2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Optimal trajectory generation for generalization of discrete movements with boundary conditions

Sebastian Herzog[1], Florentin Wörgötter[1], and Tomas Kulvicius[2]

Abstract— **Trajectory generation methods play an important role in robotics since they are essential for the execution of actions. In this paper we present a novel trajectory generation method for generalization of accurate movements with boundary conditions. Our approach originates from optimal control theory and is based on a second order dynamic system. We evaluate our method and compare it to state-of-the-art movement generation methods in both simulations and a real robot experiment. We show that the new method is very compact in its representation and can reproduce demonstrated trajectories with zero error. Moreover, it has most of the properties of the state-of-the-art trajectory generation methods such as robustness to perturbations and generalisation to new boundary position and velocity conditions. We believe that, due to these features, our method has great potential for various robotic applications, especially, where high accuracy is required, for example, in industrial and medical robotics.**

I. INTRODUCTION

Trajectory generation methods play an important role in robotics and many different methods exist ranging from conventional methods such as splines [1] to more dynamic approaches such as dynamic movement primitives (DMPs, [2]), Gaussian mixture models (GMMs, [3]), and a recent framework called probabilistic movement primitives (PMPs, [4]). DMPs, GMMs and PMPs have some advantages as compared to splines since they are robust to perturbations, can generalize to new situations and can be extended by additional coupling terms and learning. DMPs and GMMs are based on dynamic attractor systems and converge to the target (end-point) asymptotically. This means that there will be always a small error at the desired end-point of both, position and velocity profiles, which might be disadvantageous in applications where high precision is required, e.g., in industrial or medical robotics. The problem of position and velocity errors at the end points has been solved by the recent PMP approach. However, due to their probabilistic nature PMPs can not reproduce a demonstrated trajectory with zero error. Note that DMPs, GMMs and PMPs are not meant to represent demonstrated trajectories precisely but rather use those for initialisation of a model and learning.

In this paper we present a novel framework for the generation of movement primitives with boundary position and velocity conditions. We call our method *Optimal Control Primitives (OCPs)* as the framework originates and is derived from optimal control theory. The novelty of this approach is a combination of a linear-quadratic regulator (LQR) controller with a representation of trajectories utilising Chebyshev polynomials (different from splines can also use polynomials of higher order). We will show that our method can reproduce demonstrated trajectories with zero error and has most of the features of the state of the art methods such as DMPs, GMMs and PMPs. Moreover, we will demonstrate that our method has less error (deviation from the demonstrated trajectory) as compared to DMPs and PMPs when generalising to new boundary conditions, which might be advantageous for industrial or medical robotics applications.

The paper is organized as follows. We first will start by describing the formalism of our new method, which will be presented in Section II. Afterwards, in Section III we will present results from simulations where we will evaluate and compare our method to DMPs and PMPs. This will be followed by a validation and application of our method in a robot experiment. Finally, we will discuss our results and conclude our paper in Section IV.

II. METHODS

A. Formalism of Trajectory Representation

We derive our trajectory generation method from optimal control theory where we assume a second-order control system with a state vector x and a control signal u:

$$\ddot{\mathbf{x}} = \mathbf{u}, \tag{1}$$

Using assumption (1) it is possible to model the motion of an arbitrary particle in space controlled by a force $\mathbf{u}(t)$ depending on the time t, which leads to the canonical frame of a second-order control system, the double integrator:

$$\dot{\mathbf{x}} = A\mathbf{x}(t) + B\mathbf{u}(t), \tag{2}$$

where $\mathbf{x}(t) \in \mathbb{R}^2$, $\mathbf{u}(t) \in \mathbb{R}$, and $\mathbf{x}(0)$ is given. Here, $\mathbf{x} = [\xi, \dot{\xi}]^T$ is the state vector containing the position ξ and the velocity $\dot{\xi}$ of a motion in a one-dimensional space. Matrices A and B are defined as follows:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

We also require that $|\mathbf{u}(t)| \leq u_{max}$, where u_{max} is the maximal acceleration, i.e., acceleration is bounded.

*The research leading to these results has received funding from the European Communitys Seventh Framework Programme FP7/2007-2013 (Programme and Theme: ICT- 2011.2.1, Cognitive Systems and Robotics) under grant agreement no. 600578, ACAT. This work was also partially supported by funding from Centre for BioRotics, University of Southern Denmark.

[1]Sebastian Herzog and Florentin Wörgötter are with the Department of Computational Neuroscience, University of Göttingen, Germany

[2]Tomas Kulvicius is with the Centre for BioRobotics, University of Southern Denmark, Odense, Denmark `toku@mmmi.sdu.dk`

Therefore, the motion of the particle $|\mathbf{u}(t)|$ is treated as a movement trajectory where the position and the velocity are coupled. To accurately encode and reproduce a trajectory from human demonstration ($\mathbf{x}_r$) we have to assure the following condition:

$$\lim_{t->T} \|\mathbf{x}(t) - \mathbf{x}_r(t)\| = 0, \tag{3}$$

where T is the terminal time (duration of the movement trajectory). Here $\|.\|$ is the Euclidean norm. In order to fulfil the above given condition (3), we have to appropriately choose $\mathbf{u}(t)$. For simplicity the time parameter t will not be used in the following equations. Thus, to fulfil (3) *gain-scheduled* control is used such that

$$\mathbf{u} = -K(\mathbf{x} - \mathbf{x}_r). \tag{4}$$

To acquire K the linear-quadratic regulator method (LQR) is used, with the cost function

$$\hat{J} = \frac{1}{2}\int_0^T \left(\mathbf{x}^T Q_x \mathbf{x} + \mathbf{u}^T Q_u \mathbf{u}\right) dt + \frac{1}{2}\mathbf{x}^T(T) P \mathbf{x}(T). \tag{5}$$

Here, $Q_x \geq 0$, $Q_u \geq 0$, $P \geq 0$ are symmetric, positive (semi-) definite matrices. The Hamiltonian formulation for this is

$$H = \mathbf{x}^T Q_x \mathbf{x} + \mathbf{u}^T Q_u \mathbf{u} + \lambda^T \left(A\mathbf{x} + B\mathbf{u}\right).$$

But instead of solving the Hamiltonian problem the algebraic Riccati equation is considered

$$PA + A^T P - PBQ_u^{-1}B^T P + Q_x = 0, \tag{6}$$

where

$$P = \begin{bmatrix} a & b \\ b & c \end{bmatrix}, Q_x = \begin{bmatrix} q_1^2 & 0 \\ 0 & q_2^2 \end{bmatrix} \text{ and } Q_u = 1$$

are used to solve (6) such that

$$\mathbf{u} = -Q_u^{-1} B^T P \mathbf{x}. \tag{7}$$

Setting

$$K := Q_u^{-1} B^T P = \begin{bmatrix} q_1 & \sqrt{q_2^2 + 2q_1} \end{bmatrix} \tag{8}$$

gain-scheduled control is achieved

$$\mathbf{u} = -K\mathbf{x}$$

and for tracking a reference signal $\mathbf{x}_r$ with the condition (3)

$$\mathbf{u} = -K(\mathbf{x} - \mathbf{x}_r). \tag{9}$$

Applying this to (2) results in

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B(-K(\mathbf{x} - \mathbf{x}_r)) \\ &= (A - BK)\mathbf{x} + BK\mathbf{x}_r. \end{aligned} \tag{10}$$

By tuning the values q_1, q_2 it is possible to adjust the weighting between the importance of position ξ and velocity $\dot{\xi}$. This is useful in cases where $\dot{\xi}$ is not the derivative of ξ, i.e., if we want to have a velocity profile different from the velocity profile of the original trajectory. For example, increasing q_2 with respect to q_1 leads to a trajectory where the velocity profile is more accurate than the position profile, i.e., the system would try to follow a given velocity profile and would be more inaccurate in the position profile. Since in our case the velocity profile $\dot{\xi}$ is indeed the derivative of the position profile ξ, we used $q_1 = q_2 = 1$, i.e., we set equal weights for the position and velocity tracking. Here, $Q_u = 1$ acts as a stiffness parameter, where increasing this parameter leads to a stiffer behaviour of the control $\mathbf{u}$. It can be used to adjust the balance between tracking and smoothness. A small value (less than 1) for $\mathbf{u}$ indicates a strong and strict controller, which may lead to a spiky trajectory, and vice versa, a value above 1 leads to a less strict control behaviour which may result in a smoother trajectory but with bigger deviations. Note that solution of the controller is not exact, however, it is easy to implement. In order to obtain an exact solution the infinite-horizon LQ tracker can be used [5].

B. Compact Trajectory Representation

In the general case, as presented above, OCPs require a complete reference (demonstrated) trajectory in order to reproduce it, whereas DMPs, GMMs and PMPs allow encoding motion by a set of parameters, where the number of needed parameters is less than the number of points in the demonstrated trajectory [2], [3], [4]. In the following we will formalise how an equally compact representation can also be obtained for OCPs.

To obtain a compact representation of a trajectory $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, where n is the number of samples, we represent $\mathbf{u}$ by a number of interpolation nodes $\mathbf{u}^* = \{u_1^*, u_2^*, \ldots, u_{\hat{n}}^*\}$ where $\hat{n} \ll n$. By replacing $\mathbf{u}$ with $\phi(\mathbf{u}^*)$, the system (2) becomes

$$\dot{\mathbf{x}} = A\mathbf{x} + B\phi(\mathbf{u}^*), \tag{11}$$

where $\phi(.)$ is an arbitrary interpolation operator ($\phi(\mathbf{u}^*) = \mathbf{u}$). In our case an interpolation using Chebyshev polynomials of the First Kind was used [6]:

$$\phi(\mathbf{u}^*) = \sum_{k=0}^{N} c_k T_k(u^*),$$

where the coefficients c_k will converge to $a_k = \frac{2}{\pi}\int_{-1}^{1} \frac{f(u^*)T_k(u^*)du^*}{\sqrt{1-u^{*2}}}$ as $k \to \infty$. Here, $T_k(u^*) = cos(k * arccos(u^*))$ is the Chebyshev polynomial of *k-th* order.

In summary, to represent a trajectory in a compact form, instead of Gaussian kernels as used in DMPs, GMMs and PMPs, we use Chebyshev polynomials for which we only need to know $p+1$ via points (sampled from the demonstrated trajectory) if we need p polynomials in order to reconstruct $\mathbf{u}^*$ [1].

C. Generalization

The ability to generalize to new situations, i.e., to new start- and/or end-points is a relevant feature of modern trajectory generators. This generalization is implemented in the OCP framework in the following way. In general, in

[1]A Matlab demo and source code of our framework can be downloaded from http://www.dpi.physik.uni-goettingen.de/cns/index.php?page=optimal-control-primitves

order to generate a trajectory with new boundary conditions, e.g., with a new start-point $\mathbf{x}(0) = \mu$, and a new end-point $\mathbf{x}(n) = \nu$ the *two point boundary* value problem has to be solved. In order to decrease computational cost the *initial* value problem is solved two times, instead. For the new start state the equation (10) is solved with a $\mathbf{x}(0) = \mu$ and the known $\mathbf{u}$ from (4). For the new terminal state we set $\mathbf{x}(0) = \nu$ and $\mathbf{u}$ becomes

$$\hat{\mathbf{u}} = u_n, u_{n-1}, \ldots, u_1. \quad (12)$$

Let us define $\mathbf{x}^{(init)}$ as the solution of (10) for the new initial state and $\mathbf{x}^{(ter)}$ as the solution for the new terminal state. Then, the whole solution for a trajectory with changed initial and terminal states becomes

$$\mathbf{x} = x_1^{(init)}, \ldots, x_{\lfloor n/2 \rfloor}^{(init)}, x_{\lfloor n/2 \rfloor+1}^{(ter)}, \ldots, x_1^{(ter)}. \quad (13)$$

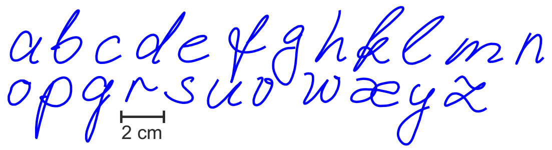


Fig. 1. Data set used for comparison of different movement generation frameworks.

III. RESULTS

In the following we will compare the performance of our framework to the performance of DMPs and PMPs with respect to three aspects: 1) accuracy in reproduction of human trajectories, 2) robustness to perturbations, and 3) position and velocity generalization. For benchmarking and evaluation we used human handwritten letters (see Fig. 1) as already used by some previous studies [2], [7]. We used the same procedure in order to obtain letter samples as used by Kulvicius et al. [7]. Data was collected by utilising a pen tablet (Wacom Intuos3 A3 Wide DTP) with a size of 48.8 cm × 30.5 cm, resolution of 5080 lpi and a sampling rate of 200 Hz.

For comparison with DMPs we used a version of DMPs for generation of hitting and batting movements [8] since it allows generalization to both position and velocity boundary conditions. For learning the weights of the Gaussian kernels we used the $\delta-rule$ as described in Kulvicius et al. [7]. For comparison with PMPs we used the framework presented in [4] and for learning PMP weights we used an expectation maximization algorithm.

A. Trajectory Reproduction

First of all we compare the performance of our framework to DMPs and PMPs with respect to trajectory reproduction. For qualitative comparison we use the letter "a" from the letter dataset as shown in Fig. 1. Here we analysed how the accuracy of trajectory reproduction is influenced by the number of the basis functions (Gaussian kernels in case of DMPs and PMPs, and Chebyshev polynomials in case of OCPs) used to represent the trajectory. Note that for DMPs

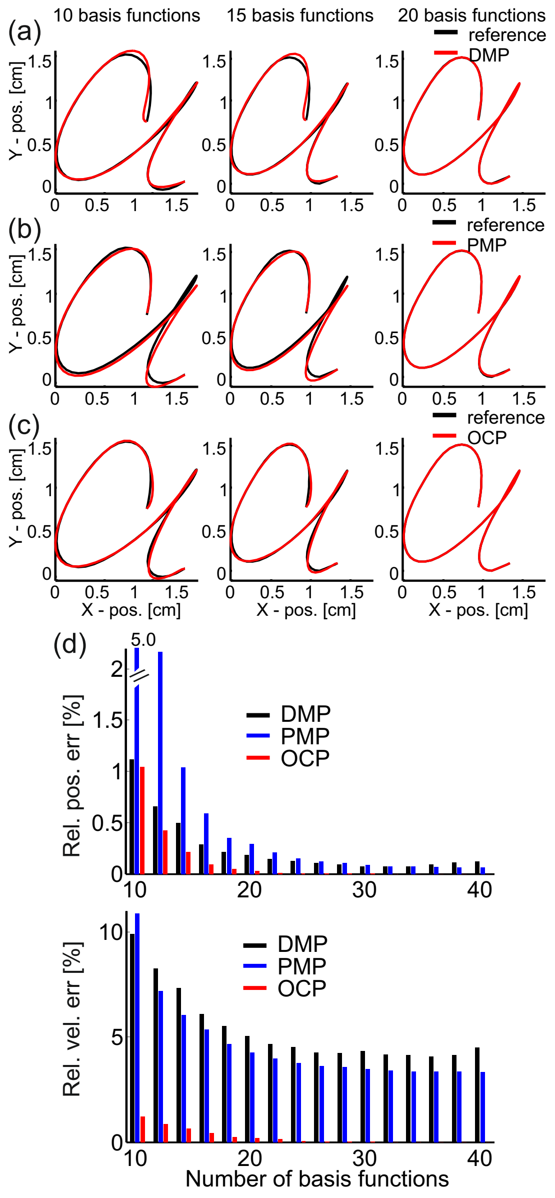


Fig. 2. Results for reproduction of human demonstration (reference) for (a) DMPs, (b) PMPs, and (c) OCPs. Position profiles are shown for each case. (d) Influence of the number of basis functions on the accuracy in the trajectory reproduction. Mean relative position (top) and velocity (bottom) error for the whole trajectory obtained from 23 letters.

and PMPs we used equally distributed kernels with equal variance where the variance was tuned in order to get best accuracy in trajectory reproduction. The results for reproduction of a human motion trajectory are shown in Fig. 2 where we used 10, 15 and 20 basis functions. We can see that OCPs can reproduce the human trajectory more accurately than the other methods when 15 and 20 basis functions are used. We also quantified the movement reproduction property of the different approaches statistically where we used all 23 letters from the data set (see Fig.1). We calculated the accuracy of trajectory reproduction for different numbers of basis functions: 10, 12, 14...40. Statistics for all three methods are presented in Fig. 2 (d) where we can see that

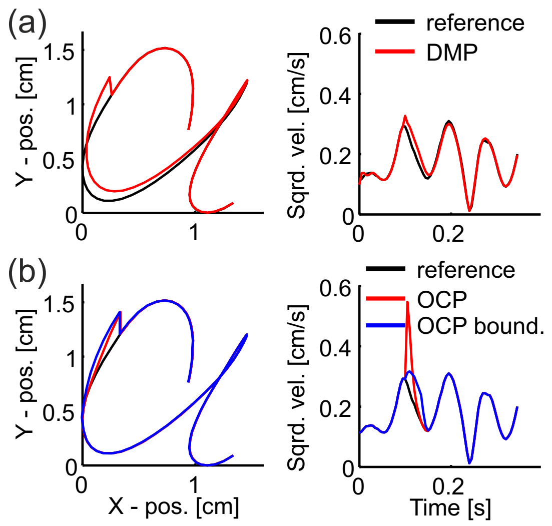


Fig. 3. Comparison of robustness to perturbation between (a) DMPs and (b) OCPs without bounded acceleration (OCP) and with bounded acceleration (OCP bound.). Position (left) and velocity (right) profiles are shown for each case.

OCPs require fewer basis function to reproduce trajectory accurately as compared to DMPs and PMPs. In case of OCPs, it was already possible to reproduce trajectories with zero error with 32 basis functions, whereas for DMPs and PMPs the error drops around 35 basis functions and is below 0.1% for position and below 5% for the velocity profile. Based on these results, in order to make a fair comparison of the different methods, in the following experiments we used 35 basis functions for all three methods.

B. Robustness to Perturbations

Next, we looked at the behaviour of our trajectory generation method with respect to perturbation robustness were we again compared our method against DMPs. We perturbed the system at some arbitrary time by shifting the position profile in both X and Y direction by 30%. The results of such a test are shown in Fig. 3 were we present the behaviour of DMPs in panel (a) and two cases for the OCPs in panel (b): 1) without bounded acceleration (OCP) and 2) with bounded acceleration (OCP bound.). Allowing the solver for the dynamic system to do arbitrary big steps for the acceleration leads to an unbounded response, bounding the acceleration by u_{max} leads to the bounded response. We can observe that in case of unbounded acceleration the OCP system produces a velocity jump, which is different from the behaviour of the DMP system and might be even dangerous for robotic applications. This is due to the fact that OCPs are optimal with respect to minimal position and velocity deviation depending on the choice of Q_u, where the system tries to come back to the original trajectory, as soon as possible. Such undesired jumps in the velocity profile can be avoided by limiting the maximal acceleration u_{max}. We can see that by limiting acceleration we can obtain much smoother response of the system. Moreover, we can also observe that the OCP system converges to the original path much sooner as compared to DMPs.

C. Position and Velocity Generalization

Finally, we analysed how well can our trajectory generation method generalize to new situations, i.e., when position or velocity boundary conditions are changed. Here we compared the performance of our framework to both DMPs and PMPs. First, we compared behaviours of the three systems when position boundary conditions are changed. To do so, we changed the end-point position (for both X and Y components) by relatively increasing the end-point value of the demonstrated trajectory by 30%. Note that in this case we kept the velocity boundary conditions the same as those for the demonstrated trajectory. Qualitative results are presented in Fig. 4 (a1-c1) where we can see that the performance of PMPs and OCPs is different from that of DMPs. In case of DMPs we get much larger deviations from the original trajectory (along the whole trajectory) as compared to PMPs and OCPs where deviations are only at the end of the trajectory. Statistics for the position generalization are shown in Fig. 4 (d1) where we show average relative position and velocity error obtained from all 23 letters (see Fig. 1). In general, we can observe that, when using DMPs, deviation from the demonstrated trajectory increases dramatically when increasing the change in end-point position. Also, the deviation of the velocity profile is much higher as compared to PMPs and OCPs. The results also demonstrate that OCPs produce smaller deviations from the demonstrated trajectory (in both position and velocity profile) as those observed with PMPs. Mean relative position error at the desired new end-point for DMPs was 0.008%, whereas for PMPs and OCPs the error was zero. This is due to the fact that DMPs converge to the end point only asymptotically [2].

Similar to position generalization, we also performed a test where we compared all three frameworks with respect to the change of velocity boundary conditions. So, here we changed the end-point velocity by scaling it relatively to the end-point velocity of the human demonstration. End-point position was kept unchanged. Qualitative results for the 30% end-point velocity change are shown in Fig. 4 (a2-c2) whereas statistics are shown in Fig. 4 (d2). Similar to the results obtained from position generalisation we observe that DMPs result in much larger position and velocity deviations as compared to PMPs and OCPs. Also as in the previous case, OCPs outperform PMPs by producing only deviations at the very end of the trajectory, which is necessary in order to meet the new boundary conditions. Similar to the case of position generalisation, relative velocity error at the desired new end-point for PMPs and DMPs was zero, whereas mean relative velocity error at the desired new end-point for DMPs was 0.434%, which again is due to the asymptotic convergence. We summarize results for position and velocity generalization in Fig. 5 (a) and (b), respectively, where we show mean relative position and velocity errors for the whole trajectory obtained from 23 letters. Results demonstrate that OCPs outperform both DMPs and PMPs in that the OCP-system can follow the demonstrated trajectory better when

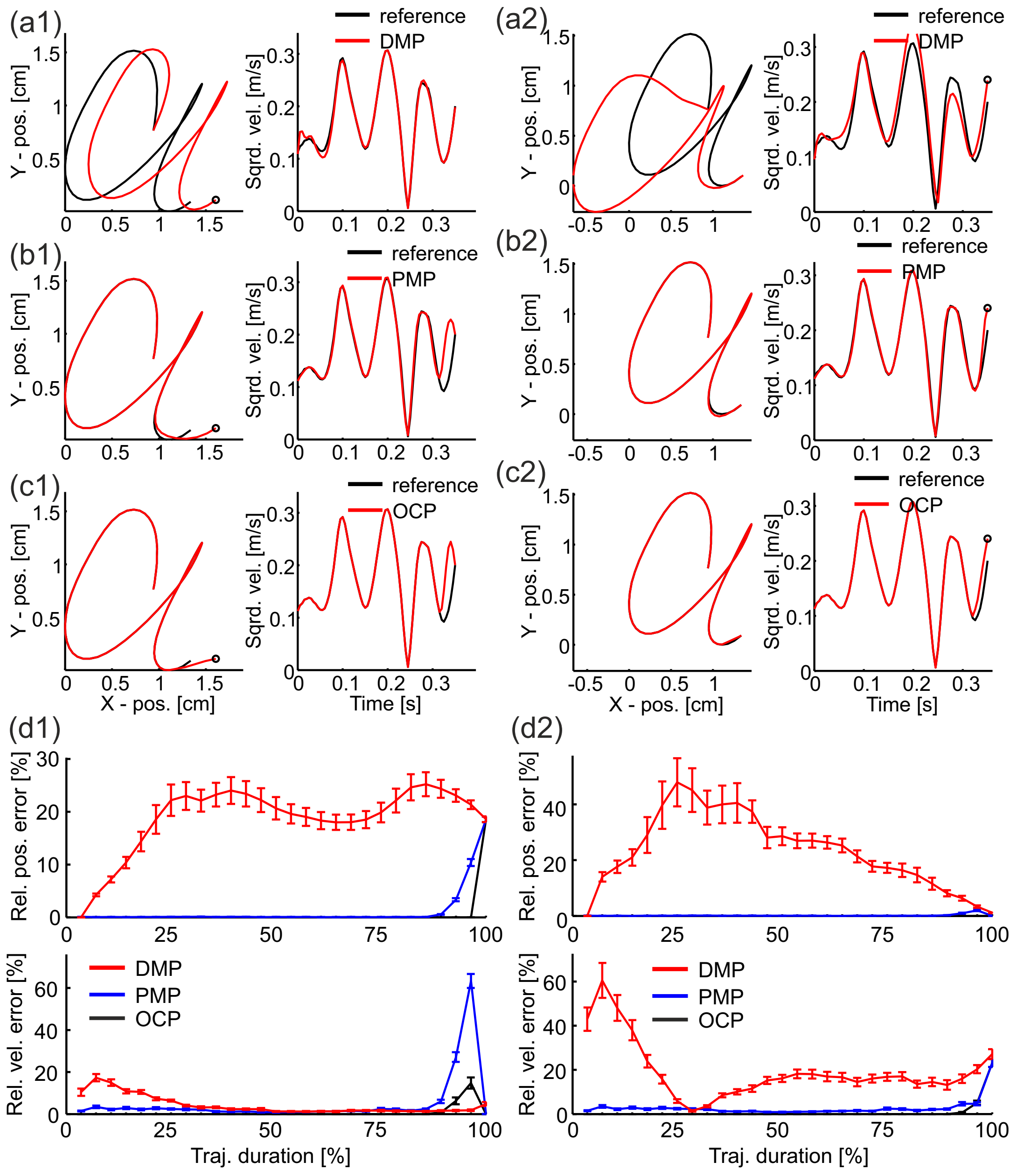


Fig. 4. Examples of (a1-c1) position and (a2-c2) velocity generalization for (a) DMPs, (b) PMPs and (c) OCPs. Here we changed position/velocity at the end-point by 30%. Position (left) and velocity (right) profiles are shown for each case. The circles denote new end-points. (d) Statistics for (d1) position and (d2) velocity generalization. Mean relative position (top) and velocity (bottom) error are shown for each case. Error bars denote standard deviation.

generalising to new boundary conditions.

D. Robot Experiment

Last but not least, we validated our method in a robot experiment, namely a pouring task, where the task for the robot was to generalise to new situations based on a human demonstrated trajectory. DMPs are currently probably still the most widely used method for the generation of dynamic, adaptive, and generalizable trajectories. Thus, we compared the performance of our approach only against DMPs. For our experiments we used a seven DOF "KUKA-LWR" robot-arm [9]. Human trajectories were recorded by using kinaesthetic guidance and were encoded with DMPs and OCPs as described above. In this experiment, a human demonstrated how to pour 50ml of sand from a container containing 200ml of sand into an empty glass (see Fig. 6 (c) and supplementary video). The task for the robot, different from human demonstration, was to empty the container completely, i.e., to pour all the sand (200ml) from the container into the glass. To do so, obviously, the robot needed to tilt the container much more as compared to human demonstration. In this case we performed the task in joint-space where we manually set the end-points of three joint position profiles in order to tilt the wrist as much as to empty the glass completely. Note that different from other

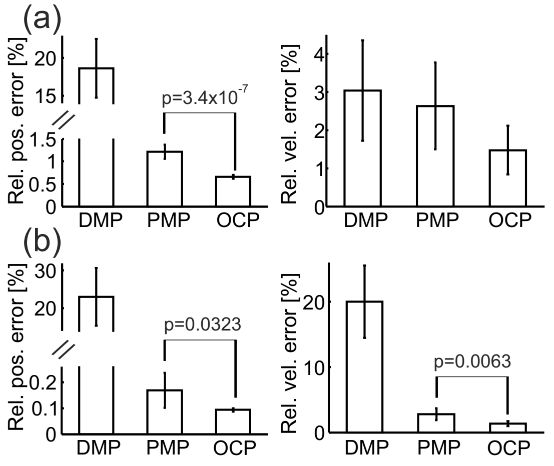


Fig. 5. Statistics for (a) position and (b) velocity generalization. Mean relative position (left) and velocity (right) error for the whole trajectory is shown for each case. Error bars denote confidence intervals (95%) of mean, where p stands for the probability of the *t-test* ($\alpha = 0.05$).

approaches used in learning to pour tasks [10], [11] we do not change parameters (weights in case of DMPs) of the trajectory generation method but only boundary conditions in order to generalise to the new situation. The resulting trajectories for DMPs and OCPs are shown in Fig. 6 (a) and (b), respectively. We can see that in the case of DMPs, as already demonstrated in Fig. 4 (a1), the trajectory is changed along the whole path when generalising to new position endpoints. This leads to the fact that the robot starts tilting the container too early and, as a consequence, spills some of the sand on the table (see Fig. 6 (d)). By contrast, OCPs change the trajectory only at the end of the movement, which enables the robot to pour the sand into the glass without spilling.

TABLE I
COMPARISSON OF DMPS, PMPS AND OCPS WITH RESPECT TO DIFFERENT FEATURES

Feature	# of param.	Built-in tracker	Robust. to pert.	Gener.
DMP	$p+1$ or $3p$	+	+	+
PMP	$p+1$ or $3p$	-	-*	+
GMM	$4p$	+	+	+/-**
OCP	$p+1$	+	+	+

* It is not possible without a tracker
** No velocity generalisation

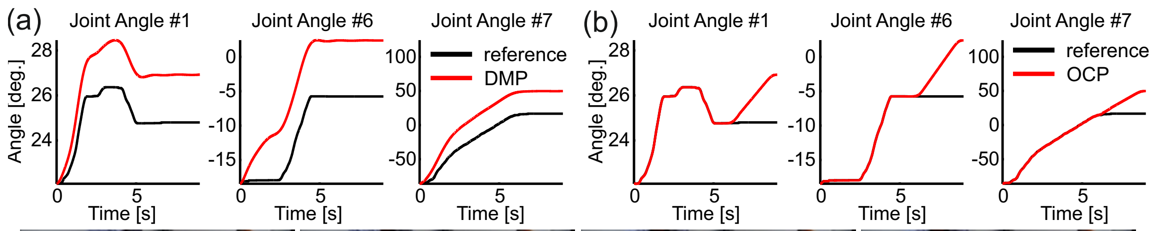
IV. DISCUSSION

In this paper we presented a novel trajectory generation method for the generation of highly accurate movements with arbitrary position and/or velocity boundary conditions. We showed that the method is comparable to the state-of-the-art methods such as DMPs [2] and PMPs [4] in terms of features such as compactness of trajectory representation, robustness to perturbations, and generalisation to new boundary conditions. Moreover, we showed that the new method generates motions which deviates less from the demonstrated trajectory as compared to DMPs or PMPs when generalising to new boundary conditions. It is very important to note that this comparison is by no means a devaluation of existing methods but rather shows behavioural differences of different trajectory methods where in some situations one method may be more preferable than the others and vice versa. For instance, we demonstrated by a robot experiment that in some situations the trajectories generated using the new method can be beneficial as compared to trajectories generated with DMPs. In general, the behaviour of OCPs is qualitatively similar to PMPs, however, different from PMPs and DMPs, OCPs can achieve higher accuracy in motion reproduction with fewer basis functions which requires less parameters to encode a trajectory. Note, that PMPs [4] due to their probabilistic nature are meant to encode trajectories from multiple demonstrations, and are not (as DMPs and GMMs) specifically designed to encode single trajectories accurately.

A comparison of DMPs, GMMs, PMPs and OCPs in terms of features is provided in Table I, where we can see that all methods are capable of generalisation to new boundary position and velocity (except GMMs) conditions. However, PMPs require an additional tracker on top of the trajectory generation model, whereas DMPs and OCPs (similar to DMPs) have a built-in tracker in the model. In general, compact trajectory representation with the OCP framework requires less parameters as compared to DMPs and PMPs. For OCPs, if we want to represent a trajectory with p polynomials, we only need to know the via points, which in total leads to $p+1$ parameters. For DMPs and PMP we would need for p Gaussian kernels a total of $3p$ parameters (mean, variance and weight for each Gaussian function). Only if all kernels are the same one would also have $p{+}1$ parameters, but many situations exist where the use of identical kernels is not optimal. For comparison, GMMs in total require $4p$ parameters in order to represent trajectory by a mixture of p 2D Gaussian kernels (in GMMs movements are represented in a phase space of position and velocity [3]). However, different from our method, GMMs can encode a set of several different trajectories, whereas our method (as DMPs) encodes only a single trajectory.

As described above our approach utilises Linear-Quadratic Regulator (LQR) control for tracking of trajectories. We have chosen LQR for the following reasons. In general, as shown in [12], [13] and [14], LQR controller can be designed as a Proportional Integral (PI) controller. Moreover, as demonstrated by [15], the more specialised LQR controller has superior properties regarding oscillations as compared to a general Proportional Integral Derivative (PID) controller. Thus, we utilised LQR and not standard PID controller due to better tracking properties.

Our approach is similar to the one presented in [16], where LQR controller is utilised to track a desired trajectory by a UAV. Different from that approach, in our case, we used a more general model, the double integrator with a mass equal to one. Since our model is simpler, there is no need



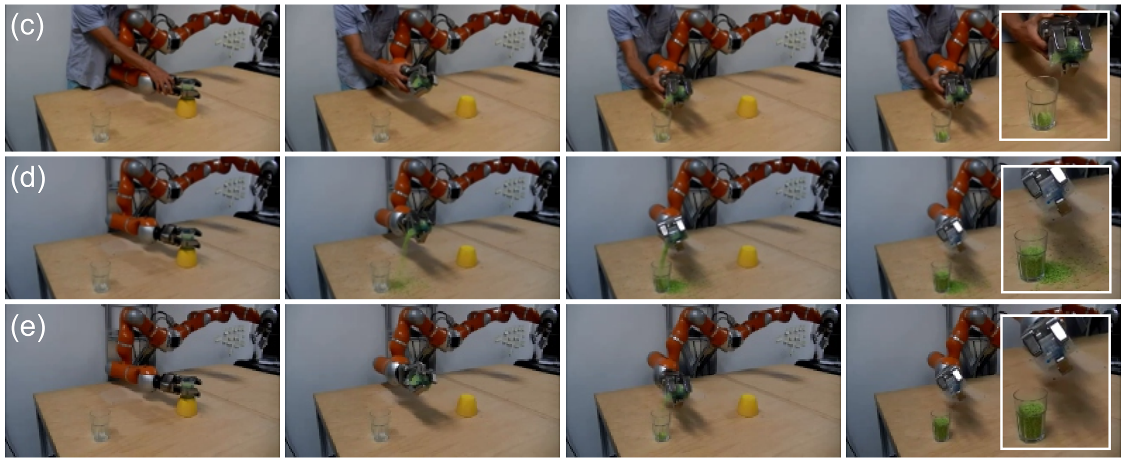


Fig. 6. Results from pouring experiment. (a,b) Resulting joint (position) trajectories for a new position end-point obtained with (a) DMPs and (b) OCPs. Note that here we show only joint trajectories which differed from human demonstration. (c-e) Selected frames from action execution performed by (c) a human, (d) a robot utilising DMPs and (e) a robot utilising OCPs.

to reduce the dimensions of the problem like it is done in [16], which allows a more intuitive use of the controller and the parametrisation. In addition, it is easier to add further constraints if necessary.

We also would like to stress that our approach is not restricted with respect to the way a trajectory is encoded, i.e., it does not necessarily have to be encoded with Chebyshev polynomials as shown in this study. In our case, different from DMPs, PMPs and GMMs, a trajectory is a (control-)signal, which means that one can use and apply any kind of control theory and/or signal processing in order to represent a trajectory. For example, a trajectory could be represented by a parabola or a sine-wave, too.

Common industrial applications, such as gluing or welding, often require highly exact position and velocity profiles for the to-be-performed trajectory. For example, the robot should not decelerate at corners, by which the gluing (or welding) track would become uneven. The attractive features of our method – compactness and high accuracy – could have a great potential especially for such applications.

References

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, planning and control*. Springer Publishing Company, 2009.
[2] J. A. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, Feb. 2013.
[3] S. M. Khansari-Zadeh and A. Billard, "Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models," *IEEE Trans. Robot.*, vol. 27, pp. 943–957, Oct. 2011.
[4] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems 26*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2616–2624.
[5] E. Barbieri and R. Alba-Flores, "On the infinite-horizon lq tracker," *Systems & Control Letters*, vol. 40, no. 2, pp. 77–82, 2000.
[6] U. W. Hochstrasser, *Orthogonal Polynomials.*, ser. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. New York: Dover, 1972.
[7] T. Kulvicius, K. J. Ning, M. Tamosiunaite, and F. Wörgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 145–157, 2012.
[8] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *Proc. 2010 IEEE Int. Conf. Robotics and Automation*, 2010, pp. 1–6.
[9] Kuka Robot Systems. [Online]. Available: http://www.kuka-robotics.com
[10] M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter, "Learning to pour combining goal and shape learning for dynamic movement primitives," *Robot. and Auton. Syst.*, vol. 59, no. 11, pp. 910–922, 2011.
[11] A. Nemec, R. Vuga, and A. Ude, "Efficient sensorimotor learning from multiple demonstrations," *Advanced Robotics*, vol. 27, no. 13, pp. 1023–1031, 2013.
[12] W. S. Levine, *The control handbook*, ser. The electrical engineering handbook series. CRC Press New York, 1996.
[13] J.-B. He, Q.-G. Wang, and T.-H. Lee, "PI/PID controller tuning via LQR approach," in *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, vol. 1, 1998, pp. 1177–1182.
[14] N. Kumari and A. N. Jha, "Automatic generation control using LQR based PI controller for multi area interconnected power system," *Advance in Electronic and Electric Engineering*, vol. 4, no. 2, 2014.
[15] A. Jose, C. Augustine, S. M. Malola, K. Chacko, *et al.*, "Performance study of PID controller and LQR technique for inverted pendulum," *World Journal of Engineering and Technology*, vol. 3, no. 02, p. 76, 2015.
[16] I. D. Cowling, W. J. F., and A. K. Cooke, "Optimal trajectory planning and LQR control for a quadrotor UAV," in *UKACC International Conference on Control*, 2006.