is replaced with

$$\Phi(t_i, t_{i-1}) = \begin{bmatrix} 1 & \Delta t_i & \frac{1}{2}(\Delta t_i)^2 \\ 0 & 1 & \Delta t_i \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

and the covariance matrix is

$$\Sigma_5 = \begin{bmatrix} \frac{1}{2}\Delta t_i^5 \mathbf{Q}_C & \frac{1}{8}\Delta t_i^4 \mathbf{Q}_C & \frac{1}{6}\Delta t_i^3 \mathbf{Q}_C \\ \frac{1}{8}\Delta t_i^4 \mathbf{Q}_C & \frac{1}{3}\Delta t_i^3 \mathbf{Q}_C & \frac{1}{2}\Delta t_i^2 \mathbf{Q}_C \\ \frac{1}{6}\Delta t_i^3 \mathbf{Q}_C & \frac{1}{2}\Delta t_i^2 \mathbf{Q}_C & \Delta t_i \mathbf{Q}_C \end{bmatrix} \tag{19}$$

### D. Kinodynamic Motion Planning Factor Graph

A factor graph is used to solve the kinodynamic motion planning problem, where constraints including start and goal pose constraints, manipulator dynamic constraints, task-dependent kinematic and dynamic constraints, and constraints that ensure a smooth and collision-free trajectory are represented as factors in the graph, as shown in Fig. 2.

## V. FAST REPLANNING WITH INCREMENTAL TECHNIQUE

We are able to solve kinodynamic motion replanning problems efficiently by taking advantage of the incremental technique with iSAM2, which uses the Bayes tree to perform incremental optimization on the factor graph. In the following kinodynamic motion replanning problem, a RR manipulator needs to find a minimum torque trajectory to move the end-effector from the given start pose to the desired goal pose while respecting both kinematic and dynamic constraints. A Bayes tree is generated by eliminating the variables in the

(a)                                                                              (b)
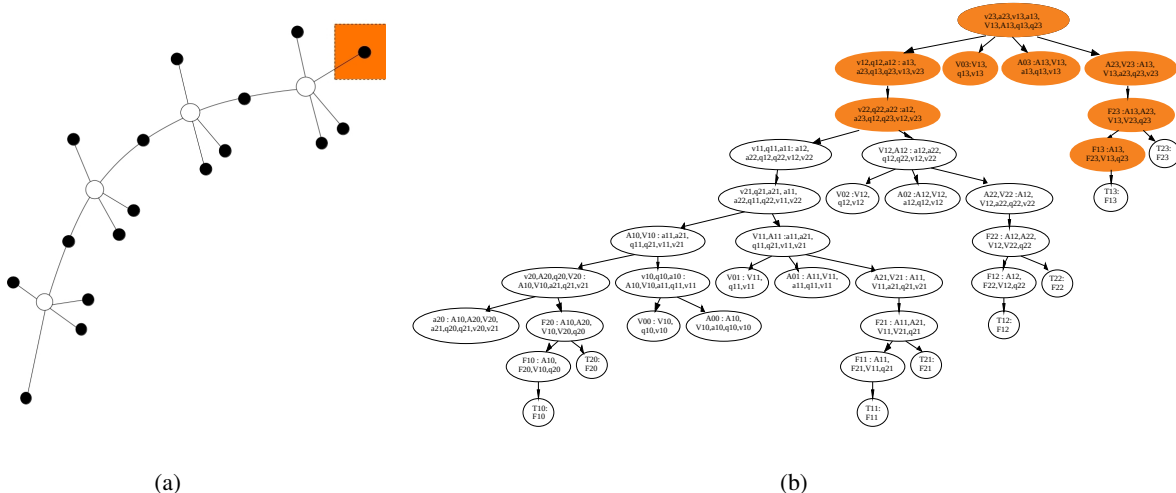
Fig. 5: (a) An example factor graph for a kinodynamic motion replanning problem of a robot with two links and two revolute joints, where the orange shaded regions show the factor changed due to a change of the goal configuration; (b) The corresponding Bayes tree with the orange shaded regions showing the parts affected by the change of factors in the graph. In this Bayes tree, the characters q, v, a, V, A, F and T represent joint angle, joint velocity, joint acceleration, link twist, link acceleration, wrench, and torque respectively. The first number ranging from 1 to 2 represents the link or joint index, and the second number ranging from 0 to 3 represents the time index.

kinodynamic motion planning factor graph from the first state to the last state. With a change of the goal configuration, only the factor in the orange shaded portion of the graph shown in Fig. 5 will be changed, and the new planning problem can be solved with the corresponding Bayes tree, where the orange shaded regions show the affected parts compared to the original Bayes tree. The optimization result obtained from the Bayes tree will only be partially changed since the majority of the Bayes tree is left unchanged.

The proposed kinodynamic motion replanning algorithm is implemented with iSAM2 incremental solver [25]. First, we check for the information which has been updated; next, we change the factors in the graph according to the updated information; finally, we run the incremental algorithm to update the Bayes tree implemented in iSAM2 to get the new solution.

## VI. EVALUATION

### A. Experiment Setup

We implemented both a batch version (DFGP) and an incremental version (iDFGP) of the method above using the GTSAM C++ library [9, 10], and ran simulations in V-REP for visualization. We used the Levenberg-Marquardt algorithm to solve the nonlinear least squares optimization problem for KMDP, with an initial value for $\lambda = 0.01$, and terminating the optimization process if either of the following conditions is satisfied: 1) it reaches a maximum of 200 iterations, or 2) the relative decrease in error is smaller than $10^{-5}$. For the incremental version, iDFGP uses the iSAM2 [25] optimizer with default settings.

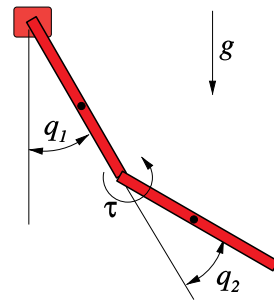### B. Kinodynamic Motion Planning for the Acrobot



Fig. 6: Acrobot

The Acrobot [45] is a planar RR manipulator with an actuator at the elbow but no actuator at the shoulder, as shown in Fig. 6. The task is to bring the Acrobot from its initial rest configuration to the upright pose where there is limited torque applied at the elbow joint. The equations of motion of the Acrobot are described in [45].

We perform kinodynamic motion planning for the Acrobot with our DFGP algorithm. Due to the lack of source code for any other optimization-based kinodynamic motion planning algorithms, we only benchmark it against some of the state-of-art sampling-based kinodynamic motion planners, such as Stable-Sparse-RRT (SST), Stable-Sparse-RRT* (SST*) and Asymptotically-Optimal-RRT (AO-RRT). All benchmarks were run on a single thread of a 4.0GHz Intel Core i7 CPU.

We used the implementations of the RRT-style methods

TABLE I: Results for Kinodynamic Motion Planning.

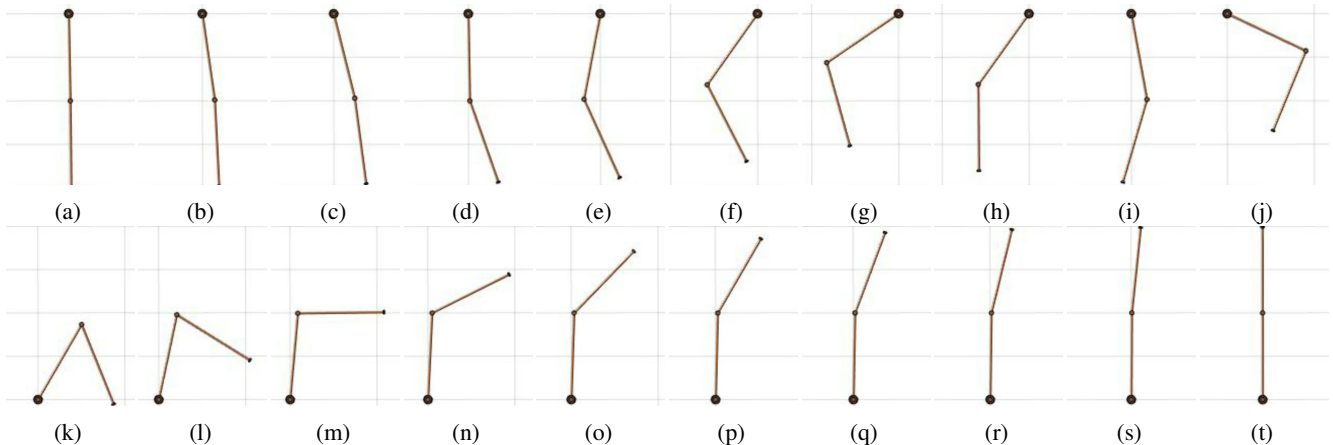| Tests | | DFGP | SST | SST* | AO-RRT |
|---|---|---|---|---|---|
| | Success Rate (%) | 100 | 98 | 100 | 90 |
| Acrobot | Average Time (s) | 1.07 | 180.55 | 229.67 | 1058.48 |
| | Maximum Time (s) | 1.07 | 1054.89 | 2152.95 | 5868.87 |
| | Success Rate (%) | 86 | – | – | – |
| Kuka Task1 | Average Time (s) | 2.15 | – | – | – |
| | Maximum Time (s) | 3.39 | – | – | – |
| | Success Rate (%) | 90 | – | – | – |
| Kuka Task2 | Average Time (s) | 1.75 | – | – | – |
| | Maximum Time (s) | 2.92 | – | – | – |



Fig. 7: A solution for the kinodynamic motion planning problem where the task is to bring the Acrobot from the rest configuration as in (a) to the goal configuration as in (t). Images (b)-(s) show the intermediate configurations along the planned trajectory.

from an open source repository [18] and implemented a test case for the Acrobot using the framework of the example problem in the repository. The parameters used for SST and SST* are suggested by the author of the original paper [49], and default settings are used for AO-RRT. We ran each of the RRT algorithms 50 times, and each time the test was terminated if any of the following events occurred: 1) found a solution with trajectory time length less than 10 seconds, or 2) reached a maximum number of 300k iterations. We counted a trial as a success if a trajectory with a time less or equal to 10 seconds was returned, and calculated the success rate and the average computation time based on the successful trials.

For the proposed DFGP algorithm, we set the trajectory length as 10 seconds and initialized the joint trajectory with an acceleration-smooth straight-line, similar to initialization methods used in [35, 36]. We ran DFGP 50 times and calculated the average computation time.

The benchmark results for the Acrobot are presented in Table I. It should not be surprising to find that RRT-style methods are mush slower than the proposed DFGP algorithm. From the results of RRT-style methods, we can see the difficulty of solving the kinodynamic motion planning problem for a under-actuated robotic system. Without taking full dynamics of the Acrobot into consideration, it is almost impossible for a motion planner to find a solution to bring the Acrobot from

its initial rest configuration to the upright pose. Fig. 7 shows a solution of the proposed DFGP algorithm, in which we observe that the Acrobot swings back and forth to gain momentum while bending the first joint to reduce the moment arm so that it can move the end effector to the goal pose.

From basic system dynamics, we know there are only two equilibrium points for the Acrobot; one is at the rest configuration, and the other one is at the upright configuration [44]. Therefore, we did not attempt any replanning experiments in the case of the Acrobot.

### C. Kinodynamic Motion Planning for a KUKA Arm

The KUKA LBR iiwa [8] is a lightweight industry robot with 7 actuated revolute joints. The tasks performed in our experiments are the following: 1) moving a block from one location to the desired goal location in the environment with obstacles, and 2) moving a block from one location to the desired goal location with minimum torque constraint.

We perform both kinodynamic motion planning tasks for the KUKA robot with the proposed DFGP algorithm, and the RRT-style methods Stable-Sparse-RRT(SST), Stable-Sparse-RRT*(SST*) and Asymptotically-Optimal-RRT(AO-RRT). For each task, we created 50 different tests with the same start configuration but randomly generated goal configurations. We ran 50 tests for each task with DFGP, SST, SST* and AO-RRT. However, due to the curse of dimensionality,
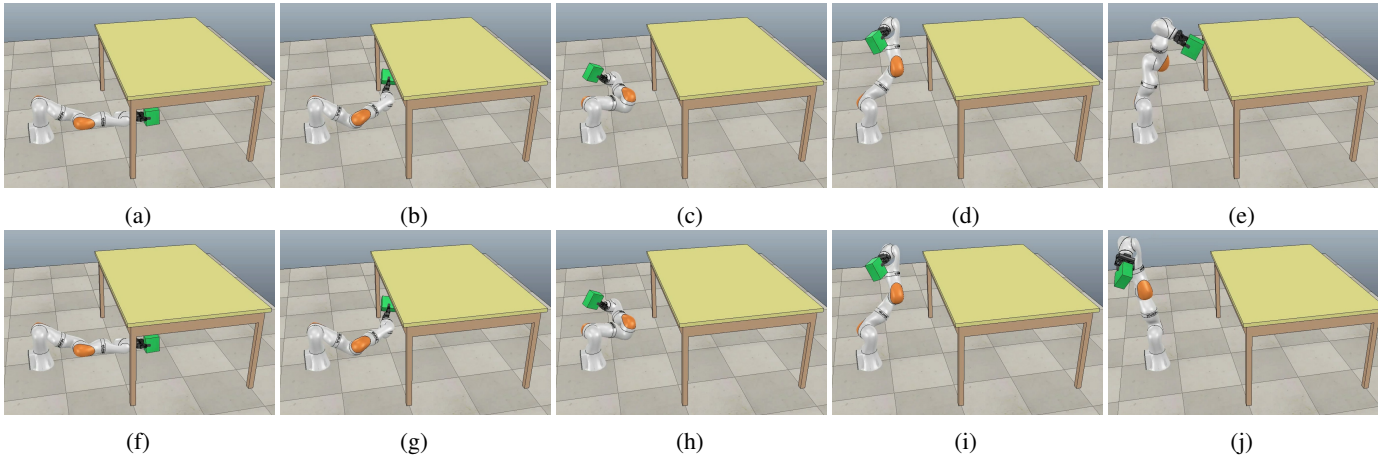
Fig. 8: (a)-(e) show a solution obtained by the DFGP algorithm for kinodynamic motion planning task 1, where the objective is to find a trajectory for the KUKA robot to bring a block from the initial location as shown in (a) to the goal location as shown in (t) while avoiding collision. (f)-(j) show a solution obtained by the iDFGP algorithm for the corresponding replanning problem with the new goal as shown in (j).
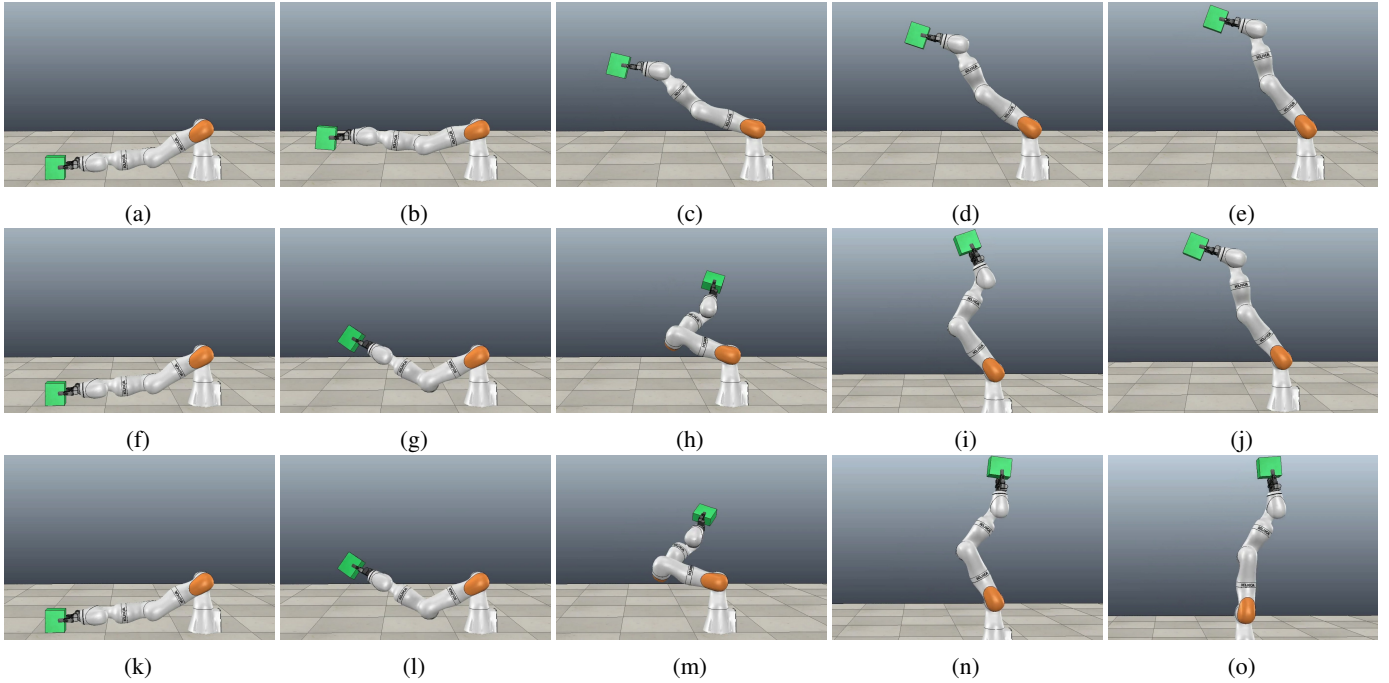


Fig. 9: (a)-(e) show a solution obtained by the DFGP algorithm for kinodynamic motion planning task 2, where the objective is to find a trajectory for the KUKA robot to bring a block from the initial location as shown in (a) to the goal location as shown in (t). (f)-(j) show a solution obtained by the DFGP algorithm for the same task but with a minimum torque constraint applied. (k)-(o) show a solution obtained by the iDFGP algorithm for the corresponding replanning problem with the new goal as shown in (o) while the minimum torque is required.

none of the RRT-style methods were able to find a solution after more than 8 hours of computation.

The benchmark results for task 1 and task 2 are shown in Table I. In task 1, we count one trial as a success if a collision free trajectory is found and all the dynamic constraints are satisfied. In task 2, one trial is considered successful if the goal is reached while all the dynamic constraints are satisfied

and the total torque (total torque for all joints in each run) is less than the one without a minimum torque constraint. The average and maximum time to success is calculated with only the successful trials. With the minimum torque constraint, the average torque based on the successful trials is 1522.25 N*m, while it is 1747.91 N*m without this constraint.

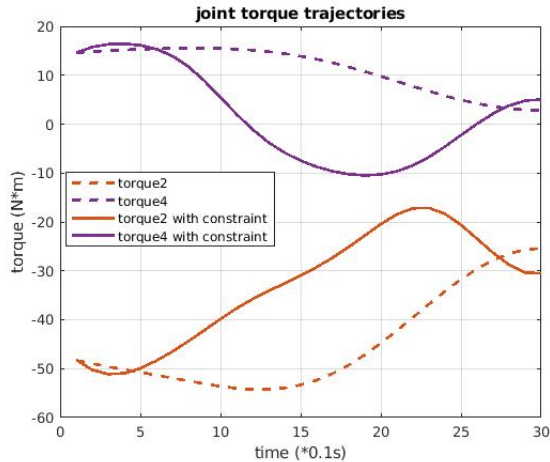Figures 8a to 8e show a solution obtained by the proposed

Fig. 10: Torque trajectories for joint 2 (orange) and joint 4 (purple) of KUKA robot in task 2, where solid lines are results with minimum torque constraint, and dashed lines are results without minimum torque constraint.

DFGP algorithm for task 1 with the goal configuration shown in Fig. 8e, from which we observe that the KUKA robot brings the block to the goal location while avoiding the desk which is initially above it.

Figures 9f to 9j show a solution obtained by the proposed DFGP algorithm for task 2 with the goal configuration shown in Fig. 9j. We observe that the KUKA robot first moves towards the center to reduce the moment arm, then pushes upwards so that it can bring the block to the goal location with less torque applied as compared to the solution shown in Fig. 9a to Fig. 9e, where the start and goal configurations are the same, but with no torque constraint applied.

The torque trajectories of joint 2 and joint 4 (the two joints that do most of the work in task 2) of the KUKA robot are plotted in Figure 10, where the solid lines and dashed lines represent the torque trajectories with and without the minimum torque constraint, respectively. From the plots in Fig. 10 we observe that a more energy efficient motion plan is obtained by the DFGP algorithm with the minimum torque constraint.

### D. Fast Kinodynamic Motion Replanning for the KUKA

TABLE II: Results for Kinodynamic Motion Replanning.

| Kuka Tests | | DFGP-L | DFGP-W | iDFGP |
|---|---|---|---|---|
| | Success Rate (%) | 86 | 92 | 80 |
| Task1 | Average Time (s) | 1.98 | 1.12 | 0.119 |
| | Maximum Time (s) | 3.33 | 3.09 | 0.200 |
| | Success Rate (%) | 90 | 98 | 96 |
| Task2 | Average Time (s) | 1.68 | 0.63 | 0.074 |
| | Maximum Time (s) | 2.98 | 1.76 | 0.078 |

We change the goal configurations generated for each kinodynamic motion planning task to new goal configurations, and solve the replanning problem with both the batch version DFGP and the incremental version iDFGP. For the batch

version DFGP, we ran the tests with two different initialization methods, one initialized with an acceleration-smooth straight-line (named DFGP-L), and the other uses a warm start with the result from the corresponding original planning problem (named DFGP-W).

Replanning benchmark results for task 1 and task 2 are summarized in Table II. The same rules are used to count the successful trials and calculate the average and maximum computation time as described in the original planning problem.

From the results, we observe that in both task 1 and task 2, the iDFGP algorithm is about 8 to 10 times faster than the DFGP algorithm with the warm start initialization, and about 20 times faster than the DFGP algorithm with an acceleration smooth straight-line initialization, with only a small loss in the success rate in task 1.

Our experimental setting for task 1 is similar to the one in [50], and the results show that the efficiency of our DFGP algorithm is about the same as their method, and our iDFGP algorithm is about 20 times faster than either batch method. The failure of iDFGP algorithm in task 1 might be caused by the fact that iDFGP uses the original solution as initialization, which can be a bad choice if the goal has been changed dramatically. However, the DFGP algorithm using the warm start initialization did not suffer from any loss in success rate due to the fact that DFGP uses the Levenberg-Marquardt optimizer, which is a more powerful optimization algorithm compared to the one used by iDFGP, and is able to provide appropriate step damping to help converge to better results [10]. The average torques calculated based on the results of the successful trials for DFGP-L, DFGP-W, and iDFGP were 1466.29 N*m, 1490.20 N*m, and 1489.03 N*m, respectively.

Figures 8f to 8j show a solution obtained by the proposed iDFGP algorithm for the replanning problem in task 1, where the goal is changed from Fig. 8e to Fig. 8j. Since the environment is the same and the goal is not dramatically changed, the iDFGP algorithm is able to update the trajectory with an incremental Bayes tree solver, and find a solution which is very close to the original trajectory with only the part near the goal being modified. A similar effect can be observed in Fig. 9, where Figures 9k to 9o show a solution by the proposed iDFGP algorithm for the change of goal replanning problem in task 2.

### VII. CONCLUSION

We consider the kinodynamic motion planning as an optimal control problem and efficiently solve using the factor-graph-based GTSAM solver, obtaining performance on par withe the state of the art for initial planning. We are able to perform fast replanning by taking advantage of the incremental Bayes tree solver, which avoids resolving the entire problem if only part of the trajectory needs to be updated.

In the future work, we would like to explore more initialization strategies to improve the success rate of the proposed kinodynamic motion planning algorithm and help the optimization solver to converge to a better local optimal solution.

REFERENCES

[1] R. E Allen and M. Pavone. A real-time framework for kinodynamic planning in dynamic environments with application to quadrotor obstacle avoidance. *Robotics and Autonomous Systems*, 115:174–193, 2019. URL https://www.sciencedirect.com/science/article/pii/S0921889017308692.

[2] N. M Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proceedings of IEEE international conference on robotics and automation*, volume 1, pages 113–120. IEEE, 1996. URL https://ieeexplore.ieee.org/abstract/document/503582.

[3] K. Bergman. *On Motion Planning Using Numerical Optimal Control*, volume 1843. Linköping University Electronic Press, 2019. URL http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1318297&dswid=2945.

[4] J. E Bobrow, S. Dubowsky, and JS Gibson. Time-optimal control of robotic manipulators along specified paths. *The international journal of robotics research*, 4(3):3–17, 1985. URL https://journals.sagepub.com/doi/abs/10.1177/027836498500400301.

[5] R. Bordalba, L. Ros, and J. M Porta. Kinodynamic planning on constraint manifolds. *arXiv preprint arXiv:1705.07637*, 2017. URL https://upcommons.upc.edu/handle/2117/110931.

[6] T. Chettibi, HE Lehtihet, and and S. Hanchi M. Haddad. Minimum cost trajectory planning for industrial robots. *European Journal of Mechanics-A/Solids*, 23(4):703–715, 2004. URL https://www.sciencedirect.com/science/article/pii/S0997753804000324.

[7] Howie M Choset, Seth Hutchinson, Kevin M Lynch, George Kantor, Wolfram Burgard, Lydia E Kavraki, and Sebastian Thrun. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[8] KUKA Robotics Corporation. KUKA LBR iiwa, 2019.

[9] F. Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012. URL https://smartech.gatech.edu/handle/1853/45226.

[10] F. Dellaert, M. Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017. URL https://www.nowpublishers.com/article/Details/ROB-043.

[11] M. Diehl, H. Bock, H. Diedam, and Wieber P-B. Fast direct multiple shooting algorithms for optimal robot control. In *Fast motions in biomechanics and robotics*, pages 65–93. Springer, 2006. URL https://link.springer.com/chapter/10.1007/978-3-540-36119-0_4.

[12] Bruce Donald, Patrick Xavier, John Canny, John Canny, John Reif, and John Reif. Kinodynamic motion planning. *Journal of the ACM (JACM)*, 40(5):1048–1066, 1993.

[13] Bruce Randall Donald and Patrick Xavier. Provably good approximation algorithms for optimal kinodynamic planning: Robots with decoupled dynamics bounds. *Algorithmica*, 14(6):443–479, 1995.

[14] A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363, April 1973. URL https://epubs.siam.org/doi/abs/10.1137/0710032.

[15] A. George and J.W.H. Liu. *Computer solution of large sparse positive definite systems*. Computational Mathematics. Prentice-Hall, Englewood Cliffs, N.J., 1981. URL https://dl.acm.org/doi/abs/10.5555/578296.

[16] E. Glassman and R. Tedrake. Lqr-based heuristics for rapidly exploring state space. In *Submitted to IEEE International Conference on Robotics and Automation. ICRA*, 2010.

[17] K. Hauser. Fast interpolation and time-optimization with contact. *The International Journal of Robotics Research*, 33(9):1231–1250, 2014. URL https://journals.sagepub.com/doi/full/10.1177/0278364914527855.

[18] K. Hauser. Optimal Motion Planning package in Python, 2019.

[19] K. Hauser and Y. Zhou. Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space. *IEEE Transactions on Robotics*, 32(6):1431–1443, 2016. URL https://ieeexplore.ieee.org/abstract/document/7588078.

[20] D. Hsu, J-C Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proceedings of International Conference on Robotics and Automation*, volume 3, pages 2719–2726. IEEE, 1997.

[21] D. Hsu, R. Kindel, J-C Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

[22] A. X Lee I. Awwal H. Bradlow J. Schulman, J. Ho and P. Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10. Citeseer, 2013. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.387.4642&rep=rep1&type=pdf.

[23] Yan J. Ho A. Lee I. Awwal H. Bradlow J. Pan S. Patil K. Goldberg J. Schulman, Y. Duan and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014. URL https://journals.sagepub.com/doi/full/10.1177/0278364914528132.

[24] E. Jelavic and M. Hutter. Whole-body motion planning for walking excavators. In *International Conference on Intelligent Robots and Systems (IROS 2019)*, 2019. URL https://doi.org/10.3929/ethz-b-000365658.

[25] 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. URL https://journals.sagepub.com/doi/abs/10.1177/0278364911430419.

[26] M. Kalakrishnan, and E. Theodorou S. Chitta, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pages 4569–4574.

IEEE, 2011. URL https://ieeexplore.ieee.org/abstract/document/5980280.

[27] L. E Kavraki, P. Svestka, J-C Latombe, and M. H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996. URL https://ieeexplore.ieee.org/abstract/document/508439.

[28] E. A. Khoury, F. Lamiraux, and M. Taix. Optimal motion planning for humanoid robots. In *2013 IEEE International Conference on Robotics and Automation*, pages 3136–3141. IEEE, 2013. URL https://ieeexplore.ieee.org/abstract/document/6631013.

[29] T. Kunz and M. Stilman. Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3713–3719. IEEE, 2014. URL https://ieeexplore.ieee.org/abstract/document/6943083.

[30] S. LaValle. *Planning algorithms*. Cambridge university press, 2006. URL https://cds.cern.ch/record/991583.

[31] S. LaValle and J. James Jr Kuffner. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001. URL https://journals.sagepub.com/doi/abs/10.1177/02783640122067453.

[32] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 1, pages 473–479 vol.1, May 1999.

[33] K. Lynch and F. Park. *Modern Robotics*. Cambridge University Press, 2017. URL https://dl.acm.org/doi/book/10.5555/3165183.

[34] A. Dragan M. Pivtoraiko M. Klingensmith C. Dellin J. Bagnelland S. Srinivasa M. Zucker, N. Ratliff. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013. URL https://journals.sagepub.com/doi/abs/10.1177/0278364913488805.

[35] M. Mukadam, X. Yan, and Boots B. Gaussian process motion planning. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 9–15. IEEE, 2016. URL https://ieeexplore.ieee.org/abstract/document/7487091.

[36] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, 2018. URL https://journals.sagepub.com/doi/full/10.1177/0278364918790369.

[37] D. Pavlichenko and S. Behnke. Efficient stochastic multicriteria arm trajectory optimization. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4018–4025. IEEE, 2017. URL https://ieeexplore.ieee.org/abstract/document/8206256.

[38] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez. Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *2012 IEEE International Conference on Robotics and Automation*, pages 2537–2542. IEEE, 2012. URL https://ieeexplore.ieee.org/abstract/document/6225177.

[39] Q.C. Pham, E. Caron, P. Lertkultanon, and Y. Nakamura. Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots. *The International Journal of Robotics Research*, 36(1):44–67, 2017. URL https://journals.sagepub.com/doi/full/10.1177/0278364916675419.

[40] M. Fallon A. Valenzuela H. Dai F. Permenter T. Koolen P. Marion S. Kuindersma, R. Deits and R. Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40(3):429–455, 2016.

[41] E. Yoshida S. Lengagne, J. Vaillant and A. Kheddar. Generation of whole-body optimal dynamic multi-contact motions. *The International Journal of Robotics Research*, 32(9-10):1104–1119, 2013. URL https://journals.sagepub.com/doi/abs/10.1177/0278364913478990.

[42] E. Schmerling and M. Pavone. Kinodynamic planning. In *Encyclopedia of Robotics*. Springer, first edition, 2019. URL http://asl.stanford.edu/wp-content/papercite-data/pdf/Schmerling.Pavone.EOR19.pdf.

[43] A. Sintov. Goal state driven trajectory optimization. *Autonomous Robots*, 43(3):631–648, 2019. URL https://link.springer.com/article/10.1007/s10514-018-9728-3.

[44] M. Spong. The swing up control problem for the acrobot. *IEEE control systems magazine*, 15(1):49–55, 1995. URL https://ieeexplore.ieee.org/abstract/document/341864.

[45] R. Tedrake. Underactuated robotics: Learning, planning, and control for efficient and agile machines: Course notes for mit 6.832. *Working draft edition*, 3, 2009. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.396.3212&rep=rep1&type=pdf.

[46] D. J Webb and J. Van Den Berg. Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics. In *2013 IEEE International Conference on Robotics and Automation*, pages 5054–5061. IEEE, 2013. URL https://ieeexplore.ieee.org/abstract/document/6631299.

[47] A. W Winkler, C D. Bellicoso, M. Hutter, and J. Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018. URL https://ieeexplore.ieee.org/abstract/document/8283570.

[48] M. Xie and F. Dellaert. A unified method for solving inverse, forward, and hybrid manipulator dynamics using factor graphs. *arXiv preprint arXiv:1911.10065*, 2019. URL https://arxiv.org/abs/1911.10065.

[49] Z. Littlefield Y. Li and K. E Bekris. Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*, 35(5):528–564, 2016. URL https://journals.sagepub.com/doi/full/10.

1177/0278364915614386.

[50] H. Lin Y. Zhao and M. Tomizuka. Efficient trajectory optimization for robot motion planning. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 260–265. IEEE, 2018. URL https://ieeexplore.ieee.org/abstract/document/8581059.