A novel approach to dynamic movement imitation based on quadratic programming

Carlos Cardoso[1], Lorenzo Jamone[1], Alexandre Bernardino[1]

***Abstract*— This paper proposes a novel approach to generate trajectories that generalize given demonstrations according to optimality criteria. By formulating the problem as a quadratic program we can efficiently incorporate constraints to adapt to new desired motion requirements while achieving the main goal of matching the acceleration profile of the demonstration. This makes our method particularly suited for the imitation and generalization of trajectories such as hitting movements, where it is crucial to maintain the dynamic traits of the demonstration while respecting strict requirements for the goals position, velocity and time. Our method draws inspiration from the Dynamical Movement Primitives (DMPs) framework, preserving its desirable properties of flexibility and rejection of disturbances during execution. Moreover, it offers an higher degree of control on the generated solution, allowing for example i) to limit the instantaneous positions, velocities and accelerations during the whole trajectory, and ii) to add intermediate way points that were not present in the demonstration. With current state-of-the-art solvers of quadratic programs, a problem with hundreds of parameters can be solved in tens of milliseconds in a standard computer, allowing practical applications. Our methodology results in trajectories with a very good approximation of the shape traits of the demonstration, with additional flexibility in specifying constraints of the generated trajectory.**

I. Introduction and Related Work

Modern robots are expected to operate alongside humans, performing complex tasks in unstructured environments, showing flexibility, adaptability and generalization capabilities. Given the diversity of situations faced by the robot, that cannot be fully pre-determined, learning techniques are going to play a major role to equip robots with complex motor skills and behaviors [1].

One practical way to allow robots to learn motor skills is through the "programming by demonstration" paradigm [2], [3], [4], also known as "learning by demonstration" or "imitation learning". The main idea is that a motor skill (i.e. the trajectory of a movement) is demonstrated to the robot (typically through kinesthetic teaching [5], [6]), and a general representation of the movement (i.e. a movement primitive) is learned from the recorded data. Using such representation the robot can replicate the demonstrated motion under different conditions, e.g. with a different starting or ending position, thus showing not only the ability to learn from a human teacher, but also to generalize and adapt what has been learned to different situations. Moreover, the learned primitive can be progressively refined through practice, using for example Reinforcement Learning (RL) techniques [7].

One key issue is therefore to find a proper representation for such movement primitives. A number of solutions have been proposed during the last decade, resorting for instance to neural networks [8], probabilistic estimation [9] and many other techniques. One of the most promising approaches draws from the theory of dynamical systems, giving raise to solutions such as Stable Estimator of Dynamical Systems (SEDS [10]), sequenced Linear Dynamical Systems [11], Implicit Dynamical Systems [12] and, most notably, Dynamic Movement Primitives (DMPs [13], [14]). This latter formulation in particular has proven to be a very effective tool for imitation learning, and has been therefore widely used in robotics and inspired many extensions to add velocity goals [15], [16] and allow uncertainty and way-points in the execution [17]. However, the DMPs approach still has a few drawbacks. One is that, despite allowing quick adaptation to new start/end positions, it may generate undesired motion profiles (especially in terms of accelerations) that are not under full control. Another one is that it lacks the possibility to impose constraints in the motion, such as joints position and velocity limits.

In this paper we propose an alternative solution in which imitation is formulated as a global optimization problem, thus allowing a much higher degree of control on the trajectory that imitates the demonstration. More specifically, the optimization procedure aims at finding the trajectory that imitates the acceleration profile of the demonstration best,

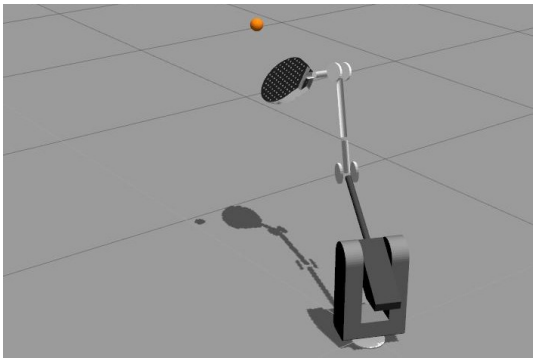

Fig. 1. A screenshot of the robotic arm in the Gazebo simulator, while performing a ball hitting task.

*This work was partially supported by the European Commission under the POETICON++ (FP7-ICT-288382) and LIMOMAN (PIEF-GA-2013-628315) projects, and by the Portuguese Science Foundation FCT projects AHA (CMUP-ERI/HCI/0046/2013), BIOMORPH (EXPL/EEI AUT/2175/2013) and LARSyS (UID/EEA/5009/2013).

[1]Carlos Cardoso, Lorenzo Jamone and Alexandre Bernardino are with the Institute for Systems and Robotics, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal carlos.cardoso@tecnico.ulisboa.pt {ljamone,alex}@isr.ist.utl.pt

given the different conditions in which the movement has to be executed - e.g. with different starting and ending positions and velocities. Moreover, by formulating the problem this way, a number of additional constraints can be easily added, allowing for instance to comply with joint limits (positions, velocities, accelerations, etc.) over the whole trajectory, or to add way-points that were not in the original demonstration. To the best of our knowledge, such features are not present in DMP formulations that generalize a single demonstration. We perform an experimental evaluation in which we compare our proposed method to a state-of-the-art DMP method. We provide a demonstration (joint space trajectories) that corresponds to a ball hitting movement performed by a 6dof robotic arm holding a tennis racket, simulated in the Gazebo [18] simulation environment (see Fig. 1). Then we adapt such demonstration to different execution conditions, comparing the two methods. More specifically, we show i) adaptation to different initial/final states and movement duration, ii) inclusion of joint limits, iii) addition of intermediate way-points and iv) robustness to noise and disturbances.

The remainder of the paper is organized as follows. In Section II we summarize the DMP framework, highlighting the main pros and cons, and in Section III we propose our alternative solution based on global optimization. Then in Section IV we presents our experimental results, and finally in Section V we report our conclusions.

II. Imitation with DMPs

As previously introduced, Dynamic Movement Primitive (DMP) is a widely adopted formalism to represent motor actions in a way that allows flexible adjustment without custom parameter tuning. After a primitive is learned from a demonstration (or a series of demonstrations) it can then be generalized to different initial and goal states while maintaining its overall shape. The original DMPs have two formulations, one for discrete movements and another for periodic motion [19]. In this paper we focus on the discrete formalization, that is needed to represent point-to-point tasks.

A. Original Formulation

The DMP system controls the motion of a scalar variable y through a point attractor g that makes the trajectory converge asymptotically to the goal and a nonlinear forcing term f that encodes the characteristics of the demonstration. The two terms are coupled by a *canonical system* z that acts as a replacement of time.

$$\begin{aligned} \tau^2\ddot{y} &= \alpha_y(\beta_y(g-y)-v)+f(z,g) \\ \tau\dot{y} &= v \qquad \tau\dot{z} = -\alpha_z z \end{aligned} \tag{1}$$

The gains α_y, β_y are chosen to make the 2^{nd} order system critically damped, ie. $\beta_y = \alpha_y/4$. The temporal scaling term τ allows the primitive to execute the movement faster or slower while preserving its shape.

Let us consider a demonstrated trajectory as $d(t)$. The objective of learning a DMP is to compute an approximation of f such that the observed profile of the trajectory is as close as possible to the demonstration. Rewriting the first equation of (1) and replacing the motion variable y by the demonstration d we have:

$$f_{target}(z,g) = \tau^2\ddot{d} - \alpha_y(\beta_y(g-d) - \tau\dot{d}) \tag{2}$$

This term can be represented by a normalized linear combination of Gaussian basis functions [14], as follows:

$$\begin{aligned} f_{target}(z,g) &= \frac{\sum_{i=1}^{N}\psi_i(z)w_i}{\sum_{i=1}^{N}\psi_i(z)}\xi(z,g) \\ \psi_i(z) &= exp(-\frac{1}{2\sigma_i^2}(z-c_i)^2) \\ \xi(z,g) &= z(g-y(0)) \end{aligned} \tag{3}$$

The Gaussian basis functions ψ_i have centers c_i along the exponential of the Canonical system so that they are equally spaced in time. The scaling term $\xi(z,g)$ makes the accelerations converge to zero near the goal and to normalize their values according to the amplitude of the movement.

The weights w_i can be learned from the samples of the original demonstration at sample times $z_p,\ p \in \{0,\cdots,P\}$, using for example Locally Weighted Regression [13]:

$$w_i = \frac{\boldsymbol{s}^T\boldsymbol{\Gamma}_i\boldsymbol{f}_{target}}{\boldsymbol{s}^T\boldsymbol{\Gamma}_i\boldsymbol{s}} \tag{4}$$

where

$$\begin{aligned} \boldsymbol{s} &= \left[\ \xi(z_0)\dots\xi(z_P)\ \right]^T \\ \boldsymbol{f}_{target} &= \left[\ f_{target}(z_0,g)\dots f_{target}(z_P,g)\ \right]^T \\ \boldsymbol{\Gamma}_i &= \begin{pmatrix} \psi_i(z_0) & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \psi_i(z_P) \end{pmatrix} \end{aligned} \tag{5}$$

The canonical system has $\alpha_z > 0$ so that it converges asymptotically to zero. This ensures that, if the weights are bounded, the forcing function's interference eventually vanishes as the movement ends, ensuring that the point attractor is free to converge to the goal. The initial value of z is usually set to 1 but any positive value can be chosen.

B. Hitting Movements using DMPs

For some situations such as striking a moving ball and throwing darts it is necessary to adapt to an end goal with a desired final velocity v_f, different from zero. A DMP formulation that adapts to goals with velocity was proposed by Kober et. al [15]. The proposed system used a shifting goal with linear velocity. This system however had some drawbacks: First, fast changes of the goal position and velocity lead to error in the final velocity. Second, when the final and starting position are close to each other the system generates large accelerations. To overcome these shortcomings Mulling et al [16] proposed an improved system, which is defined as follows:

$$\begin{aligned} \tau\dot{v} &= \alpha_y(\beta_y(g-y)+\dot{g}\tau - v) + \ddot{g}\tau^2 + \eta f(z) \\ \tau\dot{y} &= v \end{aligned} \tag{6}$$

Here the goal follows a polynomial that starts and ends with zero accelerations and the correct starting and final position and velocity, computed as:

$$g = \sum_{j=0}^{5} b_j \left(-\tau \frac{ln(z)}{\alpha_z}\right)^j \qquad \dot{g} = \sum_{j=1}^{5} j b_j (-\tau \frac{ln(z)}{\alpha_z})^{j-1}$$
$$\ddot{g} = \sum_{j=2}^{5} (j^2 - j) b_j (-\tau \frac{ln(z)}{\alpha_z})^{j-2} \tag{7}$$

The 5th order polynomial coefficients are found by setting the boundary conditions such that it has the desired initial and final positions and velocities. The accelerations are set to zero at the start and at the end of the movement.

III. Quadratic Optimization for Motion Primitives

The approach in Mulling et al [16], implemented through (6) and (7), achieves the final goal position and velocity with accuracy. However, the polynomial trajectory is computed independently of the demonstrated trajectory and may lead to accelerations profiles significantly different from the demonstration. Moreover, the acceleration vanishes at the end of the movement. This distorts the demonstrated trajectory at the most important phase in hitting movements. To have a greater control on the resulting trajectory profiles, we formulate the problem as a convex quadratic program (QP). The imitation problem is addressed by computing the trajectory that best imitates the acceleration profile of the demonstration while constrained to start and end at specific positions and velocities. Additional constraints can be incorporated, for instance via-points and limits in the joint positions, velocities or accelerations during the whole movement trajectory.

A. Formulation

We use a dynamical system as (6), but instead of having separate reference trajectory and forcing terms, we generate a single desired trajectory directly taking all constraints into account (imitation of the demonstration, initial and final states, joint limits, waypoints) and plug it into a reference tracking dynamical system. This is no longer a DMP but still ensures robustness to noise and perturbations.

To compute the desired trajectory we use a Gaussian kernel expansion as in the DMPs forcing term (3). Let $g(t),\ t \in [t_0, t_f]$ denote the desired trajectory to be executed between t_0 and t_f. Its acceleration is represented as:

$$\ddot{g}(t) = \sum_{i=1}^{N} w_i \psi_i(t) \tag{8}$$

The Gaussian basis functions $\psi_i(t)$ are the same ones used for DMPs but time warped to fit the desired time interval. Since we do not need a canonical system with exponential decay, we can have the basis functions spaced linearly in time and with constant width. The expression for the $\psi_i(t)$ is the same as in (3), but now the input is time t instead of phase z, and centers c_i and variance σ are shifted and warped according to the transformation from the demonstration time interval to the imitation time interval.

The desired trajectory position and velocity can be obtained by integration:

$$\dot{g}(t) = v_0 + \sum_{i=1}^{N} w_i \psi_i^{'}(t)$$
$$g(t) = p_0 + (t - t_0) v_0 + \sum_{i=1}^{N} w_i \psi_i^{''}(t) \tag{9}$$

where [1]

$$\psi_i^{'}(t) = \int_{t_0}^{t} \psi_i(\tau) d\tau \quad \psi_i^{''}(t) \quad = \int_{t_0}^{t} \psi_i^{'}(\tau) d\tau$$

Given a demonstration $d(t)$ and the desired initial and final times, positions and velocities, respectively $(t_0, p_0, v_0, t_f, p_f, v_f)$, the optimization problem is thus written as:

$$\begin{aligned} \underset{w_i}{\text{minimize}} \quad & \int_{t_0}^{t_f} \left(\sum_{i=1}^{N} w_i \psi_i(t) - \tau^2 \ddot{d}(t) \right)^2 dt \\ \text{subject to} \quad & p_0 + (t_f - t_0) v_0 + \sum_{i=1}^{N} w_i \psi_i^{''}(t_f) = p_f \\ & v_0 + \sum_{i=1}^{N} w_i \psi_i^{'}(t_f) = v_f \end{aligned} \tag{10}$$

where τ is the time scale due to different durations of the demonstration and imitation.

After the trajectory is computed, it can be used as a shifting goal in the acceleration-based controller:

$$\ddot{y} = \alpha_y (\beta_y (g - y) + \dot{g} - v) + \ddot{g}$$
$$\dot{y} = v \tag{11}$$

The time scaling factor is included directly in the solution of the minimization problem. Every time a trajectory has to be computed, the new $\psi_i(t)$ terms are recomputed to cover the desired time span, so it is no longer necessary to scale by τ in the dynamic system.

Note that the computed trajectory plays the role of both the forcing term and the shifting goal. Because the trajectory already incorporates the imitation, there is no need to use two different entities to implement the dynamical controller.

B. Solution

To solve the optimization problem we rearrange (10) as a quadratic program (QP) with linear constraints in standard form. Then, any existing modern QP solver can be used to obtain the result.

Expanding the cost function in (10) and defining $\psi_{ij} = \int \psi_{ij}(t)$, $\theta_j = \int \psi_j(t) \ddot{d}(t)$ and $D^2 = \int \ddot{d}^2(t)$ we get[2]:

[1] The cumulative functions $\psi_i^{'}(t)$ and $\psi_i^{''}(t)$ are, in practice, approximated by numerical integration.

[2] In practice the terms ψ_{ij} and θ_j are approximated by numerical integration

$$J = \int_{dt} \left(\sum_{j=1}^{N} w_j \psi_j(t) - \tau^2 \ddot{d}(t) \right)^2 =$$
$$J = \sum_{j=1}^{N} \sum_{i=1}^{N} w_{ij} \psi_{ij} - 2\tau^2 \sum_{j=1}^{N} w_j \theta_j + \tau^4 D^2$$

where $w_{ij} = w_i w_j$, $\psi_{ij}(t) = \psi_i(t)\psi_j(t)$ and the integrals are between t_0 and t_f.

Because the last term does not depend on the w_i's it can be left out of the optimization cost J. Therefore, representing in matrix form we get:

$$J = \mathbf{w}^T \Psi \mathbf{w} - 2\tau^2 \mathbf{w}^T \boldsymbol{\theta} \tag{12}$$

where

$$\mathbf{w} = \begin{bmatrix} w_1 \dots w_n \end{bmatrix}^T \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \dots \theta_N \end{bmatrix}^T$$
$$\Psi = \begin{pmatrix} \psi_{11} & \dots & \psi_{1N} \\ \vdots & \ddots & \vdots \\ \psi_{N1} & \dots & \psi_{NN} \end{pmatrix} \tag{13}$$

The constraints of (10) can be written as:

$$\begin{aligned} \mathbf{w}^T \boldsymbol{\psi}''(t_f) &= p_f - p_0 - (t_f - t_0) v_0 \\ \mathbf{w}^T \boldsymbol{\psi}'(t_f) &= v_f - v_0 \end{aligned} \tag{14}$$

where

$$\boldsymbol{\psi}''(t_f) = \begin{pmatrix} \psi_1''(t_f) \\ \vdots \\ \psi_N''(t_f) \end{pmatrix} \quad \boldsymbol{\psi}'(t_f) = \begin{pmatrix} \psi_1'(t_f) \\ \vdots \\ \psi_N'(t_f) \end{pmatrix} \tag{15}$$

Now we can write (10) in standard QP form:

$$\begin{aligned} &\underset{x}{\text{minimize}} \quad \frac{1}{2} \mathbf{w}^T 2\Psi \mathbf{w} + \tau^2 \boldsymbol{\theta}^T \mathbf{w} \\ &\text{subject to} \\ &\begin{pmatrix} \boldsymbol{\psi}''(\boldsymbol{t_f}) \\ \boldsymbol{\psi}'(\boldsymbol{t_f}) \end{pmatrix} \mathbf{w} = \begin{pmatrix} p_f - p_0 - (t_f - t_0) v_0 \\ v_f - v_0 \end{pmatrix} \end{aligned} \tag{16}$$

These values can be used in a standard QP solver. For adapting to new targets only p_f, p_0, v_f, v_0 and Ψ change in the resulting QP. The basis functions Ψ must be recomputed for changes in the final time. The $\boldsymbol{\theta}$ vector, obtained from the original demonstration remains unchanged and does not have to be recalculated. This vector therefore is the invariant that defines the primitive.

The minimization must be solved each time we want to execute a new motor action. However existing QP solvers can find a solution very fast. If we are controlling an arm with many DOF's, solving for each individual DOF is an intrinsically parallel problem and a solution can be found in the same time as with only one primitive given there is an execution thread available for each DOF.

Additional constraints can also be added to ensure that joint accelerations, velocities and positions are bounded within safe physical limits. For a discretization of the time interval at instants t_i the following constrains can be added:

$$\begin{aligned} \ddot{\boldsymbol{q}}_{\boldsymbol{min}} &\leq \mathbf{w}^T \boldsymbol{\psi}(t_i) \leq \ddot{\boldsymbol{q}}_{\boldsymbol{max}} \\ \dot{\boldsymbol{q}}_{\boldsymbol{min}} &\leq v_0 + \mathbf{w}^T \boldsymbol{\psi}'(t_i) \leq \dot{\boldsymbol{q}}_{\boldsymbol{max}} \\ \boldsymbol{q}_{\boldsymbol{min}} &\leq p_0 + (t_i - t_0) v_0 + \mathbf{w}^T \boldsymbol{\psi}''(t_i) \leq \boldsymbol{q}_{\boldsymbol{max}} \end{aligned} \tag{17}$$

However this requires adding 6 constraints for each sample and may increase the computation time. In our case the problem is still solved in the order of tenths of a second for trajectories with 1000 points.

IV. Experimental results

To test our proposed method we generated a ball hitting movement with our 6dofs arm (see Fig. 1), defined in joints space, to serve as a demonstration for both a DMP with polynomial shifting goal (the system proposed in [16]) and for our QP formulation. We use a python implementation with the default CVXOPT solver to optimize our QP. The choice of an optimal solver remains an open task. For both the DMP and the QP we use Euler integration with a 0.001s time step. After the sample that corresponds to the final time both the DMP and the QP accelerations are set to zero.
We performed three different experiments (Sections IV-A, IV-B and IV-C) that consisted in adapting the demonstrated motion to different execution conditions, comparing our QP method to DMP. Moreover, we discuss how some important features of the DMP approach are also preserved, namely the rejection of noise and disturbances during execution (Section IV-D). Finally, we discuss how both approaches deal with noise in the demonstration (Section IV-E). Each experiment shows one useful feature of our proposed method. For the sake of clarity we present these results in separated experiments, but in a real application all features can be used simultaneouly. Due to space constraints in the plots we display only the trajectory of one joint of our 6dof arm, but results for the other joints are analogous.

A. Adaptation to different initial/final states and movement duration

Both the DMP and the QP allow adapting the demonstrated primitive to new initial and final states composed of position, velocity and time. As it can be seen in Fig. 2 the DMP and the QP generate similar trajectories. However, the QP distributes the acceleration difference evenly throughout the trajectory. This allows a more accurate replication of the demonstrated movement shape, especially near the end of the trajectory. The non-zero final accelerations may increase the velocity error at the final instant. When a precise final velocity is critical to the task a new constraint can be added to switch off accelerations at a predetermined final fraction of the trajectory.

B. Inclusion of joint limits

Taking into account the physical limits of the joints is of crucial importance in practical operations, as every robot has position, velocities and acceleration limits that cannot be

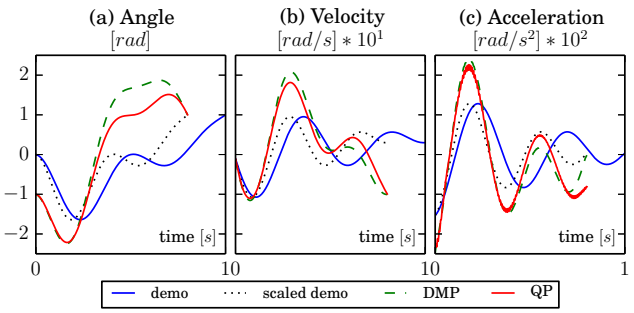

Fig. 2. Trajectory generated by adapting the demonstration to a different initial position and a different final velocity. The final time is set to be at $0.8s$ instead of the $1.0s$ of the demonstration. The movement generated by the global optimization (without additional constraints) imitates the shape of the demonstration better, especially in the final part of the motion.

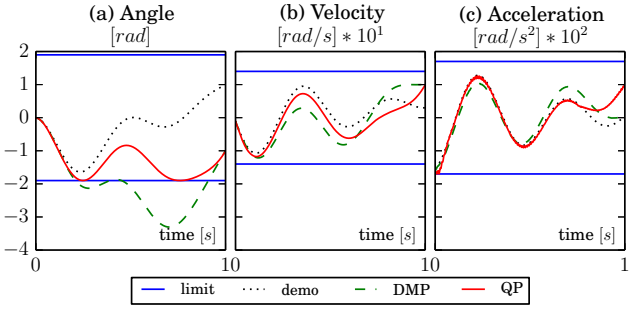

Fig. 3. Trajectory generation with joint limits. Here the joint limits were added as constraints in the QP. The position has a physical limit at $\pm 2.0 rad$. To adapt to a ney end goal ($v_f = 10 rad.s^{-1}$, $p_f = -1.0 rad$) the QP can take the limits and a safety margin into account to generate a good imitation while a DMP generates an infeasible trajectory.

violated. Moreover, in some specific cases there might be the need for setting limits that are more conservative than the maximum, nominal ones; therefore, it is extremely helpful to have the possibility of changing these limits on-the-fly, depending on the task requirements, without the overall shape of the movement being deviated too much from the demonstration. As shown in Fig. 3, with the DMP formulation the adaptation to new goals can generate trajectories that violate joint limits (even if the demonstration was bounded). Conversely, with our approach these limits can be included as constraints of the QP by adding inequalities in Equation 17. The results in Fig. 3 show that our proposed method can generate a trajectory that is feasible and still a good approximation of the original demonstration.

C. Addition of intermediate way-points

The flexibility of the QP formulation permits adding new constraints to fit the specific demands of the task. An interesting possibility is adding intermediate way-points while trying to maintain the shape of the movement. The way-points can be added in the same way as the final goal but at an intermediate time; they can be defined as a specific position and velocity occurring at a desired time instant. Among many, one practical application could be to perform

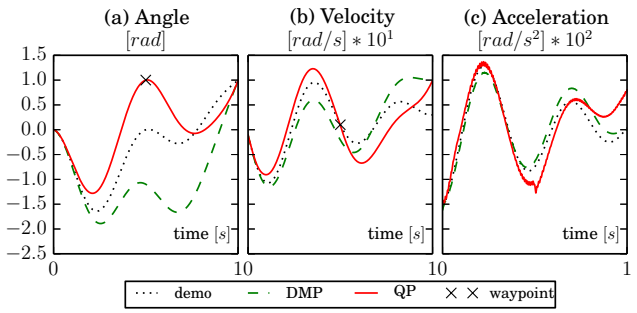

Fig. 4. Intermediate way-points can be added to a trajectory in the same way as the final goal. Here a waypoint is specified with $p_{waypoint} = 1.0 rad$ and $v_{waypoint} = 1.0 rad.s^-1$ at time $t = 0.5s$. The possibility of adding way-points while following the shape of a single primitive does not exist with the previous formulations of DMPs.

obstacle avoidance while keeping the demonstrated shape of the movement. Fig. 4 shows how the QP formulation allows to generate a trajectory with the same initial and final position of the demonstration, but with a different way-point in the middle; notably, the shape of the trajectory highly resembles the demonstrated one. Although a small discontinuity in the acceleration profile seems to occur near to the way-point, this can be mitigated by adding a jerk limiting constraint in the QP problem. Moreover, since the discontinuity happens only after the way-point and the rest of the movement is smooth, if way-points are sufficiently spaced in time the resulting trajectory is not jerky and can be executed with precision.

D. Rejection of noise and disturbances

The acceleration based controller of the DMP (see equation 11) is stable when following a polynomial goal [16]. In our case we are following an arbitrary reference instead of a polynomial: after transients, the system acts as an all-pass filter and does not diverge when given bounded inputs.
We performed an experiment in which we introduced i) noise in the sensory measurements (positions and velocities) and ii) a step disturbance in the acceleration, which can simulate for example the presence of an unexpected load. The plots in Fig. 5 show how the controller is able to reject the noise and recover the desired trajectory when the disturbance vanishes.

E. Effect of noise in the demonstration

The demonstration of a movement usually consists of a position signal, that is then differentiated to obtain the acceleration profile. The noise in the position signal is amplified by differentiation and may result in a very noisy acceleration profile unsuitable for imitation.
DMPs use the demonstrated position and velocity in addition to the acceleration for learning the forcing term. This helps to mitigate the jerky motion caused by noise in the acceleration. Our QP system looses some of the high frequency noise content by representing the demonstrated accelerations as Gaussian Basis functions, however if the position signal is very noisy the stored demonstration will still have very high

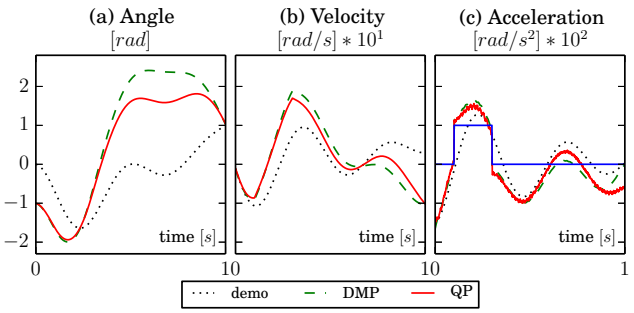

Fig. 5. The QP is disturbed by adding 1% noise in the velocity and position sensor measurements, and by adding an acceleration of 100 rad/s^2 during $t = [0.1, 0.3]$ $[s]$. The system manages to reject the noise and recovers from the disturbance accurately reaching the position and velocity goal.

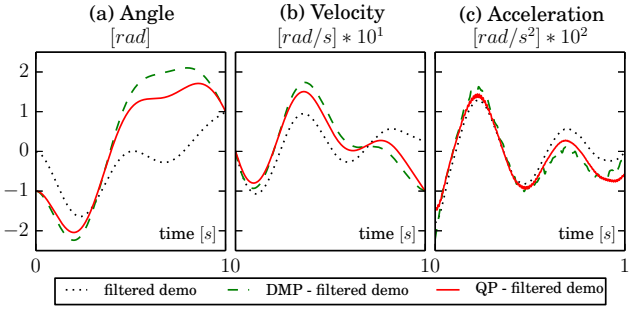

Fig. 6. Differentiation of the position signal with smoothing by a Savitzky-Golay filter results in better imitation for both the QP and the DMP. To the demonstration's position signal we added a noise of amplitude 0.01 $[rad]$ this will be amplified by the differentiation to 10 $[rad/s]$ in the velocity and to 10000 $[rad/s^2]$ in the acceleration. The QP and the DMP used the acceleration smoothed by the Savitzky-Golay filter. Filtering is very desirable for the DMP to avoid jerky motion, and necessary for any useful imitation with the QP.

accelerations, being unsuitable for imitation. Preprocessing the demonstration offline through a smoothing filter generates a smooth acceleration profile suitable for imitation. We used a Savitzky-Golay filter [20] to smooth the position signal, achieving good results in the imitation (Fig. 6).

V. Conclusions

In this paper we explore an alternative way to adapt movement primitives to new execution conditions. Expanding from modern developments on DMPs, we take a novel approach by posing the problem of imitation as a constrained global optimization.

Our experiments show that our method maintains the flexibility of the DMP formulation (i.e. possibility to change initial and final positions, velocities, time) while adding important additional features (i.e. possibility to add limits and way-points during the whole trajectory). Moreover, the generated trajectories always show the traits of the original demonstration, by closely matching its acceleration profile.

Although not explored in this paper, other constraints can be easily added, such as limitation of the maximum jerk in the joints motion.

Solving the QP requires an extra computation step with respect to the DMP formulation. However since the problem is formulated as a convex quadratic program for which fast solvers exist, the solution can be found sufficiently fast to be used even in tasks that require low reaction times.

References

[1] S. Schaal and C. Atkeson, "Learning Control in Robotics," *Robotics Automation Magazine, IEEE*, vol. 17, no. 2, pp. 20–29, June 2010.

[2] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *Proceedings of the Fourteenth Int. Conference on Mach. Learning*, ser. ICML '97. Morgan Kaufmann Publishers Inc., 1997, pp. 12–20.

[3] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, pp. 1371–1394.

[4] B. Argall, S. Chernova, B. Browning, and M. Veloso, "A survey of robot learning from demonstration." *Robotics and Autonomous Systems*, vol. 57, pp. 469–483, 2009.

[5] H. Ben Amor, E. Berger, D. Vogt, and B. Jung, "Kinesthetic bootstrapping: Teaching motor skills to humanoid robots through physical interaction," in *KI 2009: Advances in Artificial Intell.*, ser. Lecture Notes in Computer Science, B. Mertsching, M. Hund, and Z. Aziz, Eds. Springer Berlin Heidelberg, 2009, vol. 5803, pp. 492–499.

[6] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective."

[7] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[8] J. Tani and M. Ito, "Self-organization of behavioral primitives as multiple attractor dynamics: A robot experiment," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 33, no. 4, pp. 481–488, July 2003.

[9] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 37, pp. 286–298, 2007.

[10] S. M. Khansari-Zadeh and A. Billard, "Learning stable non-linear dynamical systems with gaussian mixture models," *IEEE Transaction on Robotics*, vol. 27, pp. 943–957, 2011.

[11] K. Dixon and P. Khosla, "Trajectory representation using sequenced linear dynamical systems," in *ICRA 2004*, vol. 4, April 2004, pp. 3925–3930 Vol.4.

[12] R. Krug and D. Dimitrovz, "Representing movement primitives as implicit dynamical systems learned from multiple demonstrations," in *ICAR, 2013*, Nov 2013, pp. 1–8.

[13] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," *ICRA, 2002*, pp. 1398–1403.

[14] 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, 2013.

[15] J. Kober, K. Mulling, O. Kromer, C. H. Lampert, B. Scholkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *ICRA, 2010*, May 2010, pp. 853–858.

[16] K. Mulling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The Int. J. of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[17] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *NIPS*, 2013, pp. 2616–2624.

[18] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IROS 2004. Proceedings. 2004 IEEE/RSJ Int. Conf. on*, vol. 3, Sept 2004, pp. 2149–2154 vol.3.

[19] S. Degallier and A. Ijspeert, "Modeling discrete and rhythmic movements through motor primitives: a review," *Biological Cybernetics*, vol. 103, pp. 319–338, 2010.

[20] A. Savitzky and M. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Anal. Chem.*, vol. 36, pp. 1627–39, 1964.