Timed-Elastic Bands for Manipulation Motion Planning

Bence Magyar, Nikolaos Tsiogkas, Jérémie Deray, Sammy Pfeiffer, and David Lane

***Abstract*—Motion planning is one of the main problems studied in the field of robotics. However, it is still challenging for the state-of-the-art methods to handle multiple conditions that allow better paths to be found. For example, considering joint limits, path smoothness and a mixture of Cartesian and joint-space constraints at the same time pose a significant challenge for many of them. This letter proposes to use *timed-elastic bands* for representing the manipulation motion planning problem, allowing to apply continuously optimized constraints to the problem during the search for a solution. Due to the nature of our method, it is highly extensible with new constraints or optimization objectives. The proposed approach is compared against state-of-the-art methods in various manipulation scenarios. The results show that it is more consistent and less variant, while performing in a comparable manner to that of the state of the art. This behavior allows the proposed method to set a lower-bound performance guarantee for other methods to build upon.**

***Index Terms*—Motion and path planning, mobile manipulation, collision avoidance, domestic robots.**

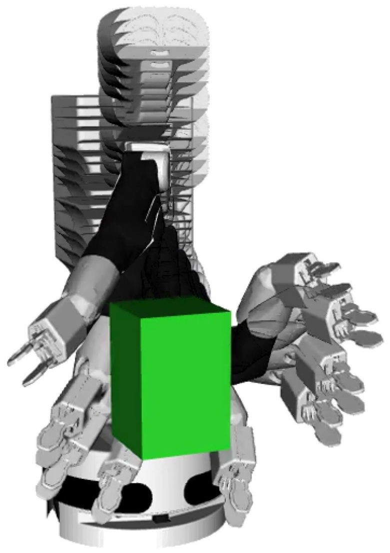


Fig. 1. TEB2MP plans around an obstacle. The robot must move the end-effector from an initial position right of the green obstacle to a final position on the left. The generated plan guides the trajectory under the obstacle allowing a successful motion to be performed.

I. Introduction

MOTION planning is one of the most important aspects of any robotic system as it enables its interaction with the real world. Operating in the real world requires fast and efficient methods that find plans for the robot to achieve its goals.

In general the motion planning problem requires solutions that allow the robot to reach a goal from a starting configuration without colliding with obstacles or itself, while respecting its joint limits and other constraints. Such a desirable property for a trajectory is smoothness. Smooth trajectories have bounded and derivable acceleration and jerk. A smooth trajectory is vital for safe robot operation as it prevents stress of the robot's parts and allows humans to feel more comfortable around the robot.

The current state-of-the-art, while producing remarkable results, still has many aspects that cannot be handled efficiently or, even worse, cannot be handled at all. For example, joint limits are usually clamped to some maximum value, leading to situations where an otherwise valid plan becomes infeasible [1]. Smoothness is often not considered at all or doing so is ill-fitted to the nature of the planner [2], leading to jerky motions and mechanical stress of the hardware. Optimization using simultaneously Cartesian and joint-space constraints is, in many cases, not possible [3].

Given the limitations of the state-of-the-art, this work proposes a method for motion planning that addresses all of the aforementioned issues. The presented approach follows the work of [4] on *Timed Elastic Bands* (TEB) for *Model Predictive Control* (MPC). TEB was first applied to mobile base trajectory optimization [5].

The experiments presented in [4] show that the TEB approach is capable of handling the computational complexity of non-linear systems. The problem is formulated as a sparse graph structure solvable by means of non-linear least-squares optimization. The system is fast enough to be run at the control loop rate. To allow dynamic obstacle avoidance it can be combined with the work presented in [5], which allows temporal information to be included in the optimization problem.

Manuscript received February 20, 2019; accepted June 26, 2019. Date of publication July 10, 2019; date of current version July 24, 2019. This letter was recommended for publication by Associate Editor F. van der Stappen and Editor N. Amato upon evaluation of the reviewers' comments. This work was supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC). *(Corresponding author: Bence Magyar.)*

B. Magyar, N. Tsiogkas, and D. Lane are with the Edinburgh Centre for Robotics, Heriot-Watt University, EH14 4AS Edinburgh, U.K. (e-mail: bence.magyar@hw.ac.uk; nt95@hw.ac.uk; d.m.lane@hw.ac.uk).

J. Deray is with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, 08005 Barcelona, Spain, and also with the PAL Robotics, 08005 Barcelona, Spain (e-mail: jderay@iri.upc.edu).

S. Pfeiffer is with the Centre for Artificial Intelligence, Innovation and Enterprise Research Lab, University of Technology Sydney, Sydney, N.S.W. 2015, Australia (e-mail: sammy.pfeiffer@student.uts.edu.au).

Digital Object Identifier 10.1109/LRA.2019.2927956

This letter proposes the use of TEB for solving manipulation motion planning problems on complex manipulator systems such as the one depicted in Fig. 1. The derived method is coined *Timed Elastic Bands for Manipulation Motion Planning* (TEB2MP).

The contributions of this work are:

- Introduction of TEB2MP, a novel trajectory optimization method for robot arms based on TEB.
- TEB2MP's trajectory continuity, ensured by a revisited formulation of interpolation on manifolds encouraging a k derivable trajectory spline.
- TEB2MP's ability to handle Cartesian and joint-space constraints in a single framework.
- A collision checking method which provides smooth, continuous gradients based on overlapping volumes, ensuring that the optimization is always able to escape colliding states.
- A rich comparison of TEB2MP and state-of-the-art methods in multiple challenging scenarios through various benchmarks.

Experimental results show a consistent behaviour with a performance comparable to the state-of-the-art for all the examined metrics. The consistency is measured through the low variance of the results and acts as a lower bound for the performance of the proposed method. In addition, TEB2MP's planning time grows linearly with the optimization iterations which is required for time-critical setups such as in a reactive planner or a controller. This behaviour guarantees that even in difficult cases, in which more iterations are required, the total time can be kept at an upper limit.

The rest of the letter is organised as follows. Sec. II presents an overview of the state-of-the-art literature on the Motion Planning problem. In Sec. III the proposed approach is presented in detail. Sec. IV presents the experimental setup, along with the evaluation metrics and results. Finally, in Sec. V the letter is concluded and future work is presented.

II. Related Work

The problem of motion planning has been actively researched in the past years. The work of [6] introduces the concept of *Probabilistic Road Maps* (PRM) for motion planning. This approach initially constructs a roadmap by randomly sampling the configuration space and creating a graph of configurations. Then multiple queries can be performed by adding a start and a goal configuration to the graph and perform a graph search in order to find a valid path. *Rapidly-exploring Random Trees* (RRT) [7] constructs a tree by randomly sampling the configuration space and trying to connect each sample to its nearest neighbour in the tree. The tree is rooted on the start configuration of the motion planning query. The search stops when the newly connected node is close enough to the goal configuration. *Rapidly-exploring Random Tree Connect* (RRTConnect) [2] extends the previous work by iteratively constructing two trees rooted on the start and goal configurations. The search stops when the two trees get connected. These sampling-based methods' main advantage is to generate trajectories in complex and high dimensional spaces. Despite their ability to generate feasible trajectories fast, the quality of the produced solution is often poor. These methods use randomness in the search procedure which often leads to redundant or jerky motions reducing solution quality. As the solution provided by the aforementioned sampling-based motion planners is usually non-optimal, new asymptotically optimal planners have been introduced in the work of [8]. It provides new methods, specifically the *Rapidly-exploring Random Tree** (RRT*) and *Probabilistic Road Maps** (PRM*) algorithms, that are almost certainly converging to the optimal path given a cost metric.

Even though sampling-based methods are able to find optimal paths, the provided solutions can potentially violate constraints imposed by the robot hardware. For example, the solution may not be feasible since they do not consider any kinematics constraints. Recently some extension to RRT* did incorporate kinematics considerations for simple models such as a mobile-base [9]. To the best of the authors knowledge, no sampling-method is able to handle the complicated kinematics chains of a robotics arm. To impose constraints that are respected by the found plan trajectory optimization techniques have been introduced. These methods take an initial trajectory as an input and optimize it based on a set of cost functions. This initial trajectory can be generated by the aforementioned sampling-based methods, interpolation or other means. A prominent approach in this category of motion planners is the *Covariant Hamiltonian Optimization for Motion Planning* (CHOMP) [10] and its variants [11], [12]. These methods optimize the cost function using covariant gradient descent. *Stochastic Optimization for Motion Planning* (STOMP), presented in [13], performs optimization on non-differentiable constraints by drawing samples stochastically from a set of noisy trajectories. Unfortunately, these methods can potentially be computationally expensive as they require a finely discretised trajectory to perform obstacle collision checks and guarantee a smooth solution and may fail to converge on even moderately hard problems. In addition, the performance of STOMP is heavily dependent on the parameters used for noise generation. In many cases a set of parameters will produce good results, while the same set will perform badly in a different problem.

To avoid the potential computational complexity of CHOMP and STOMP the Trajectory Optimization for Motion Planning (TrajOpt) method was introduced as a solution in [14]. It proposes to formulate the optimization problem as a Sequential Quadratic Programming (SQP) problem with continuous-time collision checking. The reduced computational cost is achieved by using a sparse solution, where a trajectory is represented by a small number of states and using continuous-time collision checking. In addition, the SQP formulation allows hard constraints to be imposed such that the produced plan is guaranteed to respect them. TrajOpt was also proven versatile in a series of different tasks presented in [15].

The work of [3] proposes a joint-space trajectory representation using Reproducing Kernel Hilbert Spaces. They elaborate on the optimization process and update rules with respect to smoothness and obstacle avoidance constraints and show to perform better than CHOMP in a simulated scenario. There is a strong motivation for using such representations as they make reasoning about smoothness, described by acceleration, jerk, snap etc., trivial. Unfortunately, the many open parameters make this method hard to tune for any practical application. The authors propose using different kernels which all come with their respective parameters on top of the parameters of the proposed optimization method. Unfortunately, there was no suggested method to tune the parameters making it hard to apply in a different problem or even replicate the results. Moreover, adding additional constraints in a gradient-dependent system requires adapting the optimization process itself.

The work of [1] presents a method for motion planning using Gaussian Processes (GPs). A trajectory is represented as a continuous valued function that maps time to robot states. Trajectory optimization is performed using probabilistic inference. A Gaussian Process (GP) is used to provide a prior function that encourages smoothness. Collision free trajectories are encouraged by a likelihood function. The posterior distribution of the GP prior with the likelihood function is used to calculate the *maximum a posteriori* (MAP) estimate of the trajectory. This method, coined Gaussian Process Motion Planner 2 (GPMP2) provided comparable success rates with the state-of-the-art while requiring much less computational effort. Despite the presented benefits the presented approach is limited in various ways. Joint limits are respected only by clamping their maximum values in the initial trajectory. This can lead to situations that the found trajectory may not be valid after the clamping operation and there is nothing explicitly encouraging the optimizer to respect joint limits. Moreover, trajectory smoothness is only encouraged by using a prior having no acceleration. While this may be enough in basic cases, it does not guarantee that the final generated trajectory will be smooth. Finally, the presented method does not consider the generated trajectory length in either euclidean or joint space, nor it allows for the introduction of waypoints that must be reached in various stages of the generated trajectory.

III. Methods

In this section, we describe in detail the proposed method (i.e., TEB2MP) after a brief description of TEB.

A. Timed-Elastic Band

Timed Elastic Bands was originally developed for mobile base robots [5] navigation planning. It is formulated as a sequence of $n+1$ robot poses $\in \mathrm{SE}(2)$ linked together from an initial configuration to a goal:

$$\mathbf{X} = \{\mathbf{x}_0, \ldots, \mathbf{x}_n\} \tag{1}$$

Consecutive poses are connected by time intervals,

$$\Delta T = \{\Delta t_1, \ldots, \Delta t_n\} \tag{2}$$

with Δt_i denoting the time interval required for the robot to move from a pose $\mathbf{x}_{i-1}$ to $\mathbf{x}_i$ along the trajectory.

Defining the set of pairs,

$$\mathbf{Q} = \{(\mathbf{x}_1, \Delta t_1), \ldots, (\mathbf{x}_n, \Delta t_n)\} \tag{3}$$

the TEB is formulated as a multi-objective optimization problem:

$$\mathbf{Q}^* = \underset{\mathbf{Q}}{\operatorname{argmin}} \sum_{k,i} w_k e_{k,i} \tag{4}$$

which can be solved by means of a least-squares non-linear solver. The terms w_k are weights used to balance the different contributions, and $e_{k,i}$ the i-th cost contributions of objective k during the interval Δt_i.

B. Timed-Elastic Bands for Manipulation Motion Planning

To achieve a smooth trajectory that respects joint limits and avoids obstacles, a trajectory optimization method is proposed.

Unlike [5] which TEB's states lie in Cartesian space, our states are expressed in the joint space so that we have:

$$\Theta = \{\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_n\} \tag{5}$$

the joint-space trajectory, where $\boldsymbol{\theta}_i$ is the complete joint state of the arm at time i, and

$$\mathbf{Q} = \{(\boldsymbol{\theta}_1, \Delta t_1), \ldots, (\boldsymbol{\theta}_n, \Delta t_n)\} \tag{6}$$

is our new set of pairs.

First, an initial trajectory Θ_0 is generated in order to seed the optimization. Given an initial configuration $\boldsymbol{\theta}_0$ and a final pose $\mathbf{x}_n$ of the end-effector, the desired final configuration $\boldsymbol{\theta}_n$ is retrieved by means of inverse kinematics using Trac-IK. Θ_0 is then generated using an interpolation algorithm.

The next step is to represent the problem as an optimization graph $\mathbf{G}$. It is constructed with vertices that represent robot configurations and time increments, the variables to be optimized. Vertices are then connected by edges that constrain different aspects of the robot motion (velocity, continuity, obstacle-avoidance...), and are further detailed hereafter. A vertex in the presented system is a unit of data subject to optimization. Vertices do not need to be homogeneous across the graph, in fact TEB2MP employs two types: a robot state vertex defined by the robot states $\boldsymbol{\theta}_i$; a time difference vertex Δt_i defines the time difference between two consecutive robot states. The robot state is represented by a set of joint position values, denoted by $\boldsymbol{\theta}_i$ and the corresponding Cartesian-space end-effector pose $\mathbf{x}_i$. The number of vertices used is the same as the number of points making up the initial trajectory Θ_0. These vertices are then connected by edges describing constraints in the robot's motion as error functions to be minimized. Their connections to vertices define the parameters of the functions they represent. The graph structure underlying the TEB2MP problem is highlighted in Fig. 2. It shows a sub-graph in the form of a factor graph for readability. Algorithm 1 summarizes the overall method.

The different constraints employed by the method are detailed below.

1) Joint-Space Position, Velocity, and Acceleration Limits: The first set of edges are limiting the joints positions, velocities and accelerations by enforcing upper and lower boundaries (i.e. inequality constraints).

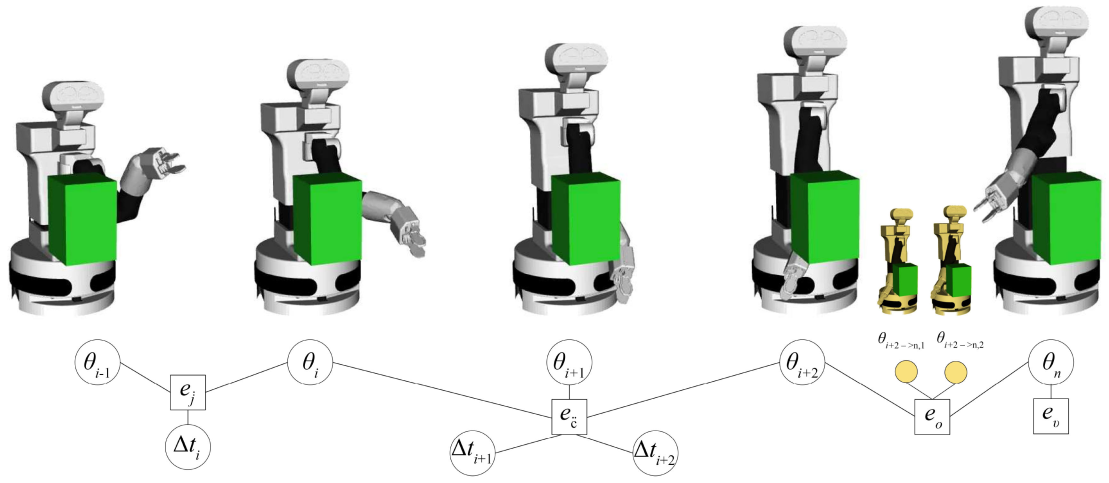


Fig. 2. An illustration of a non-specific TEB2MP sub-graph: Vertices (circles) $\boldsymbol{\theta}$ represent different configurations over time while Δt are the time intervals. Vertices are connected through factors (squares) representing the various cost functions.

Algorithm 1: Timed Elastic Bands for Manipulation Motion Planning.

```
Input: $Start, Goal$
Output: $Trajectory$
 1: $\Theta_0 \leftarrow$ GENERATEINITIALTRAJECTORY$(Start, Goal)$
 2: $\mathbf{G} \leftarrow$ BUILDGRAPH$(\mathbf{Q})$
 3: $i \leftarrow 1$
 4: $e \leftarrow$ GETERROR$(\mathbf{G})$
 5: while $e > 0$ or $i \leq NumIter$ do
 6:     $\mathbf{G} \leftarrow$ NLSOPTIMIZATION$(\mathbf{G})$
 7:     $e \leftarrow$ GETERROR$(\mathbf{G})$
 8:     $i \leftarrow i + 1$
 9: $Trajectory \leftarrow$ GETTRAJECTORY$(\mathbf{G})$
10: return $Trajectory$
```

Similarly to [4] we use a non-linear least-squares solver to optimize the problem. Inequality constraints are approximated by two-sided quadratic penalties.

The joint position limits are inspired by Eq. (83) in [16]. It is adjusted so that $P : \mathbb{R}^{dof} \to [0, 1]$ where 0 is the value given when the joint is within the limits and 1 is when outside.

$$\mathbf{e}_l = \begin{cases} 0, & \vartheta_i \in [\vartheta_i^{\mathrm{L}}, \vartheta_i^{\mathrm{U}}] \\ \left\| exp\left(\frac{(\vartheta_i - \vartheta_i^{\mathrm{L}}) * (\vartheta_i^{\mathrm{U}} - \vartheta_i)}{(\vartheta_i^{\mathrm{U}} - \vartheta_i^{\mathrm{L}})^2}\right) \right\|, & \text{otherwise} \end{cases} \tag{7}$$

where the variable ϑ_i denotes the state of a single joint which may be an angle or a position, for rotary and linear joints respectively. We define $\boldsymbol{\theta} = [\vartheta_0, \ldots, \vartheta_{\text{dof}}]^T$ to represent a full joint state configuration. Variables $\vartheta_i^{\mathrm{L}}, \vartheta_i^{\mathrm{U}}$ are the joint limits for ϑ_i, while $\mathbf{e}_l$ is a vector containing the individual error $e_{l,i}$ for each joint.

The rest of the constraints hereafter use a common penalty function,

$$\beta(\nu, \nu^{\mathrm{L}}, \nu^{\mathrm{U}}) = \begin{cases} -\nu + \nu^{\mathrm{L}}, & \text{if } \nu < \nu^{\mathrm{L}} \\ +\nu - \nu^{\mathrm{U}}, & \text{if } \nu > \nu^{\mathrm{U}} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

with ν a constrained variable, ν^{L} and ν^{U} respectively ν's lower and upper bounds, and $<, >$ are element-wise comparisons.

Velocity and acceleration are approximated by using backward finite differencing through a sliding window over TEB of, respectively, the past two and three states.

$$\dot{\boldsymbol{\theta}}_i = \frac{\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i-1}}{\Delta t_i} \tag{9}$$

$$\ddot{\boldsymbol{\theta}}_i = \frac{\frac{\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i-1}}{\Delta t_i} - \frac{\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{i-2}}{\Delta t_{i-1}}}{\Delta t_i + \Delta t_{i-1}} \tag{10}$$

The respective errors are then,

$$\mathbf{e}_{\dot{j}} = \beta(\dot{\boldsymbol{\theta}}_i, \dot{\boldsymbol{\theta}}^{\mathrm{L}}, \dot{\boldsymbol{\theta}}^{\mathrm{U}}) \tag{11}$$

$$\mathbf{e}_{\ddot{j}} = \beta(\ddot{\boldsymbol{\theta}}_i, \ddot{\boldsymbol{\theta}}^{\mathrm{L}}, \ddot{\boldsymbol{\theta}}^{\mathrm{U}}) \tag{12}$$

2) Cartesian-Space Velocity and Acceleration Limits: Similarly to Eqs. (9)–(12), TEB2MP also supports velocity/acceleration constraints in Cartesian-space where the pose of a target frame is retrieved through forward kinematics. Given,

$$\mathbf{x} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \triangleq (\mathbf{t}, \mathbf{q}) \in \mathrm{SE}(3) \tag{13}$$

with $\mathbf{t}$ a position vector and $\mathbf{R}$, $\mathbf{q}$ orientation representations (respectively rotation matrix and unit quaternion), we have,

$$\dot{\mathbf{x}}_i = \frac{\mathbf{x}_i - \mathbf{x}_{i-1}}{\Delta t_i} \tag{14}$$

$$\ddot{\mathbf{x}}_i = \frac{\frac{\mathbf{x}_i - \mathbf{x}_{i-1}}{\Delta t_i} - \frac{\mathbf{x}_{i-1} - \mathbf{x}_{i-2}}{\Delta t_{i-1}}}{\Delta t_i + \Delta t_{i-1}} \tag{15}$$

with the respective errors,

$$\mathbf{e}_{\dot{c}} = \beta(\dot{\mathbf{x}}_i, \dot{\mathbf{x}}^{\mathrm{L}}, \dot{\mathbf{x}}^{\mathrm{U}}) \tag{16}$$

$$\mathbf{e}_{\ddot{c}} = \beta(\ddot{\mathbf{x}}_i, \ddot{\mathbf{x}}^{\mathrm{L}}, \ddot{\mathbf{x}}^{\mathrm{U}}) \tag{17}$$

3) C^k Smooth Curve: The smoothness cost function aims at pushing the optimized trajectory onto a smooth (differentiable) curve on a manifold. First a few definitions are introduced.

A Lie group is an analytically differentiable (aka smooth) manifold. It implies the existence of a unique linear space tangent to any point of it. Given χ a point on a manifold space $\mathcal{M}$, a vector space tangent $\mathcal{M}$ at χ exists and is represented by $\mathcal{T}_{\mathcal{M}}(\chi)$.

In the scope of this work, we consider elements of the group $\mathrm{SE}(3)$. Its tangent space is $\mathfrak{se}(3)$ whose elements are expressed in the isomorphic Cartesian space such that $\mathbf{v} \in \mathbb{R}^6$. The mapping relating $\mathbb{R}^6$, Lie algebra $\mathfrak{se}(3)$ and Lie group $\mathrm{SE}(3)$ is as follows:

$$\mathbf{v} \in \mathbb{R}^6 \underset{\simeq}{\overset{\simeq}{\rightleftharpoons}} \mathfrak{se}(3) \underset{\log(\cdot)}{\overset{\exp(\cdot)}{\rightleftharpoons}} \mathbf{x} \in \mathrm{SE}(3) \tag{18}$$

where $\log(\cdot)$ and $\exp(\cdot)$ map the SE(3) element to/from $\mathbb{R}^6$.

We define the operators $\oplus$ and $\ominus$ such that:

$$\mathbf{x} \oplus \mathbf{v} = \mathbf{x} \circ \exp(\mathbf{v}) \tag{19}$$

$$\mathbf{v} \oplus \mathbf{x} = \exp(\mathbf{v}) \circ \mathbf{x} \tag{20}$$

$$\mathbf{x}_a \ominus \mathbf{x}_b = \log(\mathbf{x}_b^{-1} \circ \mathbf{x}_a) \tag{21}$$

The smoothness constraint proposed here results from a revisited formulation of a geometric algorithm to generate smooth curve on Lie manifold [17]. Given two poses $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, the method in [17] allows for interpolating a pose $\mathbf{x}_t$ with $\mathbf{t} \in [i, i+1]$, so that it lies on a smooth curve connecting $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$.

The interpolation algorithm is adapted so that, given three consecutive poses $\mathbf{x}_{i-1}$, $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ of the end-effector in Cartesian-space, and their associated time intervals Δt_i and Δt_{i+1}, the cost term can be described as follows:

$$\mathbf{l} = \mathbf{x}_{i-1} \oplus (t \cdot \mathbf{v}_{i-1}) \tag{22}$$

$$\mathbf{r} = \mathbf{x}_{i+1} \oplus ((t-1) \cdot \mathbf{v}_{i+1}) \tag{23}$$

$$\boldsymbol{\rho} = \log(\mathbf{r} \circ \mathbf{l}^{-1}) \tag{24}$$

$$\hat{\mathbf{x}}_i = (\phi(t) \cdot \boldsymbol{\rho}) \oplus \mathbf{l} \tag{25}$$

where $t = \Delta t_i / (\Delta t_i + \Delta t_{i+1})$ and $\mathbf{v}_i$ is the tangent to the curve at point i:

$$\mathbf{v}_i = \mathbf{x}_i \ominus \mathbf{x}_{i-1} \tag{26}$$

The resulting error function is then,

$$\mathbf{e}_s = \hat{\mathbf{x}}_i \ominus \mathbf{x}_i \tag{27}$$

The smooth aspect of Eq. (25) lies partly in the real valued smoothing function $\phi(t)$. The reader can refer to [17] for more detailed reference and proofs of these critical properties.

4) Collision Avoidance: This set of edges aims at avoiding collisions with the environment as well as with the robot itself. To detect collisions the robot is modeled by Axis-Aligned Bounding Box (AABB) representations and collisions are detected using Flexible Collision Library (FCL) [18]. This method efficiently approximates complex shapes using bounding box segments in otherwise expensive computations.

TEB2MP implements the collision error as a cost function. It is computed as the sum of the overlapping area of the AABB with the environment and the distance of the robot with the closest obstacle Eq. (29). To ensure continuity in collision checking, TEB2MP proposes inter-vertex collision cost computation by linearly interpolating N_c configurations between the two corresponding vertices, computing the cost over these states and the first vertex. A visual example is shown on the right side of Fig. 2 (smaller intermediate yellow robots). The collision cost function reads,

$$g(\boldsymbol{\theta}) = \breve{V}(\boldsymbol{\theta}) + \beta(\Gamma(\boldsymbol{\theta}), d^{\mathrm{L}}) \tag{28}$$

$$\mathbf{e}_o = [g(\boldsymbol{\theta}_1),\, g(\boldsymbol{\theta}_{1\to 2,1}), \ldots,\, g(\boldsymbol{\theta}_{1\to 2,N_c})]^T \tag{29}$$

where $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ are the two waypoint vertices associated to each instance of this edge, $\boldsymbol{\theta}_{1\to 2,i}$ denotes the i-th linearly interpolated intermediate state between $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, $\breve{V}$ is the overlapping volume of the robot at state $\boldsymbol{\theta}$, Γ is the distance to the closest collision point and d^L denotes the minimum distance to keep from obstacles.

5) Cartesian Viapoint: Viapoints are used to push the robot end-effector to pass through specific points ($\mathbf{x}_v$) in Cartesian-space. The cost function reads,

$$\mathbf{e}_v = \begin{bmatrix} \|\mathbf{p}_i, \mathbf{p}_v\| \\ d(\mathbf{q}_i, \mathbf{q}_v) \end{bmatrix} \tag{30}$$

where $\mathbf{x}_v$ is a Cartesian pose of the end-effector, $\mathbf{x}_i = (\mathbf{p}_i, \mathbf{q}_i)$ and $d(\mathbf{q}_1, \mathbf{q}_2)$ is the unit quaternion distance ([19], Eq. 10). Currently, TEB2MP uses it to implement goal tolerance.

IV. Experiments

To validate the proposed TEB2MP method a series of simulated experiments were designed. This section describes the experimental setup along with results extracted from simulations.

TEB2MP is compared against state-of-the-art planners, RRT-Connect, STOMP and TrajOpt in three different scenarios. Every scenario uses the *TIAGo*[1] robot which provides an 8 *Degrees of Freedom* (DoF) manipulator system: a 7 DoF arm on an elevating torso. In the first scenario the robot has to move its arm toward an end-effector goal position in an empty world. In the second scenario the robot has to plan a path around a free-standing box placed in front of it (Fig. 3). In the third scenario the robot has to perform a manipulation task in an industrial setting first introduced in [1]. In this setting, the robot has to reach the top of a shelf on it right-hand side and move to a bench on its left-hand side as depicted in Fig. 4. Our TEB2MP implementation relies on the *General Graph Optimization* (g2o) library [20] for building and solving the non-linear least-square problem and on the *manif* library [21] for the Lie theory aspect. All the scenarios are simulated using *Robot Operating System* (ROS) and *MoveIt!*, the output of these planners are directly applicable to real robots using *ros_control* [22].

Given the probabilistic nature of some planners multiple trials are performed for each scenario and statistical results are presented. Specifically, each planner is executed a 100 times for each scenario and results were compiled from these trials. For TrajOpt and STOMP a linear initial trajectory is used that is generated by linearly interpolating between the start and end

[1]http://wiki.ros.org/Robots/TIAGo

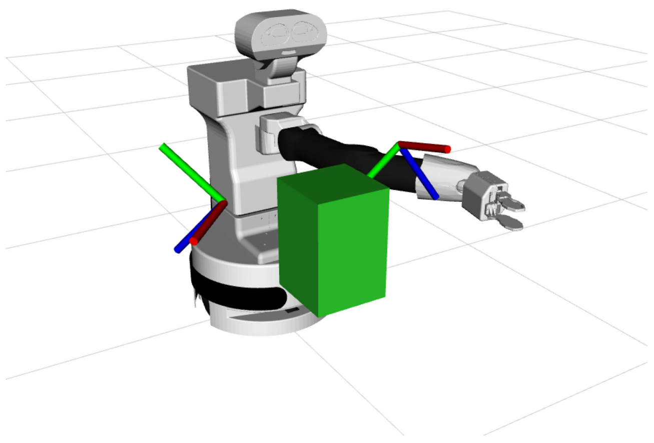


Fig. 3. Box scenario: Starting from the right hand side of the box, the robot has to reach the goal on the left side of the box.

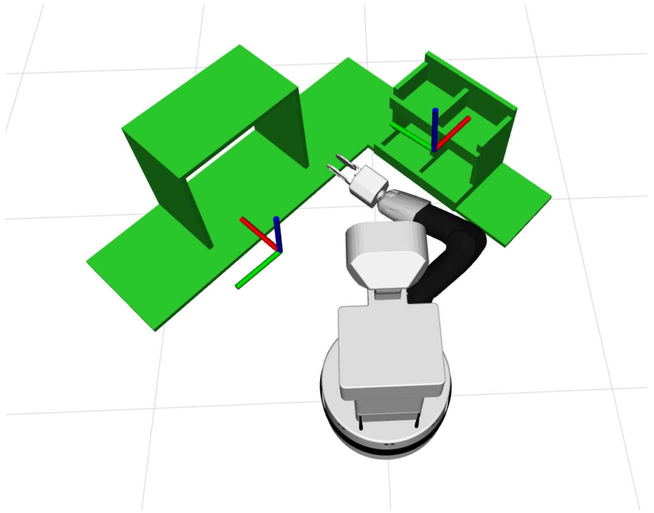


Fig. 4. Industrial manufacturing scenario: Starting from the top right shelf, the robot has to reach the goal on the left of the bench.

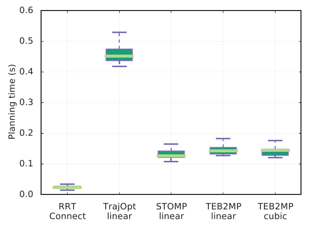
joint states. For TEB2MP both a linear and a cubic polynomial interpolation method is benchmarked. This is done to see the behaviour of the proposed approach using different initialization methods. Additionally, to achieve a fair comparison TEB2MP, STOMP and TrajOpt are initialized with the same length of initial trajectories, each one having 20 points.

To ensure equal ground for the algorithms all experiments were conducted using the *MoveIt!* framework [23] and were ran on a system having an Intel i7-4710MQ, 2.5 GHz CPU, 8 GB of RAM and running on Ubuntu 16.04.

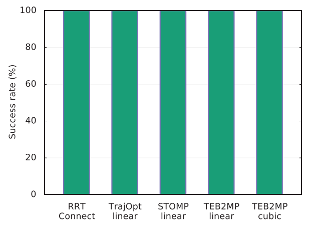
The first evaluation metric is the time required to find a valid plan. As the robot operates in the real environment it is important to be able to find plans in a timely manner. The second metric is the success rate of each planner at each scenario. It is used to demonstrate the ability of each planner to find a path. The third metric is the smoothness of each generated path measured in rad/s^2. Smoother trajectories require less acceleration or deceleration, therefore putting less stress on the mechanical parts of the robot. It must be noted that for the smoothness metric, the lower the score the better. Finally, the last metric is distance from joint limits. This metric is used to show how far on average each

TABLE I
METRICS USED IN OUR EXPERIMENTS

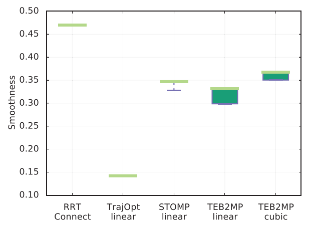
Metric	Description
Planning time in s	τ
Success rate in %	s_r
Smoothness in rad/s^2	$\ddot{\boldsymbol{\theta}}$
Distance from joint limits	$\dfrac{\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\vartheta_{i,j}-\vartheta_{i,j}^{\min}}{\vartheta_{i,j}^{\max}-\vartheta_{i,j}^{\min}}}{n\cdot m}$



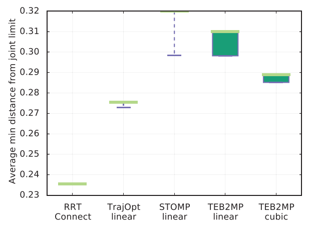
(a) Planning time



(b) Success rate



(c) Smoothness



(d) Distance from joint limits

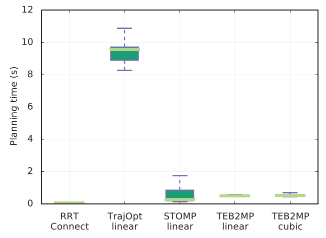
Fig. 5. Empty scenario experiment results from 100 queries with each planner. All planners manage to find a plan in every query. RRTConnect is performing better regarding planning time but worse in the other metrics. TrajOpt is the best in smoothness, while STOMP is the best in joint limit distance. The proposed TEB2MP method performs on par with the other methods.

joint is from its limits. Being further away from the joint limits is beneficial as plans that are close to the joint limits may be harder to execute due to hardware lock-in or precision errors. In addition, operating close to the joint limits, may have a negative effect in the manipulability [16] and the ability to find new plans.

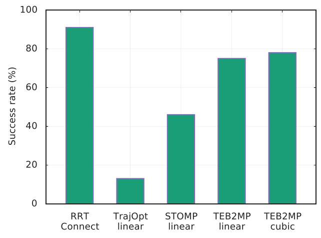
A. Empty Environment

This section describes the results of the empty world scenario which acts as a baseline experiment. The start and the goal states for the robot are marked by coordinate frames in Fig. 3 but the collision box was removed.

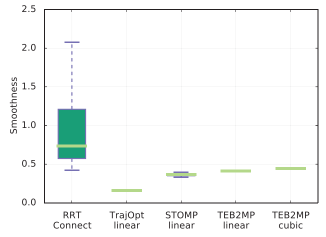
Statistical results for the various tested planners can be seen in Fig. 5. Regarding the planning time metric we can see that RRTConnect performs better than any other planner. STOMP and TEB2MP perform on average six times slower than RRT-Connect. TrajOpt comes last requiring on average 2.5 times the time of STOMP and TEB2MP. Success rate is 100% for every planner. An expected result since the scene is empty. Regarding the smoothness metric TrajOpt outperforms every other method. STOMP and the two variants of TEB2MP are performing equally good, while RRTConnect performs worse than any other method. Finally, regarding the average normalized minimum joint limit distance STOMP performs best. TEB2MP with linear initialization is second, while cubic initialization comes third.



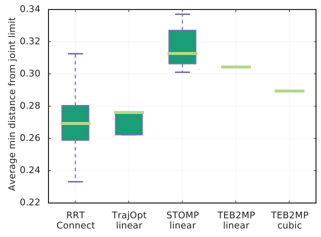
(a) Planning time



(b) Success rate

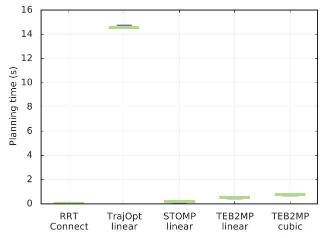


(c) Smoothness

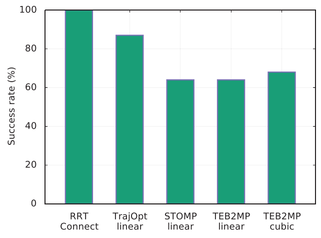


(d) Distance from joint limits

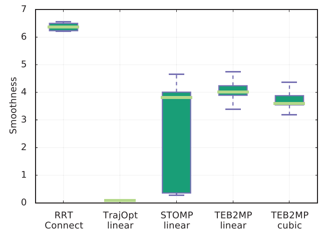
Fig. 6. Floating box scenario experiment results from 100 queries with each planner. RRTConnect has the best planning time and success rate but the worst smoothness and distance from joint limits. TrajOpt has the best smoothness, while STOMP has the best distance from joint limits. Both have limited success rate though. TEB2MP provides an overall good performance that is consistently close to the best in all metrics.



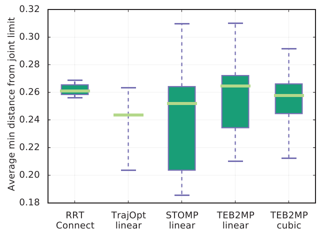
(a) Planning time



(b) Success rate



(c) Smoothness



(d) Distance from joint limits

Fig. 7. Industrial scenario experiment results from 100 queries with each planner. RRTConnect is the best planner regarding planning time and success rate. TrajOpt performs better in terms of smoothness. Regarding the distance from joint limits all methods perfom equally good. TEB2MP is performing consistently good in all the cases.

TrajOpt performs a bit worse than the cubic TEB2MP, while RRTConnect performs the worst of all.

B. Box Environment

Fig. 3 depicts the second scenario with a floating box where the planners were tested. The results for this experiment are presented in Fig. 6.

RRTConnect is still the best regarding the planning time. STOMP has high variance performing better than TEB2MP in at least 50% of the cases but there were cases that it performed much worse. Both flavours of TEB2MP required more time on average but performed more consistently than STOMP. TrajOpt again required the most time to find a plan. Regarding the success rate of finding a plan, RRTConnect managed to find a plan in 90% of the cases being the top planner in that metric. TEB2MP is second achieving a score of almost 80%. STOMP provided a success rate close to 45%, while TrajOpt had the lowest score. When it comes to smoothness, TrajOpt performs better than the other planners. STOMP performs marginally better than TEB2MP. On the other hand RRTConnect performs much worse and shows big variance in its results. Finally, regarding the joint limit distance, STOMP performs better than any other method. TEB2MP is slightly worse but shows a much more consistent performance, while RRTConnect and TrajOpt are the worst.

C. Industrial Environment

The last and most complex scenario shown in Fig. 4 is an industrial setting presented in [1]. The results can be seen in Fig. 7.

As with the previous experiments RRTConnect performs better than any other planner in terms of planning time. STOMP is second, with the TEB2MP following. The least performing in this metric is again TrajOpt. Regarding the success rate RRTConnect manages to find a path in all the cases. TrajOpt is second, with TEB2MP and STOMP following a bit behind. For the smoothness criterion TrajOpt performs the best. STOMP is coming second, showing a better behaviour than TEB2MP in half of the paths that it found. The other 50% performs the same as the TEB2MP. RRTConnect performs much worse in this metric. Finally, in the joint limits distance all the methods perform comparably good. It is noteworthy that RRTConnect performs consistently good in all the plans found.

Since the examined environment mimics one of the environments presented in [1] a comparison can be attempted with TEB2MP. Although our experiments do not include GPMP2 with the *TIAGo* robot, an approximate comparison can be made as the *PR2* robot is fairly similar from a motion planning perspective. In the same environment with the *PR2* robot GPMP2 generated plans reported an average planning time of 20 ms, adding this to the computation time for the *Signed Distance Field* (SDF) used there, a GPMP2 plan takes 800 ms, similarly to the proposed method of this work method without using SDF. Regarding the success rate [1] reports a single number for the *PR2* robot while two different problem sets and scenes were used. Therefore, a direct comparison cannot be made. Nevertheless, even in the case that GPMP2 has a higher success rate, no guarantee is provided regarding the smoothness of the trajectory or the distance from joint limits.

V. Conclusions and Future Work

The work presented in this letter studies the manipulation motion planning problem. It proposes the *Timed-Elastic Band for Manipulation Motion Planning* (TEB2MP) planner. This planner allows for multiple aspects of the motion plan to be constrained or optimized during planning. Examples include, position, acceleration or velocity constraints in Cartesian and joint-space and path smoothness. The proposed approach is compared against state-of-the-art methods in three different scenarios of increasing difficulty. Results show that in the worst case it performs in a comparable manner with the state-of-the-art in every chosen metric. In general it provides an overall improved performance which is consistently delivered with a lower variance.

Despite the demonstrated benefits of the proposed approach there are several directions for future exploration that could further improve it. Firstly, the optimization could be improved by using a *Sequential Quadratic Programming* (SQP) solver with hard constraints as proposed in the recent work of [24]. Compared to the currently used *Linear Block Solver* (LBS), the use of SQP will move towards a pure *Model Predictive Control* (MPC) formulation which can generalize better. Additionally, the planner's performance in highly cluttered environments could be improved by dynamically increasing the resolution of the planner around often colliding points. Different initialization methods could improve the success rate of the planner as well. For example, instead of using linear or cubic interpolation, joint or Cartesian trajectories from other motion planners such as RRTConnect could be used to initialize the planner. Finally, the planning speed can be improved by introducing more efficient collision checking methods, such as precomputed SDFs as done in the work of [1] or locally updated collision environments.

As further research targets one could look at the nature of the original TEB implementation. Given that there is the notion of time in the planning procedure one could extend the planners capabilities by introducing dynamic information for both obstacles and goals. This way a fully dynamic environment can be supported. In addition, the planner could be extended to allow sequential tasks by adding multiple goals. This would allow for planning more complex, multi-step tasks. Finally, the presented approach could be integrated with systems having complex dynamics as the ones presented in [25]. The limits in the acceleration and the velocity of the manipulator imposed by the proposed approach can potentially facilitate the control task of such a highly coupled dynamic system.

References

[1] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using Gaussian processes and factor graphs," *Robot.; Sci. Syst.*, vol. 12, 4 pp., 2016.

[2] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, vol. 2, pp. 995–1001.

[3] Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa, "Functional gradient motion planning in reproducing kernel Hilbert spaces," *Robot.; Sci. Syst. XII*, vol. 12, 4 pp., 2016.

[4] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *Proc. Eur. Control Conf.*, 2015, pp. 3352–3357.

[5] C. Röesmann, W. Feiten, T. Wöesch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *Proc. ROBOTIK 2012; 7th German Conf. Robot.*, 2012, pp. 1–6.

[6] L. V. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[7] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, Tech. Rep. 98-11, 1998.

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

[9] D. J. Webb and J. V. D. Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," *IEEE Int. Conf. Robot. Autom.*, 2013, pp. 5054–5061.

[10] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa,"CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 489–494.

[11] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox, "Space-time functional gradient optimization for motion planning," in *Proc. Int. Conf. Robot. Autom.*, 2014, pp. 6499–6506.

[12] K. He, E. Martin, and M. Zucker, "Multigrid CHOMP with local smoothing," in *Proc. 13th IEEE-RAS Conf. Humanoid Robots (Humanoids)*, 2013, pp. 315–322.

[13] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 4569–4574.

[14] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," *Robot.; Sci. Syst.*, vol. 9, no. 1, 2013, pp. 1–10.

[15] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, 2014.

[16] M.-J. Tsai, "Workspace geometric characterization and manipulability of industrial robots," Ph.D. dissertation, Dept. Mech. Eng., The Ohio State University, Columbus, OH, USA, 1986.

[17] J. Jakubiak, F. S. Leite, and R. C. Rodrigues, "A two-step algorithm of smooth spline generation on Riemannian manifolds," *J. Comput. Appl. Math.*, vol. 194, no. 2, pp. 177–191, 2006.

[18] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 3859–3866.

[19] A. Ude, "Filtering in a unit quaternion space for model-based object tracking," *Robot. Auton. Syst.*, vol. 28, no. 2–3, pp. 163–172, 1999.

[20] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3607–3613.

[21] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," Institut de Robòtica i Informàtica Industrial, Barcelona, Spain, Tech. Rep. IRI-TR-18-01, 2018. [Online]. Available: http://arxiv.org/abs/1812.01537

[22] S. Chitta *et al.*, "ros_control: A generic and simple control framework for ROS," *J. Open Source Softw.*, vol. 2, 2017, Art. no. 456.

[23] I. A. Sucan and S. Chitta, "Moveit!," 2013. [Online]. Available: http://moveit. ros. org

[24] M. Biel and M. Norrlöf, "Efficient trajectory reshaping in a dynamic environment," in *Proc. IEEE 15th Int. Workshop Adv. Motion Control*, 2018, pp. 54–59.

[25] K. Baizid *et al.*, "Experiments on behavioral coordinated control of an unmanned aerial vehicle manipulator system," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 4680–4685.