Multiresolution Laplacian Trajectory Replanning

Thomas Nierhoff, Sandra Hirche, and Yoshihiko Nakamura

1. Introduction

Programming by Demonstration (PbD) [1] is a promising way for teaching a robot complex movements. Here the robot imitates an expert's movement either through physical guidance also known as kinesthetic teaching [2] or through suitable mapping algorithms [3]. A major challenge of PbD are its adaption capabilities to changed environments, requiring modifications of the original robot movement. Focusing on discretized endeffector trajectories, there exists one related field of research beside robotics: Laplacian mesh editing in computer graphics [4, 5]. By interpreting a discretized trajectory as a simple mesh, it is possible to adapt mesh editing methods to trajectory deformation [6].

Still there are two drawbacks that have to be overcome: The computational complexity is high as the method includes a matrix inversion. In addition, Laplacian mesh editing can handle only small trajectory deformations as it treats every dimension independently. Existing approaches are either computationally expensive [7] or allow only thoughtful modifications and need manual adjustment [8].

In order to overcome both drawbacks, the method presented in this paper uses a combination of Laplacian mesh editing and a multiresolution approach [9], thus performing all computationally expensive operations only on a downsampled trajectory. Large trajectory deformations are handled through an iterative scheme based on the tangent vector of the downsampled trajectory. Experiments show the applicability of the presented method to typical robotic problems.

2. Problem Statement and Conceptual Approach

Given an initial discretized trajectory, the goal of this paper is to find a real-time capable method to deform it in such a way that

- the trajectory passes defined waypoints
- the local trajectory deformation stays minimal

In the presented approach we propose to perform the time-consuming deformation computations on a subset of so called *support sampling points*. The final trajectory is then retrieved by interpolating the missing sampling points using Laplacian mesh editing. All examples in this paper are based on sinusoidal trajectories. This type of trajectory has been chosen intentionally as it gives an intuitive feeling of the deformation process.

3. Approach

3·1 Laplacian Trajectory Deformation

We assume a trajectory given the combination of a path, described by an ordered set of sampling points $\mathbf{P} = [\mathbf{p}(t_1), \mathbf{p}(t_2), \ldots, \mathbf{p}(t_n)]^T \in \mathbb{R}^{n\times 3}$ and associated temporal information t_i represented as time $t_i \in \mathbb{R}$, $\mathbf{p}(t_i) \in \mathbb{R}^3$. For simplicity, $\mathbf{P} = [\mathbf{p}(t_1), \mathbf{p}(t_2), \ldots, \mathbf{p}(t_n)]^T$ 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 being associated with one sampling point $\mathbf{p}_i \to v_i$, i.e. $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$. The neighbor set $\mathcal{N}_i$ of the vertex v_i is the set of all adjacent vertices v_j. 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 edge length irregularity $l_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|_2$, scale-dependent umbrella weights $w_{ij} = 1/l_{ij}$ can be used instead of uniform umbrella weights $w_{ij} = 1$.

Instead of working in absolute Cartesian coordinates, the discrete Laplace-Beltrami operator $\boldsymbol{\delta}$ is used for specifying the local trajectory deformation. For vertex v_i it is defined as

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

The topology of the graph is defined by the Laplacian matrix matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$ with

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

By concatenating all $\boldsymbol{\delta}_i$-values (named *Laplacian coordinates*) into a single vector as $\boldsymbol{\Delta} = [\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \ldots, \boldsymbol{\delta}_n]^T$, it can be rewritten as

$$\mathbf{LP} = \boldsymbol{\Delta} \tag{4}$$

As the equation system is underdetermined, the sampling point positions cannot be uniquely reestablished using the inverse of $\boldsymbol{\Delta}$. Instead, a set of p additional positional constraints in the form $\mathbf{p}_i = \mathbf{c}_i$ has to be defined by adding them to the linear equation system (4) as

$$\begin{pmatrix} \mathbf{L} \\ \omega\bar{\mathbf{P}} \end{pmatrix} \mathbf{P}_n = \begin{pmatrix} \boldsymbol{\Delta} \\ \omega\mathbf{C} \end{pmatrix}, \tag{5}$$

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

$$\bar{P}_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and } \mathbf{p}_i = \mathbf{c}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

$$C_{i:} = \begin{cases} [p_{ix}, p_{iy}, p_{iz}] & \text{if } \mathbf{p}_i = \mathbf{c}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

and the scalar weighting factor ω. For $p > 1$ positional constraints this results in an overdetermined linear equation system that can be solved for $\mathbf{P}_n$ using least squares. Due to the additional constraints imposed by $\bar{\mathbf{P}}$ and $\mathbf{C}$, the resulting trajectory $\mathbf{P}_n$ differs from $\mathbf{P}$. Note that positional constraints are treated only in the least-squares sense, i.e. the higher the value of the weighting factor ω the better those constraints are satisfied.

3·2 Trajectory Similarity

In order to compare two trajectories $\mathbf{P}$ and $\mathbf{P}_s$, a measure of deformation indicating the similarity is needed. Inspired by [7], it is defined using the summed quadratic deviation E for the original sampling points $\mathbf{p}_j, \mathbf{p}_i$ and the resulting sampling point positions $\mathbf{p}_{js}, \mathbf{p}_{is}$ as

$$E = \sum_{i=1}^{n} \sum_{j\in\mathcal{N}_i} w_{ij} \|(\mathbf{p}_j - \mathbf{p}_i) - \mathbf{R}_{is}(\mathbf{p}_{js} - \mathbf{p}_{is})\|_2^2. \tag{8}$$

with $\mathbf{R}_{is}$ being the rotational matrix described in [10] in order to minimize E.

3·3 Trajectory Downsampling

For speeding subsequent calculations up, the trajectory is downsampled first by selecting only a subset of all sampling points $\mathbf{P}$. An heuristic method found out to work better than just selecting every q-th sampling point is to consider the lowpass-filtered version $\mathbf{P}''$ of the trajectory $\mathbf{P}$ and looking for the points with minimal distance. The lowpass can be either a simple moving-average filter (SMA) or in the frequency domain using FFT.

In case additional fixed positional constraints (defined by $\bar{\mathbf{P}}$ and $\mathbf{C}$ in Eq. (5), those points have to be included in $\mathbf{P}'$.

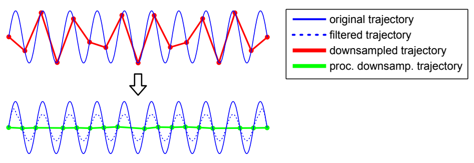


Fig.1 Downsampled trajectory based on every q-th sampling point (red) and the lowpass-filtered version (green)

3·4 Trajectory Adaption

Having downsampled the trajectory, this section covers the trajectory adaption algorithm that constitutes the core concept of the proposed method. As mentioned before, rotational effects cannot be represented properly by a single least squares solution according to Eq. 5. Therefore, an iterative scheme alternately solving the least squares problem for the downsampled trajectory $\mathbf{P}'$ and rotating the corresponding *Laplacian coordinates* is presented. The *Laplacian coordinates* $\mathbf{\Delta}' \in \mathbb{R}^{m\times 3}$ of the downsampled trajectory are calculated as

$$\mathbf{\Delta}' = \mathbf{L}'\mathbf{P}', \tag{9}$$

depending on the scale-dependent umbrella weighted Laplacian matrix $\mathbf{L}' \in \mathbb{R}^{m\times m}$.

Following the idea described in [7], the rotation has to determined such that it minimizes

$$\sum_{l\in\mathcal{N}_k} w_{kl} \|(\widehat{\mathbf{p}}_l - \widehat{\mathbf{p}}_k) - \widehat{\mathbf{R}}_k(\mathbf{p}'_l - \mathbf{p}'_k)\|_2^2 \tag{10}$$

with respect to the retargeted downsampled trajectory $\widehat{\mathbf{P}} = [\widehat{\mathbf{p}}_1, \widehat{\mathbf{p}}_2, \dots, \widehat{\mathbf{p}}_n]^T$. If the edge lengths $\widehat{e}_k$ of the transformed downsampled trajectory differ strongly from the edge lengths e' of the original downsampled trajectory, it is advantageous to propagate the amount of change $\widehat{s}_k = \frac{\widehat{e}}{e'}$ to the differential coordinates of the original trajectory during the reconstruction step.

Mathematically, calculation of the tangential vectors $\mathbf{t}'_k$ and edge lengths e'_k is as follows:

$$\mathbf{t}_{tmp,k} = \frac{\mathbf{p}'_k - \mathbf{p}'_{k-1}}{\|\mathbf{p}'_k - \mathbf{p}'_{k-1}\|_2} + \frac{\mathbf{p}'_{k+1} - \mathbf{p}'_k}{\|\mathbf{p}'_{k+1} - \mathbf{p}'_k\|_2}, \tag{11}$$

$$\mathbf{t}'_k = \frac{\mathbf{t}_{tmp,k}}{\|\mathbf{t}_{tmp,k}\|_2}, \tag{12}$$

$$e'_k = \|\mathbf{p}'_k - \mathbf{p}'_{k-1}\|_2. \tag{13}$$

The iterative process with $\mathbf{P}', \mathbf{T}'$ and $\mathbf{e}'$ as input, the transformed sampling point positions $\widehat{\mathbf{P}}$, tangential vector $\widehat{\mathbf{T}}$, scaling vector $\widehat{\mathbf{s}} \in \mathbb{R}^{m-1}$ and the list of local rotation matrices $\widehat{\mathcal{R}} = [\widehat{\mathbf{R}}_1, \widehat{\mathbf{R}}_2, \dots, \widehat{\mathbf{R}}_m]^T \in \mathbb{R}^{m\times 3\times 3}$ as output is shown in Tab. 1.

This way, the transformed absolute coordinates in $\widehat{\mathbf{P}}$ act as fixed *support sampling points* for the reconstruction process. The local rotation of the *Laplacian coordinates* is stored in $\widehat{\mathcal{R}}$ and a possible stretching/compression of the trajectory is encoded in the scaling vector $\widehat{\mathbf{s}}$.

3·5 Trajectory Reconstruction

During reconstruction one has to calculate the position of the remaining sampling points in between the *support sampling points.* It consists of two parts, the determination of suitable boundary constraints to ensure C^1-continuity and the rotation of the *Laplacian coordinates* of all sampling points in $\mathbf{P}$.

```
• Input
    P', T', Δ', e'
• Output
    $\widehat{\mathbf{P}}, \widehat{\mathbf{T}}, \widehat{\mathcal{R}}, \widehat{\mathbf{s}}$
• Repeat $u$ time
    - If first iteration: $\widehat{\mathbf{\Delta}} = \mathbf{\Delta}'$
    - Calculate new sampling points $\widehat{\mathbf{P}}$ as
          $\widehat{\mathbf{P}} = \begin{pmatrix} \mathbf{L}' \\ \omega\bar{\mathbf{P}} \end{pmatrix}^{-1} \begin{pmatrix} \widehat{\mathbf{\Delta}} \\ \omega\mathbf{C} \end{pmatrix}.$
    - Calculate new tangential vectors $\widehat{\mathbf{T}}$ and edge lengths $\widehat{\mathbf{e}}$ depending on $\widehat{\mathbf{P}}$
    - Calculate the rotation matrix $\widehat{\mathbf{R}}_i$ between every element in $\mathbf{T}'$ and in $\widehat{\mathbf{T}}$. Store it in $\widehat{\mathcal{R}}$.
    - Calculate the scaling $\widehat{s}_i$ factor between every element in $\mathbf{e}'$ and in $\widehat{\mathbf{e}}$ as $\widehat{s}_i = \frac{\widehat{e}}{e'}$. Store it in $\widehat{\mathbf{s}}$.
    - Rotate the Laplacian coordinates by $\widehat{\mathcal{R}}$ as $\widehat{\mathbf{\Delta}} = [\widehat{\mathbf{R}}_1\boldsymbol{\delta}'_1, \widehat{\mathbf{R}}_2\boldsymbol{\delta}'_2, \dots, \widehat{\mathbf{R}}_m\boldsymbol{\delta}'_m]^T$
```

Table 1 Trajectory adaption algorithm

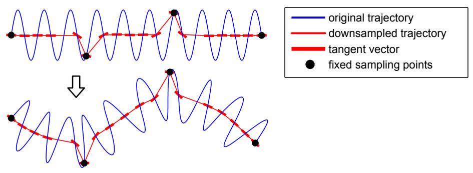


Fig.2 Visualization of the tangent vector before (up) and after deformation (down)

Regarding the boundary constraints, the first order derivatives for each *support sampling point* are calculated based on **P** and then rotated according to $\widehat{\mathcal{R}}$. Let the two first order derivatives at the k-th *support sampling point* from the left and right side be denoted $\boldsymbol{\sigma}_k^-$ and $\boldsymbol{\sigma}_k^+$. They have to be similar for C^1-continuity at the *support sampling points*. Consequently, they can be split up into vectors $\boldsymbol{\sigma}_{k\|}^-, \boldsymbol{\sigma}_{k\|}^+$ collinear to the tangential vector $\widehat{\mathbf{t}}_k \in \widehat{\mathbf{T}}$ and vectors $\boldsymbol{\sigma}_{k\perp}^-, \boldsymbol{\sigma}_{k\perp}^+$ perpendicular to it. Having calculated $\widehat{\mathbf{s}}$, one can scale $\boldsymbol{\sigma}_{k\|}^-, \boldsymbol{\sigma}_{k\|}^+$ accordingly, thus obtaining $\mathbf{C}_s$ for a single trajectory segment as

$$\boldsymbol{\sigma}_k^- = \widehat{\mathbf{R}}_k(\mathbf{p}_{k-1} - \mathbf{p}_k), \tag{14}$$

$$\boldsymbol{\sigma}_k^+ = \widehat{\mathbf{R}}_k(\mathbf{p}_{k+1} - \mathbf{p}_k), \tag{15}$$

$$\boldsymbol{\sigma}_k^{\{-,+\}} = \boldsymbol{\sigma}_{k\|}^{\{-,+\}} + \boldsymbol{\sigma}_{k\perp}^{\{-,+\}}, \tag{16}$$

$$\boldsymbol{\sigma}_{k\|}^{\{-,+\}} \parallel \widehat{\mathbf{t}}_k, \;\; \boldsymbol{\sigma}_{k\perp}^{\{-,+\}} \perp \widehat{\mathbf{t}}_k, \tag{17}$$

$$\mathbf{C}_s = \begin{pmatrix} \widehat{\mathbf{p}}_k \\ \widehat{\mathbf{p}}_k + \boldsymbol{\sigma}_{k\perp}^+ + \frac{\widehat{s}_{k-1}+\widehat{s}_k}{2}\boldsymbol{\sigma}_{k\|}^+ \\ \widehat{\mathbf{p}}_{k+1} + \boldsymbol{\sigma}_{k+1\perp}^- + \frac{\widehat{s}_{k+1}+\widehat{s}_k}{2}\boldsymbol{\sigma}_{k+1\|}^- \\ \widehat{\mathbf{p}}_{k+1} \end{pmatrix}, \tag{18}$$

Rotation of the *Laplacian coordinates* in **P** is based on $\widehat{\mathcal{R}}$ with the rotation matrix being linearly interpolated for every sampling point based on the two surrounding *support sampling points*. This results in the rotated *Laplacian coordinates* $\mathbf{\Delta}_s$ for a single trajectory segment. Having calculated the other matrices/vectors calculated $\mathbf{L}_s$ according to Eq. (3), $\mathbf{C}_s$ according to Eq. (18) and $\bar{\mathbf{P}}_s$ fixing the first/last two sampling points of the trajectory, the resulting equation system can be solved according to Eq. (5) for every trajectory segment.

A similar scaling process as in Eq. 18 can be applied for **C** at the trajectory adaption stage for better results as shown in Fig. 3.

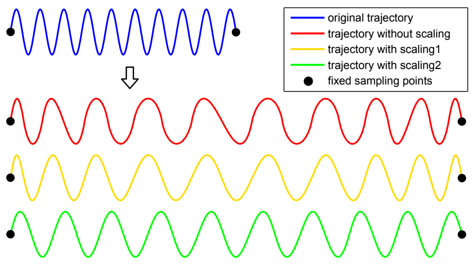


Fig.3 Effect of the scaling process. Scaling1 scales the *Laplacian coordinates* only during the reconstruction step based on $\widehat{\mathbf{s}}$. Scaling2 scales them also during the adaption step

4. Experiments

All calculations are performed using a Intel Core2Duo T7500 CPU with 4GB RAM, implemented in Matlab in combination with OpenRAVE [11].

4·1 Comparison with other Approaches

As stated in the introduction, reducing the computational complexity is one of the principal goals of the proposed method. For this purpose, the processing time is compared in Fig. 4 with a conventional, non multiresolution method called "as-rigid-as-possible" (ARAP) approach [7]. A spatial comparison is presented in Fig. 5

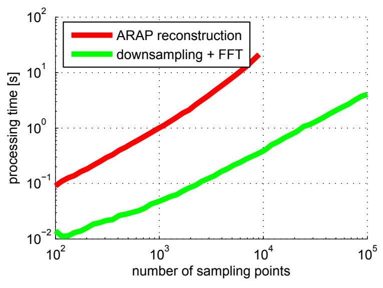


Fig.4 Processing time comparison

4·2 Demonstration Scenarios

The presented scenario optimizes a predefined trajectory (neglecting the endeffector orientation) for

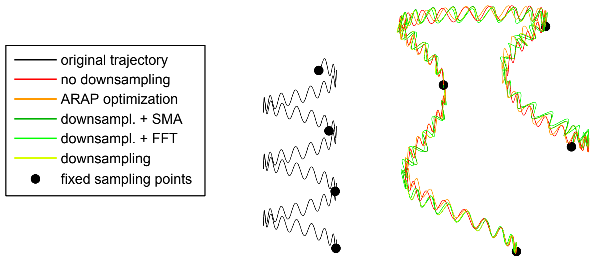
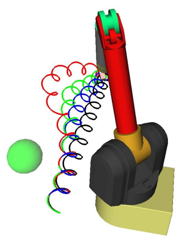


Fig.5 Spatial comparison

a Barrett WAM4 manipulator while avoiding an obstacle represented by a green ball. The approach consists of a two-staged optimization. On the lower level the trajectory is retargeted using Laplacian editing. Following the trajectory is achieved using the method described in [12]. On the higher level, a set of *support sampling points* are repositioned through a continuous gradient descent optimization minimizing the cost function c_τ based on the quadratic joint velocities $\dot{\theta}$ over all links $1, \ldots, \alpha$ and time steps $t_1, \ldots, t_n$ as

$$c_\tau = \sum_{i=1}^{n} \sum_{j=1}^{\alpha} \dot{\theta}_{ij}^2. \tag{19}$$

To limit the deviation between retargeted trajectory and original trajectory, a breaking criterion defined in terms of the deviation of the downsampled trajectory E' is used. Results for different E' values are shown in Fig. 6.



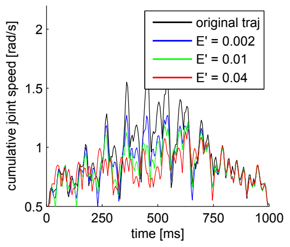


Fig.6 Optimized trajectories for different E' values (left) and summed absolute joint velocities (right)

5. Discussion and Conclusion

This paper presents a novel approach for trajectory retargeting. By using a multiresolution approach, computational complexity can be reduced by up to 2 magnitudes with respect to existing approaches while obtaining similar results. Being easy to handle and numerically robust, it can be safely embedded in an optimization routine.

A challenge of the method consists of finding an optimal number of *support sampling points*. In the non-optimal case either not enough tangent vectors can be calculated to capture the local trajectory deformation properly or the resulting downsampled trajectory may not represent the underlying structure of the trajectory properly, resulting in aliasing effects during the reproduction step.

Future work will be focused on an improved version not depending on any tunable parameters anymore, making it suitable for an even wider class of problems.

References

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, “Robot programming by demonstration,” in *Springer Handbook of Robotics*, 2008, pp. 1371–1394.

[2] D. Lee and C. Ott, “Incremental kinesthetic teaching of motion primitives using the motion refinement tube,” *Auton. Robots*, vol. 31, no. 2-3, pp. 115–131, 2011.

[3] S. Schaal, “Dynamic movement primitives - a framework for motor control in humans and humanoid robots,” in *International Symposium on Adaptive Motion of Animals and Machines*, 2003.

[4] M. Alexa, “Differential coordinates for local mesh morphing and deformation,” *The Visual Computer*, vol. 19, no. 2, pp. 105–114, 2003.

[5] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, “Laplacian mesh optimization,” in *ACM SIGGRAPH*, 2006, pp. 381–389.

[6] T. Nierhoff and S. Hirche, “Fast trajectory replanning using laplacian mesh optimization,” in *IEEE ICARCV*, 2012.

[7] O. Sorkine and M. Alexa, “As-rigid-as-possible surface modeling,” in *EUROGRAPHICS*, 2007, pp. 109–116.

[8] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, “Mesh editing with poisson-based gradient field manipulation,” in *ACM SIGGRAPH*, 2004, pp. 644–651.

[9] L. Kobbelt, T. Bareuther, and H.-P. Seidel, “Multiresolution shape deformations for meshes with dynamic vertex connectivity,” *Comput. Graph. Forum*, vol. 19, no. 3, 2000.

[10] 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.

[11] R. Diankov, “Automated construction of robotic manipulation programs,” Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, 2010.

[12] K. Yamane and Y. Nakamura, “Natural motion animation through constraining and deconstraining at will,” *IEEE TVCG*, vol. 9, pp. 352–360, 2003.