Online deformation of optimal trajectories for constrained nonprehensile manipulation

Alexander Pekarovskiy[1,2], Thomas Nierhoff[3], Jochen Schenek[1], Yoshihiko Nakamura[4], Sandra Hirche[3] and Martin Buss[1,2]

Abstract— **This paper discusses an online dynamic motion generation scheme for nonprehensile object manipulation by using a set of predefined motions and a trajectory deformation algorithm capable of incorporating positional and velocity boundary constraints. By creating optimal trajectories offline and deforming them online, computational complexity during execution is reduced considerably. As tight convex hulls of the deformed trajectories can be found, possible obstacles or workspace boundaries can be circumnavigated precisely without collision. The approach is verified through experiments on an inclined planar air-table for volleyball scenario using two 3-DoF robots.**

I. Introduction

When using a robotic system to manipulate objects, the object is usually grasped firmly to predict and control its state precisely during the entire task execution. Yet, this approach has also drawbacks as the pose of the object is limited by the workspace limitations of the robot. Another option is to manipulate an object without grasping, which is called nonprehensile manipulation [1]. The advantage is that the possible workspace can be greatly extended. Typical examples include batting [2] and juggling [3]. All these tasks are either one-off or periodic motions and characterized by presence of the flight phase and impact point, where an instantaneous state transition occurs. Due to missing form or force closure grasp and the short time period where the manipulator is in contact with the object, its motion needs to be both well planned and precisely executed. Analytical solution for motion planning based on predicted object trajectory works for some applications, however they are not always available. For instance, Senoo *et al.* [4] analyzed the derivation of polynomial solutions for hitting motion with refinement of the hitting point based on the high speed vision feedback.

For creating a feasible trajectory one can distinguish between two main approaches:

On the one hand, there are numerical optimal control methods which recalculate the entire trajectory for every new object motion [5], [6]. They can readily incorporate constraints required for a successful hitting motion, but they are limited as they rely on a good initial guess in order to converge to a feasible solution and are too slow to find an optimal trajectory online. It makes sense to replace optimal control with optimization in some cases due to requirements on algorithm execution time and rate of convergence. Gradient optimization methods were implemented for obstacle avoidance based on trajectory sampling in [7], [8] and for robust nonprehensile balancing based on polynomial trajectory representation in [9]. Sequential quadratic programming was used instead of gradient descent for getting better convergence in [10].

On the other hand, there are learning methods based on movement imitation. Here the general idea is to learn some prototypic motion offline based on a human demonstration and adapt the learnt motion online to match the task-specific constraints. Various methods exist in literature, for example, Dynamic Movement Primitives as described in [11]. An extension overcoming the prior problem of accounting only for rest-to-rest motions is presented in [12]. By allowing position and velocities to be specified for a predefined point, the approach can be used for general hitting motions. Another option is to use Gaussian Mixture Regression [13] to learn motions through imitation. Still the approaches are only suitable for free space motions as they cannot guarantee collision avoidance. In addition, the fixed number of internal states limits the extent of possible deformations.

Another method from Yamane and Nakamura [14] produces physically plausible motions from possibly infeasible ones, where a trial-and-error procedure is used to find proper unilateral force constraints of the contacts with the environment. Here calculations are done independently for every single time step and the resulting motion behavior is highly dependent on the choice of the gains.

An approach capable of overcoming drawbacks of the discussed methods is Laplacian Trajectory Editing [15] minimizing the acceleration deviation to a given reference path through a least squares approach. Our approach modifies the entire trajectory with a feasible contact point online - not just a single time step. As the method does not rely on any kernel/internal state, its resolution is solely limited by the number of trajectory sampling points. In addition, extensions for collision avoidance are presented in [16].

The contribution of the paper is thus the combination of a numerical optimal control method and Laplacian Trajectory Editing (LTE) for online deformation of a previously calculated optimal trajectories. Results of numerical optimal control methods are in general locally optimal, but further

[1]Chair of Automatic Control Engineering, Technische Universität München, D-80333 Munich, Germany
{a.pekarovskiy,j.schenek, mb} at tum.de

[2]TUM Institute for Advanced Study, Technische Universität München, Lichtenbergstr. 2a, 85748 Garching, Germany

[3]Chair of Information-oriented Control, Technische Universität München, D-80333 Munich, Germany {tn, hirche} at tum.de

[4]Department of Mechano-Informatics, University of Tokyo, Tokyo 113-8656, Japan. {nakamura} at ynl.t.u-tokyo.ac.jp

in the paper we will call them optimal for brevity. If the amount of deformation is small, the optimality properties of undeformed trajectory also hold for the deformed trajectory up to a certain extent. It is shown that for a specific type of deformation tight boundaries of the deformed trajectory can be derived, making the approach an ideal choice for motion adaptation in constrained environments. To achieve a tradeoff between trajectory similarity and a large deformation, a hierarchical collision avoidance approach iteratively increases the amount of deformation and varies trajectory segmentation size until a collision-free trajectory is obtained.

The remainder of the paper is as follows: Sec. II describes the trajectory deformation process and the collision avoidance scheme. In Sec. III the boundary constraints for trajectory deformation are derived through a precise physical model of the given task. Experiments for a 2D volleyball scenario on an inclined air-table are conducted in Sec. IV. Finally, Sec. V and Sec. VI discuss the presented approach and suggest ideas for further expansion.

II. Trajectory Deformation

For each new motion, initial and end boundary points of the trajectory are assigned based on the task goal. Once both new target position and velocity are known, a feasible trajectory that moves the robot from a starting position to the target position and back again while avoiding workspace constraints has to be found.

Whereas in previous papers [5], [6] optimal trajectories have been produced with direct collocation method, its high computational complexity and sensitivity towards a good initial guess are major drawbacks. In contrast, this paper assumes that optimal trajectories have been calculated beforehand and will be deformed online to meet additional requirements. Although giving up the optimality of the direct collocation method, it results in close-to-optimal trajectories if the amount of deformation is small. On the plus side, computational complexity is reduced considerably. For each new motion, initial and end boundary points of the trajectory are assigned based on the task goal. Once both new target position and velocity are known, a feasible trajectory has to be found that moves the robot from a rest position A to the hit position B and back again while avoiding workspace constraints.

A. Deforming Trajectories Through Laplacian Trajectory Editing

A generic method for deforming trajectories through a least-squares approach is Laplacian Trajectory Editing. It assumes that the trajectory consists of n equitemporally spaced sampling points $\mathbf{P} = [\mathbf{p}(t_1), \mathbf{p}(t_2), \ldots, \mathbf{p}(t_n)]^T \in \mathbb{R}^{n\times m}$, $t_{i+1} - t_i = \Delta t \ \ \forall i \in \{1, \ldots, n-1\}$. For simplicity, original optimal trajectory is denoted as $\mathbf{P}^o = [\mathbf{p}_1^o, \mathbf{p}_2^o, \ldots, \mathbf{p}_n^o]^T$. One can then calculate the acceleration along the original trajectory for the i-th sampling point through the finite difference

$$\delta_i = \frac{\mathbf{p}_{i+1}^o - 2\mathbf{p}_i^o + \mathbf{p}_{i-1}^o}{\Delta t^2}. \tag{1}$$

The key idea of LTE during trajectory deformation is to calculate local trajectory properties, resulting in a linear system of equations. When adding boundary constraints, the resulting overdetermined system of equations can be solved using least squares. By introducing weighting factors for each constraint, they can be prioritized to fit the user requirements. Note that the acceleration in this paper is just a special case of the *Laplacian coordinates* in [15] for equitemporally spaced sampling points.

Only positional constraints of the form $\mathbf{p}_j = \mathbf{c}_j$ are considered, pinning a specific sampling point $\mathbf{p}_j$ to a desired position $\mathbf{c}_j$. With the weighting factor w it can be rewritten as

$$w\mathbf{p}_j = w\mathbf{c}_j, \ \ j \in \{1, 2, \ldots, n\}. \tag{2}$$

Writing everything in matrix form, one obtains

$$\begin{bmatrix}\mathbf{L}\\ \bar{\mathbf{P}}\end{bmatrix}\mathbf{P} = \begin{bmatrix}\boldsymbol{\Delta}\\ \bar{\mathbf{C}}\end{bmatrix}, \tag{3}$$

with

$$\mathbf{L} = \frac{1}{\Delta t^2}\begin{bmatrix}1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1\end{bmatrix}, \ \ \boldsymbol{\Delta} = \begin{bmatrix}\delta_2\\ \delta_3\\ \vdots\\ \delta_{n-2}\\ \delta_{n-1}\end{bmatrix}, \tag{4}$$

and the matrices $\bar{\mathbf{P}}$ and $\bar{\mathbf{C}}$ accounting for the weighted positional constraints as described in (2). The equation system can be solved for the deformed trajectory $\mathbf{P}_s$ as

$$\mathbf{P}_s = \begin{bmatrix}\mathbf{L}\\ \bar{\mathbf{P}}\end{bmatrix}^+ \begin{bmatrix}\boldsymbol{\Delta}\\ \bar{\mathbf{C}}\end{bmatrix}, \tag{5}$$

using least squares. For weighting factors $w \gg \frac{1}{\Delta t^2}$ the error $\epsilon = \mathbf{p}_i - \mathbf{c}_i$ is negligible for practical purposes and can be approximated with $\epsilon \approx 0$.

Straightforward as it is, the approach is also relatively slow as the least squares solution requires a matrix inversion. For a trajectory with n sampling points and p positional constraints, the dimension of the matrix $\left[\mathbf{L}^T\bar{\mathbf{P}}^T\right]^T$ is $n + p - 2 \times n$, making the approach not feasible for real-time deformation of large trajectories.

It is however possible to decompose the solution in (5) into two parts as

$$\mathbf{P}_s = \underbrace{\begin{bmatrix}\mathbf{L}\\ \bar{\mathbf{P}}\end{bmatrix}^+ \begin{bmatrix}\boldsymbol{\Delta}\\ \bar{\mathbf{C}}_1\end{bmatrix}}_{\{1\}} + \underbrace{\begin{bmatrix}\mathbf{L}\\ \bar{\mathbf{P}}\end{bmatrix}^+ \begin{bmatrix}\mathbf{0}\\ \bar{\mathbf{C}}_2\end{bmatrix}}_{\{2\}}, \tag{6}$$

$$\text{s.t. } \bar{\mathbf{C}}_1 + \bar{\mathbf{C}}_2 = \bar{\mathbf{C}}. \tag{7}$$

Through an adequate choice of the positional constraints in $\bar{\mathbf{C}}_1$, the term $\{1\}$ results in a sole translation of the original trajectory $\mathbf{P}^o$ which can be obtained without the need to calculate the least squares solution. In the continuous domain the equivalent solution for $\{2\}$ is a cubic spline interpolation of the form

$$\mathbf{P}_j = \sum_{k=0}^{3} \mathbf{a}_{j,k} t^k, \quad t \in [0, 1], \tag{8}$$

for the j-th trajectory interval. A proof is given in the Appendix. This way, the equation system to be solved does not consist of $n + p - 2$ but only $4(p - 1)$ variables which is advantageous if $n \gg p$.

B. *Collision Detection Using Bézier Splines*

There are numerous ways to represent a cubic spline curve, for example Bézier splines. In this case one can rewrite (8) for the j-th trajectory interval as

$$\mathbf{P}_j = \sum_{k=0}^{3} \mathbf{c}_{j,k} \mathbf{B}_k^n(t), \tag{9}$$

with the weighting factors $\mathbf{c}_{j,k}$ and $\mathbf{B}_k^n(t)$ as Bernstein polynomials

$$\mathbf{B}_k^n(t) = \binom{n}{k} (1-t)^{n-k}\, t^k, \quad t \in [0,1]. \tag{10}$$

Due to the non-negativity of each Bernstein polynomial for $t \in [0,1]$, every point $\mathbf{p} \in \mathbf{P}_j$ is a convex combination of the control points $\mathbf{c}_{j,k}$. This means that the trajectory segment $\mathbf{P}_j$ always stays within the convex hull of its control points.

It is thus possible to find convex hulls $conv(\{1\})$ and $conv(\{2\})$ both for $\{1\}$ and for $\{2\}$. The convex hull of the deformed trajectory $\{1\} + \{2\}$ is then bounded from above by

$$\begin{aligned} &conv(\{1\} + \{2\}) \leq \\ &conv(\{1\} \oplus \{2\}) = conv(\{1\}) \oplus conv(\{2\}), \end{aligned} \tag{11}$$

where the operator $\oplus$ denotes the Minkowski sum of two convex hulls. It is also possible to calculate the convex hull based on $conv(\{1\} + \{2\})$ without using the Minkowski sum. Although generally resulting in tighter bounds, it is not applicable for real-time deformations of large trajectories due to its $\mathcal{O}(n \log(n))$ complexity. Yet the Minkowski sum enables one to calculate $conv(\{1\})$ and $conv(\{2\})$ independently. Knowing that the term $\{1\}$ is a sole translation of the original trajectory, the convex hull of the original trajectory is calculated offline and just has to be shifted accordingly to obtain $conv(\{1\})$. The convex hull $conv(\{2\})$ is given by the four control points of the associated Bézier curve.

C. *Hierarchical Collision Avoidance Through Iterative Subdivision*

Having calculated the convex hull of the deformed trajectory, collisions are detected. Unfortunately, the Minkowski sum provides rather conservative bounds of the deformed trajectory, detecting collisions even if the underlying trajectory is non-colliding. To overcome the problem, the original trajectory is iteratively split into smaller trajectory intervals for which a new convex hull is calculated. Through Jensen's inequality it can be proven that if a trajectory interval $\mathbf{P}_j$ is split up into two smaller intervals $\mathbf{P}_{j,1}$ and $\mathbf{P}_{j,2}$, it is for the resulting convex hull

$$conv(\mathbf{P}_j) \geq conv(\mathbf{P}_{j,1}) + conv(\mathbf{P}_{j,2}), \tag{12}$$

thus subdividing the trajectory generally results in tighter convex hulls, see Fig. 1.

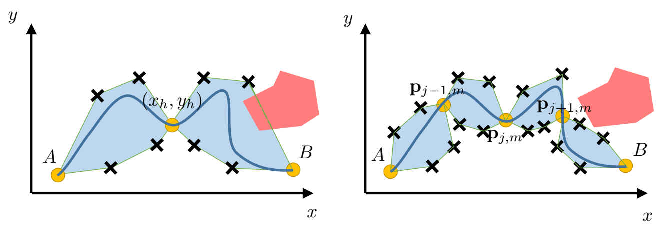

Fig. 1: Two consecutive steps for safe check. Minkowski sum of the deformed non-periodic trajectory in the workspace of the robot. Red polygon represents the static obstacle. If the intersection is found the more refined segmentation is applied.

In case of a collision the boundary sampling points of m-th iteration $\mathbf{p}_{j-1,m}, \mathbf{p}_{j,m} \in \mathbf{P}_j$ of every trajectory interval are shifted depending on the maximum intersection depth $\mathbf{d}_{j,m}$ of the convex hulls as

$$\begin{aligned} \mathbf{p}'_{j-1,m} &= \mathbf{p}_{j-1,m} + \max(\mathbf{d}_{j,m}, \mathbf{d}_{j-1,m}), \\ \mathbf{p}'_{j,m} &= \mathbf{p}_{j,m} + \max(\mathbf{d}_{j,m}, \mathbf{d}_{j+1,m}), \end{aligned} \tag{13}$$

resulting in the new sampling points position $\mathbf{p}'_{j-1,m}, \mathbf{p}_{j,m}$, see Fig. 2. The shifting process is then repeated until a collision-free trajectory is obtained.

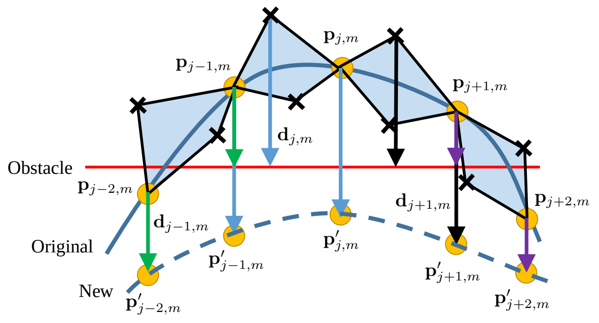

Fig. 2: Shifting the segmentation points (orange circles) outside of the static obstacle (red line). The new deformed trajectory is depicted with the dashed line.

When subdividing the trajectory, two contradicting objectives have to be fulfilled: On the one hand, we want the granularity as fine as possible (many trajectory intervals) in order to let the resulting convex hull be as tight as possible. On the other hand, a coarse granularity reduces computational complexity and leads to smoother deformations. A viable tradeoff is found by iteratively subdividing the trajectory and performing a number of shifting operations after each subdivision until a collision-free trajectory is obtained.

III. Impact Modeling

In this paper, we present a scenario that shows full dynamic capability of the introduced method by tackling the problem of playing planar volleyball against a human with a planar 3DOF robot, see Fig. 3. Note that only the planar case is considered, although the entire approach can be readily extended to the three-dimensional case. For the robot, the task consists of hitting an incoming volleyball such that the ball flies over the net (in orange) and lands in the opponent's field (in green). When modeling the impact of the ball one has to find the full state vector of the end effector for the desired ball motion, described by the hit position (x_h, y_h) along the table, the fixed end effector angle Φ_h and its translational velocity (v_{xh}, v_{yh}).

Under the influence of gravity the ball moves along a parabolic trajectory described by

$$\begin{aligned} x_h &= x_0 + v_{x0} t_h, \\ y_h &= y_0 + v_{y0} t_h - \frac{1}{2} g_m t_h^2, \end{aligned} \tag{14}$$

where $g_m = g \sin\alpha$ is a modified gravity on the surface of the air table tilted at an angle α. By representing the workspace constraints of the 3R robot by a circle, the intersection point (x_h, y_h) for hitting the ball at time t_h is found by solving (14) subject to the constraint

$$(x_h - x_c)^2 + (y_h - y_c)^2 - r_c^2 = 0, \tag{15}$$

with workspace-dependent variables x_c, y_c, r_c. The velocity of the ball immediately before the hit is denoted by (v_x, v_y) and the rotational speed by ω.

To find the remaining unknowns, one has to specify the velocities (v'_x, v'_y) and ω' immediately after the hit. In the absence of disturbances, the ball moves similar to (14) on a parabolic trajectory after the hit. By defining two waypoints (x_1, y_1) and (x_2, y_2) along the parabola the ball has to pass, one can solve the resulting system of equations

$$\begin{aligned} x_1 &= x_h + v'_x t_1, \\ y_1 &= y_h + v'_y t_1 - \frac{1}{2} g_m t_1^2, \\ x_2 &= x_h + v'_x t_2, \\ y_2 &= y_h + v'_y t_2 - \frac{1}{2} g_m t_2^2, \end{aligned} \tag{16}$$

for the four unknowns t_1, t_2, v'_x, v'_y. The variables t_1 and t_2 denote the time when passing the two waypoints.

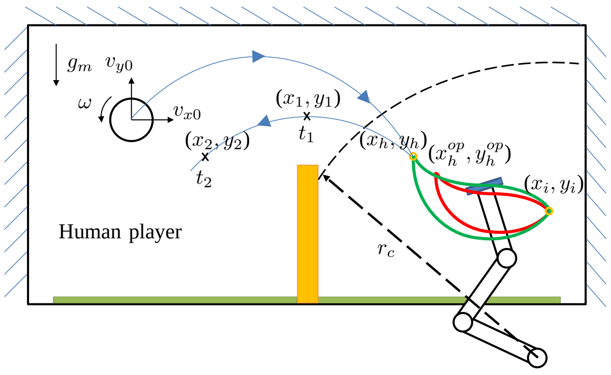

Fig. 3: Scenario overview.

Knowing the state of the ball immediately before and after the hit, one can derive the required end-effector angle and velocity during the hit. To do so the ball velocities are first decomposed into relative normal components $\mathbf{v}_\perp, \mathbf{v}'_\perp$ and relative tangential components $\mathbf{v}_\parallel, \mathbf{v}'_\parallel$ before and after the hit as

$$\begin{aligned} \mathbf{v}_\perp &= \mathbf{P}_\perp \left(\begin{bmatrix} v_x \\ v_y \end{bmatrix} - \begin{bmatrix} v_{xh} \\ v_{yh} \end{bmatrix} \right), \quad \mathbf{v}_\parallel = \mathbf{P}_\parallel \left(\begin{bmatrix} v_x \\ v_y \end{bmatrix} - \begin{bmatrix} v_{xh} \\ v_{yh} \end{bmatrix} \right), \\ \mathbf{v}'_\perp &= \mathbf{P}_\perp \left(\begin{bmatrix} v'_x \\ v'_y \end{bmatrix} - \begin{bmatrix} v_{xh} \\ v_{yh} \end{bmatrix} \right), \quad \mathbf{v}'_\parallel = \mathbf{P}_\parallel \left(\begin{bmatrix} v'_x \\ v'_y \end{bmatrix} - \begin{bmatrix} v_{xh} \\ v_{yh} \end{bmatrix} \right), \end{aligned} \tag{17}$$

with the projection matrices $\mathbf{P}_\perp, \mathbf{P}_\parallel$ depending on the end effector angle Φ as

$$\begin{aligned} \mathbf{P}_\parallel &= [\cos(\Phi_h)\ \sin(\Phi_h)]^T [\cos(\Phi_h)\ \sin(\Phi_h)], \\ \mathbf{P}_\perp &= \mathbf{I} - \mathbf{P}_\parallel. \end{aligned} \tag{18}$$

Fig. 4 gives a detailed illustration of the impact model.

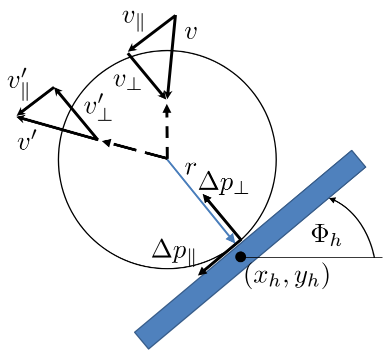

Fig. 4: Impact model between the flat end effector and the ball.

The hit between ball and end effector is modeled as a partially elastic collision with restitution coefficient ϵ along the normal direction, resulting in the change of normal impulse $\Delta\mathbf{p}_\perp$. In tangential direction it is assumed that the change of impulse $\Delta\mathbf{p}_\parallel$ during the hit just reduces the relative tangential velocity $\mathbf{v}_{\parallel r}$ between ball and end effector to zero. The tangential impulse also leads to an angular momentum $\mathbf{L}_\parallel$. Mathematically it can be described by

$$\begin{aligned} \Delta\mathbf{p}_\perp &= -(1+\epsilon)\mathbf{p}_\perp, \\ \Delta\mathbf{p}_\parallel &= -m\mathbf{v}_{\parallel r} = -m(\mathbf{v}_\parallel + \boldsymbol{\omega} \times \mathbf{r}), \\ \Delta\mathbf{L}_\parallel &= \mathbf{r} \times \Delta\mathbf{p}_\parallel, \end{aligned} \tag{19}$$

with m as the mass of the ball, $\mathbf{r}$ as the vector from the center of the ball to the contact point, $\boldsymbol{\omega}$ as the vectorized version of ω and the operator $\times$ as the pseudo cross product in 2D. The resulting velocities before and after the hit can then be related to each other as

$$\begin{aligned} \mathbf{v}'_\perp &= \frac{1}{m}(\mathbf{p}_\perp + \Delta\mathbf{p}_\perp) = -\epsilon\mathbf{v}_\perp, \\ \mathbf{v}'_\parallel &= \frac{1}{m}(\mathbf{p}_\parallel + \Delta\mathbf{p}_\parallel) = -\boldsymbol{\omega} \times \mathbf{r}, \\ \omega' &= \omega + \frac{1}{\Theta}\mathbf{r} \times \Delta\mathbf{p}_\parallel, \end{aligned} \tag{20}$$

with Θ as the moment of inertia and $\mathbf{p}_\perp = m\mathbf{v}_\perp$, $\mathbf{p}_\parallel = m\mathbf{v}_\parallel$ as the normal/tangential impulse of the ball before the hit. Due to the nonlinearity of the projection matrices $\mathbf{P}_\parallel$ and $\mathbf{P}_\perp$, the resulting system of equations (20) has to be solved numerically for the variables v_{xh}, v_{yh}, Φ_h.

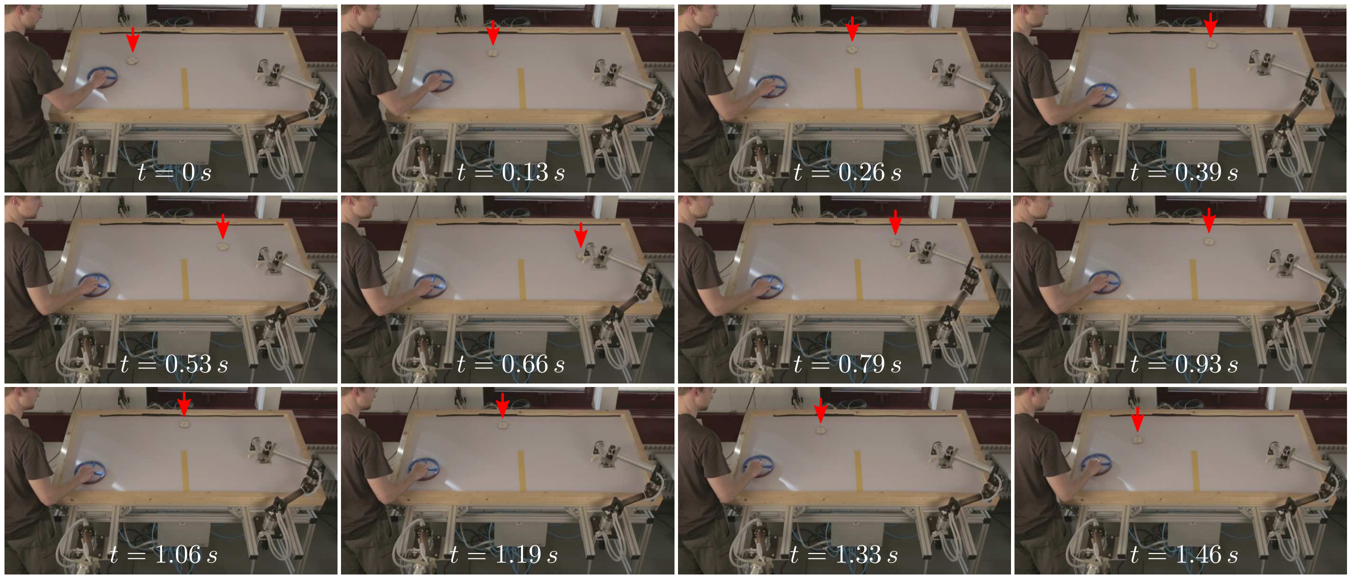

Fig. 5: Frames of the experiment with the human playing against the robot.

IV. Experimental Verification

Experimental verification of the proposed approach is performed on a tilted air-table, see Fig. 5. The ball (puck) is tracked using a Qualisys motion capture system, operating at a framerate of 250Hz. The robot manipulator's update rate is 1kHz. To obtain an optimal trajectory that will be defined afterwards, the numerical optimal control direct collocation method DirCol [5] is used. The workspace of the robot is bounded on each side by a wooden board, constituting obstacles the robot manipulator must avoid. Due to tight real-time constraints only one subdivision step is performed in case the deformed trajectory collides, resulting in trajectory intervals of 50 ms. Fig. 5 shows the frames for a typical batting movement of the robot manipulator. The ball is thrown by a human opponent from the left side and batted back by the robot manipulator while avoiding the net (yellow line) in the middle of the table.

A large amount of this paper is dedicated to deriving an alternative representation for LTE using splines. The main reason is the reduced computational complexity. Fig. 6 evaluates the improvement through simulations. The computation time for a single trajectory deformation step for a varied number of sampling points is shown. All calculations are performed using an Intel Core2Duo T7500 CPU with 4GM RAM with Matlab R2010a running under Windows7. For small trajectories ($n < 500$) the spline representation is slower than the straightforward matrix inversion according to (5) due to the computational overhead of the built-in Matlab function for calculating splines (csape). For large trajectories the spline representation is about one magnitude faster than LTE. When executing the algorithm on the robot, computation time is further reduced for both methods as the code is run as an executable C++ program.

The online trajectory deformation is illustrated more in detail in Fig. 7. Because the time to reach the hit point

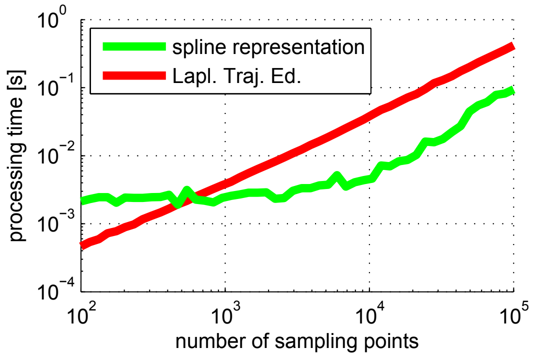

Fig. 6: Reduced computational complexity using a spline representation.

and the time back to the resting position are similar for all trajectories (381ms and 673ms), one can see well how the end effector has to move faster to reach a hit position further away from the resting position. Although the acceleration difference to the original trajectory is minimized by using splines, the absolute acceleration can still become large for big trajectory deviations in task space. As the acceleration is coupled to the torques through the dynamical model of the manipulator, minimizing the acceleration also minimizes the torques up to a certain extent. This is effect becomes more dominant for fast movements where the gravitational term plays a minor role. Fig. 8 illustrates this effect. Shown on top is a true-to-scale model of the table with an overlaid contour plot. The color of the contour corresponds to the maximum end effector acceleration of the robot depending on the position of the (x_h, y_h) for the deformed trajectory. All other parameters like the original trajectory, the angle Φ_h and the hit velocity (v_{xh}, v_{yh}) and time t_h are kept constant. The bottom figure shows the maximum joint torque of the manipulator, illustrating the dependence on the end effector acceleration.

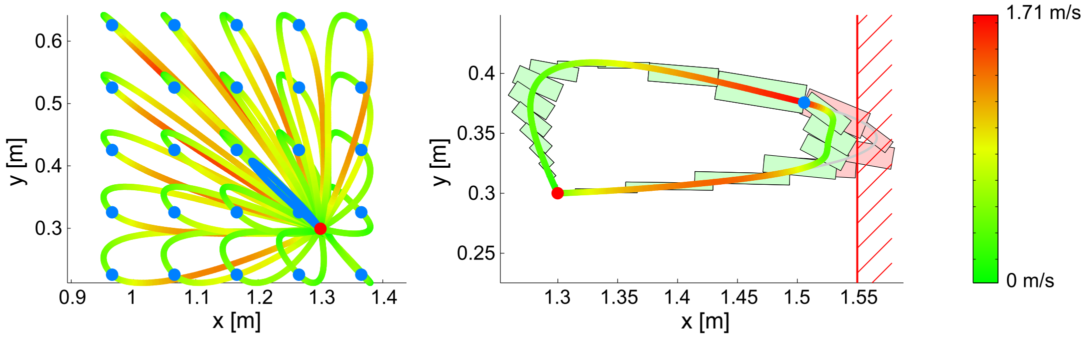

Fig. 7: Trajectory deformation for the volleyball scenario. Left side: Multiple possible deformations for a given hit position (blue dots) and start/end position (red dot) with corresponding velocity along the trajectory (path color). The blue line marks the undeformed reference trajectory calculated with DirCol. Right side: Collision avoidance scheme using convex hulls (red rectangles) for a given workspace boundary (red line). Rectangular convex hulls were used here to simplify calculations. As they enclose the trajectory, the trajectory is deformed through the collision avoidance scheme of Sec. II-B such that the convex hulls do not intersect with any obstacle (green rectangles).

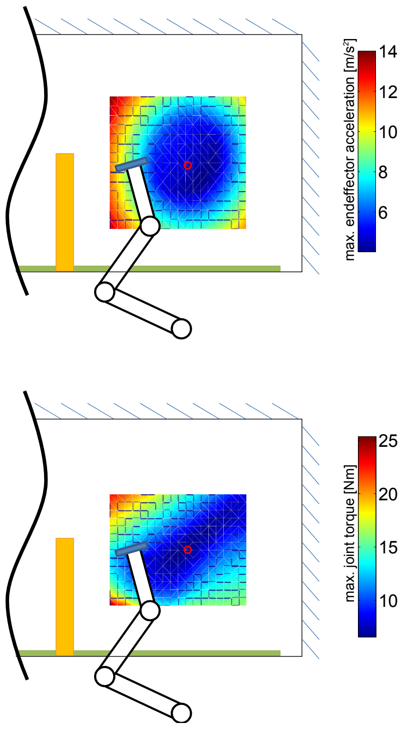

Fig. 8: Maximum end effector acceleration and joint torques depending on the exact hit position in task space. A red dot depicts the hit position of the original undeformed trajectory.

V. Discussion

Experiments show that the presented approach works reliably and is executed online. Right now only minimum acceleration deformation is considered in the paper. This is chosen intentionally as the resulting cubic splines offer a good tradeoff between generalization and stability without suffering from oscillations occurring with higher order polynomials. Still it is theoretically possible to use splines of any degree for the deformation process. For 5-th order splines it results in minimum jerk deformation, for 7-th order splines in minimum snap deformation etc. In addition, only one reference trajectory is deformed. As the experiments show this becomes problematic if the trajectory deformation gets large, leading to high end effector accelerations and joint torques. Another option is to use multiple trajectories and select the one that needs to be deformed less for hitting the ball. Yet this raises the question about proper selection strategies and is thus not further investigated here.

Although not implemented in the current experiment, the manipulator trajectory can be recalculated every millisecond, allowing for fast adaptation strategies in case of sudden disturbances. This is especially important if the air drag can not be neglected anymore, requiring either a very precise model of the environment or a continuous recalculation of the hitting movement.

Whereas large parts of the approach are very stable, we encounter some problems in finding a proper number of trajectory intervals for the collision avoidance procedure. As mentioned before it is a tradeoff between contradicting objectives, thus manual adjustment is necessary. Even if the presented collision avoidance scheme has proven to work reliably for the given scenario, it is unknown how it behaves in very cluttered environments as convergence of the collision avoidance algorithm is not proven yet.

VI. Conclusion and Future Work

This paper presents a novel method for retargeting previously calculated optimal trajectories online. The approach can be executed online and in real time together with the feasibility check on kinematic and dynamic constraints of the generated trajectories. Special structure of the LTE deformation allows to modify separate nodes with preserving, as much as possible, desired acceleration profile of the precalculated optimal trajectory. Deformation algorithm is used for avoiding task space obstacles. Simulation results were experimentally validated with planar volleyball scenario. The method shows very high repeatability and robustness of the produced motions in constrained environment.

Future work will be focused on extending the presented approach to 5-th or higher order polynomials to cover a wider variety of retargeting problems. In addition it is still an open question whether a collision avoidance algorithm with guaranteed avoidance can be found for the given trajectory representation.

Appendix

It is stated in Sec. II-A that the term

$$\begin{bmatrix}\mathbf{L}\\ \bar{\mathbf{P}}\end{bmatrix}^{+}\begin{bmatrix}\mathbf{0}\\ \bar{\mathbf{C}}_2\end{bmatrix}, \tag{21}$$

can be expressed in the continuous domain by a cubic spline. As the matrix $\mathbf{L}$ consists of second order finite differences along the entire trajectory and the matrices $\bar{\mathbf{P}}, \bar{\mathbf{C}}_2$ specify waypoints, the term in (21) can then be interpreted as minimizing the acceleration along the trajectory while passing a set of waypoints. In the continuous domain this corresponds to minimizing the acceleration $\ddot{x}$. An optimal trajectory x_{opt} is then calculated as

$$x_{opt} = \min_x I(x) = \min_x \frac{1}{2}\int_0^T \ddot{x}(t)^2 dt. \tag{22}$$

Through the calculus of variation we consider the disturbed trajectory $x(t) + \epsilon\eta$ with the scalar ϵ and η as an arbitrary function fulfilling the boundary conditions

$$\begin{aligned}\eta(0) = 0, \quad \eta(T) = 0,\\ \dot{\eta}(0) = 0, \quad \dot{\eta}(T) = 0,\end{aligned} \tag{23}$$

This results in

$$I(x+\epsilon\eta) = \frac{1}{2}\int_0^T (\ddot{x} + \epsilon\ddot{\eta})^2 dt, \tag{24}$$

$$\frac{dI(x+\epsilon\eta)}{d\epsilon} = \int_0^T (\ddot{x} + \epsilon\ddot{\eta})\ddot{\eta} dt. \tag{25}$$

For x to minimize $I(x+\epsilon\eta)$, the following condition has to be fulfilled. In all other cases the trajectory x is not optimal. Note that we write $^{(i)}$ for the i-th time derivative.

$$\left.\frac{dI(x+\epsilon\eta)}{d\epsilon}\right|_{\epsilon=0} = 0 = \int_0^T \ddot{x}\ddot{\eta} dt = \int_0^T x^{(2)}\eta^{(2)}. \tag{26}$$

Through partial integration we obtain

$$\int_0^T x^{(2)}\eta^{(2)} = \underbrace{x^{(2)}\eta^{(1)}\Big|_0^T}_{=0} - \int_0^T x^{(3)}\eta^{(1)} = -\int_0^T x^{(3)}\eta^{(1)}. \tag{27}$$

Applying partial integration another time yields

$$-\int_0^T x^{(3)}\eta^{(1)} = -\underbrace{x^{(3)}\eta\Big|_0^T}_{=0} + \int_0^T x^{(4)}\eta = \int_0^T x^{(4)}\eta, \tag{28}$$

giving

$$\int_0^T x^{(4)}\eta = 0, \tag{29}$$

as the resulting condition that must hold for any function η. This is the case for every function fulfilling

$$x^{(4)} = 0 \;\; \forall t \in [0, T]. \tag{30}$$

Any third order polynomial of the form

$$x = \sum_{k=0}^{3} \mathbf{a}_k t^k, \tag{31}$$

fulfills this condition. Note that the proof is a modified version of the one for minimum jerk trajectories in [17]. Fore more details on the work related to the proof one can see [18].

Acknowledgment

This work is supported in part within the ERC Advanced Grant SHRINE Agreement No. 267877 (www.shrine-project.eu) and in part by the Technische Universität München - Institute for Advanced Study (www.tum-ias.de), funded by the German Excellence Initiative.

References

[1] K. M. Lynch and M. T. Mason, “Dynamic nonprehensile manipulation: Controllability, planning and experiments,” *International Journal of Robotics Research*, vol. 18, No. 1, pp. 64–92, 1999.

[2] K. Mülling, J. Kober, O. Kroemer, and J. Peters, “Learning to select and generalize striking movements in robot table tennis,” *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[3] M. Buehler, D. E. Koditschek, and P. J. Kindlmann, “Planning and control of robotic juggling and catching tasks,” *International Journal of Robotics Research*, vol. vol.13, no.2, pp. pages 101–108, April 1994.

[4] T. Senoo, A. Namiki, and M. Ishikawa, “High-speed batting using a multi-jointed manipulator,” in *Robotics and Automation. Proceedings. IEEE Int. Conf. on*, vol. 2, pp. 1191–1196, 2004.

[5] O. Von Stryk and M. Schlemmer, “Optimal control of the industrial robot manutec r3,” *Computational optimal control, International series of Numerical Mathematics*, vol. 115, pp. 367–382, 1994.

[6] A. Pekarovskiy and M. Buss, “Optimal control goal manifolds for planar nonprehensile throwing,” *In IEEE/RSJ Int Conf. on Intelligent Robots and Systems*, 2013.

[7] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, “Chomp: Gradient optimization techniques for efficient motion planning,” in *IEEE Int. Conf. on Robotics and Automation*, pp. 489–494, 2009.

[8] S. Quinlan, *Real-time modification of collision-free paths*. PhD thesis, Stanford University, 1994.

[9] A. Pekarovskiy, F. Stockmann, M. Okada, and M. Buss, “Hierarchical robustness approach for nonprehensile catching for rigid objects,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014.

[10] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, “Finding locally optimal, collision-free trajectories with sequential convex optimization.,” in *Robotics: Science and Systems*, vol. 9, pp. 1–10, Citeseer, 2013.

[11] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Movement imitation with nonlinear dynamical systems in humanoid robots,” in *IEEE Int. Conf. on Robotics and Automation*, pp. 1398–1403, 2002.

[12] J. Kober, K. Mulling, O. Kromer, C. H. Lampert, B. Scholkopf, and J. Peters, “Movement templates for learning of hitting and batting,” in *IEEE Int. Conf. on Robotics and Automation*, pp. 853–858, 2010.

[13] S. Calinon, E. Sauser, A. Billard, and D. Caldwell, “Evaluation of a probabilistic approach to learn and reproduce gestures by imitation,” in *IEEE Int. Conf. on Robotics and Automation*, pp. 2381–2388, 2010.

[14] K. Yamane and Y. Nakamura, “Dynamics filter - concept and implementation of online motion generator for human figures,” *Robotics and Automation, IEEE Transactions on*, vol. 19, pp. 421–432, June 2003.

[15] T. Nierhoff and S. Hirche, “Fast trajectory replanning using laplacian mesh optimization,” in *IEEE Int. Conf. on Control, Automation, Robotics and Vision*, 2012.

[16] T. Nierhoff, S. Hirche, and Y. Nakamura, “Variable positional constraints for laplacian trajectory editing,” in *DGR-Tage*, 2013.

[17] T. Flash and N. Hogan, “The coordination of arm movements: An experimentally confirmed mathematical model,” *The Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[18] G. Wahba, *Spline models for observational data*, vol. 59. Siam, 1990.