Online Motion Generation for Mirroring Human Arm Motion

Roman Weitschat*, Alexander Dietrich, Jörn Vogel*

***Abstract*— Motion planning in robotics is a very large field of research. Many different approaches have been developed to create smooth trajectories for robot movement. For example there are optimization algorithms, which optimize kinematic or dynamic properties of a trajectory. Furthermore, nonlinear programming methods like e.g. optimal control, or polynomial based methods are widely used for trajectory generation. Most of these techniques are used to calculate a trajectory in advance, or they are limited to create point-to-point motions, where the robot needs to stop when switching to the next target point, especially, when interpolating in rotational space. In this paper, we combine a low-pass filter and spherical linear interpolation to realize a velocity-limited online trajectory generator for robot orientations in quaternion space. We use the developed motion generator for mirroring a human arm motion with a robot, recorded by a low frequency visual tracking. Using the proposed method, we can replicate the motion of the operator's arm with very little delay and thereby achieve an easy-to-use interface. Furthermore, as we can strictly limit the velocity of the generated motion, the approach can safely be used in human robot collaboration applications.**

I. Introduction

Fig. 1. The DLR Light-Weight-Robot - a sensitive torque controlled robot for safe human robot collaboration

The current trend in robotics is moving to a very close interaction of humans and robots. Whether at home or in industrial applications, robots will find their way in our everyday life [1]. Human robot interaction will at some point be very similar to the interaction of humans among themselves. We are talking to robots, gesticulate with them, or we show the robot what it has to do by demonstrating the task. The latter, namely learning by demonstration, is a very intuitive and therefore easy-to-use approach to program a robotic task. Two methods of demonstration are usually applied: One, in which the human operator physically moves the robot to the target configurations, the other, in which the motion of the human operator is recorded and then transformed into trajectories for the robot. This trajectory generation is usually performed offline. To make the interaction between humans and robots more intuitive, robots should be able to imitate human motions. New vision-based sensors allow to capture human motion but are able to process the data in a low frequency only. Furthermore, one important goal of generating trajectories for these motions is to limit the speed and acceleration along the path, in order to comply with the dynamic capabilities of the robot and to guarantee safety in human robot collaboration.

In the beginning, research was primarily concerned with generating trajectories by using geometric approaches such as linear interpolation, circular interpolation or polynomial splines [2] [3] [4]. Furthermore, optimization techniques and nonlinear programming methods were developed to ensure a collision-free path from an initial position to a desired final configuration [5], [6], [7], [8]. These optimization methods also allow to define constraints as for example velocity, acceleration and jerk constraints. Furthermore, time-optimal or energy-optimal solutions for a path can be found.

While these optimization techniques are very well suited methods for planning robot paths they do have one disadvantage: The process of finding an optimal path within constraints needs a lot of computation time and therefore can not easily be performed online. However, nowadays it is getting more important to generate reactive trajectories in real time. Therefore, optimization approaches are useful in combination with generalization methods [9] [10]. Similar approaches, which are based on predefined trajectories or trajectories extracted from learning by demonstration, modify the path online caused by sensor events e.g. external forces or vision based force fields. These methods can be found in the reflex motion library [11] or in [12], [13], where the limitation of velocities, accelerations and jerk is treated. However, in another recently presented approach [14], trajectories are generated in consideration of the robot dynamics and its torque limits.

All these aforementioned methods are basically limited to Cartesian positions and cannot be used for generation of Cartesian orientations. A common way to interpolate scalar signals, is to filter the data using a low pass filter or similar methods. However, due to the mathematical properties of SO(3), these classical filtering methods cannot be directly

* Joern Vogel and Roman Weitschat contributed equally to this work. All authors are with Robotics and Mechatronics Center, DLR - German Aerospace Center, Wessling, Germany

applied to orientations. Filtering of orientations can be used for averaging, e.g. in sensor data fusion, but when applied to interpolation, it usually leads to notable deviations because the components of a rotation representation are dependent and cannot be individually filtered.

A very nice approach for interpolating orientations originates from computer animation. Shoemake [15] introduces an approach for spherical linear interpolation (SLERP) which generates the shortest path between the current and the desired orientation. Furthermore, this linear interpolation method allows to define a desired time for the motion and thereby a maximum rotary velocity can easily be realized. However, there is a discontinuity in the rotational velocity, when concatenating multiple segments of SLERP-trajectories.

In this paper we present a real-time trajectory generation method, which can be used for demonstrating tasks or interactive programming in a very simple way. In our approach, we command the robot online via visual tracking of the operator's arm. To achieve this, the low frequency tracking signal (e.g. 30Hz when using a Microsoft Kinect) needs to be up-sampled to the 1kHz control loop of the robotic system at runtime. At the same time the velocity of the generated trajectory needs to be limited in order to ensure safety for the environment and the robotic system itself. Our approach allows to directly move the robot in tandem with the arm of a human operator while guaranteeing a smooth continuous behavior.

This paper is organized as follows. Section II describes the SLERP algorithm introduced by Shoemake and a filter design using quaternions. A proof for satisfying the limitation of the angular velocity is done. Sec. III provides simulation and the results of a comparison of each method and the combination of both. In Sec. IV experiments using a VICON tracking system and grasping a ball from a stand are shown. Section V concludes the paper.

II. Smooth motion generation using SLERP

In this section we introduce the general spherical linear interpolation (SLERP) formalism given a starting orientation and a desired final orientation [15]. Then, we explain a method for filtering the SLERP output for generating smooth motions with quaternions and finally prove the adherence of the maximum velocity limitation to the combined trajectory generation. The overall scheme of our method is depicted in Fig. 2.

A. Slerp

When interpolating between two points in space, one is interested in following the shortest path between these points. In Euclidian space, this shortest path is obviously given by a straight line. However, in non-Euclidian spaces, like SO(3), the shortest path is not that intuitively found. In [15], this problem is solved using quaternion-based linear interpolation. Given the starting orientation as $\mathbf{q}_0$ and the desired orientation $\mathbf{q}_s$ the interpolating path can be computed

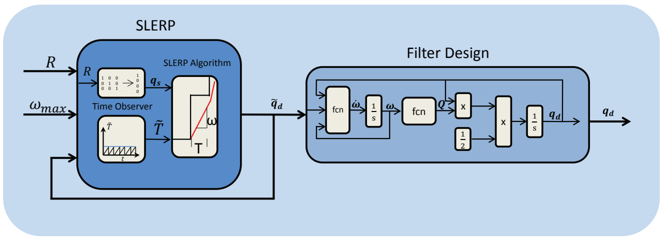

Fig. 2. Approach - Combining two methods for smooth motion generation in quaternion space. A low-frequency sensor signal is used as input command for the robot. In combination with a method known from the dynamic movement primitives paradigm, a suitable orientation generation allows to smoothly and constrained move the robot in Cartesian space.

as

$$\mathrm{Slerp}(\mathbf{q}_0, \mathbf{q}_s, u) = \mathbf{q}_0(\mathbf{q}_0^{-1}\mathbf{q}_s)^u, \tag{1}$$

where $0 \leq u \leq 1$ denotes the interpolation parameter. In quaternion space, the quaternion $\mathbf{q}_s$ and $-\mathbf{q}_s$ describe the same orientation. However, when calculating the spherical linear trajectory towards $\mathbf{q}_s$ there are two solutions, which can be seen as clockwise and counter-clockwise rotations. To actually compute the shortest possible path from a starting orientation to a target orientation, either $\mathbf{q}_s$ or $-\mathbf{q}_s$ has to be chosen according to:

$$\mathbf{q}_s = \begin{cases} \mathbf{q}_s & , \ \mathbf{q}_0 \cdot \mathbf{q}_s \geq 0 \\ -\mathbf{q}_s & , \ \mathbf{q}_0 \cdot \mathbf{q}_s < 0 \end{cases} \tag{2}$$

Equation (1) can be reformulated to:

$$\tilde{\mathbf{q}}_\mathbf{d}(t) = \frac{sin[(1-u(t))\Theta]}{sin(\Theta)}\mathbf{q}_0 + \frac{sin(u(t)\Theta)}{sin(\Theta)}\mathbf{q}_s, \tag{3}$$

where the total amount of rotation is given by

$$\Theta = acos(\mathbf{q}_0 \cdot \mathbf{q}_s). \tag{4}$$

Since SLERP corresponds to linear interpolation (i.e. constant velocity) we can use Θ to calculate an interpolation duration T_m, which limits the rotational velocity to ω_{max}.

$$T_m = \frac{\Theta}{\omega_{max}} \tag{5}$$

However, in order to prevent the robot from moving faster than the source signal, we define the final interpolation time T_q as:

$$T_q = max(T_m, T_s), \tag{6}$$

where T_s is the sampling time of the source signal, which can be computed as

$$T_s = \frac{1}{\mathrm{f}_s}, \tag{7}$$

where f_s is the frequency of the sensor signal. Finally, having the interpolation duration, we can formulate the interpolation parameter u as:

$$u(t) = \begin{cases} \zeta\Delta u & , \ \zeta < T_q \\ 1 & , \ \zeta \geq T_q \end{cases} \tag{8}$$

with $\zeta = t - t_l$ where t denotes the time and t_l is the time when the desired goal orientation changed for the last time and $\Delta u = \frac{1}{T_q}$. Furthermore, for each goal update at time t, the initial orientation is updated as $\mathbf{q}_0 = \tilde{\mathbf{q}}_d(t)$ and the interpolation parameter u is reset to zero.

One main problem when concatinating trajectories generated using the SLERP algorithm is getting discontinuities in the velocities, which lead to very high motor commands. In the following section we describe how to smoothen the SLERP output using a second-order differential equation with quaternions.

B. Dynamic Filter Design

Generating smooth trajectories is very important to achieve safe and stable robot behavior. Differential equations of second order are well known, and it is possible to asymptotically stabilize the system. In order to design a filter for quaterinons, we use an approach introduced in the dynamic movement primitives (DMPs) community by [16]. There, quaternions are used to represent non-singular rotations in $\mathbb{R}^3$. In order to use SLERP as a motion generator for direct and shortest rotational movements, we need to smoothen the output signal. Different approaches exist [17] parabolic blends are used to smoothen the trajectory. But, as we are interested in close human robot interaction the maximum desired velocity must never be exceeded, which is not in the focus of other methods.

1) Differential equation for quaternions: The general formulation introduced by [16] using quaternions for dynamic movement primitives is

$$\begin{aligned}\tau\dot{\boldsymbol{\vartheta}} &= K(\mathbf{g}-\mathbf{x}) - D\boldsymbol{\vartheta} + K\boldsymbol{f}(s)\\ \tau\dot{\mathbf{x}} &= \boldsymbol{\vartheta}\end{aligned} \tag{9}$$

where $\mathbf{x}$ and $\boldsymbol{\vartheta}$ denote the position and the velocity. The stiffness of the system is defined by K, the damping term D leads to a non-oscillating system. A convergence to the goal g is guaranteed and τ is a temporal scaling factor. The force term $\boldsymbol{f}(s)$ is basically used for learning the trajectories.

In general, movement primitives can be used for generating motions in joint or cartesian space. However, in order to apply for rotational motions with quaternions the formalism described in [16] has to be used:

$$\tau\dot{\boldsymbol{\omega}} = K\mathbf{e}_0(\{\eta_d, \mathbf{q}_d\}\{\tilde{\eta}_d, \tilde{\mathbf{q}}_d\}) - D\boldsymbol{\omega}, \tag{10}$$

for the quaternion low-pass-filter design, where $\boldsymbol{\omega}$ denotes the angular velocity, and $\mathbf{e}_0$ the error between the actual quaternion and the goal quaternion. As our main intend is to converge to the goal orientation and not generating predefinded trajectories by gaussian basis, we choose the force $\mathbf{f}(s)$ to be zero.

For the following we denote our quaternion vector as $\mathbf{q} = [\eta\ \tilde{\mathbf{q}}]^T$, where η denotes the scalar element and $\tilde{\mathbf{q}} = [q_i\ q_j\ q_k]^T$ the vector element of the quaternion. The error between the actual orientation and the goal orientation can be written as

$$e_0(\{\eta_1, \mathbf{q}_1\}, \{\eta_2, \mathbf{q}_2\}) = \eta_1\mathbf{q}_2 - \eta_2\mathbf{q}_1 - \mathbf{q}_1^\times\mathbf{q}_2, \tag{11}$$

where $\mathbf{q}_1$ denotes the actual orientation, $\mathbf{q}_2$ the desired goal position and $\mathbf{q}^\times$ denotes a skew-symmetric matrix

$$\mathbf{q}^\times = \begin{bmatrix} 0 & -q_k & q_j \\ q_k & 0 & -q_i \\ -q_j & q_i & 0 \end{bmatrix}. \tag{12}$$

To finally calculate quaternions from the rotary velocity as shown in Fig. 2 the equation of the time derivative of quaternions is

$$\frac{d\mathbf{q}}{dt} = \begin{bmatrix}\dot{\eta}\\ \dot{\tilde{\mathbf{q}}}\end{bmatrix} = \frac{1}{2}\mathbf{Q}(\boldsymbol{\omega})\begin{bmatrix}\eta\\ \tilde{\mathbf{q}}\end{bmatrix}, \tag{13}$$

with

$$\mathbf{Q}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & \boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -\boldsymbol{\omega}^\times \end{bmatrix}. \tag{14}$$

The resulting quaternion is the desired value as input for the torque-controlled LWR III. In the following section we prove that the input velocity of the dynamical system can never be exceeded.

2) Maximum velocity proof: Due to the properties of the SLERP algorithm, the input to the filter has a limited velocity. Now we have to find the right set of parameters to ensure that the output velocity of the second-order differential equation with quaternion input is never higher than the velocity of the input. To achieve this, we use the one-degree-of-freedom formulation for rotation as

$$\tau\dot{\boldsymbol{\omega}} + D\boldsymbol{\omega} + K\mathbf{e}_0(\boldsymbol{\omega}, \boldsymbol{\omega}_d) = 0 \tag{15}$$

where the error between actual orientation and goal orientation can be expressed as the subtraction of the integrated angular goal velocity and actual rotary velocity, which can be written as

$$\mathbf{e}_0 = \int\boldsymbol{\omega}dt - \int\boldsymbol{\omega}_d dt. \tag{16}$$

We obtain the derivative of the first-order differential equation which leads to the second-order differential equation

$$\tau\ddot{\boldsymbol{\omega}} + D\dot{\boldsymbol{\omega}} + K\boldsymbol{\omega} - K\boldsymbol{\omega}_d = 0. \tag{17}$$

Transformation to Laplace domain yields

$$\frac{\Omega(s)}{\Omega_d(s)} = \frac{K}{\tau s^2 + Ds + K}. \tag{18}$$

A non-oscillating dynamic system is defined by non-imaginary poles which can be calculated as

$$p_{1,2} = -\frac{D}{2\tau} \pm \sqrt{\frac{D^2}{4\tau^2} - \frac{K}{\tau}} \tag{19}$$

With a constant stiffness factor K we caluclate the damping term as $D \geq \sqrt{4K\tau}$ for a non-oscillating system. Thus, setting the damping coefficient D accordingly guarantees that the maximum output angular velocity is $\boldsymbol{\omega} \leq \boldsymbol{\omega}_{d_{max}}$, where $\boldsymbol{\omega}_{d_{max}}$ is the maximum absolute angular velocity.

III. Simulation

In this section we simulate the application of the filter algorithm as a method for direct low frequency sensor input. Then, we examine the behavior of using only SLERP as trajectory generation without further smoothing. Finally, the simulation results of the combined algorithm are shown.

A. Simulation results using only filter on the input signal

For the simulation we generated an artificial source signal ($\mathbf{q}_{\text{target}}$) sampled with a frequency of 25 Hz. When feeding this low frequency signal to the filter algorithm the output shows an oscillating behavior, as shown in Fig. 3). This behavior is expected, since the low sampling of the source signal results in infinite high velocities at each sampling point. The second-order differential equation actually smoothes the signal to some extent, but the output is not applicable on a robotic system as the resulting torques are not feasible. Furthermore, the application of this signal to a robotic simulator results in a very unnatural and jerky movement behavior of the robot. This unnatural behavior makes it difficult to control the robot by navigating via tracking sensors. Tuning the parameters to reduce the oscillation would lead to a very high delay. Application of a source signal with even lower frequency ($\leq 10Hz$) results in a stop-motion-like behavior of the robot similar to the stick-slip effect known from friction. Another issue when using only the quaternion filter method is the exceedance of the desired maximum velocity as one can see in Fig. 3 in the right plot. The final velocity of the generated trajectory is much higher than the maximum desired velocity of 2 rad/s. Especially when humans and robots coexist in a shared workspace, limiting the velocity is a safety requirement and renders the filter approach inapplicable.

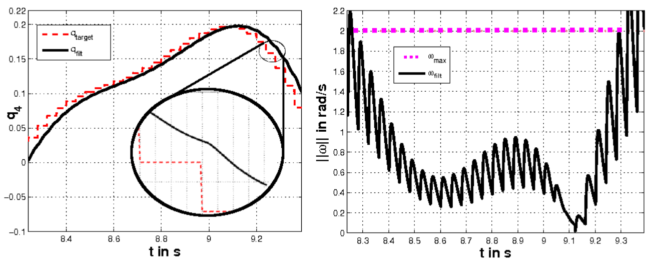

Fig. 3. Direct signal input to the quaternion filter. (left) One component of the resulting quaternion is depicted. The red dashed line depicts the original sensor signal. The black solid line depicts the resulting trajectory. (right) The maximum desired velocity is depicted as dotted magenta line. The resulting velocity of the trajectory generation exceeds the maximum velocity and a strong oscillating behavior is apparent.

B. Simulation results using only SLERP

Velocity limitation in orientation interpolation can be achieved using the SLERP algorithm, which was introduced by Shoemake as a method for interpolating orientations in quaternion space [15]. The main features of this approach are the generation of the shortest motion from one initial orientation to a desired final orientation and the possibility to bound the angular velocity. In Fig. 4 the same artificial target signal, depicted as dashed red line, was used for analyzing the behavior of the SLERP trajectory generation. The resulting trajectory is depicted by the blue solid line. The trajectory of the SLERP-generated quaternions appears to be smooth, but the resulting velocity is not continuous (see Fig. 4 right side), which is inapplicable as an input signal for a robot. Therefore, we use the combined trajectory generation which is analyzed in the following.

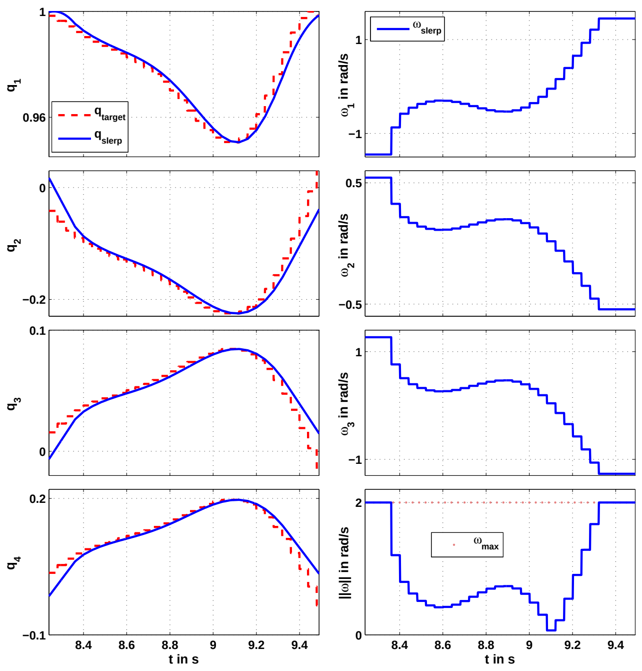

Fig. 4. Left row: Comparison of artificially generated input and the generated trajectory using SLERP only. Right row: Angular velocities generated from the SLERP algorithm. The right bottom plot depicts the absolute angular velocity (blue) and the velocity limit of 2 rad/s which is never exceeded.

C. Filtered SLERP

As seen, each single method is not sufficient when applied on its own. However, a combination of both methods leads to the desired result. In Fig. 5, the target trajectory, as used in the plots before, is depicted by a red dashed line. The resulting trajectory of the filtered SLERP algorithm is depicted as black solid line. As one can see the velocity in the right row never exceeds the maximum desired velocity and the resulting trajectory is smooth and continuous. Application of this signal on a real robotic system is presented in the following chapter.

IV. Experiments

For validating the filtered SLERP combination on a real robotic system, we used the DLR light-weight-robot LWR III equipped with the DLR-HIT-hand II. The goal of the experiment was to mirror the motion of the human online (see Fig. 9). In particular the human dynamically grasped a ball from a stand, without slowing down and the robot was doing exactly the same task in parallel. In order to trigger the grasp command we used a surface electromyography (sEMG) sensor placed on the forearm of the operator. From the EMG activity opening and closing of the operators hand could be detected and commanded to the robot's hand accordingly. We used the vision-based tracking system VICON as a tracking device for the human motion. The tracking marker was placed at the backhand of the human operator. A schematic

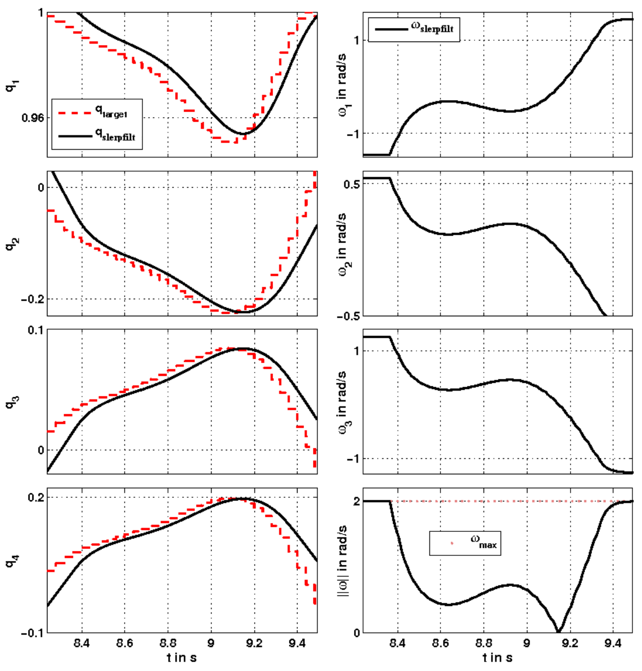

Fig. 5. Left row: Comparison of artificially generated input data of low frequency with the filtered SLERP-generated trajectory. Right row: Angular velocities generated from the filtered SLERP algorithm. The right bottom plot depicts the absolute angular velocity (blue) and the velocity limit of 2 rad/s which is never exceeded.

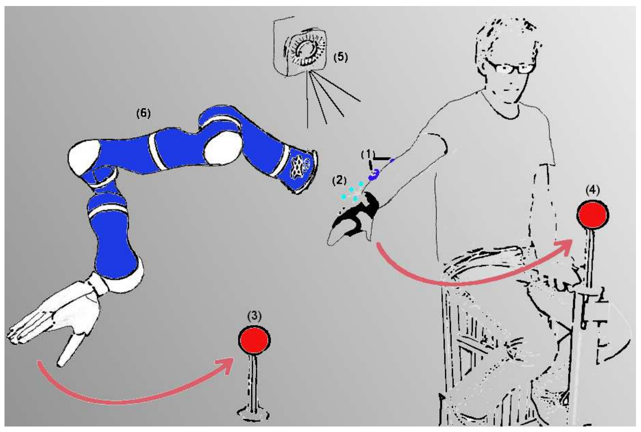

Fig. 6. Schematic setup of the experiment - The human participant is equipped with EMG sensors (1) for triggering the grasp command and tracking markers (2) for tracking the position and orientation of the human arm with a VICON tracking system (5). The task in this experiment is to dynamically grasp the ball (4) without slowing down while the LWR III robot (6) is exactly mirroring the task (3).

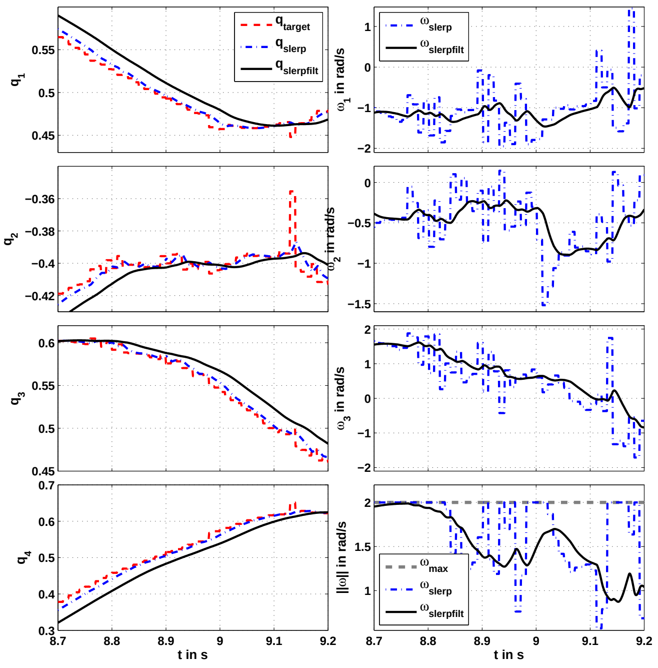

Fig. 7. Left row: Comparison of 100 Hz sensor data (red), as received from the VICON system, the SLERP-generated trajectory (blue) and the final interpolator output (black). Right row: Comparison of angular velocities generated from the SLERP algorithm (blue) and the results of the filtered SLERP output (black). The right bottom plot depicts the absolute angular velocity (SLERP in blue, filtered SLERP in black) and the velocity limit of 2 rad/s (grey), which is never exceeded.

overview is depicted in Fig. 6. The frequency of the position and orientation data transfer was set to 100 Hz but was modifiable to a lower frequency. At this point we didn't want to set the value of the input signal frequency for Eq. (7) manually. Therefore, we implemented a signal frequency observer, which detects changes in the input signal and calculates the sampling time T_s over a window of 4 samples. In Fig. 7 (left row) the target values of each quaternion-component is depicted as red, dashed line. In contrast to the artificially generated signal, the real sensor data is more noisy and contains outliers. As one can see the SLERP algorithm compensates for the low quality of the sensor data effectively. Also the maximum velocity is never exceeded, as depicted in Fig. 7 on the right. In combination with the quaternion filter algorithm we achieve the final trajectory as depicted by the black, solid line in Fig. 7. In a second experiment we modified the data transfer to a frequency of 10 Hz as shown in Fig. 8. In order to also move the robot translationally, we implemented a similar method for the online generation of position trajectories. Having the same algorithm for position and orientation interpolation allows e.g. for synchronization, i.e. the rotation and positions reach the desired values at the same time.

V. Conclusion

In this paper we presented a new approach for online smooth trajectory generation. We used this approach to have a robot mirror the human arm motion, which was recorded online via a low-frequency visual tracking. The robot follows the human motion with limited velocities and smooth acceleration. The SLERP algorithm introduced in [15] was modified in order to continuously interpolate incoming data towards a changing target. In combination with an approach from the dynamic movement primitives, we are able to generate orientation trajectories online in quaternion space. We proved the adherence of the limited angular velocity

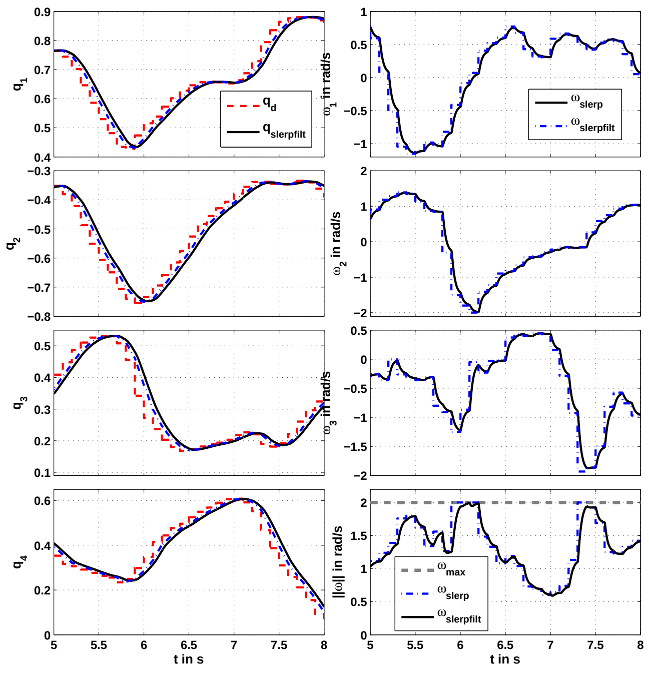

Fig. 8. Left row: Comparison of 10 Hz sensor data (red), as received from the VICON system, the SLERP-generated trajectory (blue) and the final interpolator output (black). Right row: Comparison of angular velocities generated from the SLERP algorithm and the results of the filtered SLERP output. The right bottom plot depicts the absolute angular velocity (SLERP in blue, filtered SLERP in black) and the velocity limit of 2 rad/s (grey), which is never exceeded.

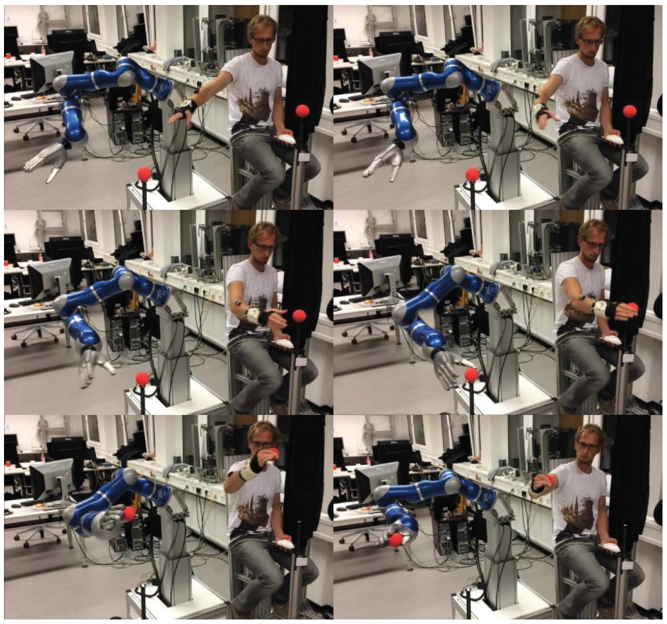

Fig. 9. Grasping sequence where the human picks a ball from a stand and the robot mirrors the motion and the grasping.

and showed the behavior in simulation. Additionally, in our experiments we dynamically grasped a ball from a stand while the light weight robot LWR III exactly mirrored the operator's movement. In the future we will look for a way to limit the accelerations and extend this algorithm to a via-point interpolator for Cartesian movements with the robot.

REFERENCES

[1] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, "The dlr lightweight robot - lightweight design and soft robotics control concepts for robots in human environments," Industrial Robot Journal, vol. 34, no. 5, pp. 376–385, 2007.

[2] L. van Aken and H. van Brussel, "On-line robot trajectory control in joint coordinates by means of imposed acceleration profiles," Robotica, pp. 185–195, July 1988.

[3] B. Cao, G. I. Dodds, and G. W. Irwin, "Time-optimal and smooth constrained path planning for robot manipulators," International Conference on Robotics and Automation (ICRA), pp. 1853–1858, May 1994.

[4] T. Kröger, "On-line trajectory generation: Straight-line trajectories," IEEE Trans. on Robotics, vol. 27, no. 5, pp. 1010–1016, October 2011.

[5] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, S. Thrun, and L. Kavraki, Principles of Robot Motion: Theory, Algroithms, and Implementation. Cambridge: MIT Press, 2005.

[6] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," Robotics and Automation, IEEE Transactions on, vol. 12, no. 4, pp. 566–580, 1996.

[7] S. M. LaValle, J. H. Yakey, and L. E. Kavraki, "A probabilistic roadmap approach for systems with closed kinematic chains," 1999.

[8] S. M. LaValle and J. J. K. Jr., "Randomized kinodynamic planning," in IEEE Int. Conf. on Robotics and Automation, 1999, pp. 473–479.

[9] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," International Conference on Robotics and Automation (ICRA), pp. 566–580, May 2011.

[10] R. Weitschat, S. Haddadin, F. Huber, and A. Albu-Schäffer, "Dynamic optimality in real-time: A learning framework for near-optimal robot motions," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5636–5643, 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6697173&tag=1

[11] T. Kröger and M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," IEEE Transactions on Robotics, vol. 26, no. 1, pp. 94–111, February 2010.

[12] J. A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2002, pp. 958–963.

[13] F. Lange and M. Suppa, "Predictive path-accurate scaling of a sensor-based defined trajectory," International Conference on Robotics and Automation (ICRA), 2014.

[14] R. Katzschmann, T. Kröger, T. Asfour, and O. Khatib, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," IEEE Transactions on Robotics, vol. 26, no. 1, pp. 94–111, February 2010.

[15] K. Shoemake, "Animating rotation with quaternion curves," SIGGRAPH '85 Proceedings of the 12th annual conference on Computer graphics and interactive techniques, vol. 19, no. 3, pp. 245–254, 1985. [Online]. Available: http://www.engr.colostate.edu/ECE555/reading/article_8.pdf

[16] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaption based on prevoius sensor experiences," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 365–371, 2011. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6095059

[17] N. Dantam and M. Stilman, "Spherical parabolic blends for robot workspace trajectories," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014. [Online]. Available: http://www.neil.dantam.name/papers/dantam2014spherical.pdf