LexTOR: Lexicographic Teach Optimize and Repeat Based on User Preferences

Mladen Mazuran Christoph Sprunk Wolfram Burgard Gian Diego Tipaldi

***Abstract*— In the last years, many researchers started to consider teach-and-repeat approaches for reliable autonomous navigation. The paradigm, in all its proposed forms, is deeply rooted in the idea that the robot should autonomously follow a route that has been demonstrated by a human during a teach phase. However, human demonstrations are often inefficient in terms of execution time or may cause premature wear of the robot components due to jittery behavior or strong accelerations. In this paper, we propose the concept of *teach, optimize and repeat*, which introduces a trajectory optimization step between the teach and repeat phases. To address this problem, we further propose LexTOR, a constrained trajectory optimization method for teach and repeat problems, where the constraints are defined according to user preferences. At its core, LexTOR optimizes both the execution time and the trajectory smoothness in a lexicographic sense. The experiments show that LexTOR is very effective, both qualitatively and quantitatively, in terms of execution time, smoothness, accuracy and bound satisfaction.**

I. Introduction

Over the last years, *teach and repeat* has proven to be a successful paradigm for the execution of repetitive tasks that do not require high-level reasoning on behalf of a robotic platform. Teach and repeat comprises two main phases. In the *teach phase*, the user manually steers the robot along a route and the system builds a representation of the environment from the sensor data it acquires. In the *repeat phase* the robot aligns its sensor data to the built representation, in order to execute the demonstrated trajectory as accurately as possible.

The main assumption of current teach-and-repeat strategies is that the quality of the demonstrated trajectory is good enough to perform the task at hand. However, in most cases, the route is demonstrated by a human operator. Introducing the human factor in the loop will, in general, result in sub-optimal trajectories, particularly in case of non-expert users. Problems may include jittery behavior in the demonstration, needlessly large accelerations, unnecessary stops or cusps in the path. This often results in either long execution times or in premature wear and tear of the robot components.

With this work, we propose a novel twist to teach and repeat and we refer to it as *teach, optimize and repeat* (TOR). Our goal is to bridge the gap between the field of teach and repeat and trajectory optimization. We envision a scenario in which the user specifies a maximum allowance in deviation from the demonstrated trajectory, e.g., the bounds shown in Fig. 1, with possibly further limitations in velocity or acceleration. Without requiring further input, the system then computes an improved trajectory that will be executed during the repeat phase.

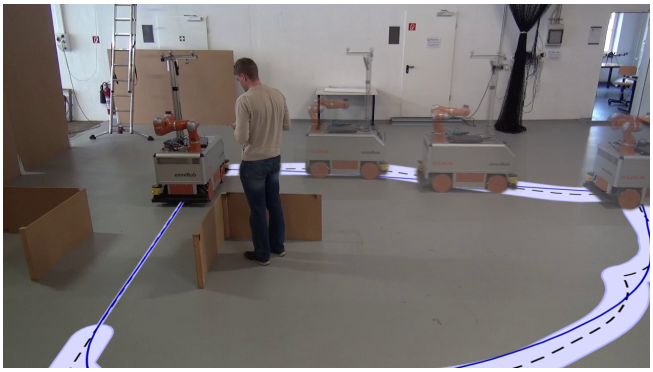

Fig. 1. View of the omniRob during the teach phase. The figure shows the taught path (dashed) and the optimized one (solid), together with the user-defined bounds on position (shaded).

Furthermore, we propose LexTOR, a novel trajectory optimization technique to jointly optimize both execution time and trajectory smoothness in a lexicographic sense. LexTOR solves the optimization problem iterating between solving two convex problems in a block coordinate descent fashion. For a second order approximation of the trajectory, we prove that the two problems reduce to second order cone programs (SOCPs), for which efficient solvers exist [3]. We extensively evaluate our approach on a holonomic robot, and show the effectiveness of LexTOR in terms of trajectory execution time, smoothness, accuracy and bound satisfaction.

II. Related work

The teach-and-repeat paradigm has been successfully applied to various scenarios, ranging from underground tramming [14], to sample-and-return missions on foreign planets [7]. Teach and repeat can be model-based [1], appearance-based [15], or a hybrid of the two [14]. Model-based approaches require a metric-map for navigation, which often proves to be a costly endeavor. Appearance-based and hybrid approaches, on the other hand, rely more on sensor data and have experienced widespread success.

Pioneering work [15] associated camera images to particular actions of the robot and later replayed them by following the sequence of recorded images. Due to the lack of metrical information from camera-only sensing, authors employed either bearing-only control laws [12] or visual servoing [6, 5]. To increase accuracy, many researchers started to employ proximity sensors. Furgale and Barfoot [7] proposed a stereo-camera approach able to handle long-range navigation in

All authors are with the University of Freiburg, Institute of Computer Science, 79110 Freiburg, Germany. This work has partly been supported by the European Commission under ERC-AG-PE7-267686-LIFENAV, FP7-610603-EUROPA2, and H2020-645403-ROBDREAM.

challenging environments. Sprunk et al. [21] propose a data-driven approach using laser scanners and demonstrate millimeter-level accuracy without building any map of the environment. Cherubini and Chaumette [5] introduced the use of a laser range finder for obstacle avoidance. Marshall et al. [14] use a 3D laser sensors and build a hybrid map of the environment. McManus et al. [16] extend [7] to reflectance imagery coming from a 3D LIDAR. Krüsi et al [13] build a topological/metrical map and use ICP together with obstacle avoidance in the repeat phase. Ostafew et al. [17] employ model predictive control (MPC) and Gaussian processes to improve tracking accuracy, and at the same time devise a set of heuristics for improving the velocity profile.

All those approaches aim at an accurate reproduction of a path that is carefully demonstrated by the user. In this work, we distinguish ourselves in that we introduce an additional optimization step in the process.

The majority of trajectory optimization approaches relies on a globally consistent map of the environment and, to the best of our knowledge, have not been used in teach and repeat settings. Khatib [11] first introduced the use of potential functions for computing collision free trajectories. Quinlan and Khatib [18] later devised an approach which enables avoiding obstacles by deforming the trajectory as a collection of elastic bands.

The latter inspired the CHOMP algorithm [26] which uses covariant functional gradient descent to plan collision free trajectories. Kalakrishnan [10] proposed a variant of CHOMP that uses derivative-free stochastic optimization. Schulman et al. [20] use convex relaxation and sequential quadratic programming to compute a collision-free trajectory. Those approaches consider optimization in path coordinates, without explicitly considering time, and need to be followed by a motion retiming step [2, 25]. Such a decoupled approach, however, may lead to suboptimal trajectories in the vicinity of obstacles. To avoid this Byravan et al. [4] explicitly introduce timing in the parametrization of CHOMP. In their work, however, they require a fixed execution time, while our approach takes it as the main objective to optimize.

Trajectory planning has been also considered in the context of optimal control, where differential dynamic programming methods [9] have been successful in a number of applications. Similarly, iLQG [22] allowed handling nonlinear control costs with a quadratic approximation. This was improved on by Toussaint [24] who used approximate inference to efficiently generalize non-linear-quadratic-gaussian systems.

Our approach has teach and repeat as a target, and differs from most works in trajectory optimization. Since we are only interested in finding a perturbation about a demonstrated trajectory, the problem is well represented by introducing convex constraints in the functional space of trajectories. Furthermore, with our approach we marry both trajectory *and* time optimization in a single, iterative, algorithm.

III. Lexicographic Trajectory Optimization

Robot motion planning can be expressed as a multi-objective optimization problem, whose goal is to minimize a number of objective functions under a set of constraints. Typical objective functions include the execution time of the trajectory, path smoothness or distance to obstacles. The problem has often been tackled by replacing the multiple objectives $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \dots, f_k(\mathbf{x})]$ with a weighted sum $\boldsymbol{\lambda}^\top \mathbf{f}(\mathbf{x})$. This renders the objective scalar, and the problem can be solved with standard optimization procedures.

In most cases, the appropriate weighting $\boldsymbol{\lambda}$ might be arbitrary and hard to determine, especially for a non-expert user. In this paper, we follow an alternative approach and we only assume that the user can give a set of preferences $f_1 \succ f_2 \succ \ldots \succ f_k$ on the priorities of the objectives.

Problems of this kind are known as lexicographic optimization [19], and, given a feasible set $\mathcal{X}$, can be compactly stated as follows:

$$\begin{aligned} &\text{lex}\min \quad \{f_1(\mathbf{x}), \dots, f_k(\mathbf{x})\} \\ &\text{subject to} \quad \mathbf{x} \in \mathcal{X}. \end{aligned} \tag{1}$$

The problem is equivalent to an ordered sequence of optimization problems for $i = 1, \dots, k$:

$$\begin{aligned} &\text{minimize} \quad f_i(\mathbf{x}) \\ &\text{subject to} \quad \mathbf{x} \in \mathcal{X}, \quad f_j(\mathbf{x}) \leq f_j^* \quad \forall j < i, \end{aligned} \tag{2}$$

where f_j^* is the j-th optimal value of the previous problem.

Given an m-dimensional trajectory $\mathbf{q}(t) = \mathbf{q}(s(t))$, we formulate the trajectory optimization problem as computing a path $\mathbf{q} : [0,1] \to \mathbb{R}^m$ and a monotone non-decreasing time-scaling function $s : \mathbb{R}_{\geq 0} \to [0,1]$. We consider two objective functions: execution time and trajectory smoothness.

Let time T be expressed by the following objective, which only depends on s:

$$T = \int_0^T dt = \int_0^1 \frac{ds}{\dot{s}}. \tag{3}$$

Let us express the trajectory smoothness through a convex functional $\phi : \mathcal{H} \to \mathbb{R}$, bounded from below, where $\mathcal{H}$ denotes the Hilbert space of trajectories in $\mathbb{R}^m$. Finally, let $\mathcal{Q} : [0,1] \to \mathcal{P}(\mathbb{R}^m)$, $\mathcal{V} : [0,1] \to \mathcal{P}(\mathbb{R}^m)$, and $\mathcal{A} : [0,1] \to \mathcal{P}(\mathbb{R}^m)$ be functions that map path coordinates to convex feasible sets, respectively on position, velocity, and acceleration. Let the first and second derivative of $\mathbf{q}(t)$ with respect to time be:

$$\dot{\mathbf{q}}(t) = \mathbf{q}'(s)\dot{s}, \quad \ddot{\mathbf{q}}(t) = \mathbf{q}''(s)\dot{s}^2 + \mathbf{q}'(s)\ddot{s}, \tag{4}$$

where we denote with the apostrophe derivatives with respect to the path coordinate s, and with the dot derivatives with respect to the time t.

We thus formulate the problem as

$$\text{lex}\min \ \left\{ \int_0^1 \frac{ds}{\dot{s}},\ \phi\left[\mathbf{q} \circ s\right] \right\} \tag{5}$$

$$\text{subject to} \ \mathbf{q}(s) \in \mathcal{Q}(s) \qquad \forall s \in [0,1], \tag{6}$$

$$\mathbf{q}'(s)\dot{s} \in \mathcal{V}(s) \qquad \forall s \in [0,1], \tag{7}$$

$$\mathbf{q}''(s)\dot{s}^2 + \mathbf{q}'(s)\ddot{s} \in \mathcal{A}(s) \qquad \forall s \in [0,1], \tag{8}$$

$$\mathbf{q}(0) = \mathbf{q}_0, \qquad \mathbf{q}(1) = \mathbf{q}_T, \tag{9}$$

$$\mathbf{q}'(0)\dot{s}(0) = \dot{\mathbf{q}}_0, \quad \mathbf{q}'(1)\dot{s}(T) = \dot{\mathbf{q}}_T, \tag{10}$$

where $\mathbf{q}_0$, $\mathbf{q}_T$, $\dot{\mathbf{q}}_0$ and $\dot{\mathbf{q}}_T$ denote the configuration and the velocity of the trajectory at the boundaries.

A. Optimization via block coordinate descent

Unfortunately, problem (5)–(10) is highly non-convex. This is because optimization is carried out with respect to both $\mathbf{q}$ and s, which appear composed in the problem. We choose to iteratively solve the problem via block coordinate descent, under the assumption that an initial feasible guess $\breve{\mathbf{q}}$, $\breve{s}$ is available. This is equivalent to alternating between solving (5)–(10) for $\mathbf{q}$ with s fixed (*trajectory smoothing*) and solving (5)–(10) for s with $\mathbf{q}$ fixed (*time scaling*). Our choice is motivated by the fact that, under the assumption that $\mathcal{Q}$, $\mathcal{V}$, and $\mathcal{A}$ return convex sets and for certain convex $\phi[\mathbf{f}]$ functionals, both steps can be individually solved via convex optimization. Moreover, the convergence of the algorithm can be trivially proven by exploiting the monotone convergence theorem.

B. Convexity of time scaling and trajectory smoothing

Let us consider the optimization of (5)–(10) for $\mathbf{q}$. Since the execution time only depends on s, the lexicographic objective reduces to the smoothness functional. By assumption, s is fixed and $\mathcal{Q}$, $\mathcal{V}$, and $\mathcal{A}$ return convex sets. Since convexity is preserved under linear transformation, (7) and (8) are convex. Similarly, (6), and (9)–(10) are trivially convex, and so is the smoothness functional $\phi[\mathbf{q} \circ s]$ for any convex functional $\phi[\mathbf{f}]$. Therefore the trajectory smoothing problem is convex.

When solving (5)–(10) with respect to the time scaling s, we need to consider the two functionals separately, due to the lexicographic framework. Following Verscheure et al. [25], we take the nonlinear substitution:

$$a(s) = \ddot{s}, \qquad b(s) = \dot{s}^2. \tag{11}$$

These functions are related by the differential property $b'(s) = 2a(s)$, which follows from noting that $\dot{b}(s) = b'(s)\dot{s}$ and $\dot{b}(s) = \frac{d}{dt}\left\{\dot{s}^2\right\} = 2\dot{s}\ddot{s} = 2a(s)\dot{s}$.

If we include this constraint in (5)–(10) and enforce the non-negativity of $b(s)$, we can rewrite the problem of minimizing the time functional as a function of $a(s)$ and $b(s)$ alone:

$$\begin{aligned}
\text{minimize} \quad & \int_0^1 \frac{ds}{\sqrt{b(s)}} && && (12)\\
\text{subject to} \quad & b(s) \geq 0, \quad b'(s) = 2a(s) && \forall s \in [0,1], && (13)\\
& \mathbf{q}'(s)\sqrt{b(s)} \in \mathcal{V}(s) && \forall s \in [0,1], && (14)\\
& \mathbf{q}''(s)b(s) + \mathbf{q}'(s)a(s) \in \mathcal{A}(s) && \forall s \in [0,1], && (15)\\
& b(0) = 1, \quad b(1) = 1. && && (16)
\end{aligned}$$

Here, we removed (6) and (9), due to their independence of the time scaling function, and replaced (10) by (16), without loss of generality. The cost (12) is convex in $b(s)$, (13) and (16) are trivially convex, while the convexity of (15) is established due to invariance under linear transformations. The constraint (14) can be proven to be convex exploiting the scalar nature of $b(s)$, for chosen s.

We rewrite the problem of minimizing the smoothness functional in a similar way to (12)–(16), with the difference that we replace (12) with $\phi[\mathbf{q} \circ s]$ and we add the lexicographic constraint:

$$\int_0^1 \frac{ds}{\sqrt{b(s)}} \leq T^*, \tag{17}$$

where T^* is the execution time computed by the preceding time minimization. While (17) is convex, $\phi[\mathbf{q} \circ s]$ is not always so under substitution (11). In this work we consider the squared ℓ^2 norm of $\ddot{\mathbf{q}}(t)$ as smoothness functional, since lower accelerations lead to smoother velocities and lower energy consumption. In this case we have:

$$\phi[\mathbf{q} \circ s] = \int_0^1 \frac{\|\mathbf{q}''(s)b(s) + \mathbf{q}'(s)a(s)\|_2^2}{\sqrt{b(s)}}\, ds, \tag{18}$$

which is indeed convex in both $a(s)$ and $b(s)$.

IV. Teach Optimize and Repeat with LexTOR

In this section, we describe LexTOR, our proposed algorithm for teach, optimize and repeat. Intuitively, LexTOR inserts an *optimize* step between the *teach* and *repeat* phase, in order to compute an optimized trajectory in terms of execution time and smoothness. This entails defining a trajectory parameterization, identifying the constraints (6)–(10) and formulating the problem with respect to the chosen parametrization.

A. Trajectory parameterization and constraints

We represent the trajectory $\mathbf{q}(t)$ as a piecewise quadratic function with $\mathbf{q} \in \mathcal{C}^1(\mathbb{R})$, composed of N distinct intervals with non-uniform sizes $\Delta t_n \ \forall n \in \{1, \ldots, N\}$. Hence, we compactly represent $\mathbf{q}(t)$ as a collection of N time intervals Δt_n and of $N+1$ vectors $\mathbf{q}_n$, $\mathbf{v}_n$, $\mathbf{a}_n$, where:

$$\mathbf{q}(t_n) = \mathbf{q}_n, \quad \dot{\mathbf{q}}(t_n) = \mathbf{v}_n, \quad \ddot{\mathbf{q}}(t_n) = \mathbf{a}_n, \tag{19}$$

for all $n \in \{1, \ldots, N+1\}$, with $t_n = \sum_{i=1}^{n-1} \Delta t_i$.

Regarding the constraints, we consider $\mathcal{Q}(s)$, $\mathcal{V}(s)$, and $\mathcal{A}(s)$ to be sampled at $N+1$ distinct points $\mathcal{Q}_n$, $\mathcal{V}_n$, and $\mathcal{A}_n$ which respectively constrain $\mathbf{q}_n$, $\mathbf{v}_n$, and $\mathbf{a}_n$. We compute the set $\mathcal{Q}_n$ based on user preferences on the deviation from the taught trajectory $\mathbf{q}_o(t)$. The sets $\mathcal{V}_n$ and $\mathcal{A}_n$ express the maximum velocities and accelerations the robot is allowed to achieve.

Unfortunately, constraining the x and y velocity in the robot frame is not convex. Hence, we enforce convexity by bounding the tangential velocity. Denoting by $v_{\max}$ and $\omega_{\max}$ respectively the maximum spatial and angular velocity, we impose:

$$\sqrt{\dot{x}^2 + \dot{y}^2} \leq v_{\max}, \quad |\dot{\theta}| \leq \omega_{\max}. \tag{20}$$

Similarly, we compute a conservative bound in acceleration, by taking into account the inequality:

$$\frac{d}{dt}\sqrt{\dot{x}^2 + \dot{y}^2} \leq \sqrt{\ddot{x}^2 + \ddot{y}^2} \quad \forall x, y \in \mathcal{C}^1(\mathbb{R}). \tag{21}$$

Hence, if $a_{\max}$ and $\alpha_{\max}$ denote respectively the maximum spatial and angular acceleration we can impose:

$$\sqrt{\ddot{x}^2 + \ddot{y}^2} \leq a_{\max}, \quad |\ddot{\theta}| \leq \alpha_{\max}. \tag{22}$$

The constraints are second order cone (SOC) constraints and can be expressed in terms of $\mathbf{q}_n$, $\mathbf{v}_n$, and $\mathbf{a}_n$.

B. Trajectory smoothing

During the trajectory smoothing step of block coordinate descent, the scaling function s is fixed and known. Therefore, we directly optimize in temporal space, as this greatly simplifies both exposition and implementation. Given the piecewise quadratic parameterization, we formulate the trajectory smoothing problem as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{n=1}^{N} \|\mathbf{a}_n\|_2^2 \, \Delta t_n && (23)\\
\text{subject to} \quad & \mathbf{q}_{n+1} = \mathbf{q}_n + \mathbf{v}_n \Delta t_n + \frac{1}{2}\mathbf{a}_n \Delta t_n^2, && (24)\\
& \mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}_n \Delta t_n, && (25)\\
& \mathbf{q}_n \in \mathcal{Q}_n, \quad \mathbf{v}_n \in \mathcal{V}_n, \quad \mathbf{a}_n \in \mathcal{A}_n, && (26)\\
& \mathbf{q}_1 = \mathbf{q}_0, \quad \mathbf{q}_{N+1} = \mathbf{q}_T, && (27)\\
& \mathbf{v}_1 = \dot{\mathbf{q}}_0, \quad \mathbf{v}_{N+1} = \dot{\mathbf{q}}_T. && (28)
\end{aligned}$$

Here, we consider the Δt_n as the time coordinates from the previous time scaling optimization, and optimize for $\mathbf{q}_n$, $\mathbf{v}_n$, and $\mathbf{a}_n$, $\forall n \in \{1, \dots, N\}$. The constraints (24) and (25) are taken $\forall n \in \{1, \dots, N\}$, while (26) $\forall n \in \{1, \dots, N+1\}$.

All the constraints are either SOC or linear. Note that minimizing the objective function is equivalent to minimizing the value of a helper variable γ with the constraint

$$\left\| \begin{bmatrix} \mathbf{a}_1^\top \sqrt{\Delta t_1} & \mathbf{a}_2^\top \sqrt{\Delta t_2} & \cdots & \mathbf{a}_N^\top \sqrt{\Delta t_N} \end{bmatrix}^\top \right\|_2 \leq \gamma. \tag{29}$$

The trajectory smoothing is therefore a SOCP problem, for which efficient solvers exist [3].

C. Time scaling

During the time scaling step, the path $\mathbf{q}$ is fixed and known. Following [25], we assume $b(s)$ to be a continuous piecewise linear function of s, with N fixed intervals. As a consequence, $a(s)$ is piecewise constant, given the differential relationship $b'(s) = 2a(s)$. We compactly represent $a(s)$ and $b(s)$ as a collection of N path coordinate intervals Δs_n and $N+1$ samples $a_n = a(s_n)$, $b_n = b(s_n)$, whose spacing depends on the Δt_n of (23). Under the piecewise linearity assumption, the problem (12)–(16) becomes

$$\begin{aligned}
\text{minimize} \quad & \sum_{n=1}^{N} \frac{2\Delta s_n}{\sqrt{b_{n+1}} + \sqrt{b_n}} && (30)\\
\text{subject to} \quad & b_n \geq 0, && (31)\\
& b_{n+1} - b_n = 2a_n \Delta s_n, && (32)\\
& \mathbf{v}_n \sqrt{b_n} \in \mathcal{V}_n, && (33)\\
& \mathbf{v}_n a_n + \mathbf{a}_n b_n \in \mathcal{A}_n, && (34)\\
& \mathbf{v}_{n+1} a_n + \mathbf{a}_n b_{n+1} \in \mathcal{A}_n, && (35)\\
& b_1 = 1, \quad b_N = 1, && (36)
\end{aligned}$$

where we optimize for b_n and a_n $\forall n \in \{1, \dots, N+1\}$ and consider (31), (33), and (34) $\forall n \in \{1, \dots, N+1\}$. Constraints (32) and (35) are taken $\forall n \in \{1, \dots, N\}$.

We include the constraint (35) to bound the acceleration $\forall s \in [s_n, s_{n+1}]$. The scaled acceleration is linear in s, therefore we can check inclusion only at s_n and s_{n+1}.

Similarly to trajectory smoothing, the entire problem can be reformulated as SOCP. We know that (33) is a convex constraint of a scalar variable, therefore it is in the form $b_n^{\min} \leq b_n \leq b_n^{\max}$, for some $b_n^{\min}$ and $b_n^{\max}$. SOC constraints are closed with respect to linear transformations, hence (34) and (35) are SOC. The problem is equivalent to minimizing, for the helper variables c_n and d_n [25],

$$\sum_{n=1}^{N} 2\Delta s_n d_n, \tag{37}$$

with the additional constraints:

$$\left\| \begin{bmatrix} 2c_n & b_n - 1 \end{bmatrix}^\top \right\|_2 \leq b_n + 1, \tag{38}$$

$$\left\| \begin{bmatrix} 2 & c_{n+1} + c_n - d_n \end{bmatrix}^\top \right\|_2 \leq c_{n+1} + c_n + d_n, \tag{39}$$

where we take the first constraint $\forall n \in \{1, \dots, N+1\}$ and the second $\forall n \in \{1, \dots, N\}$. Finally, we compute a new set of time intervals Δt_n for the succeeding iteration as $\Delta t_n \mapsto 2\Delta s_n d_n$.

A similar approach can be followed in order to optimize the smoothness of the trajectory with respect to the time scaling. Consistently with Sec. III-B we supplement (30)–(36) with the lexicographic constraint:

$$\sum_{n=1}^{N} 2\Delta s_n d_n \leq \sum_{n=1}^{N} \Delta s_n, \tag{40}$$

where we take Δs_n from the solution of (30)–(36).

Under our parameterization for $\mathbf{q}$ and s, we approximate (18) as:

$$\sum_{n=1}^{N} \frac{\|\mathbf{a}_n b_n + \mathbf{v}_n a_n\|^2}{\sqrt{b_n}} \Delta s_n, \tag{41}$$

which is a fair choice so long as $\Delta s_n \ll 1$ $\forall n \in \{1, \dots, N\}$. The whole problem can be expressed in SOCP form by introducing N additional variables e_n, replacing (30) with the objective $\sum_{i=1}^{N} e_n$, and introducing the following constraints $\forall n \in \{1, \dots, N\}$:

$$\left\| \begin{bmatrix} 2\sqrt{\Delta s_n} \left(\mathbf{a}_n^\top b_n + \mathbf{v}_n^\top a_n\right) & e_n - c_n \end{bmatrix}^\top \right\|_2 \leq e_n + c_n. \tag{42}$$

D. Execution of the optimized trajectory

For the execution of the trajectory, we build upon our previous work on teach and repeat [21]. During the teach phase, the system records pose and velocity of the robot from the wheel encoders. The system also records the measurements from an on-board laser range finder at so-called anchor points. We express the reference trajectory as relative offset to these anchor points and, during the repeat phase, we employ a scan matching routine to compute the error signal for the feedback controller. By expressing the

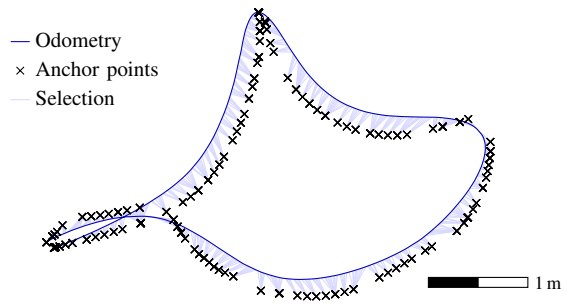

Fig. 2. Anchor points, along the demonstrated path, that were selected when tracking the optimized trajectory of the Edgy scenario, with 20 cm distance constraints. Slight holes in anchor point selection may occur due to differences in orientation, which are not shown here. The path is expressed in odometry frame as it is internally computed by the robot, which is not aware of its global position.

reference trajectory relative to anchor points, our framework alleviates the drift in odometry without the need of building a globally consistent metric map in a setup phase.

This work presents two main differences. First, the anchor points now no longer necessarily fall onto the optimized trajectory, an effect that is visualized in Fig. 2. Second, we replace the recorded velocity with the optimized one in the feed-forward part of the controller.

E. Practical aspects

Both optimization problems described in Sec. IV-C result in a piecewise quadratic time scaling function. For any continuous trajectory model, the composition with the scaling function results in higher order terms, like jerk or snap. This high order effects may render the trajectory smoothing problem infeasible, since we are only considering Δt_n time intervals. This is unfortunately inevitable for any computation that relies on finite precision. When this happens, we circumvent this problem by choosing a small $\varepsilon > 0$ and feed to the trajectory smoothing a set of scaled time intervals $(1 + \varepsilon)\Delta t_n$. As the sampling of the trajectory gets denser, jerk and snap become negligible, and in the limiting case of $N \to \infty$ we have that $\varepsilon \to 0$.

As an additional note, in our experiments we have found that optimizing the trajectory smoothness with respect to the time scaling function will not, in general, improve the smoothness cost significantly. As a rule of thumb it can thus be completely omitted, especially if computational efficiency is of dire importance.

V. Experimental Evaluation

We evaluated LexTOR on a holonomic KUKA omniRob equipped with two SICK S300 laser scanners with a 270° field of view and 541 beams. Fig. 1 depicts the KUKA omniRob in action, as well as the environment in which the tests were performed, although different obstacles were placed for different trajectory demonstrations. We recorded the robot trajectory with Motion Analysis' Cortex motion capture studio for the ground-truth evaluation. To reduce the noise due to the vibration of the pole on which the markers are mounted, we smooth the recorded data with a normal

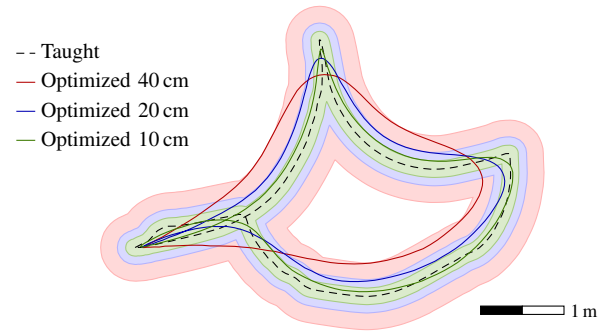

Fig. 3. Taught and optimized trajectories with location boundaries for the Edgy test scenario, according to the motion capture studio. Three levels of maximum distance are shown, respectively 10 cm, 20 cm, and 40 cm.

kernel with 20 ms of standard deviation. We implemented the lexicographic optimization in MATLAB with CVX [8] as front-end convex optimizer and SDPT3 [23] as back-end. For the recording and execution of trajectories we relied on the teach-and-repeat approach proposed by Sprunk et al. [21]. For the latter we used the reported parameters without further fine-tuning control gains or time delays. As shown by Sprunk et al. [21], the teach-and-repeat approach is accurate enough to not require the use of both laser scanners; we thus use only one scanner for the sake of simplicity.

A. Experimental setup

We test four distinct trajectories in order to validate our approach under different scenarios, shown in Fig. 3 and Fig. 4. For ease of exposition, we name them *FigureEight*, *Loop*, *Passage* and *Edgy*. All of them are round-courses for pure ease of practical evaluation, as this avoids the need to reposition the robot for multiple executions.

We devise the scenarios to showcase different behaviors. FigureEight provides a reference trajectory that is well suited for smooth execution. Edgy evaluates the behavior of the approach in the presence of strong cusps in the path of the robot. Loop provides a test where both smooth execution and a cusp are present. Passage showcases an instance where the robot needs to pass through a narrow passage and hence contains bounds of different size. We bounded all executions to maximum velocities of 0.6 m/s in translation and 0.5 rad/s in rotation. We set the bounds on the acceleration according to the hard limits of the firmware, namely 0.8 m/s^2 in translation and 1 rad/s^2 in rotation. We demonstrated the trajectories with a joystick, resulting in irregular velocity and acceleration profiles. The execution time of the demonstrated trajectories ranges from 29 to 79 seconds.

We devise both qualitative and quantitative tests, with the goal of verifying execution time, smoothness, accuracy, and satisfaction of location constraint. For a fair comparison with the human demonstration we optimize the input trajectories with the same velocity limits of 0.6 m/s in translation and 0.5 rad/s in orientation, while also bounding the maximum acceleration to be no larger than 0.4 m/s in translation and 0.4 rad/s in rotation, which are, in fact, more restrictive than those of the taught trajectories.

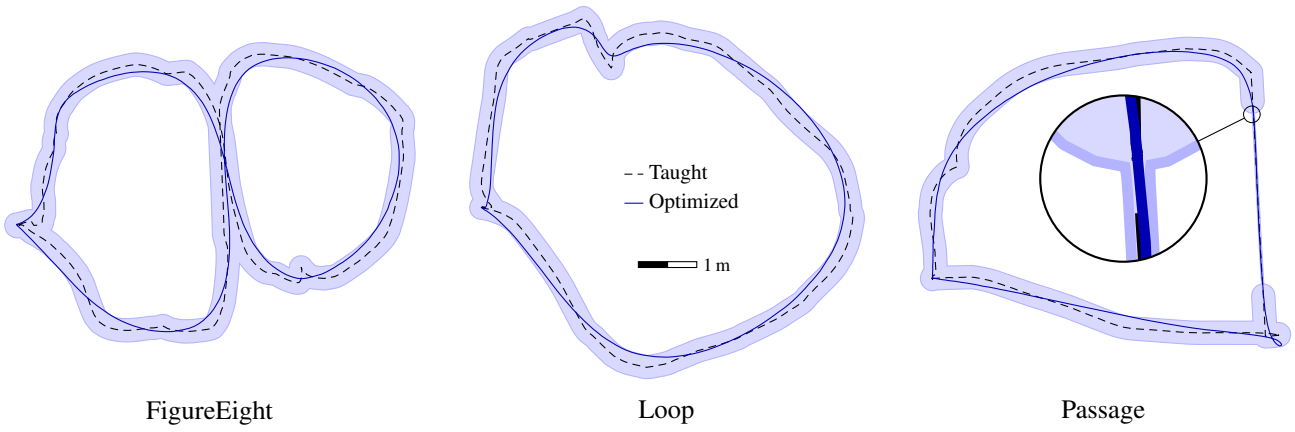

Fig. 4. Taught and optimized trajectories with location boundaries (shaded) for the FigureEight, Loop and Passage test scenarios according to the motion capture studio. All three scenarios share the same scale reported in the center.

We impose a maximum Euclidean distance radius of 20 cm and a maximum angular distance of 10° from each odometry measurement of the reference trajectory. For the Passage trajectory, we consider non-uniform distance bounds: we change the latter values to be respectively 2 cm and 1°, in the narrow passage portion of the sequence. In an effort to determine the behavior of our approach under more and less restrictive location constraints, for the Edgy trajectory we also consider two extra tests, in one we decrease the Euclidean distance to 10 cm and angular distance to 5°, in the other we increase them to 40 cm and 20°, respectively.

B. Qualitative analysis

Fig. 3 and Fig. 4 show the taught trajectories and a sample execution of the optimized ones, as captured by the motion capture studio, as well as the bounds in position. The optimized trajectories always remains within the given bounds, without the use of any map of the environment.

As the optimized trajectory does not follow exactly the taught one, the anchor points might be further away from the optimized trajectory. We are, nevertheless, able to accurately repeat the trajectory by matching anchor points based on a combined norm of rotational and spatial differences. Fig. 2 reports which anchor points were selected for the 20 cm optimized trajectory of the Edgy scenario.

The resulting trajectories are not only smooth in position, but also in velocity. Fig. 8 shows a comparison between the recorded and optimized velocity profiles for the Loop trajectory. The erratic behavior of the velocities demonstrated by the human operator result in degraded execution times and put significantly more strain on the robot, which may result in faster wear of its components.

Our approach is not equivalent to a mere trajectory smoothing followed by a retiming step. Changing the velocity or acceleration limits will not only affect the execution time, but also the computed trajectory. Fig. 5 depicts an instance of such a behavior, where we report the optimized version of the Loop trajectory together with another optimized version with halved maximum speeds.

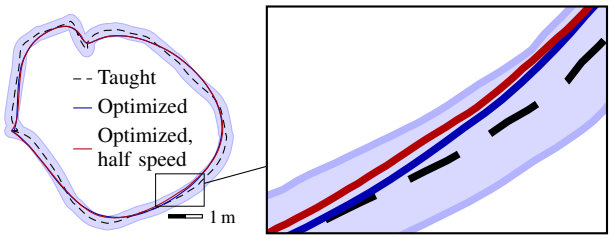

Fig. 5. Close-up of the taught and optimized trajectories for different velocity limits, as recorded by the motion capture. Halving the maximum velocities not only affects the temporal execution of the trajectory, but also the actual computed path.

C. Quantitative analysis

Fig. 6 shows the real execution times for the taught trajectories against the optimized ones with 20 cm bounds. We achieve a reduction ranging from 23% to 45% when compared to the human demonstrations, despite lower acceleration allowances. For the Edgy trajectory we achieve timings consistent with the location constraints: 25 s with a reduction of 14% for the more constricted configuration and 20 s with a reduction of 32% for the less constricted one.

In addition to the execution time, we evaluated the accuracy in tracking for both taught and optimized trajectories. Given the absence of a ground-truth demonstration for the optimized trajectories, we compute it from the taught one. We synchronize the motion capture recording with the odometry of the taught trajectory and then, for each sample of the odometry, we apply to the motion capture recording the rigid body transformation between the taught odometry and the optimized one. Clearly, this approach introduces additional errors, due to time misalignments and the leaver-arm effect for the rigid body transformations. As a consequence, we take the synthetic ground-truth to be a rough tool to evaluate lower bounds on the accuracy of the approach.

Fig. 7 shows a box plot of the translational errors for both taught and 20 cm optimized trajectories. Despite the inaccuracies introduced by the synthetic ground-truth, we consistently obtain median errors lower than 5 mm.

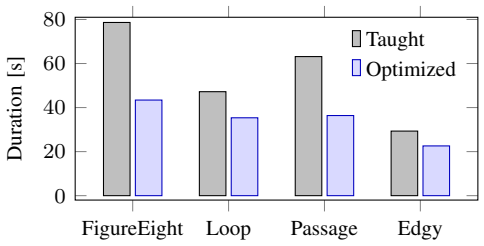

Fig. 6. Average execution times of taught and optimized trajectories with 20 cm distance constraints for all four scenarios.

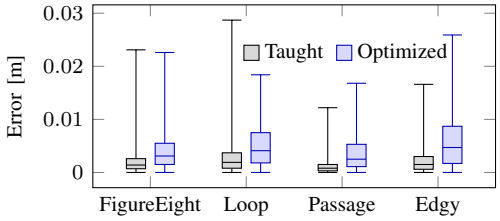

Fig. 7. Translational error of taught and 20 cm optimized trajectories. The boxes show minimum, maximum, and median errors, as well as upper and lower quartiles. We evaluated approximately 10 executions per trajectory, with a total of 90 executions. The errors for the optimized trajectories provide an upper bound on the actual error, due to absence of ground-truth.

VI. Conclusion

In this paper, we proposed *teach, optimize and repeat (TOR)*, a novel paradigm for navigation that extends teach and repeat methods. The paradigm is able to counter the human factor during demonstration by including an additional optimization step between the teach and repeat phase, based on user preferences. This results in repeat phases that require less execution time and are less prone to wear or tear robot components, while still remaining within chosen bounds of the original demonstration.

We further devised a constrained trajectory optimization framework, LexTOR, which optimizes both execution time and trajectory smoothness in the lexicographic sense. Each step of the algorithm can be expressed as a second order cone program, thus allowing for efficient implementations. Finally, with the aid of a motion capture system, we showed the effectiveness of our approach, by thoroughly evaluating it on a holonomic robot. Experiments show that LexTOR produces trajectories that are faster and smoother than those taught by the user, that are within the specified bounds, and that can be executed with an accuracy of a few millimeters.

References

[1] E. T. Baumgartner and S. B. Skaar. An autonomous vision-based mobile robot. *IEEE Transactions on Automatic Control*, 1994.

[2] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. *Int. J. of Robotics Research*, 1985.

[3] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

[4] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox. Space-time functional gradient optimization for motion planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2014.

[5] A. Cherubini and F. Chaumette. Visual navigation of a mobile robot with laser-based collision avoidance. *Int. J. of Robotics Research*, 32 (2):189–205, 2013.

[6] A. Cherubini, F. Chaumette, and G. Oriolo. Visual servoing for path reaching with nonholonomic robots. *Robotica*, 2011.

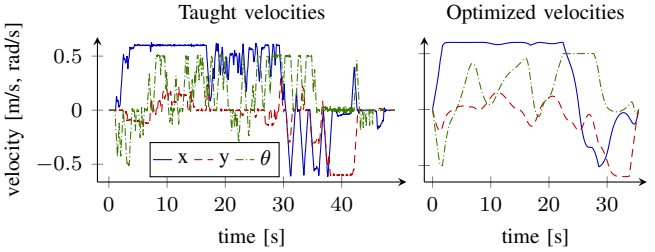

Fig. 8. Velocities expressed in the robot reference frame for the taught and optimized trajectories of the Loop scenario. The erratic behavior of the taught velocities is smoothed out through optimization, also achieving a lower execution time.

[7] P. Furgale and T. D. Barfoot. Visual teach and repeat for long-range rover autonomy. *J. of Field Robotics*, 2010.

[8] M. C. Grant and S. P. Boyd. Graph implementations for nonsmooth convex programs. In *Recent advances in learning and control*. 2008.

[9] D. H. Jacobson and D. Q. Mayne. *Differential dynamic programming*. North-Holland, 1970.

[10] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2011.

[11] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. of Robotics Research*, 1986.

[12] O. Koch, M. R. Walter, A. S. Huang, and S. Teller. Ground robot navigation using uncalibrated cameras. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2010.

[13] P. Krüsi, B. Bücheler, F. Pomerleau, U. Schwesinger, R. Siegwart, and P. Furgale. Lighting-invariant adaptive route following using iterative closest point matching. *J. on Field Robotics*, 2014.

[14] J. Marshall, T. Barfoot, and J. Larsson. Autonomous underground tramming for center-articulated vehicles. *J. on Field Robotics*, 2008.

[15] Y. Matsumoto, M. Inaba, and H. Inoue. Visual navigation using viewsequenced route representation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 1996.

[16] C. McManus, P. Furgale, B. Stenning, and T. D. Barfoot. Lighting-invariant visual teach and repeat using appearance-based lidar. *J. on Field Robotics*, 2013.

[17] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2014.

[18] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 1993.

[19] M. J. Rentmeesters, W. K. Tsai, and K.-J. Lin. A theory of lexicographic multi-criteria optimization. In *Proc. of the IEE Int. Conf. on Engineering of Complex Computer Systems*, 1996.

[20] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *Int. J. of Robotics Research*, 2014.

[21] C. Sprunk, G. Tipaldi, A. Cherubini, and W. Burgard. Lidar-based teach-and-repeat of mobile robot trajectories. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.

[22] E. Todorov and W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proc. of the American Control Conference*, 2005.

[23] K.-C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 - a MATLAB software package for semidefinite programming, version 1.3. *Optimization methods and software*, 1999.

[24] M. Toussaint. Robot trajectory optimization using approximate inference. In *Proc. of the Int. Conf. on Machine Learning*, 2009.

[25] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl. Time-energy optimal path tracking for robots: a numerically efficient optimization approach. In *Proc. of the IEEE Int. Workshop on Advanced Motion Control*, 2008.

[26] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. CHOMP: Covariant hamiltonian optimization for motion planning. *Int. J. of Robotics Research*, 2013.