# Sampling-based Trajectory Imitation in Constrained Environments using Laplacian-RRT*

Thomas Nierhoff, Sandra Hirche, and Yoshihiko Nakamura

***Abstract*— This paper presents an incremental sampling-based approach for trajectory imitation in cluttered environments using the RRT* algorithm. Inspired by the discrete Laplace-Beltrami operator the underlying distance metric is based upon the difference from a reference trajectory through a quadratic distance term incorporating velocity and acceleration deviations along the trajectory. Mathematically-backed approximations in combination with a task-space bias make it possible to use standard nearest neighbor methods in task space when expanding the RRT*-tree. It is shown that metric-consistent biases considerably increase the convergence speed. The proposed approach is validated in simulations in a 2D environment and in experiments using a HRP-4 humanoid robot.**

## I. Introduction

With continuously increasing computing power sampling-based motion planning has become very popular over the last two decades. This is due to various reasons: Sampling-based planners are generally easy to implement, can handle multiple types of constraints and converge quickly to a feasible solution. Probably the most prominent planners are Probabilistic Roadmaps (PRM) [1], Rapidly Exploring Random Trees (RRT) [2] and a modification of RRT called RRT* [3]. All three algorithms can be seen rather as a framework than a specific realization, allowing one to modify various core components. As such they have been applied to different robotic problems, including but not limited to kinematic motion planning [4] focusing on shortest path solutions, kinodynamic motion planning [5], [6] mainly interested in minimal-time trajectories, nonholonomic path planning [7] and guaranteed stability for nonlinear systems [8].

An open problem for sampling-based planners is the adaption of a trajectory to a similar one in terms of velocity/acceleration in the presence of obstacles. Approaches tackling similar kind of problems are found in the field of imitation learning and inverse optimal control. Yet velocity and acceleration similarity is only indirectly addressed by the underlying dynamical system [9], [10] or by the deviation from a reference path [11], [12] together with robot kinematics. Obstacles along the way are either avoided through a reactive control approach using Potential Fields [10] or through a sampling-based planner [12] based upon a distance metric measuring the absolute distance to a reference trajectory. An approach able to incorporate the deviation from a reference path in terms of finite differences is presented in [13]. Based on the discrete Laplace-Beltrami operator originating from differential geometry [14], Laplacian Trajectory Editing (LTE) minimizes the curvature deviation from a reference path through a least squares approach. Still it can only be applied to the unconstrained case without obstacles along the path.

The contribution of this paper is the combination of LTE and RRT* - called Laplacian-RRT* - for finding similar-shaped trajectories in constrained environments. The similarity measure based upon the velocity/acceleration deviation and minimized by LTE serves as the distance metric for the RRT* algorithm. Due to the high dimensionality of the configuration space, the Laplacian-RRT* algorithm operates in task space. Special focus is drawn on reasonable simplifications allowing one to use standard nearest neighbor methods in task space, thus avoiding a brute-force approach or heuristic approximations. Simulations investigate the convergence properties, imitation quality and computational complexity of the method. A experiment involving a HRP-4 humanoid platform shows the applicability of the approach to a real-life scenario and the comparison with potential field methods.

The paper is organized as follows: Sec. III describes the basics of LTE and RRT* which are combined in Sec. IV and evaluated in Sec. V. Sec. VI and Sec. VII discuss the presented approach and present ideas for future expansion.

**Notation:** Throughout the paper scalars are written in non-bold letters (e.g $a$), vectors in bold lower case letters (e.g. $\mathbf{a}$) and matrices in bold capital letters (e.g. $\mathbf{A}$). Accessing a specific element of a matrix/vector is denoted by curly subscript brackets in a Matlab-like notation (e.g. $\mathbf{A}_{\{3,:\}}$ for the entire third row of $\mathbf{A}$).

## II. Problem Statement and Conceptual Approach

This paper looks at the problem of how to find a trajectory in a constrained environment with similar local trajectory properties as a reference trajectory. We define local trajectory properties as the velocity $\boldsymbol{\gamma}$ and acceleration $\boldsymbol{\delta}$ along the trajectory. For a trajectory consisting of $n$ sampling points the cost function to be minimized then takes the form

$$\bar{E} = \sum_{i=1}^{n} (\boldsymbol{\gamma}_{i,s} - \boldsymbol{\gamma}_i)^2 + \sum_{k=1}^{n} (\boldsymbol{\delta}_{i,s} - \boldsymbol{\delta}_i)^2, \qquad (1)$$

that is the summed squared difference of velocity/acceleration between a reference trajectory and the new trajectory (subscript $_s$). In order to find optimal trajectories in the presence of obstacles, the idea is to use

Thomas Nierhoff and Sandra Hirche are with the Institute of Automatic Control Engineering (LSR), Faculty of Electrical Engineering, Technische Universität München, D-80290 München, Germany {tn, hirche}@tum.de.
Thomas Nierhoff and Yoshihiko Nakamura are with the Department of Mechano-Informatics, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, 113-8656 Tokyo, Japan nakamura@ynl.t.u-tokyo.ac.jp.

the cost function $\bar{E}$ as the underlying distance metric for the RRT* algorithm. Whereas in theory any distance metric can be used, the one in (1) is special as it can be minimized at relatively low computational cost for the unconstrained case using LTE.

## III. Background of LTE and RRT*

This section both briefly reviews LTE and RRT* and presents the extension of the original LTE method for being combined with RRT*.

### *A. Laplacian Trajectory Editing*

Let a trajectory be the combination of a path, described by an ordered set of sampling points $\mathbf{P} = [\mathbf{p}(t_1), \mathbf{p}(t_2), \ldots, \mathbf{p}(t_n)]^T \in \mathbb{R}^{n\times m}$ in task space and associated temporal information $T = [t_1, t_2, \ldots, t_n] \in \mathbb{R}^n$ represented as time $t_i \in \mathbb{R}$, $\mathbf{p}(t_i) \in \mathbb{R}^m$. For simplicity, $\mathbf{P} = [\mathbf{p}(t_1), \mathbf{p}(t_2), \ldots, \mathbf{p}(t_n)]^T$ is rewritten as $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n]^T$.

In case the sampling points are spaced equitemporally as $t_{i+1} - t_i = \Delta t \;\; \forall i \in \{1, \ldots, n-1\}$, the acceleration along the trajectory for the $i$-th sampling point can be described by the finite difference

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

The same accounts for the velocity along the trajectory, described by

$$\frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{\Delta t}. \tag{3}$$

The key idea of LTE is to calculate local trajectory properties, resulting in a linear equation system. When adding boundary constraints, the resulting overdetermined equation system 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 only a special case of the *Laplacian coordinates* in [15] for equitemporally spaced sampling points. With weighting factor $\omega_{i,2}$ instead of $\frac{1}{\Delta t^2}$, (2) can be rewritten as

$$\boldsymbol{\delta}_i = \omega_{i,2}(\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1}) = \omega_{i,2}\mathbf{c}_{i,2}, \tag{4}$$

In a similar manner (3) becomes

$$\boldsymbol{\gamma}_i = \omega_{i,1}(\mathbf{p}_i - \mathbf{p}_{i-1}) = \omega_{i,1}\mathbf{c}_{i,1}, \tag{5}$$

with weighting factor $\omega_{i,1}$. For the remainder of the paper we only consider positional constraints in the form

$$\omega_{i,0}\mathbf{p}_i = \omega_{i,0}\mathbf{c}_{i,0}. \tag{6}$$

as additional constraints, fixing the sampling point $\mathbf{p}_i$ to its desired position $\mathbf{c}_{i,0}$. Again, the constraint is weighted with a scalar $\omega_{i,0}$.

Writing everything in matrix form, one obtains

$$\begin{pmatrix} \bar{\mathbf{P}}_0 \\ \bar{\mathbf{P}}_1 \\ \bar{\mathbf{P}}_2 \end{pmatrix} \mathbf{P}_s = \begin{pmatrix} \mathbf{C}_0 \\ \mathbf{C}_1 \\ \mathbf{C}_2 \end{pmatrix}, \tag{7}$$

with

$$\begin{pmatrix} \bar{\mathbf{P}}_0 \\ \bar{\mathbf{P}}_1 \\ \bar{\mathbf{P}}_2 \end{pmatrix} = \left(\begin{array}{ccccc} \omega_{1,0} & & & & \\ & \ddots & & & \\ & & \omega_{l,0} & & \\ & & & & \omega_{n,0} \\ \hline \omega_{2,1} & -\omega_{2,1} & & & \\ & & \ddots & & \\ & & & \omega_{n,1} & -\omega_{n,1} \\ \hline \omega_{2,2} & -2\omega_{2,2} & \omega_{2,2} & & \\ & & \ddots & & \\ & & \omega_{n-1,2} & -2\omega_{n-1,2} & \omega_{n-1,2} \end{array}\right), \tag{8}$$

and

$$\begin{pmatrix} \mathbf{C}_0 \\ \mathbf{C}_1 \\ \mathbf{C}_2 \end{pmatrix} = \left(\begin{array}{c} \omega_{1,0}\mathbf{c}_{1,0} \\ \vdots \\ \omega_{l,0}\mathbf{c}_{l,0} \\ \omega_{n,0}\mathbf{c}_{n,0} \\ \hline \omega_{1,1}\mathbf{c}_{1,1} \\ \vdots \\ \omega_{n,1}\mathbf{c}_{n,1} \\ \hline \omega_{1,2}\mathbf{c}_{1,2} \\ \vdots \\ \omega_{n,2}\mathbf{c}_{n,2} \end{array}\right). \tag{9}$$

The different terms in (7)-(9) account for weighted velocity ($\bar{\mathbf{P}}_1$,$\mathbf{C}_1$) and acceleration ($\bar{\mathbf{P}}_2$,$\mathbf{C}_2$) along the reference trajectory $\mathbf{P}$. The terms $\mathbf{P}_0$ and $\mathbf{C}_0$ specify positional constraints for the first $l$ sampling points and the last sampling point. The deformed trajectory $\mathbf{P}_s = [\mathbf{p}_{1,s}, \mathbf{p}_{2,s}, \ldots, \mathbf{p}_{n,s}]^T \in \mathbb{R}^{n\times m}$ can then be calculated using least squares. As the least-squares solution minimizes the cost term $\hat{E}$ defined as

$$\hat{E} = \sum_{k=1}^{m} \left\| \begin{pmatrix} \bar{\mathbf{P}}_0 \\ \bar{\mathbf{P}}_1 \\ \bar{\mathbf{P}}_2 \end{pmatrix} \mathbf{P}_{s\{:,k\}} - \begin{pmatrix} \mathbf{C}_0 \\ \mathbf{C}_1 \\ \mathbf{C}_2 \end{pmatrix}_{\{:,k\}} \right\|_2^2, \tag{10}$$

the term $\hat{E}$ is a natural choice for a similarity measure. Under the assumption of $\omega_{i,0} \gg \omega_{i,1}, \omega_{i,2}$ the unweighted positional offset $\|\mathbf{p}_{i,s} - \mathbf{c}_{i,0}\|_2^2$ is negligible. Then an approximate measure is given as

$$\hat{E} \;\approx\; E \;=\; \sum_{k=1}^{m} \left\| \begin{pmatrix} \bar{\mathbf{P}}_1 \\ \bar{\mathbf{P}}_2 \end{pmatrix} \mathbf{P}_{s\{:,k\}} - \begin{pmatrix} \mathbf{C}_1 \\ \mathbf{C}_2 \end{pmatrix}_{\{:,k\}} \right\|_2^2 \tag{11}$$
$$, \qquad \text{s.t. } \mathbf{p}_{i,s} = \mathbf{c}_{i,0} \;\; \forall i = \{1, \ldots, l, n\},$$

In case the elements of $[\mathbf{C}_1^T \mathbf{C}_2^T]^T$ correspond to the undeformed trajectory $\mathbf{P}$ as $\boldsymbol{\gamma}_i$, $\boldsymbol{\delta}_i$ and the elements in $[\bar{\mathbf{P}}_1^T \bar{\mathbf{P}}_2^T]^T \mathbf{P}_s$ to the deformed trajectory $\mathbf{P}_s$ as $\boldsymbol{\gamma}_{i,s}$, $\boldsymbol{\delta}_{i,s}$, the similarity between the two trajectories is rewritten as

$$\begin{aligned} E \;=\; & \sum_{k=1}^{m}\sum_{i=2}^{n} (\boldsymbol{\gamma}_{i,s\{k\}} - \boldsymbol{\gamma}_{i\{k\}})^2 \qquad (12) \\ + \; & \sum_{k=1}^{m}\sum_{i=2}^{n-1} (\boldsymbol{\delta}_{i,s\{k\}} - \boldsymbol{\delta}_{i\{k\}})^2, \\ & \text{s.t. } \mathbf{p}_{i,s} = \mathbf{c}_{i,0} \;\; \forall i = \{1, \ldots, l, n\}. \end{aligned}$$

Note that the term similarity is slightly misleading as a high similarity between two trajectories corresponds to low $\hat{E}, \bar{E}$ values. In addition, the weighting terms $\omega_{i,0}, \omega_{i,1}, \omega_{i,2}$ can be chosen arbitrarily.

### *B. RRT**

A drawback of LTE is that it can modify trajectories only for the unconstrained case and at once, i.e. non-incrementally. In contrast, RRT* makes it possible to find optimal trajectories in an incremental way for the constrained case with guaranteed asymptotic optimality. Its pseudocode (in black letters) is given as follows:

**Algorithm 1:** Laplacian-RRT* $(V, E, N)$

```
for $f = 1, \dots, N$ do
    $x_{rand}, id_{rand}$ ← Sample;
    $x_{nearest}$ ← Nearest($V, x_{rand}, id_{rand}$);
    $x_{new}$ ← Approach($x_{nearest}, x_{rand}$);
    if CollisionFree($x_{nearest}, x_{new}$) then
        $X_{near}$ ← Near($V, x_{new}$);
        $X_{near} \leftarrow V$;
        $(V, E)$ ← ParentId($V, E, X_{near}, x_{new}, id_{rand}$);
        $(V, E)$ ← RewireId($V, E, X_{near}, x_{new}, id_{rand}$);
return $G = (V, E)$
```

**Algorithm 2:** ParentId$(V, E, X_{near}, x_{new}, id_{rand})$

```
$V \leftarrow V \cup x_{new}$;
minCost ← $\infty$; $x_{min}$ ← NULL; $\sigma_{min}$ ← NULL;
for $x_{near} \in X_{near}$ do
    if $id(x_{near}) = id_{rand} - 1$ then
        $\sigma$ ← Steer($x_{near}, x_{new}$);
        if Cost($x_{near}$) + Cost($\sigma$) < minCost and
        CollisionFree($\sigma$) then
            minCost ← Cost($x_{near}$) + Cost($\sigma$);
            $x_{min} \leftarrow x_{near}$; $\sigma_{min} \leftarrow \sigma$;
$E \leftarrow E \cup \{x_{near}, x_{new}\}$;
return $(V, E)$
```

**Algorithm 3:** Rewire$(V, E, X_{near}, x_{new})$

```
for $x_{near} \in X_{near}$ do
    $\sigma$ ← Steer($x_{new}, x_{near}$);
    if Cost($x_{new}$) + Cost($\sigma$) < Cost($x_{near}$) and
    CollisionFree($\sigma$) then
        $x_{parent}$ ← Parent($x_{near}$);
        $E \leftarrow E \setminus \{x_{parent}, x_{near}\}$;
        $E \leftarrow E \cup \{x_{new}, x_{near}\}$;
return $(V, E)$
```

Its core components are:

- Cost: Optimal cost of a given state based on the underlying distance metric $dist(x, x')$ between two states $x$ and $x'$ in the configuration space $C$.
- Near/Nearest: Returns a set of states $X \in V$ resp. the nearest state $x_{nearest}$ from the vertices $V$ of the graph $(V, E)$ within a sphere of fixed radius centered at the state $x$ and based on the distance metric $dist(x, x')$.
- Steer: Gives a path $\sigma$ connecting two states $x$ and $x'$.
- Approach: Approaches the desired position $x'$ from $x$ by a predefined amount $\alpha$.
- CollisionFree: Checks if the path $\sigma$ between two states $x$ and $x'$ lies entirely in the free (i.e. non-colliding) configuration space $C_{free} \subseteq C$.
- Sample: Independent uniformly random sampling of $C_{free}$.

**Algorithm 4:** RewireId$(V, E, X_{near}, x_{new}, id_{rand})$

```
for $x_{near} \in X_{near}$ do
    if $id(x_{near}) = id_{rand} + 1$ or $id(x_{near}) = id_{rand} + 2$
    then
        $i = id_{near}$;
        Check any possible combination $\{\mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i\}$
        for lower cost/collision and rewire when necessary
return $(V, E)$
```

When expanding the graph $(V, E)$ with edges $E$ the algorithm first samples a new state $x_{rand}$ from the free configuration space and approaches it from the closest node $x_{nearest}$, resulting in $x_{new}$. In case the path between $x_{nearest}$ and $x_{new}$ is non-colliding, a parent node $x_{par}$ is selected from the set of nearest nodes $X_{near}$ as described in the Parent function such that it connects to $x_{new}$ with minimal cost. Having added $x_{new}$ and the edge between $x_{par}$ and $x_{new}$ to the graph, the Rewire function checks whether a node $x_{near} \in X_{near}$ can be connected via $x_{new}$ with less than its current cost and rewires the graph accordingly.

## IV. Combining Laplacian Trajectory Editing and RRT*

Both RRT* and LTE have different advantages. LTE is faster than RRT* and able to minimize the cost function in (11)-(12) in a computationally efficient way. RRT* on the other hand also works in the presence of obstacles. This section explains how to combine LTE with RRT* to combine the advantages of both algorithms. It also presents several biases/approximations for faster convergence and reduced computational complexity.

### *A. Modifications to RRT**

A key concept of RRT* is the underlying distance metric $dist(x, x')$ between two states $x$ and $x'$. Differing from other approaches, the used distance metric for equitemporally spaced sampling points is inspired by (12) as

$$\begin{aligned} dist(x, x') &= dist(\mathbf{p}_i, \mathbf{p}_{i,s}), \\ &= \sum_{k=1}^{m} (\gamma_{i,s\{k\}} - \gamma_{i\{k\}})^2 \\ &+ \sum_{k=1}^{m} (\boldsymbol{\delta}_{i-1,s\{k\}} - \boldsymbol{\delta}_{i-1\{k\}})^2. \end{aligned} \quad (13)$$

that is the weighted difference of velocity and acceleration between two points in configuration space. The index $i - 1$

for the acceleration vector is intentional due to the one-sided incremental nature of RRT*, making it necessary to use backward finite differences. As the acceleration vectors are calculated from three sampling points according to (3), each state $x \in C$ of the configuration space consists of three subsequent sampling points $\{\mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i\}$. In order to reduce the dimensionality of the problem from $\mathbf{R}^{3m}$ to $\mathbf{R}^m$ and simplify calculations, the Laplacian-RRT* algorithm operates in task space using an adequate bias [5], [16]. Thus every velocity/acceleration vector is not associated with a state $x, x'$ in configuration space but with an individual sampling point $\mathbf{p}_i, \mathbf{p}_{i,s}$ in task space. The distance metric is only defined for sampling points with similar index $i$. Implicitly it is assumed that $\mathbf{P}$ and $\mathbf{P}_s$ have the same number of sampling points. Because the sampling points are indexed, the original RRT* algorithm has to be modified, ensuring that connected nodes of the graph correspond to subsequent sampling points of the trajectory.

The modifications are described in Alg. 1-4 (in red letters): Due to indexed sampling points, the `Sample` function outputs both, a sample position and an index $id_{rand} \in \{1, 2, \ldots, n\}$. Because a distance $dist(x, x') = 0$ does not necessarily correspond to sampling points close together in task space, $X_{near}$ now consists of all vertices $V$, resulting in a brute-force approach. Differing from the `Parent` function, the `ParentId` function also checks the index of the parent node. The mutual dependence of subsequent acceleration vectors along the same trajectory causes the `RewireId` function to check every possible combination $\{\mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i\} \mid i \in \{id_{rand}+1, id_{rand}+2\}$ for better cost/collision and rewire in case.

## B. Introducing Bias For Faster Convergence

The algorithm in Sec. IV-A is both slow and slowly converging because the set $X_{near}$ for the nearest neighbor search consists of all vertices in $V$. For faster convergence this subsection presents a task space bias [5], [16] consistent with the used distance metric and a nearest neighbor approximation, allowing one to use nearest neighbor methods in task space for finding the closest node in configuration space.

*1) Task Space Bias:* When operating in Task Space instead of Configuration Space, a suitable bias is essential for the RRT* algorithm to converge quickly. Yet a common problem when using complex distance metrics and/or unintuitive configuration spaces is the difficulty of finding a proper goal-driving heuristic. The Laplacian Trajectory Editing framework has the invaluable advantage of providing a very good approximation of the optimal next node with respect to (13). In general when expanding the RRT*-tree, one is given a branch $\mathbf{P}_b = [\mathbf{p}_{1,b}, \mathbf{p}_{2,b}, \ldots, \mathbf{p}_{l,b}]^T$ of the tree, an original trajectory $\mathbf{P}$ that has to be resembled as good as possible and a known start/end sampling point $\mathbf{p}_s = \mathbf{p}_{1,b}/\mathbf{p}_e$. When expanding the branch with another sampling point with index $l+1$, its optimal position $\hat{\mathbf{p}}_{l+1,b}$ is calculated as

$$\mathbf{P}_s = \begin{pmatrix} \bar{\mathbf{P}}_0 \\ \bar{\mathbf{P}}_1 \\ \bar{\mathbf{P}}_2 \end{pmatrix}^+ \begin{pmatrix} \mathbf{C}_0 \\ \mathbf{C}_1 \\ \mathbf{C}_2 \end{pmatrix}, \quad (14)$$

$$\hat{\mathbf{p}}_{l+1,b} = \mathbf{P}_{s\{l+1,:\}}, \quad (15)$$

$$\text{with} \quad \mathbf{C}_0 = \begin{pmatrix} \mathbf{P}_b \\ \mathbf{p}_e \end{pmatrix}, \quad (16)$$

$$\mathbf{C}_1 = \bar{\mathbf{P}}_1 \mathbf{P}, \quad (17)$$

$$\mathbf{C}_2 = \bar{\mathbf{P}}_2 \mathbf{P}. \quad (18)$$

The resulting vector $\hat{\mathbf{p}}_{l+1,b} - \mathbf{p}_{l,b}$ is used as an offset for the `Approach` function in order to bias the tree growth towards the goal position while being consistent with the used distance metric, see Fig. 1.

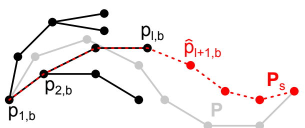

Fig. 1. Task Space Bias. Original trajectory $\mathbf{P}$ (grey), Laplacian-RRT*-tree (black) and calculated optimal trajectory $\mathbf{P}_s$ for a single branch of the tree (red) with corresponding optimal next sampling point $\hat{\mathbf{p}}_{l+1,b}$

*2) Task Space Nearest Neigbors:* Because the Laplacian-RRT* algorithm operates in task space, it relies on a nearest neighbor search in task space. Otherwise - as shown in Sec. IV-A - the set $X_{near}$ has to consist of all vertices $V$ for finding the optimal node. Looking at (11), it is visible that the similarity measure $\bar{E}$ depends quadratically on the elements of $\mathbf{P}_s$. It is assumed that the next sampling point's position $\mathbf{p}_{l+1,b}$ of the branch $\mathbf{P}_b$ is split up as $\mathbf{p}_{l+1,b} = \hat{\mathbf{p}}_{l+1,b} + \Delta\mathbf{p}_{l+1,b}$ such that

$$\hat{\mathbf{p}}_{l+1,b} = \underset{\mathbf{p}_{l+1,b}}{\text{argmin}}(E), \quad (19)$$

i.e. the position $\hat{\mathbf{p}}_{l+1,b}$ leads to maximal similarity. Then the similarity measure $\bar{E}$ depends quadratically on $\Delta\mathbf{p}_{l+1,b}$.

Given different branches $\{\mathbf{P}_{b1}, \ldots, \mathbf{P}_{bm}\}$ of the tree with corresponding next sampling points $\{\mathbf{p}_{l+1,b1}, \ldots, \mathbf{p}_{l+1,bm}\}$ having approximately the same shape due to a task space bias and thus leading to approximately the same similarity as

$$\min_{\mathbf{P}_{bi}, \mathbf{p}_{l+1,bi}} (E) \approx const. \ \forall\, i \in \{1, \ldots, m\}, \quad (20)$$

the similarity $\bar{E}$ is mainly influenced by the offset $\Delta\mathbf{p}_{l+1,bi}$, allowing one to search for minimal $\Delta\mathbf{p}_{l+1,bi}$ in task space using standard nearest neighbor methods (Fig. 2).

Hence when given a random sampling point $x_{rand} = \mathbf{p}_{rand}$ with index $id_{rand} = l+1$, calculating the nearest neighbor $x_{nearest} = \mathbf{p}_{nearest}$ simplifies as

$$\mathbf{p}_{nearest} = \underset{\mathbf{p}_{l+1,bi}}{\text{argmin}} \|\mathbf{p}_{rand} - \hat{\mathbf{p}}_{l+1,bi}\|_F, \quad (21)$$

that is the Frobenius norm between every optimal next sampling point $\hat{\mathbf{p}}_{l+1,bi}$ and the random sampling point $\mathbf{p}_{rand}$.

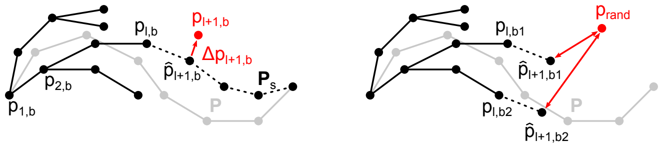

Fig. 2. Task Space Nearest Neighbors. Left side: Deviation of the next sampling point $\mathbf{p}_{l+1,b}$ from its optimal position $\hat{\mathbf{p}}_{l+1,b}$ by $\Delta\mathbf{p}_{l+1,b}$. Right side: Approximate nearest neighbor calculation based on the distance between optimal sampling points $\hat{\mathbf{p}}_{l+1,bi}$ and a random sampling point $\mathbf{p}_{rand}$

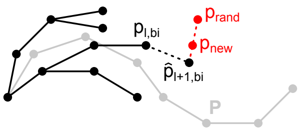

Fig. 4. Growing Exploration Ratio. Due to the task space bias the new sampling point $\mathbf{p}_{new}$ is located along the line between $\hat{\mathbf{p}}_{l+1,bi}$ and $\mathbf{p}_{rand}$ and not along the line between $\mathbf{p}_{l,bi}$ and $\mathbf{p}_{rand}$

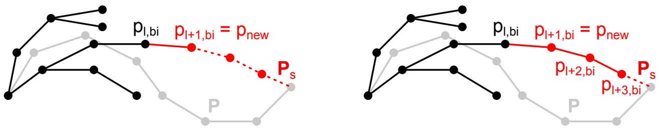

Fig. 5. Multiple Sampling Point Expansion. Left side: Conventional approach extending only one node. Right side: Accelerated approach improving multiple nodes ($\sigma = 3$ for the given illustration)

When adding a new sampling point $\hat{\mathbf{p}}_{l+1,bi}$ with possibly suboptimal position $\mathbf{p}_{l+1,bi} \neq \hat{\mathbf{p}}_{l+1,bi}$ to the branch $\mathbf{P}_{bi}$, it automatically becomes the last sampling and the index $l$ increases by one. By defining the sensitivity $s_{l,n}$ of $\bar{E}$ depending on $\Delta\mathbf{p}_{l,bi}$ for a trajectory with $n$ sampling points as

$$s_{l,n} = \left.\frac{\partial E^2}{\partial^2 \Delta\mathbf{p}_{l,bi}}\right|_{\Delta\mathbf{p}_{l,bi}=0}, \tag{22}$$

one sees in Fig. 3 that the sensitivity is independent of the choice of $l$ and $n$ over a wide range of $l$ and $n$. This gives rise to the simplification of using the nearest neighbor search regardless of each node's index for the `Nearest` function.

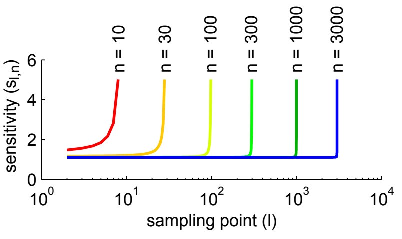

Fig. 3. Sensitivity $s_{l,n}$ for multiple trajectories with different number of sampling points $n$, evaluated over varied sampling point indices $l$. The used parameterization is $\omega_i^0 = 1e6$, $\omega_i^1 = 0.1$ $\omega_i^2 = 1$

*3) Growing Exploration Ratio:* As described in Sec. III-B, the `Approach` function of the Laplacian-RRT* algorithm depends on a variable $\alpha$ specifying the amount of exploration. It has been found practical to increase the variable $\alpha$ with increasing number of iteration $f$, see Fig. 4. The used heuristic for the `Approach` function with $\mathbf{p}_{l,bi}$ as the nearest node and $\hat{\mathbf{p}}_{l+1,bi}$ as its optimal next node is

$$\gamma = \begin{cases} \alpha\, f^{\beta} & \text{if } \alpha\, f^{\beta} \leq 1, \\ 1 & \text{else,} \end{cases} \tag{23}$$

$$\mathbf{p}_{new} = \underbrace{\hat{\mathbf{p}}_{l+1,bi}}_{\text{task space bias}} + \underbrace{\gamma(\mathbf{p}_{rand} - \hat{\mathbf{p}}_{l+1,bi})}_{\text{growing exploration ratio}}. \tag{24}$$

*4) Multiple Sampling Point Expansion:* To increase the amount of exploration per iteration and therefore reduce the number of computationally expensive nearest neighbor/task space bias calculations, the branch $\mathbf{P}_{bi}$ can be expanded with multiple sampling points $\mathbf{p}_{l+1,bi}, \ldots, \mathbf{p}_{l+\sigma,bi}$ in every iterations based on the task space bias trajectory $\mathbf{P}_s$. Fig. 5 illustrates the idea.

*5) Reduced Rewiring:* For large Laplacian-RRT*-trees with possibly ten thousands of nodes, checking every combination $\{\mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i\}$ of three sampling points with subsequent indices as described in Alg. 4 becomes quickly unfeasible: It is known that one of the three sampling points has to be $x_{new} = \mathbf{p}_{new}$. Under the assumption of $n_t$ sampling points in the tree and all sampling point indices being identically distributed over the range $\{1, \ldots, n\}$, the amount of checks $c$ per iteration is approximately

$$c = 3\left(\frac{n_t}{n}\right)^2, \tag{25}$$

that is $O\left(\left(\frac{n_t}{n}\right)^2\right)$ complexity. Following the same argumentation as in Sec. IV-B.2, when decomposing the three sampling points as

$$\mathbf{p}_q = \hat{\mathbf{p}}_q + \Delta\mathbf{p}_q, \quad q = \{i, i-1, i-2\}, \tag{26}$$

the similarity $\bar{E}$ is mainly influenced by $\Delta\mathbf{p}_q$, that is the deviation from the optimal position $\hat{\mathbf{p}}_q$. As one of the three sampling points is always $x_{new} = \mathbf{p}_{new}$, a good approximate solution is obtained by finding the nearest neighbors of $\mathbf{p}_{new}$ when looking for candidate sampling points $\{\mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i\}$. In addition, the heuristic of just rewiring candidate sampling points with $id(x_{near}) = id_{rand} + 1$ is used in order to reduce complexity to $O\left(\frac{n_t}{n}\right)$.

## V. Experimental Evaluation

Experiments are conducted in two ways, through computational simulations and using a HRP-4 humanoid robotic platform.

### A. Simulations

Simulations using a 2D example illustrate the principle procedure when expanding the Laplacian-RRT*-tree. Shown in Fig. 6 are two versions of Laplacian-RRT*, the unbiased version according to Sec. IV-A on the left side and the biased version as described in Sec. IV-B on the right side. The task consists in finding a trajectory not colliding with obstacles (in red) and resembling the reference trajectory (in orange) as good as possible in terms of (11). For all simulations the parameters have been set to $\alpha = 2e-4$, $\beta = 0.5$, $\sigma = 3$, $\omega_{i,0} = 1e6$, $\omega_{i,1} = 0.1$ $\omega_{i,2} = 1$. To reduce calculation time, the reduced rewiring technique in Sec. IV-B.5 is also used for the unbiased version and only 5000 iterations are run (10000 for all other versions). Clearly the unbiased version fails to come up with good results whereas the optimal trajectory of the biased version resembles the reference trajectory well.

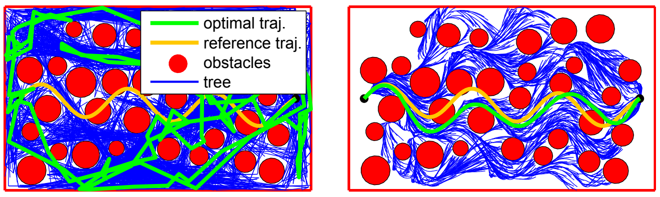

Fig. 6. Spatial comparison of different Laplacian-RRT* approaches without bias (left) and with bias (right)

Fig. 7 compares the cost $\bar{E}$ of the unbiased and biased version over 30 rollouts. Costs are only plotted after a feasible solution connecting the first and last sampling point has been found. It is visible that the cost of the biased version is about three magnitudes smaller than the unbiased one. In addition the cost graphs of the biased version stay almost constant: Caused by the growing exploration ratio in combination with the task space bias, near-optimal solutions are found rather quickly at the cost of a reduced exploration ratio.

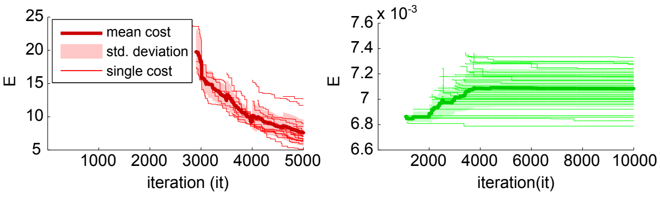

Fig. 7. Comparison of the cost $\bar{E}$ of different Laplacian-RRT* approaches without bias (left) and with bias (right)

Shown in Fig. 8 are computation times based on 30 rollouts. The algorithm is implemented in Matlab R2013b (single-threaded) on a i7-3635QM/8GB notebook using Matlab's `knnsearch` nearest neighbor search method. The x-axis shows the computation time at intermediate steps of the algorithm after every 1000-th iteration. The total processing time (grey) with corresponding standard deviation (black bars) is split up both into algorithms (`ParentId`/`RewireId`) and functions (LTE/collision detection/nearest neighbor search) causing the main computational load. In addition, the number of tree nodes and its standard deviation is displayed in blue. It is visible that most computation time is spent for the nearest neighbor search and collision detection.

It is stated in Sec. III-A that the used version of LTE minimizes the squared deviation from the reference trajectory in terms of weighted velocity and acceleration terms. Fig. 9 shows velocity and acceleration plots of both, reference and optimal trajectory. No major deviations from the reference trajectory can be observed.

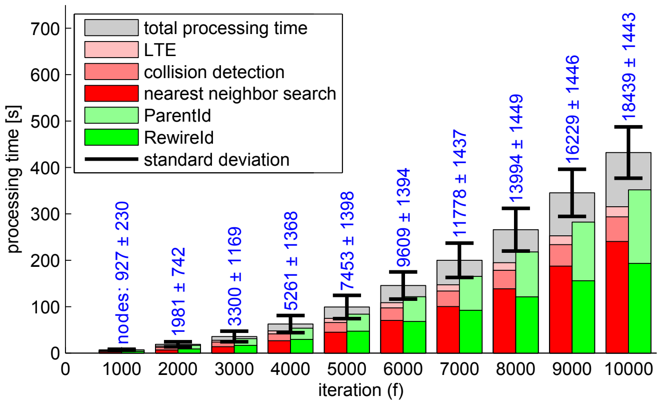

Fig. 8. Processing time evaluation

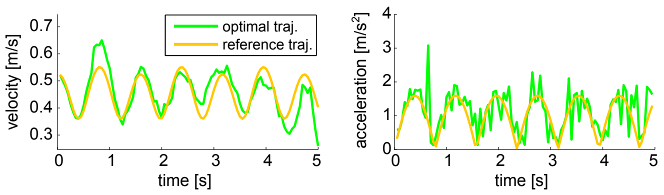

Fig. 9. Velocity and acceleration plots of the optimal trajectory of the biased Laplacian-RRT*

## B. *Experiments On Humanoid Robot HRP4*

Real-life experiments for a pick-and-place scenario use a HRP4 humanoid robot as shown in Fig. 10. For a physically feasible full-body robot movement based on the endeffector trajectory, a hierarchical inverse kinematics method as described in [17] is adopted. The task consists in placing a cup upright on the table by following a reference trajectory (orange) that has been previously recorded from a human demonstration using a Vortex motion capture system. Obstacles (a pile of books) along the reference trajectory force the robot to find a new suboptimal trajectory (green). Two approaches for obstacle avoidance are compared, the Laplacian-RRT* method and a potential field approach [18]. The used parameters for the Laplacian-RRT* algorithm are $\alpha = 1e-3$, $\beta = 0.5$, $\sigma = 10$, $\omega_i^0 = 1e6$, $\omega_i^1 = 0.1$ $\omega_i^2 = 1$. As shown in the figure, the similarity $\bar{E}$ of the Laplacian-RRT* based trajectory with respect to the reference trajectory is about $5\times$ better (=lower) than the potential field based trajectory. Shown in Fig. 11 are velocity and acceleration plots for the optimal trajectory found by Laplacian-RRT*, the resulting potential field trajectory and the reference trajectory. Whereas the potential field trajectory has large velocity/acceleration errors for $t > 2.5s$, there are only small errors for the Laplacian-RRT* trajectory over the entire timespan due to its stochastic nature.

## VI. Discussion

Differing from all known sampling-based approaches, this is the first method defining optimality not based on an extrinsic measure like "shortest path" or "minimum time"

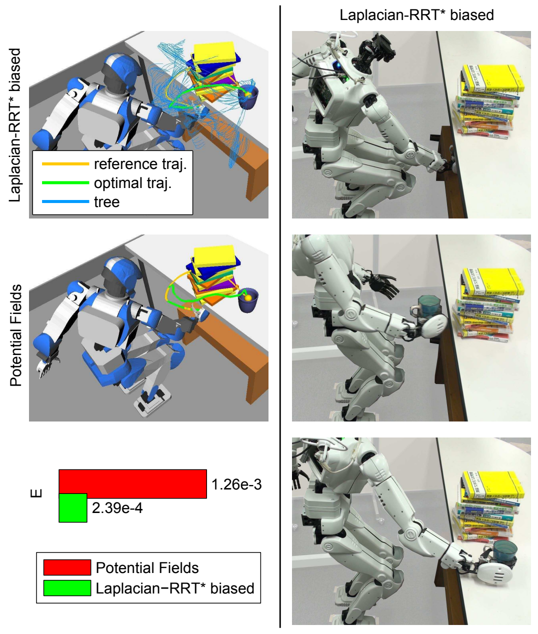

Fig. 10. Simulation results (right) and conducted experiment (left) using HRP4. A bar graph shows the similarity measure $\bar{E}$ of the two compared methods

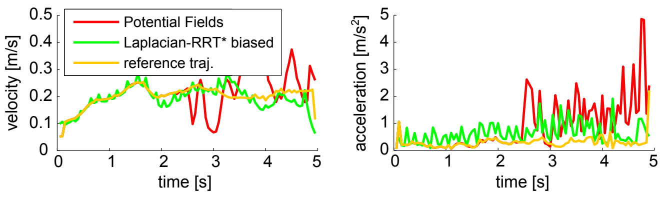

Fig. 11. Velocity (left) and acceleration plot (right) of the compared methods

but with respect to another trajectory, making it well suited for motion imitation tasks.

The method suffers from a few drawbacks: The used heuristics for biasing exploration towards the goal node make the problem computationally tractable but reduce the amount of exploration. As a least-squares problem has to be solved in every iteration step, the straightforward approach is only applicable for small trajectories, e.g. $n < 200$. Otherwise multiresolution techniques as described in [13] are better suited. In addition, the distance metric is only well defined for trajectories with similar number of sampling points.

The approach can be readily extended in a number of ways: So far, only first and second order derivatives are considered. Yet higher-order derivatives can be added in a similar manner. Another option are via points that have to be passed on the way to the goal position. By adding an additional positional constraint to the matrix $\bar{\mathbf{P}}_0$, they can be incorporated.

## VII. Conclusion and Future Work

This paper presents Laplacian-RRT* as a novel tool for finding similar-shaped trajectories in a constrained environment using a sampling-based approach. The similarity measure is based upon the local trajectory properties as velocity and acceleration along the trajectory. Mathematically derived approximations allow one to perform a standard nearest neighbor search in task space when looking for the closest node of the tree to be expanded, thus increasing speed considerably.

Future work will be focused on combining Laplacian-RRT* with full-body movement schemes for humanoid robots in order to derive an optimal controller for dynamical movements in partially blocked environments.

## Acknowledgement

This work is supported in part within the DFG excellence research cluster *Cognition for Technical Systems - CoTeSys* (`www.cotesys.org`) and by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (S), 2008-2012, 20220001, "Establishing Human-Machine Communication through Kinesiology and Linguistics Integration" (PI: Y. Nakamura)

## References

[1] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," Tech. Rep., 1994.

[2] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," No. 98-11, Tech. Rep., 1998.

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *IJRR*, vol. 30, no. 7, pp. 846–894, 2011.

[4] L. Jaillet and J. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *TRO*, vol. 29, no. 1, pp. 105–117, 2013.

[5] J. Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *CDC*, 2011, pp. 3276–3282.

[6] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints," *CoRR*, vol. abs/1205.5088, 2012.

[7] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *ICRA*, 2013, pp. 5041–5047.

[8] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *IJRR*, vol. 29, no. 8, pp. 1038–1052, 2010.

[9] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Advanced Robotics*, vol. 21, no. 13, pp. 1521–1544, 2007.

[10] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields." in *Humanoids*, 2008, pp. 91–98.

[11] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard, "Imitation learning with generalized task descriptions." in *ICRA*, 2009, pp. 3968–3974.

[12] G. Ye and R. Alterovitz, "Demonstration-guided motion planning," *ISRR*, 2011.

[13] T. Nierhoff, S. Hirche, and Y. Nakamura, "Multiresolution laplacian trajectory replanning," in *Annual Conference of the RSJ*, 2013.

[14] H. Zhang, O. van Kaick, and R. Dyer, "Spectral mesh processing," *Comput. Graph. Forum*, vol. 29, no. 6, pp. 1865–1894, 2010.

[15] T. Nierhoff and S. Hirche, "Fast trajectory replanning using laplacian mesh optimization," in *ICARCV*, 2012.

[16] A. C. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space voronoi bias." in *ICRA*, 2009, pp. 2061–2067.

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

[18] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *IJRR*, vol. 21, no. 12, pp. 1031–1052, 2002.