Fast Trajectory Replanning Using Laplacian Mesh Optimization

Thomas Nierhoff and Sandra Hirche
Institute of Automatic Control Engineering
Technische Universität München
D-80290 München, Germany
Email: {tn,hirche}@tum.de

***Abstract*—Adjusting to new situations by changing the shape of a prerecorded trajectory is an important aspect for robot manipulation in a constrained environment. For being recognized as a distinctive trajectory, the goal of any trajectory modification is to keep local and global properties as similar as possible compared to the reference trajectory. This paper presents a framework that can alter the shape of a trajectory by defining the position of a set of sampling points while maintaining local properties in a least-squares manner. The method consists of a three-staged approach first modifying the global shape of the trajectory and subsequently taking local features into account. Inspired by mesh processing used for 3D surface editing, differential coordinates based on the discretized Laplacian operator are used for measuring and maintaining local trajectory properties when deforming the trajectory. Last, a post-processing step based on a relaxed "as-rigid-as-possible" principle allows local deformations and length modifications of the trajectory for a better tradeoff between preserving local and global properties. Experiments verifying the applicability of the proposed algorithm are conducted using a 7-DoF anthropomorphic arm following a previously recorded and modified trajectory.**

I. Introduction

Robotic systems are increasingly used for assisting humans in daily life tasks. Even though some highly specialized commercial systems like Roomba or Automower [1], [2] exist already today, it is still a largely open challenge to design a full-scale humanoid robot that can interact with any human in a natural way. One core component of natural interaction are natural movements. Here, programming by demonstration and movement adaption [3] are considered being a promising approach. Rather than programming every single possible movement manually, a class of typical movements is encoded as a prototypic movement and then modified in a adequate manner. This way, only a few movements are required to be able to fulfil a large variety of tasks by suiting them to situation-dependent circumstances.

For both techniques - programming by demonstration and movement adaption - different methods are proposed. Examples are Hidden Markov Models [4], Gaussian mixture- and regression models [5] being able to extract one representative trajectory from multiple demonstrations and dynamic movement primitives [6] suited well for low-dimensional trajectory representation. All three methods have in common they can both encode and reproduce trajectories and allow trajectory modifications by varying certain model parameters.

However, in certain situations the user wants to modify some prerecorded discretized trajectory not in the parameter space of an underlying model but directly in the task space of the trajectory. This can be both, faster and more efficient, as the trajectory does not have to be generated first. In addition it provides a more intuitive and possibly flexible way than the parameter space if the number of sampling points of the trajectory is large enough.

A still ongoing problem is a proper trajectory adaption in case a single sampling point of the trajectory is displaced for collision avoidance or task adaption. In this case, a set of adjacent sampling points has to be displaced as well in order to maintain the local trajectory properties. Whereas maintaining local trajectory properties (curvature) and global trajectory properties (fixed sampling points, trajectory shape) appears to be a challenging optimization problem at first glance, methods do exist in computer graphics for this type of problem. Being a standard tool in 3D surface editing and mesh processing, Laplacian mesh optimization provides a fast and intuitive way to change the global shape of a mesh while preserving local features and considering positional constraints [7], [8]. Based on the discrete version of the Laplace-Beltrami-operator, the method tries to maintain the local mesh curvature if vertices of the mesh are displaced. This is also known as "as-rigid-as-possible" (ARAP) transformation [9]–[11]. Resulting in a linear least-squares problem, it can be solved in real-time even for large meshes. By interpreting a path (a trajectory without time information) as a very primitive mesh, this method can be adapted for trajectories as well.

The contribution of this paper is the adaption of Laplacian mesh optimization for modifying trajectories while keeping local and global properties similar to a reference trajectory for application in robotics. As existing methods partially fail if the trajectory deformation is large, a novel optimization approach is presented. This makes it possible to preserve characteristic properties of the trajectory even if the trajectory is deformed by adding positional constraints on a subset of sampling points of the trajectory. Experiments with a 7-DoF anthropomorphic robot manipulator performing imitation of a human movement and the subsequent adaption of the recorded trajectory is performed in order to react properly to environmental changes. The results show the suitability of the proposed approach for online trajectory replanning.

The remainder of this paper is organized as follows: Section IV illustrates the general approach. In section V experiments show the movement imitation and adaption on an anthropomorphic arm together with a critical analysis of the approach, discussing both its potential and limitations. Last, section VI concludes with a final statement and possible expansions for the future.

II. Objective and Conceptual Approach

This paper adresses the general issue of how to transform a sampled reference trajectory into another one if the position of a set of sampling points has to be set to a fixed value. In case enough sampling points are fixed, no perfect matching can be achieved in general and an approximated solution has to be used instead. Here, the fundamental question is how to achieve a suitable approximation as there is no unique best solution. The adaption of Laplacian mesh optimization for trajectory editing constitutes the core component for modifying trajectories (section IV-B). As Laplacian mesh optimization only preserves local features, a preprocessing step modifies the global shape of the trajectory through an affine transformation to achieve a coarse matching for the fixed sampling points (section IV-A). In addition, a post-processing step (section IV-C) ensures a good tradeoff between preserving local and global properties.

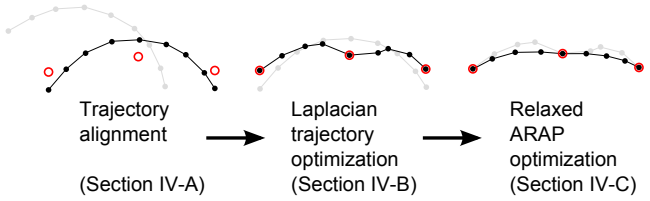

Figure 1. Overview of the three-staged approach presented in this paper. Red circles around certain sampling points indicate the desired position of fixed sampling points, grey lines mark the trajectory before the corresponding modification stage and black lines a schematic resulting trajectory.

III. Laplacian Trajectory Rrepresentation

Each trajectory is given by an ordered set of sampling points $\mathbf{P} = [\mathbf{p}(t_1), \mathbf{p}(t_2), \ldots, \mathbf{p}(t_n)]^T \in \mathbb{R}^{n\times 3}$ where t_i represents the time $t_i \in \mathbb{R}$, $\mathbf{p}(t_i) \in \mathbb{R}^3$. The set of sampling points $\mathbf{P} = [\mathbf{p}(t_1), \mathbf{p}(t_2), \ldots, \mathbf{p}(t_n)]^T$ is rewritten as $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n]^T$. The trajectory is represented as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each vertex v_i is associated with one sampling point $\mathbf{p}_i \rightarrow v_i$, i.e. $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$. The neighbor set of the vertex v_i, i.e. the set of adjacent vertices, is given by $\mathcal{N}_i = \{v_j | j \in \{i+1, i-1\}; i, j \in \{1, , n\}\}$. Accordingly the edge set is defined as $\mathcal{E} = \{e_{ij}\}, i, j \in \{0, .., n\}$ with

$$e_{ij} = \begin{cases} w_{ij} & \text{if } j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Depending on the irregularity of the edge lengths $l_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|$, different weighting terms w_{ij} are presented in literature for mesh optimization: Uniform umbrella weights $w_{ij} = 1$ or scale-dependent umbrella weights $w_{ij} = 1/l_{ij}$. As cotangent weights described in [12] are only defined for triangular meshes structures, they cannot be applied for trajectories. Uniform umbrella weights proved to be sufficient in all performed experiments and in literature [13], so they are used for the remainder of this paper.

Instead of working in absolute Cartesian coordinates, the discrete Laplace-Beltrami operator $\boldsymbol{\delta}$ is used for specifying the local properties of the trajectory [14]. For vertex $\mathbf{v}_i$ it is defined as

$$\boldsymbol{\delta}_i = \sum_{j\in\mathcal{N}_i} \frac{w_{ij}}{\sum w_{ij}} (\mathbf{p}_i - \mathbf{p}_j), \tag{2}$$

The topology of the graph can be represented as a tridiagonal Laplacian matrix matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$ with

$$\mathbf{L_{ij}} = \begin{cases} 1 & \text{if } i = j, \\ -\dfrac{w_{ij}}{\sum\limits_{j\in\mathcal{N}_i} w_{ij}} & \text{if } j \in \mathcal{N}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

IV. Approach

A. Trajectory Alignment

Given a reference trajectory that has to be realigned through an affine transformation such that some sampling points of the trajectory minimize the distance to predefined positions, this section describes an SVD-based approach to achieve this goal.

To apply the affine transformation it is assumed that a subset $\mathbf{P}_s = [\mathbf{p}_{s1}, \mathbf{p}_{s2}, \ldots, \mathbf{p}_{sk}]^T \in \mathbf{P}$ of sampling points - with a special meaning like being the start pointt of the trajectory, end point, grasping point or interaction point - shall be mapped as good as possible onto desired positions $\mathbf{P}_d = [\mathbf{p}_{d1}, \mathbf{p}_{d2}, \ldots, \mathbf{p}_{dk}]^T$. Formally

$$\mathbf{p}_{di} = c\mathbf{R}\mathbf{p}_{si} + \mathbf{t} + \mathbf{n}_o, \quad i = 1, \ldots, k, \tag{4}$$

with c as a scalar scaling factor, $\mathbf{R} \in \mathbb{R}^{3\times 3}$ as a rotation matrix, $\mathbf{t}$ as a translational vector and $\mathbf{n}_o$ as a noise term due to matching imperfection that has to be minimized. For this purpose, least squares fitting using SVD as presented in [15] and [16], can be applied: The first step consists of the calculation of the centroids $\bar{\mathbf{p}}_s$ and $\bar{\mathbf{p}}_d$ of $\mathbf{P}_s$ and $\mathbf{P}_d$ as

$$\bar{\mathbf{p}}_s = \frac{1}{k}\sum_{i=1}^{k} \mathbf{p}_{si}, \qquad \bar{\mathbf{p}}_d = \frac{1}{k}\sum_{i=1}^{k} \mathbf{p}_{di}. \tag{5}$$

Then $\mathbf{P}_s$ and $\mathbf{P}_d$ have to be centered around $\bar{p}_s$ and $\bar{p}_d$

$$\mathbf{p}'_{si} = \mathbf{p}_{si} - \bar{\mathbf{p}}_s, \quad \mathbf{p}'_{di} = \mathbf{p}_{di} - \bar{\mathbf{p}}_d, \quad i = 1, \ldots, k. \tag{6}$$

The covariance matrix $\mathbf{Q}$ is calculated as

$$\mathbf{Q} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{p}'_{si}\mathbf{p}'^{T}_{di}, \tag{7}$$

and the variance σ_s^2 as

$$\sigma_s^2 = \frac{1}{k}\sum_{i=1}^{k} \|\mathbf{p}'_{si}\|^2 . \tag{8}$$

The SVD of $\mathbf{Q}$ is calculated such that $\mathbf{Q} = \mathbf{U}_q\mathbf{S}_q\mathbf{V}_q^T$. Then c, $\mathbf{R}$ and $\mathbf{t}$ can be computed as

$$\mathbf{R} = \mathbf{V}_q\mathbf{S}'_q\mathbf{U}_q^T, \tag{9}$$

$$c = \frac{1}{\sigma_s^2} tr(\mathbf{S}_q\mathbf{S}'_q), \tag{10}$$

$$\mathbf{t} = \bar{\mathbf{p}}_d - c\mathbf{R}\bar{\mathbf{p}}_s. \tag{11}$$

with $\mathbf{S}'_q$ preventing mirrored mappings

$$\mathbf{S}'_q = \begin{cases} \mathbf{I} & \text{if } det(U_q)det(V_q) = 1, \\ diag(1,\ldots,1,-1) & \text{if } det(U_q)det(V_q) = -1. \end{cases} \tag{12}$$

Remark 1: It is also possible to use multilateration instead of an SVD-based affine transformation: Given enough sampling points in $\mathbf{P}_s$ and $\mathbf{P}_d$ such that (4) applies with $\mathbf{n}_o = \mathbf{0}$, i.e. a perfect affine transformation is possible, multilateration can be applied. However, multilateration fails easily if the mapping between $\mathbf{P}_s$ and $\mathbf{P}_d$ is not affine or the basis formed for multilateration of $\mathbf{P}_s$ or $\mathbf{P}_d$ is ill-defined.

B. Laplacian Trajectory Optimization

After the trajectory is aligned depending on the task as described in IV-A, this section decribes how to ensure positional constraints while maintaining shape similarity. When concatenating all $\boldsymbol{\delta}_i$-values into a single vector as $\boldsymbol{\Delta} = [\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \ldots, \boldsymbol{\delta}_n]^T$, one can write the equation system

$$\mathbf{L}\mathbf{P}' = \boldsymbol{\Delta}. \tag{13}$$

with $\mathbf{P}' = [\mathbf{p}'_1, \mathbf{p}'_2, \ldots, \mathbf{p}'_n]^T \in \mathbb{R}^{n\times 3}$ for the desired sampling point positions of the modified trajectory. As the equation system is underdetermined (the differential coordinates $\boldsymbol{\Delta}$ are translational invariant), one additional positional constraint is needed in order to get a unique solution. A set of k additional positional constraints in the form $\mathbf{p}_i = \mathbf{c}_i$ can be defined by adding them to the linear equation system (13)

$$\begin{pmatrix} \mathbf{L} \\ \omega\bar{\mathbf{P}} \end{pmatrix} \mathbf{P}' = \begin{pmatrix} \boldsymbol{\Delta} \\ \omega\mathbf{C} \end{pmatrix}, \tag{14}$$

with the definition of $\bar{\mathbf{P}} \in \mathbb{R}^{n\times n}$ and $\mathbf{C} \in \mathbb{R}^{n\times 3}$ as follows

$$\bar{P}_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and } \mathbf{p}_i = \mathbf{c}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

$$C_{i:} = \begin{cases} [p_{ix}, p_{iy}, p_{iz}] & \text{if } \mathbf{p}_i = \mathbf{c}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

For $k > 1$ positional constraints this results in an overdetermined linear equation system defining both the local curvature of the trajectory and additional positional constraints of possibly displaced vertices. The equation system can be solved for $\mathbf{P}'$ using least squares. The weighting factor ω determines the weight of the positional constraints with respect to the curvature of the trajectory. Note that the positional constraints are treaten in the least-squares sense, i.e. they are not matched perfectly and have have to be reassigned manually for a perfect match. If the trajectory is bent too much, artifacts as illustrated in section V-A can occur during the least-squares optimization and have to be enhanced during the post-processing step.

C. Relaxed ARAP Optimization

Any calculated mesh as described in section IV-B still has the disadvantage not including any local rotation of the trajectory as the x, y and z-components of the resulting vertex positions are calculated independently. In addition, it might be necessary to patch artifacts occurring during the least-squares optimization. To overcome both problems, [9] presented a SVD-based ARAP optimization method which iteratively tries to rotate the differential coordinates $\boldsymbol{\delta}$ for preserving the absolute edge lengths as good as possible. This is problematic as any trajectory is inherently more flexible than a surface mesh due to the lower valence of each vertex. Therefore, conventional ARAP optimizations used for modifying trajectories may fail in case the trajectory is bent too much.

The algorithm presented in this paper is a more relaxed version of the SVD-based ARAP optimization, located in between a native Laplacian optimization not allowing any rotations at all and a ARAP optimization rotating differential coordinates freely. Whereas the SVD-based method tries to maintain edge lengths using differential coordinates, our method tries to maintain angles and weighted relative distances between adjacent sampling points in order to model the relation between adjacent sampling points better without forcing absolute distance constraints.

Let $\mathbf{c}_i = \mathbf{p}'_{i+1} - \mathbf{p}'_i$ be the connection vector between two adjacent sampling points and $\mathbf{P}' = [\mathbf{p}'_1, \mathbf{p}'_2, \ldots, \mathbf{p}'_n]^T \in \mathbb{R}^{n\times 3}$ be the set of all sampling points after Laplacian optimization. The angle between adjacent connection vectors is calculated using the dot product

$$\theta'_i = arccos\left(\frac{\mathbf{c}_i \cdot \mathbf{c}_{i-1}}{\|\mathbf{c}_i\|\,\|\mathbf{c}_{i-1}\|}\right). \tag{17}$$

The relative distance between adjacent sampling points is defined as

$$\gamma'_i = \frac{\|\mathbf{c}_i\|}{\|\mathbf{c}_{i-1}\|}. \tag{18}$$

All angles θ'_i and relative distances γ'_i are characteristic measures of every trajectory. Thus, if a reference trajectory (subscript $_r$) is altered, resulting in a modified trajectory (subscript $_m$), the resulting optimization problem can be formulated as keeping angles and relative distances between reference trajectory and modified trajectory as similar as possible. The angular difference θ between the modified trajectory and the reference trajectory of vertex i is defined as

$$\theta_i = \theta'_{ri} - \theta'_{mi}, \tag{19}$$

and the relative distance difference γ_i is defined as

$$\gamma_i = \max\left(\frac{\gamma'_{ri}}{\gamma'_{mi}}, \frac{\gamma'_{mi}}{\gamma'_{ri}}\right) - 1. \tag{20}$$

It is $\max\left(\frac{\gamma'_{ri}}{\gamma'_{mi}}, \frac{\gamma'_{mi}}{\gamma'_{ri}}\right) \geq 1$. The term -1 ensures that a perfect match of relative distances will result in a relative distance difference of 0. The resulting optimization problem can be formulated as

$$\mathbf{P}'' = \underset{\mathbf{P}'}{\operatorname{argmin}} \left(\sum_{i=1}^{n-1} \theta_i^2 + \mu \gamma_i^2 \right), \tag{21}$$

subject to the equality constraints

$$\theta_i = \theta'_{ri} - \theta'_{mi}(\mathbf{P}'), \tag{22}$$

$$\gamma_i = \max\left(\frac{\gamma'_{ri}}{\gamma'_{mi}(\mathbf{P}')}, \frac{\gamma'_{mi}(\mathbf{P}')}{\gamma'_{ri}}\right) - 1. \tag{23}$$

with the term μ as a weighting factor. This is a nonlinear least-squares problem depending on the 3D position of all vertices in $\mathbf{P}'$. However, as the optimization search space is $3n$-dimensional for a 3D space it becomes quite inefficient as soon as the number of sampling points of the trajectory increases.

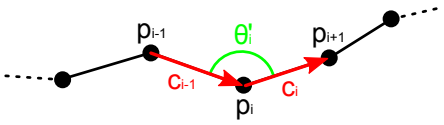

Figure 2. Optimization is based on the angle θ_i between adjacent sampling points and the relation between the distances of adjacent sampling points using the vectors $\mathbf{c}_i$ and $\mathbf{c}_{i-1}$.

A fast approximate solution (relaxed fast ARAP optimization) can be obtained under two conditions:

- To reduce the dimensionality of the search space, gradients for vertex i are only allowed to be computed in the plane defined by $\mathbf{c}_i$ and $\mathbf{c}_{i-1}$. This reduces the search space to $2n$ dimensions, regardless of the task space dimension.
- Due to the structure of the trajectory and the type of cost function, gradients at optimization step o have to be computed only for sampling points adjacent to other sampling points which are updated at optimization step $o-1$.

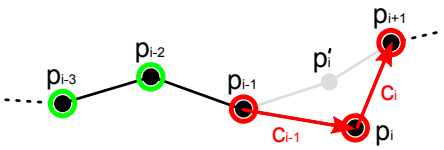

Figure 3. Independence of the optimization terms θ_i and γ_i: When changing the position of the sampling point $\mathbf{p}'_i$ during the optimization step o, only the gradients for $\mathbf{p}_{i+1}$, $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$ (red circles) have to be calculated at step $o+1$. All other sampling points (green circles) stay the same as they do not depend on $\mathbf{c}_i$, $\mathbf{c}_{i-1}$ or θ_i.

In combination with a threshold below which the gradient is set to zero, it increses the execution speed of gradient-based algorithms at the cost of a slighty reduced accuracy.

The first point limits the search space of the optimization algorithm for finding better sampling point positions. As the result of the Laplacian trajectory optimization is generally close to the desired shape of the trajectory, the search space reduction helps preventing the optimization algorithm from finding results differing too much from the Laplacian optimized trajectory.

The second point takes advantage the fact that certain terms of the optimization are independent of other terms. As the optimal solution of (21) is the sum of single optimal partial solutions θ_i and γ_i, each one depending only on a certain number of sampling point positions $\mathbf{p}_i$, gradients have to be computed during an optimization step only for variables which are either not yet at their optimal value or dependent of sampling point positions that changed during the last optimization step. The gradient threshold prevents variables from moving towards a more optimal point for small gradients. This helps reducing the number of gradient evaluations of dependent variables, see Fig. 3.

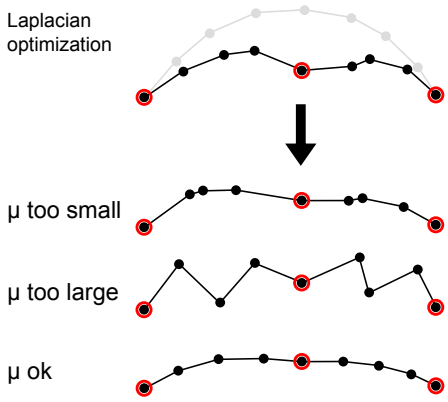

Figure 4. Schematic effect of varying the μ parameter for fixed sampling point positions (red dots): Topmost trajectory: Resulting trajectory after Laplacian optimization. Lower three images: Resulting trajectory after relaxed ARAP optimization.

Determination of a proper value for μ is crucial. When being set to a too small or too large value, too much weight is given on matching the angles respectively the relative distances between modified trajectory and reference trajectory, see Fig. 4.

V. Experimental Evaluation

A. Experimental Results

Experiments are conducted in two ways: In the first set of experiments (Fig. 5 - 7) an original trajectory is modified by fixing some of its sampling points at certain positions. Subsequently the methods presented in this paper are compared with other approaches. In the second experiment (Fig. 8), the applicability to a real-life problem is shown.

A first experiment in Fig. 5 shows the difference between a piecewise shearing of the reference trajectory and Laplacian optimization, as they do look similar at a first glance. One can

see that Laplacian optimization offers the advantage of maintaining local properties like a smooth curvature characteristics also at the fixed sampling points.

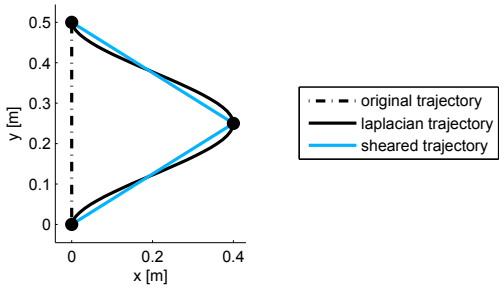

Figure 5. Comparison between Laplacian optimization and piecewise shearing of a straight reference trajectory. Fixed sampling points are marked by black dots. Whereas shearing preserves the local features of a straight line between the fixed sampling points better, it results in a kink at the central point.

The difference between Laplacian mesh optimization, relaxed ARAP optimization and SVD-based ARAP optimization is shown in Fig. 6: Based on a reference trajectory first Laplacian optimization is performed and subsequently either SVD-based ARAP optimization, relaxed ARAP optimization or relaxed fast ARAP optimization. Whereas the SVD-based ARAP optimization ensures better similarity of local features, it differs from Laplacian optimization at global scale. The two different relaxed ARAP optimizations with parameter $\mu = 0.1$ produce almost similar results while the approximated version is around two magnitudes faster. For all experiments presented in this section, relaxed fast ARAP optimization took between 30ms and 100ms whereas relaxed ARAP optimizations took between 1s and 10s on a Core2Duo T7500 with 4GB Ram using Matlab.

The difference between relaxed ARAP optimization and smoothing is shown in Fig. 7: Here, relaxed ARAP optimization is compared to a single iteration of Laplacian smoothing, an iterative process that computes the resulting position $\tilde{\mathbf{p}}_i$ of vertex i as the centroid of all neighboring vertices

$$\tilde{\mathbf{p}}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{p}_j. \tag{24}$$

Even one iteration of Laplacian smoothing produces unacceptable results both at global scale where the trajectory differs much from the Laplacian trajectory and at local scale where distinctive corners have got a round shape.

Real-life experiments are conducted by tracking the 3D trajectory of a yellow marker from a human demonstration using a Microsoft Kinect. Reproduction of the trajectory is performed using a 7-DoF anthropomorphic manipulator with a Schunk PG-70 gripper. Fifth-order polynomials are used for interpolation between adjacent sampling points of the trajectory. As shown in Fig. 8, the reproduced trajectories are modified in three different ways: Whereas the first trajectory has only been aligned according to section IV-A to match

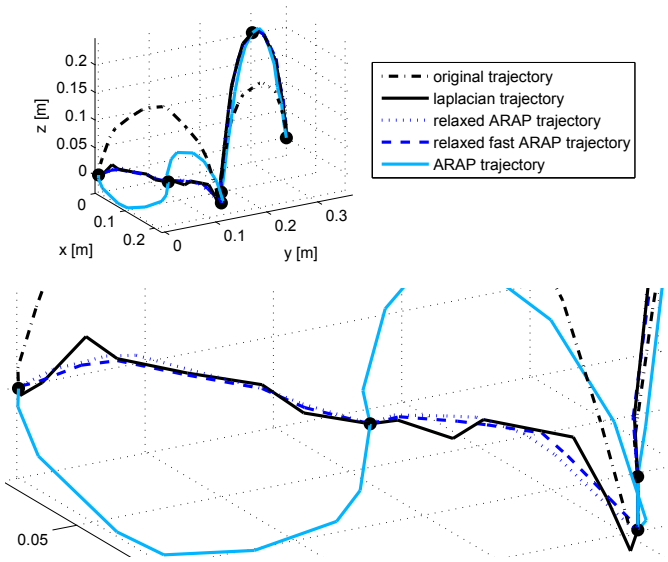

Figure 6. Drawbacks of pure Laplacian optimization and SVD-based ARAP optimization at global scale (top image) and local scale (bottom image). Fixed sampling points are marked by black dots.

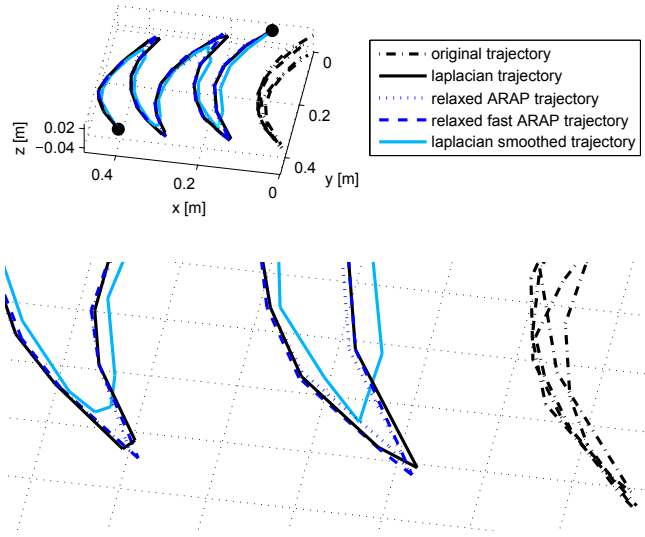

Figure 7. Comparison of relaxed ARAP optimization and one iteration of Laplacian smoothing at global scale (top image) and local scale (bottom image). Fixed sampling points are marked by black dots.

the green waypoints on the table, one waypoint is moved for the second trajectory. For the third trajectory, the obstacle in between the waypoints is removed partially. As a consequence, is is now possible to traverse between two of the waypoints in a straight line. For the second and third trajectory, Laplacian trajectory optimization and relaxed ARAP optimization with fixed waypoint positions are performed in order to obtain the smooth trajectories and to fulfill the positional waypoint constraints.

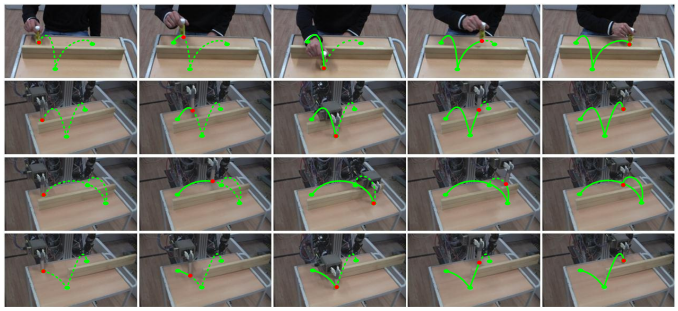

Figure 8. Trajectory imitation using an anthropomorphic manipulator. Top row: Human demonstrated trajectory. Second row: Replayed trajectory. Third row: Trajectory with modified waypoint. Fourth row: Trajectory with removed obstacle. Columns (from left to right): Chronological order.

B. Discussion

Experiments show Laplacian trajectory optimization in combination with a post-processing step (relaxed ARAP optimization) and preprocessing (trajectory alignment) works well for manipulating trajectories. Depending on the required execution speed, the post-processing step can be neglected. In this case, the total execution time for trajectory alignment and Laplacian optimization is smaller than 1ms for the presented experiments.

Considering computational complexity, current solvers for sparse matrices depend only on the number of nonzero entries. As this number depends linearly on the number of sampling points, the computational complexity for the Laplacian optimization scales linearly with the number of sampling points.

Whereas "stretching" a trajectory, i.e. increasing l_{ij} compared to the reference trajectory, causes only minor differences between a Laplacian optimized trajectory, ARAP optimized trajectory and relaxed ARAP optimized trajectory, the differences and therefore the need to apply our method increase when a trajectory is being "compressed", i.e. l_{ij} decreases with respect to the reference trajectory. Whereas a correct parameterization of μ is crucial for the relaxed ARAP optimization to work properly, varied μ parameters within a rather large range of 0.01-1 produced satisfying results.

Though it is tested only with two- and three-dimensional trajectories, the algorithm is theoretically able to work in an arbitrary number of dimensions. One still existing challenge is a proper selection of sampling points for alignment and repositioning. Whereas start and end point of the trajectory simple to determine, other points usually have to be determined taking also the environment into account.

VI. Conclusion and Future Work

This paper presents a framework for possible online trajectory modifications with the goal of preserving local and global properties as good as possible. Inspired by Laplacian optimization from computer graphics, we present a novel approach to modify the shape of a trajectory by introducing positional constraints on some sampling points of the trajectory. However, instead of performing an "as-rigid-as possible" optimization focusing only on the local features, a novel more relaxed optimization is presented preserving global features during trajectory modification better. Modifications leading to an approximate but much faster solution are presented. Experiments with an anthropomorphic robot with an anthropomorphic robot manipulator with seven degrees of freedom show the approach is suitable for application in robotics.

Laplacian mesh optimization shall not be understood as a competitive algorithm to existing trajectory interpolation methods [17]. Rather it works as an extension to existing methods as the resulting sampling points still have to be interpolated either using polynomial functions, Bézier curves or splines for smooth and high resolution trajectories.

Future work will be focused on the combination of the proposed algorithm with a motion imitation and recognition framework, allowing a robot to interact dynamically with a human counterpart.

Acknowledgement

This work is supported in part within the DFG excellence research cluster *Cognition for Technical Systems - CoTeSys* (`www.cotesys.org`) and by the DAAD (`www.daad.de`).

References

[1] http://www.irobot.com/en/us/robots/home/roomba.aspx.
[2] http://www.husqvarna.com/us/products/robotic-mowers/husqvarna-robotic-mowers-for-homeowners/.
[3] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, 2008, pp. 1371–1394.
[4] J. R. Medina, M. Lawitzky, A. Mortl, D. Lee, and S. Hirche, "An experience-driven robotic assistant acquiring human knowledge to improve haptic cooperation," in *IEEE IROS*, 2011, pp. 2416–2422.
[5] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.
[6] S. Schaal, "Dynamic movement primitives - a framework for motor control in humans and humanoid robotics," *Adaptive Motion of Animals and Machines*, pp. 261–280, 2006.
[7] M. Alexa, "Differential coordinates for local mesh morphing and deformation," *The Visual Computer*, vol. 19, no. 2, pp. 105–114, 2003.
[8] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *ACM SIGGRAPH*, 2006, pp. 381–389.
[9] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *EUROGRAPHICS*, 2007, pp. 109–116.
[10] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," in *ACM SIGGRAPH*, 2005, pp. 1134–1141.
[11] M. Alexa, D. Cohen-Or, and D. Levin, "As-rigid-as-possible shape interpolation," in *ACM SIGGRAPH*, 2000, pp. 157–164.
[12] U. Pinkall, S. D. Juni, and K. Polthier, "Computing discrete minimal surfaces and their conjugates," *Experimental Mathematics*, vol. 2, pp. 15–36, 1993.
[13] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *ACM SIGGRAPH*, 2004, pp. 175–184.
[14] Y. Lipman, O. Sorkine, M. Alexa, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel, "Laplacian framework for interactive mesh editing," *IJSM*, vol. 11, no. 1, pp. 43–62, 2005.
[15] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," in *IEEE TPAMI*, vol. 9, no. 5, 1987, pp. 698–700.
[16] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," in *IEEE TPAMI*, vol. 13, no. 4, 1991, pp. 376–380.
[17] B. Lau, C. Sprunk, and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *IEEE IROS*, 2009, pp. 2427–2433.