

Trajectory Planning for Manipulators based on the Optimal Concatenation of LQ Control Primitives

Michael Steinegger*, Benjamin Passenberg*, Marion Leibold*, and Martin Buss*

Abstract—A trajectory planning method for robotic systems consisting of kinematic chains is introduced based on the concatenation of control primitives. The parameterized, optimal motion primitives are derived from a parametric, linear-quadratic optimal control problem, which is formulated for the input-to-state and input-to-output linearized robot dynamics. The primitives can be concatenated, such that the resulting trajectory is optimal with respect to desired intermediate points. Here, sub-optimal intermediate points are found by a heuristic motion planning algorithm and are iteratively inserted, if necessary, to avoid collisions with obstacles in the robot workspace. All parameters for concatenated primitives are uniquely determined by the solution of a system of parameterized linear equations. In comparison to ordinary approaches based on optimal control, the computational effort for trajectory planning is reduced, since the system of linear equations can be solved on-line by algebraic computations.

I. INTRODUCTION

Trajectory planning is one of the basic problems in robotics. The planning process is often carried out in high dimensional configuration spaces and adaptations to changes in the robot workspace have to be determined in real-time. Furthermore, the generated trajectory is often requested to minimize a desired cost criterion under the restriction that obstacles are avoided on the way to the desired final state.

One method to address the trajectory planning problem consists in applying optimal control theory. Optimal control provides powerful methods to compute optimal trajectories for linear as well as nonlinear, high-dimensional systems while taking the system dynamics and other possible constraints or limitations into account. However, motion planning based on optimal control is in general only feasible in the case of obstacle-free environments and for non-real-time applications if the system dynamics are nonlinear and highly complex.

In order to reduce the complexity of the system dynamics and the computational effort for the planning process, there has been an increasing interest in robot control based on control primitives. The idea of primitive-based control is to generate trajectories out of several primitives encoding simple and stereotypical motions. To achieve complex trajectories, primitives are often adapted by parameters and sequenced or superimposed. Primitives based on dynamical

systems encoding the desired trajectory in a landscape of attractors are introduced in [1] and specified in [2] as *Dynamic Movement Primitives* (DMP). The trajectory profiles can be adapted by parameters which makes DMPs attractive for imitation learning approaches as in [3]–[5]. In [6], movement-patterns are used for motion capturing and learning, where intermediate points are extracted from a human movement trajectory. Spline optimization is applied in order to compute the trajectory, passing through the extracted fixed intermediate points. Nonlinear contraction theory is applied in [7] for a smooth concatenation of DMP trajectories for an unmanned aerial vehicle (UAV). A similar approach is presented in [8], where drawing tasks are reproduced by switching between linear dynamical systems delivering different trajectory samples.

Motion primitives inspired by experimental observations on the motion generation of vertebrates are introduced in [9]. There, primitives are defined as linear systems with different equilibria. By a suitable linear combination of the system outputs, desired motions of a manipulator with two degrees of freedom (DOF) are generated. In [10], a primitive-based hybrid control approach is presented, in which the dynamics of nonlinear systems are quantized in terms of time-parameterized trajectory primitives. The primitives are concatenated by a *maneuver automaton* to generate feasible trajectories for an UAV. A method based on the non-optimal concatenation of feedback LQ regulators around set-points of the linearized dynamics is presented in [11] for planning stabilizing trajectories.

In this paper, an approach is presented based on optimal control primitives for collision-free trajectory planning. In [12], it is shown that the optimal solution of a finite-horizon linear-quadratic optimal control problem is a parametrized primitive that is given in the form of a feedback-feedforward control. This paper introduces how several of these primitives can be optimally concatenated, such that intermediate points are met. When applying the concept to robotics, intermediate points are determined with a heuristic motion planner to avoid collisions with obstacles. Since the intermediate points are not optimal, the overall trajectory is sub-optimal. The optimal concatenation of primitives is calculated in real-time by evaluating a set of algebraic equations.

The paper is organized as follows: Sec. II introduces parametric LQ control primitives, defined for linearized robot dynamics. Furthermore, constraint equations for the optimal concatenation of primitives in joint space are derived. In Sec. III, the primitives are extended for motions in task space. The

*Michael Steinegger, Benjamin Passenberg, Marion Leibold, and Martin Buss are with the Institute of Automatic Control Engineering, Technische Universität München, 80290 München, Germany. michael.steinegger@mytum.de, {passenberg, marion.sobotka}@tum.de, m.buss@ieee.org

planning scheme for collision-free trajectories is described in Sec. IV and evaluated in Sec. V.

II. JOINT SPACE PRIMITIVES FOR LINEAR AND LINEARIZABLE SYSTEMS

We consider the linear time-invariant dynamical system

$$\begin{cases} \dot{z}(t) = Az(t) + Bv(t) \\ y(t) = Cz(t) \end{cases}, \quad (1)$$

where $z(t) \in \mathbb{R}^m$ is the system state, $v(t) \in \mathbb{R}^n$ is the control input, $y(t) \in \mathbb{R}^s$ is the output, and $m = 2n$. The system matrices are given by $A \in \mathbb{R}^{m \times m}$, $B \in \mathbb{R}^{m \times n}$, and $C \in \mathbb{R}^{s \times m}$. The objective of the LQ optimization is to find an optimal control input $v^*(t)$ for a given time interval $t \in [t_0, t_f]$, such that the quadratic performance index

$$\Gamma(z, v) = \Theta_0 + \mu_0^T \Pi_0 + \Theta_f + \mu_f^T \Pi_f + \int_{t_0}^{t_f} \Phi(z, v) dt \quad (2)$$

is minimized. The weighting function $\Phi(z, v)$ is defined as

$$\Phi(z, v) = \frac{1}{2} z^T(t) Q z(t) + z^T(t) S v(t) + \frac{1}{2} v^T(t) R v(t), \quad (3)$$

with the weighting matrix $S \in \mathbb{R}^{m \times n}$, the symmetric, positive semi-definite matrix $Q \in \mathbb{R}^{m \times m}$, and the symmetric, positive definite matrix $R \in \mathbb{R}^{n \times n}$. The functions Θ_k , $k \in \{0, f\}$ penalize the deviation from the initial and final states and they are defined with symmetric, positive semi-definite matrices $W_k \in \mathbb{R}^{m \times m}$ as

$$\Theta_k = \frac{1}{2} (z(t_k) - z_k)^T W_k (z(t_k) - z_k). \quad (4)$$

The affine constraints Π_k on the initial and final state

$$\Pi_k := Cz(t_k) - y_k = 0_{s \times 1} \quad (5)$$

are adjoined by constant multipliers $\mu_k \in \mathbb{R}^s$. Note that if the initial and final states are fully assigned, the functions (4) are equal to zero since $z(t_0) = z_0$ and $z(t_f) = z_f$.

Applying calculus of variations yields the necessary conditions for a minimum of the performance index

$$\dot{z}(t) = Az(t) + Bv(t) \quad (6)$$

$$\dot{\lambda}(t) = -Qz(t) - Sv(t) - A^T \lambda(t) \quad (7)$$

$$0 = Rv(t) + S^T z(t) + B^T \lambda(t) \quad (8)$$

$$\lambda(t_0) = -W_0 z(t_0) + W_0 z_0 - C^T \mu_0 \quad (9)$$

$$\lambda(t_f) = W_f z(t_f) - W_f z_f + C^T \mu_f. \quad (10)$$

The state and costate differential equations can be written as

$$\begin{bmatrix} \dot{z}^T(t) & \dot{\lambda}^T(t) \end{bmatrix}^T = H \begin{bmatrix} z^T(t) & \lambda^T(t) \end{bmatrix}^T, \quad (11)$$

where $\lambda(t) \in \mathbb{R}^m$ and H denotes the Hamiltonian matrix

$$H = \begin{bmatrix} A - BR^{-1}S^T & -BR^{-1}B^T \\ -Q + SR^{-1}S^T & SR^{-1}B^T - A^T \end{bmatrix}.$$

It has been shown [12, Theorem 1], that all solutions of (11) can be parameterized in terms of two parameter vectors $\eta \in \mathbb{R}^m$ and $\rho \in \mathbb{R}^m$ under the assumptions that (A, B)

is controllable and H has no eigenvalues on the imaginary axis. The parametric state and costate trajectories are then defined as

$$\begin{bmatrix} z(t) \\ \lambda(t) \end{bmatrix} = \begin{bmatrix} I_m \\ P_\oplus \end{bmatrix} e^{A_\oplus t} \eta + \begin{bmatrix} I_m \\ P_\ominus \end{bmatrix} e^{A_\ominus (t-t_f)} \rho, \quad (12)$$

where I_m is the identity matrix of size $m \times m$. The matrix $P_\oplus \in \mathbb{R}^{m \times m}$ denotes the positive semi-definite and symmetric maximal solution and $P_\ominus \in \mathbb{R}^{m \times m}$ denotes the negative semi-definite and symmetric minimal solution of the continuous-time algebraic Riccati equation (CARE)

$$PA + A^T P - (S + PB)R^{-1}(S^T + B^T P)Q = 0. \quad (13)$$

With the two extremal solutions, the closed-loop system matrices $A_h \in \mathbb{R}^{m \times m}$, $h \in \{\oplus, \ominus\}$ can be written as

$$A_h = A - BK_h, \quad (14)$$

where the gain matrices $K_h \in \mathbb{R}^{n \times m}$ are defined as

$$K_h = R^{-1}(S^T + B^T P_h), \quad (15)$$

and the eigenvalues of A_\oplus and A_\ominus are stable and anti-stable, respectively. The main idea of the parametric solution is to combine the stabilizing solution P_\oplus and anti-stabilizing solution P_\ominus of the CARE to achieve an optimal point-to-point movement from an initial to a desired final point. The resulting parameterized optimal control law is [12]

$$v^*(t) = -K_\oplus e^{A_\oplus t} \eta - K_\ominus e^{A_\ominus (t-t_f)} \rho, \quad (16)$$

which we define as a control primitive for linear systems.

A. Intermediate Point Constraints

The LQ primitive (16) drives the linear system (1) from its initial state z_0 to the desired final state z_f . Now, we introduce a state variable $x(t) \in \mathbb{R}^n$ and define the system state as $z(t) = [x^T(t) \quad \dot{x}^T(t)]^T$. The new state $x(t)$ is considered as a position vector and $\dot{x}(t)$ as the associated velocities.

If the trajectory $z(t)$ should be forced to pass through some specific points or if obstacles should be avoided on the way to the final state, it is necessary to include intermediate points to the trajectory planning process. The intermediate points specify the desired position values $x^d = x(t^d)$ at a desired time t^d and $z(t^d)$ is considered as the transition state between two control primitives. The velocities $\dot{x}(t^d)$ are not specified, such that the optimization chooses them optimally to guarantee a minimum value of the performance index.

To derive the necessary conditions at the intermediate points for the optimal concatenation of several primitives,

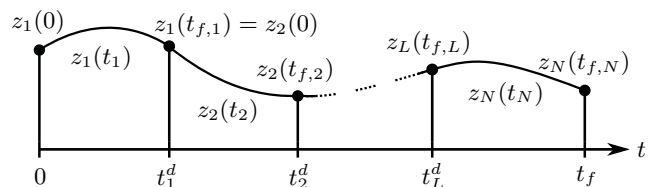


Fig. 1. Concatenation of N trajectories for the intermediate point problem.

we assume that a number of L intermediate positions $x_j^d = x(t_j^d)$, $j \in \{1, \dots, L\}$ must be met on the way to the final state $z(t_f)$. The desired time at which the corresponding intermediate point x_j^d is passed, is denoted by t_j^d .

Referring to Fig 1, a trajectory $z(t)$ is now divided into $N = L + 1$ sub-trajectories $z_i(t_i)$, $i \in \{1, \dots, N\}$ which should be generated by the associated control primitives

$$v_i(t_i) = -K_{\oplus} e^{A_{\oplus} t_i} \eta_i - K_{\ominus} e^{A_{\ominus} (t_i - t_{f,i})} \rho_i . \quad (17)$$

Each primitive $v_i(t_i)$ is parameterized by the pair of parameter vectors (η_i, ρ_i) and we define a time horizon $t_i \in [0, t_{f,i}]$ for each primitive, where $t_i = t - t_{i-1}^d$ and $t_{f,i} = t_i^d - t_{i-1}^d$ with $t_0^d = t_{f,0} = t_0 = 0$ and $t_{f,N} = t_f$. Furthermore, the states $z_j(t_{f,j})$ and $z_{j+1}(0)$ are equal to guarantee a smooth transition between two sub-trajectories.

An optimal trajectory for a two-point boundary-value problem can be parameterized by η and ρ . In the case of a boundary-value problem with L intermediate points, in total $2N$ parameter vectors have to be specified with a total of $2Nm$ unknowns. The boundary conditions

$$z_1(0) = z_0 \quad (18)$$

$$z_N(t_{f,N}) = z_f , \quad (19)$$

the affine constraints at the j^{th} intermediate point with the matrix $\Omega = [I_n \quad 0_{n \times n}]$

$$Z(z(t_{f,j})) := \Omega z_j(t_{f,j}) - x_j^d = 0_{n \times 1} \quad (20)$$

$$\Omega z_{j+1}(t_{j+1} = 0) - x_j^d = 0_{n \times 1} , \quad (21)$$

and the requirement of a continuous transition velocity at each intermediate point

$$\Psi(z_j(t_{f,j}) - z_{j+1}(t_{j+1} = 0)) = 0_{n \times 1} , \quad (22)$$

$\Psi = [0_{n \times n} \quad I_n]$, provide in total $(2m + 3Ln)$ equations. The Ln missing constraints, which are required to specify all $2Nm$ parameters, are derived from the theory of optimal control problems with intermediate constraints (see [13, pp. 101] for details). Adjoining (20) to the performance index (2) by some multipliers $\kappa_j \in \mathbb{R}^n$ and applying the calculation of variations yields the optimality condition for the costates $\lambda_j(t_j)$ and $\lambda_{j+1}(t_{j+1})$ at the transition point x_j^d

$$\lambda_j^T(t_{f,j}) = \lambda_{j+1}^T(t_{j+1} = 0) + \kappa_j^T \frac{\partial Z(z(t))}{\partial z(t)} \Big|_{t=t_j^d} . \quad (23)$$

Due to the fact that the affine constraints (20) at the intermediate point only affect the position values, the derivatives of $Z(z(t_{f,j}))$ with respect to the velocities $\dot{x}(t)$ is equal to zero. This gives the Ln additional constraints

$$\Psi(\lambda_j(t_{f,j}) - \lambda_{j+1}(t_{j+1} = 0)) = 0_{n \times 1} . \quad (24)$$

This requires the Lagrangian multipliers (costates) for the velocities to be continuous at $t = t_j^d$.

B. Parameter Computation

The $2N$ parameter vectors for the given optimization problem can be specified by use of (18)-(22) and (24). To simplify the notation, we define $E_{\{\oplus,i\}} = e^{A_{\oplus} t_{f,i}}$ and $E_{\{\ominus,i\}} = e^{-A_{\ominus} t_{f,i}}$. First, the state equation in (12) is evaluated at the boundary conditions (18)-(19), which yields

$$z_1(0) = z_0 = \eta_1 + E_{\{\ominus,1\}} \rho_1 \quad (25)$$

$$z_N(t_{f,N}) = z_f = E_{\{\oplus,N\}} \eta_N + \rho_N . \quad (26)$$

With the affine constraints (20) and (21) at the intermediate point x_j^d and (12), the following is obtained:

$$x_j^d = \Omega(E_{\{\oplus,j\}} \eta_j + \rho_j) \quad (27)$$

$$x_j^d = \Omega(\eta_{j+1} + E_{\{\ominus,j+1\}} \rho_{j+1}) . \quad (28)$$

The requirement of a continuous velocity at the transition between two sub-trajectories $z_j(t_j)$ and $z_{j+1}(t_{j+1})$ gives

$$0_{n \times 1} = \Psi((E_{\{\oplus,j\}} \eta_j + \rho_j) - (\eta_{j+1} + E_{\{\ominus,j+1\}} \rho_{j+1})) , \quad (29)$$

and the last constraint equation (24) with (12) results in

$$0_{n \times 1} = \Psi((P_{\oplus} E_{\{\oplus,j\}} \eta_j + P_{\ominus} \rho_j) - (P_{\oplus} \eta_{j+1} + P_{\ominus} E_{\{\ominus,j+1\}} \rho_{j+1})) . \quad (30)$$

Without loss of generality, we restrict the notation of (25)-(30) to the case $N = 2$ for a compact representation

$$[z_0^T \quad (x_j^d)^T \quad (x_j^d)^T \quad 0_{2n \times 1}^T \quad z_f^T]^T = \Xi \begin{bmatrix} \eta_1 \\ \rho_1 \\ \eta_2 \\ \rho_2 \end{bmatrix} , \quad (31)$$

$$\Xi = \begin{bmatrix} I_m & E_{\{\oplus,1\}} & 0_{m \times m} & 0_{m \times m} \\ \Omega E_{\{\oplus,1\}} & \Omega & 0_{n \times m} & 0_{n \times m} \\ 0_{n \times m} & 0_{n \times m} & \Omega & \Omega E_{\{\oplus,2\}} \\ \Psi E_{\{\oplus,1\}} & \Psi & -\Psi & -\Psi E_{\{\oplus,2\}} \\ \Psi P_{\oplus} E_{\{\oplus,1\}} & \Psi P_{\ominus} & -\Psi P_{\oplus} & -\Psi P_{\ominus} E_{\{\oplus,2\}} \\ 0_{m \times m} & 0_{m \times m} & E_{\{\oplus,2\}} & I_m \end{bmatrix} .$$

The unknown parameters are uniquely determined by the solution of (31) since the matrix Ξ is non-singular.

In the presented example of the $(L + 2)$ -point boundary-value problem, the initial and final states were fully assigned and thus $\Theta_{0,1} = \Theta_{f,N} = 0$. Note that $\Theta_{k,i}$ obtained from (4) is the weighting function of the initial state $z_i(0)$ or final state $z_i(t_{f,i})$ associated with the i^{th} primitive. However, the method can also be applied to other LQ optimization problems, e.g. fully assigned initial state and quadratically weighted final state. In this case, the weighting function $\Theta_{f,N}$ (4) must also be considered in the performance index (2) for the N^{th} primitive, which yields the additional condition $\lambda_N(t_{f,N}) = W_{f,N} z_N(t_{f,N}) - W_{f,N} z_f$ from (10) for a minimum. With (12) this condition can be written as

$$(P_{\oplus} - W_{f,N}) E_{\{\oplus,N\}} \eta_N + (W_{f,N} - P_{\ominus}) \rho_N + W_{f,N} z_f = 0 . \quad (32)$$

Since only the initial state is assigned, condition (26) and therefore the last m rows of (31) are replaced by (32). If the

initial state is weighted the necessary minimum conditions $\lambda_1(0) = -W_{0,1}z_1(0) + W_{0,1}z_0$ obtained from (9) for the first primitive are considered. This constraint equation together with (12) replaces (25) and the first m rows of (31).

The previously defined time t_j^d , at which the intermediate position x_j^d is passed, was assumed to be prescribed up to now. If t_j^d is not explicitly specified, the Hamiltonian $\mathcal{H}(z, v, \lambda, t) = \Phi(z, v) + \lambda^T(t)(Az(t) + Bv(t))$ must be continuous at t_j^d [13, p. 101], which yields the constraint $\mathcal{H}_j(z_j, v_j, \lambda_j, t_{f,j}) = \mathcal{H}_{j+1}(z_{j+1}, v_{j+1}, \lambda_{j+1}, 0)$. With (12), (16), and $t_{f,j} = t_j^d - t_{j-1}^d$, this additional constraint can be solved in parallel with (31) for the unknowns t_j^d , η_i , and ρ_i .

C. Optimal Cost

The integral term of (2) can be analytically solved (see [12, Section 4.1]). With the parametric equations (12), the value of (2) for the i^{th} control primitive is computed as

$$\Gamma_i = \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}^T \begin{bmatrix} D^T G & -D^T F \\ -FD & F \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}, \quad (33)$$

where $\alpha_i = [\eta_i^T \ \rho_i^T]^T$, $\beta_i = [z_i^T(0) \ z_i^T(t_{f,i})]^T$, and the matrices D , F , G are given by

$$D = \begin{bmatrix} I_m & E_{\{\ominus, i\}} \\ E_{\{\oplus, i\}} & I_m \end{bmatrix}, \quad F = \begin{bmatrix} W_{0,i} & 0_{m \times m} \\ 0_{m \times m} & W_{f,i} \end{bmatrix},$$

$$G = \begin{bmatrix} W_{0,i} + P_{\oplus} & (W_{0,i} + P_{\ominus})E_{\{\ominus, i\}} \\ (W_{f,i} - P_{\oplus})E_{\{\oplus, i\}} & W_{f,i} - P_{\ominus} \end{bmatrix}.$$

The entire costs are computed by solving (33) and summing up the costs of the sub-trajectories. Note that $W_{0,1}$ and $W_{f,N}$ are zero in the case of fully assigned initial and final states. If the initial or final state is quadratically weighted in (2) then $W_{0,1} > 0$ or $W_{f,N} > 0$. If the number of primitives is $N > 2$ all matrices $W_{k,b}$ of (4) with $k \in \{0, f\}$, $b \in \{2, \dots, L\}$ are equal to zero and the final state $z_i(t_{f,i})$ can be obtained by solving (12) with the i^{th} pair of parameters.

So far, the planning method based on the concatenation of LQ primitives was defined for linear dynamics. To control a manipulator with primitives (17), we introduce the nonlinear robot dynamics and the required feedback linearization.

D. Joint Space Model

The dynamics of a manipulator in joint space are given by a second-order differential equation [14, p. 171]:

$$\tau(t) = M(q(t))\ddot{q}(t) + c(q(t), \dot{q}(t)) + g(q(t)), \quad (34)$$

where $q(t) \in \mathbb{R}^n$ is the vector of joint angles, $M(q(t)) \in \mathbb{R}^{n \times n}$ is the positive-definite and symmetric inertia matrix, $c(q(t), \dot{q}(t)) \in \mathbb{R}^n$ is the vector of centrifugal and coriolis torques, $g(q(t)) \in \mathbb{R}^n$ is the vector of gravity torques and $\tau(t) \in \mathbb{R}^n$ is the vector of torques applied to the joints.

By introducing an additional state variable $p(t) = \dot{q}(t)$ the system (34) can be converted to the first-order system

$$\begin{cases} \begin{bmatrix} \dot{q}(t) \\ \dot{p}(t) \end{bmatrix} = \begin{bmatrix} p(t) \\ -M(q(t))^{-1}(\gamma(q(t), p(t)) - u(t)) \end{bmatrix}, \\ y(t) = \zeta(q(t)) \end{cases} \quad (35)$$

where $\gamma(q(t), p(t)) = c(q(t), p(t)) + g(q(t))$, $u(t) = \tau(t)$, and $y(t) \in \mathbb{R}^s$, $s \leq n$ is the output of the system. The function $\zeta(\cdot)$ is known as forward kinematics, specifying the transformation from joint space to task space coordinates.

E. Input-to-State Feedback Linearization

Controlling the system state of a manipulator with control primitives of the form (16) requires a linearization of the nonlinear state equation. It can be shown that the robot dynamics given by (35) are input-to-state feedback linearizable ([15, pp. 213]) and the feedback control results in

$$u(t) = M(q(t))v(t) + \gamma(q(t), p(t)), \quad (36)$$

where $v(t)$ is the control input for the linearized system and the linear relation is given by $v(t) = \dot{p}(t)$. The dynamics of the state variable $z(t) = [q^T(t) \ p^T(t)]^T$ with respect to the new control input can be written in the form (1) with

$$A = \begin{bmatrix} 0_{n \times n} & I_n \\ 0_{n \times n} & 0_{n \times n} \end{bmatrix}, \quad B = \begin{bmatrix} 0_{n \times n} \\ I_n \end{bmatrix}. \quad (37)$$

Based on the performance index (2) and the linearized system dynamics with the state space realization (1) and (37), the two extremal solutions of (13) and the closed-loop matrices (14) with (15) can be computed. Furthermore, it is possible to directly apply the trajectory planning method of Sec. II-A and Sec. II-B for the linearized manipulator dynamics, since $x(t) = q(t)$ and $\dot{x}(t) = p(t)$.

F. Control Primitives for Nonlinear Dynamics

The concatenation of primitives derived from (17) yields the overall trajectory, which is optimal with respect to the included intermediate points. To apply the LQ primitives to the nonlinear system (35), (17) is inserted in (36), which leads to the i^{th} primitive for the manipulator (35)

$$u_i(t_i) = M(q(t_i))v_i(t_i) + \gamma(q(t_i), p(t_i)) \quad (38)$$

in each time interval $t_i \in [0, t_{f,i}]$.

An optimal solution delivers minimal costs for a weighted combination of the state $z(t)$ and input trajectories $v(t)$. This means for the manipulator (35) that the joint angles $q(t)$, the angular velocities $p(t)$, and the angular accelerations $\dot{p}(t)$ are minimized over time in configuration space, since $z(t) = [q^T(t) \ p^T(t)]^T$ and $v(t) = \dot{p}(t)$. However, it is not possible to optimize the control $u(t)$ of the nonlinear system.

III. TASK SPACE PRIMITIVES

The derivation of task space control primitives driving the linear system (1) from an initial state $z(0)$ to a final state $z(t_f)$ such that the output $y(t_f)$ reaches a desired value is similar to the procedure presented in the previous section.

To apply the primitive-based control method in the case of controlling the end-effector of a manipulator towards a desired Cartesian position, linearization of the mapping between the input and the output of a nonlinear system is

required. It can be shown, that the system (35) is input-to-output linearizable with relative degree $r_l = 2$, $l \in \{1, \dots, s\}$. The result of the feedback linearization for multiple-input and multiple-output systems is given by

$$\ddot{y}(t) = \xi(q(t), p(t)) + \Lambda(q(t))u(t), \quad (39)$$

where the vector $\xi(q(t), p(t))$ and matrix $\Lambda(q(t))$ contain the Lie-derivatives of (35) as defined in [15, pp.229]. The computation yields the solutions

$$\xi(q(t), p(t)) = \dot{J}(q(t))p(t) - J(q(t))M(q(t))^{-1}\gamma(q(t), p(t)) \quad (40)$$

$$\Lambda(q(t)) = J(q(t))M(q(t))^{-1} \quad (41)$$

with the Jacobian matrix $J(q(t)) = \frac{\partial \zeta(q(t))}{\partial q(t)}$. The input-to-output linearization provides the linear relation $v(t) = \ddot{y}(t)$ between the Cartesian acceleration of the end-effector and the new control input $v(t)$ for the linearized system. This linear equation can be written in the state space form of a linear system (1) with the state variable $z(t) = [y^T(t) \quad \dot{y}^T(t)]^T$, the matrices A and B as defined in (37), and the output matrix $C = [I_n \quad 0_{n \times n}]$. The procedure presented in Sec. II is applied to compute the parameters for task space primitives. The only difference is that the state variable now contains the Cartesian position and velocity and the intermediate point x_j^d now specifies the position of the end-effector instead of the joint angles. Solving (39) for $u(t)$ and applying a control primitive (17) yields

$$u_i(t_i) = \Lambda(q(t_i), p(t_i))^{-1}(v_i(t_i) - \xi(q(t_i), p(t_i))), \quad (42)$$

which represents the i^{th} control primitive for the nonlinear robot system. In the case of task space primitives, optimality refers to the position $y(t)$, velocity $\dot{y}(t)$, and acceleration $\ddot{y}(t)$ of the end-effector in Cartesian coordinates.

IV. OBSTACLE AVOIDANCE

In the previous sections, it has been shown how primitives can be concatenated in order to obtain an optimal trajectory with respect to a finite number of intermediate points. Adding intermediate points with free or fixed time, it can be ensured that collision-free trajectories are found in the presence of static or moving obstacles in the manipulator workspace.

It is in general unknown where to place the intermediate points to obtain a collision-free trajectory. Therefore, the presented control method is combined with a heuristic motion planning algorithm as presented in [16]. The main idea is to depict samples of kinematic paths derived from a *Rapidly-Exploring Random Tree* (RRT) if collisions were detected. The complete algorithm proceeds as follows: In a first step, the RRT algorithm finds a collision-free kinematic path from the initial to the final state. Afterwards, one single primitive is considered generating the trajectory between the initial state and final state and the resulting trajectory is checked for collisions. If a collision was detected at time step t_c , the configuration-time sample from the kinematic path whose time stamp is closest to t_c is added as an intermediate point.

Now, the trajectory generated by the concatenation of two control primitives is computed and another intermediate point is added to the planning process if a collision occurred. This procedure is repeated until a collision-free overall trajectory is found. The resulting trajectory is an optimal one for the specific choice of intermediate points. However, the intermediate points are not chosen optimally, such that the trajectory is sub-optimal.

V. SIMULATION EXAMPLE

The proposed method is applied to a simple manipulation task of a 3-DOF-manipulator, where the planning process is carried out in joint space under consideration of two static obstacles in the workspace. Referring to Fig. 2, the task objective is to control the manipulator from its initial state $z_0 = [q^T(0) \quad p^T(0)]^T$ to the final state $z_f = [q^T(t_f) \quad p^T(t_f)]^T$ and the trajectory must pass through the intermediate configuration $q(t_a)$. The initial and final joint velocities $p(0)$ and $p(t_f)$ are equal to zero. The joint velocities $p(t_a)$ at the intermediate point are not specified and optimized during the planning process. The times t_j^d for the intermediate points are fixed since they were derived from the configuration-time samples from the RRT-algorithm.

The resulting state trajectories, joint accelerations, as well as the intermediate configurations added by the heuristic planning algorithm (marked as red crosses) are illustrated in Fig. 3. There, the joint accelerations show corners at all intermediate states but are continuous. This is related to the fact that the intermediate configurations are assigned and the angular velocities are continuous by virtue of (22). Solving (8) for $v(t)$ considering (24) and (37) in the resulting equation, and with the linear relation $v(t) = \dot{p}(t)$ obtained from the input-to-state linearization, it can be concluded that the angular accelerations are always continuous.

Table I lists the computational effort for the parameter computation of all primitives generating the overall trajectory and the time needed for the computation of the resulting trajectories, collision checking and integration of the differential equation (35). The small computational effort is related to the fact that the closed-loop matrices A_\ominus and A_\oplus as well as the extremal solutions P_\oplus and P_\ominus of the CARE (13) are computed off-line. Since the constraint equations (25) - (30) for the primitives can be automatically derived and arranged in the form of (31), if intermediate trajectory points must be met, the only computations that must be carried out on-line are to evaluate the matrix exponentials in (31), solve the system of linear equations for the unknown parameters, and then check the resulting trajectory for collisions.

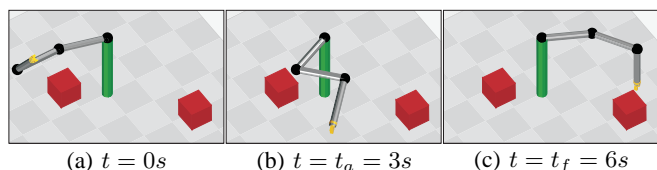


Fig. 2. Initial, intermediate, and final configuration of the manipulator.

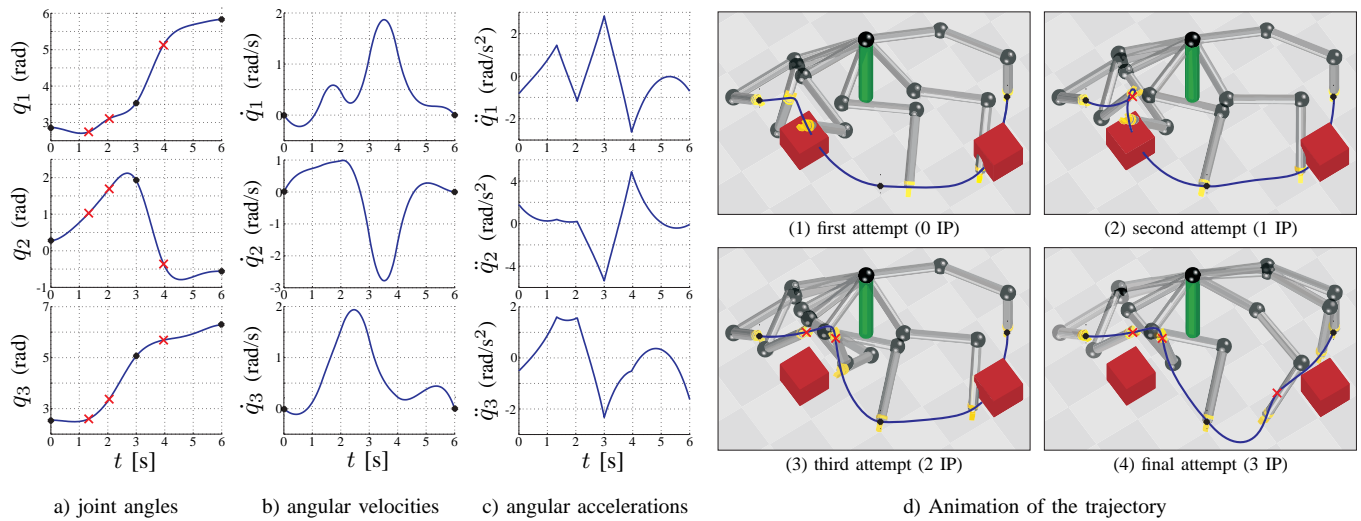


Fig. 3. Joint angles (a), angular velocities (b), and angular accelerations (c) obtained from concatenated primitives with boundary values and intermediate configuration (black dots). The intermediate points (IP) added by the planning algorithm to guarantee a collision-free trajectory are marked with red crosses. The resulting task space trajectories with the iteratively added IP are shown in (d1)-(d4), where (d4) corresponds to the joint angles (a).

TABLE I
AVERAGE COSTS AND COMPUTATIONAL TIMES.

Description	Quantity
Kinematic path	9.4 s
Parameter computation	16.5 ms
Trajectory computation	0.81 s
Collision Checking	17.7 s
Total computational time	28 s
Costs Γ	62.5

(Computations were carried out with MATLAB2008a on a 2GHz Intel Core 2 Duo T5750 processor, 2GB RAM).

VI. CONCLUSIONS

The proposed trajectory planning method for linear and linearizable systems combines primitive-based robot control and a heuristic planning approach. Primitives are defined in terms of parametric LQ control laws and an algebraic computation strategy to obtain the parameters for concatenated primitives is shown. To preserve optimal trajectories in the presence of obstacles in the operational environment, intermediate points are derived from kinematic paths computed by a RRT algorithm and included into the trajectory planning method as transition points between primitives.

The trajectory planning approach shows good results in view of computational effort for the computations of concatenated primitives and the resulting trajectory. However, the optimality of the solutions strongly depends on the choice of the intermediate points and therefore on the solution of the RRT algorithm. The approach can be extended to calculate collision-free trajectories in real-time, e.g. only one primitive in the extreme case.

REFERENCES

- [1] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems*. MIT Press, 2003, pp. 1523–1530.
- [2] S. Schaal, "Dynamic movement primitives - a framework for motor control in humans and humanoid robots," in *The Int. Symp. on Adaptive Motion of Animals and Machines*, 2003.
- [3] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Control, planning, learning, and imitation with dynamic movement primitives," in *Workshop on bilateral Paradigms on Humans and Humanoids, IEEE Int. Conf. on Intelligent Robots and Systems*, 2003.
- [4] J. Nakanishi, J. Morimoto, G. Endo, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion with dynamical movement primitives," in *Workshop on Robot Learning by Demonstration, IEEE Int. Conf. on Intelligent Robots and Systems*, 2003.
- [5] J. Kober, B. J. Mohler, and J. Peters, "Learning perceptual coupling for motor primitives," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 834–839.
- [6] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato, "A kendama learning robot based on bi-directional theory," *Neural Networks*, vol. 9, no. 8, pp. 1281–1302, 1996.
- [7] B. E. Perk and J.-J. E. Slotine, "Motion primitives for robotic flight control," *Computing Research Repository*, 2006.
- [8] D. Del Vecchio, R. M. Murray, and P. Perona, "Decomposition of human motion into dynamics based primitives with application to drawing tasks," *Automatica*, vol. 39, pp. 2085–2098, 2003.
- [9] F. Nori and R. Frezza, "Nonlinear control by a finite set of motion primitives," *6th IFAC Symp. on Nonlinear Control Systems*, 2004.
- [10] E. Frazzoli, "Robust hybrid control for autonomous vehicle motion planning," Ph.D. dissertation, Massachusetts Inst. of Technology, 2001.
- [11] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sum-of-squares verification," *Int. Journal of Robotics Research*, vol. 29, pp. 1038–1052, 2010.
- [12] A. Ferrante, G. Marro, and L. Ntogramatzidis, "A parametrization of the solutions of the finite-horizon LQ problem with general cost and boundary conditions," *Automatica*, vol. 41, pp. 1359–1366, 2005.
- [13] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control - Optimization, Estimation, and Control*. Taylor & Francis, 1975.
- [14] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, 1st ed. CRC Press, March 1994.
- [15] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Prentice-Hall, 1991.
- [16] H. Ding, G. Schnattinger, B. Passenberg, and O. Stursberg, "Improving motion of robotic manipulators by an embedded optimizer," in *IEEE Conf. on Automation Science and Engineering*, 2010, pp. 204–209.