2013 European Conference on Mobile Robots (ECMR)
Barcelona, Spain, September 25-27, 2013

Efficient Trajectory Optimization using a Sparse Model

Christoph Rösmann[1], Wendelin Feiten[2], Thomas Wösch[2], Frank Hoffmann[1] and Torsten Bertram[1]

Abstract— **The "timed elastic band" approach optimizes robot trajectories by subsequent modification of an initial trajectory generated by a global planner. The objectives considered in the trajectory optimization include but are not limited to the overall path length, trajectory execution time, separation from obstacles, passing through intermediate way points and compliance with the robots dynamic, kinematic and geometric constraints. "Timed elastic bands" explicitly consider spatial-temporal aspects of the motion in terms of dynamic constraints such as limited robot velocities and accelerations. The trajectory planning operates in real time such that "timed elastic bands" cope with dynamic obstacles and motion constraints. The "timed elastic band problem" is formulated as a scalarized multi-objective optimization problem. Most objectives are local and relate to only a small subset of parameters as they only depend on a few consecutive robot states. This local structure results in a sparse system matrix, which allows the utilization of fast and efficient optimization techniques such as the open-source framework "g2o" for solving "timed elastic band" problems. The "g2o" sparse system solvers have been successfully applied to VSLAM problems. This contribution describes the application and adaptation of the g2o-framework in the context of trajectory modification with the "timed elastic band". Results from simulations and experiments with a real robot demonstrate that the implementation is robust and computationally efficient.**

I. Introduction

Trajectory planning finds an optimal collision free trajectory that complies with the robots kinematic and dynamic motion constraints. This paper focuses on trajectory modification assuming that a global planner generated an initial feasible path beforehand [1]. In particular in the context of service robotics the dynamic modification of a preplanned path is preferable over offline trajectory planning. Online modification copes with changes of a dynamic environment by incorporating the most recent sensor data for local refinement of the trajectory. In most realistic applications the model of the environment is subject to continuous change due to partial, incomplete maps and dynamic obstacles. Furthermore, the (re-)computation of a large scale global path is often not feasible in real-time applications. This observation leads to approaches which modify a path locally, such as the "elastic band" proposed by [2], [3].

Later the original approach was extended to nonholonomic kinematics [4], [5], [6] and robotic systems with many degrees of freedom [7]. [8] proposed a method, where an initial path is deformed using optimization techniques. The trajectory, i.e. the velocities along the path, are not optimized. The time parameter is used to control the modifications of the path as the optimization proceeds. The planner considers the nonholonomic constraints.

[9] deals with the deformation of trajectories rather than paths by an explicit consideration of temporal information. The deformation is decomposed into an obstacle avoidance step using repulsive forces and a connectivity maintenance step. Based on this work [10] proposes a single step approach that combines external deformation with internal connectivity forces. Both methods support general state transition models and allow for spatial-temporal obstacle avoidance. In contrast, our approach is based on a graph-optimization formulation, operates with general optimization solvers and time optimality is an explicit objective. Other methods which directly optimize trajectories are presented in [11], [12]. In our case, a parametric path is augmented with velocity profiles that respect the kinodynamic constraints of the platform. The approach starts with an initial path found by a global planner and represents it by a compact spline-based path model also used in [13]. This path model exposes a set of higher-level parameters to the optimization that iteratively adapts the shape of the curvature continuous path to reduce an objective function such as the time of travel. The main difference to our work is that it trades in the precision of the analytic model for a discretized trajectory model that allows it to employ a highly sophisticated, efficient optimization algorithm, enabling trajectory refinement in real-time.

Most recent approaches for trajectory modification dealing with robot arms with many degrees of freedom use a discretized representation of the trajectory in configuration space (see [15], [16]). The proposed objective function contains a finite difference matrix to smooth the resulting trajectory and to additionally satisfy constraints like obstacle avoidance. The CHOMP algorithm relies on a covariant gradient descent method which explicitly requires the gradients of each objective, whereas STOMP uses a stochastic trajectory optimization technique without explicit knowledge of gradients. Both approaches include temporal information only in implicit manner by defining a specific discretization and task duration. The differences to our approach are detailed in Section IV.

In [17] the authors introduced a new approach called "timed elastic band" which explicitly augments "elastic band" with temporal information. The proposed extension allows the consideration of the robot's dynamic constraints and direct modification of trajectories rather than paths. The "timed elastic band" is formulated as a scalarized multi-objective optimization. The structure of the underlying optimization problem is sparse as most objectives are local in that they only depend on a few consecutive configurations

[1]Institute of Control Theory and Systems Engineering, Technische Universität Dortmund, Germany
[2]Siemens Corporate Technology, Research Group Robotics, Germany

978-1-4799-0263-7/13/$31.00 ©2013 IEEE

rather than the entire trajectory.

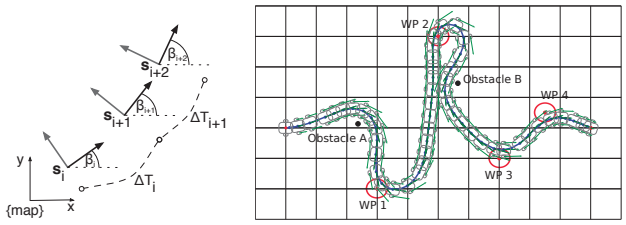
Numerical mathematic provides efficient algorithms for optimization problems with sparse structures that have been applied successfully to tasks such as "visual simultaneous localizing and mapping" (vSLAM) or "sparse bundle adjustment" (SBA) [18]. [19] introduces an open-source C++ framework called "general (hyper-)graph optimization" (g2o) which solves graph based nonlinear optimization problems. An obvious advantage of using a multi-objective optimization framework is the modular formulation of the objectives and constraints.

This paper presents the "timed elastic band" (TEB) approach according to a hyper-graph based nonlinear optimization problem and the implementation with g2o on a mobile robot with a differential drive. The robot moves in a planar environment with three global and two local degrees of freedom. In general, the TEB is suitable for high dimensional state spaces. By considering the temporal information, TEB explicitly considers and controls the robot velocities and accelerations.

We first introduce the general concept of TEB described in [17] in more detail, in particular mapping the problem into a hyper-graph representation, determining initial conditions of the TEB and the algorithmic implementation. Section III presents the connection between the "timed elastic band" and the g2o-framework. Section IV presents and analyzes experimental results. Although the experiments in this presentation describe a non-holonomic robot, the approach is not limited to any particular robot kinematic or dynamic structure. Finally, section V summarizes the results of the TEB and provides an outlook on further work.

II. Timed Elastic Band

A. Definition of Timed Elastic Band (TEB)



(a) (b)

Fig. 1. (a) TEB: sequences of configurations and time differences and (b) Large scenario with consideration of way-points and obstacles

The classic "elastic band" is described in terms of a sequence of n intermediate robot poses $\mathbf{s}_i = [x_i, y_i, \beta_i]^T \in \mathbb{R}^2 \times S^1$, in the following denoted as a configuration defined by the robots position x_i, y_i and orientation β_i in a global frame ($\{\text{map}\}$, Fig. 1(a)):

$$Q = \{\mathbf{s}_i\}_{i=0\ldots n} \quad n \in \mathbb{N} \tag{1}$$

The TEB augments this representation by incorporating the time intervals between two consecutive configurations, resulting in a sequence of $n-1$ time intervals ΔT_i:

$$\tau = \{\Delta T_i\}_{i=0\ldots n-1} \tag{2}$$

Each time interval denotes the time that the robot requires to transit from the current configuration to the next configuration in the sequence Q (Fig. 1(a)). The TEB is defined as a tuple of both sequences:

$$B := (Q, \tau) \tag{3}$$

The key idea is to adapt and optimize the TEB in terms of both configurations and time differences by a scalarized multi-objective optimization red using the weighted sum model:

$$f(B) = \sum_k \gamma_k f_k(B) \tag{4}$$

$$B^* = \underset{B}{\operatorname{argmin}} f(B) \tag{5}$$

in which B^* denotes the optimized TEB and $f(B)$ denotes the underlying global objective function.

Suitable component objective functions f_k for TEB are presented in [17] and belong to two basic types: constraints such as velocity and acceleration limits formulated in terms of penalty functions and objectives functions with respect to the trajectory such as shortest path, fastest execution time or clearance from obstacles. Sparse constrained optimization algorithms are not readily available in robotic frameworks (e.g. ROS) in a freely usable implementation. This motivates the adoption of the g2o-framework in which constraints are formulated as objectives in terms of piecewise continuous, differentiable cost functions that penalize the violation of a constraint.

B. Problem representation as a Hyper-Graph

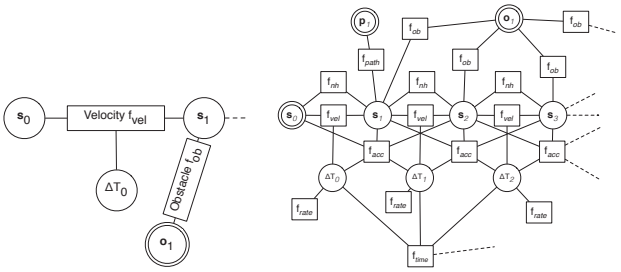
According to equations (4), (5), the TEB is defined as a scalarized multi-objective optimization problem. Most of the required objective functions rely on parameters that only depend on a subset of neighboring configurations of the band:

- Velocity (acceleration) constraints depend on two (three) consecutive configurations and one (two) time differences.
- Clearance from obstacles and homing on intermediate way-points effect a single configuration and its k nearest neighbors (in practice about 3-5).
- Compliance with the robot's non-holonomic constraints involves two adjacent configurations, which are required to be located on a common arc of constant curvature.

Fastest and shortest path are exceptions to the local structure as these objectives that globally depend on all parameters. The fastest path with temporally uniformly spaced configurations is obtained by minimizing the square of the sum of all time differences or alternatively the sum of squared time differences.

This property of locality of TEB results in a sparse system matrix which is represented by a hyper-graph, where the nodes correspond to the configurations and time intervals. The nodes containing parameters that contribute to the same objective function are connected by a corresponding multi-edge. In the following, equation (4) is transformed into a hyper-graph. The definition of a hyper-graph implies that

the number of nodes that are connected by a single edge is not limited to two nodes as in a conventional graph. A hyper-edge is an extension of a conventional edge as its connects multiple nodes with each other. Each objective functions depends on a subset of TEB-states (configurations and time differences), a hyper-edge represents the objective function f_k and connects nodes which correspond to the configurations and time differences that occur as parameters in its evaluation.



(a) Velocity and obstacle objective function formulated as a hyper-graph

(b) Simplified example structure of TEB hyper-graph

Fig. 2. Hyper-Graph structures: nodes (circles) and multi-edges (rectangles)

Fig. 2(a) shows a simple example of a hyper-graph representing a constraint and an objective related to two subsequent configurations $\mathbf{s}_0$, $\mathbf{s}_1$, their time difference ΔT_0 and an obstacle $\mathbf{o}_1 \in \mathbb{R}^2$. Notice, that our specific implementation partitions each configuration node into a position node and an orientation node to facilitate the modular activation and deactivation of objective functions. The velocity objective function puts an upper bound on the distance between $\mathbf{s}_0$ and $\mathbf{s}_1$ that the robot is able to travel within the allocated time ΔT_0. Velocity-multi-edges (f_{vel}) and acceleration-multi-edges (f_{acc}) capture dynamic aspects. The distance between an obstacle and its nearest configuration (in the example $\mathbf{o}_1$ and $\mathbf{s}_1$) is bounded from below by a minimal separation that guarantees a collision free path. The obstacle location is inferred from sensor data provided from the perceptual layer within the robot's architecture. The corresponding obstacle is not subject to graph optimization and is illustrated by a double circle in Fig. 2(a). A larger extract of the TEB hyper-graph with respect to most of the implemented objective functions is illustrated in Fig. 2(b). The objective function in each multi-edge is represented in the overall objective function according to its weight. In addition to the fixed obstacle node, the way-point configurations $\mathbf{p}$ and the initial state $\mathbf{s}_0$ are fixed, too. In our application we use the planner in the control loop, so the initial state is given by the robot's current configuration.

C. Control flow

Fig. 3 shows the control flow of the implemented TEB. In the initialization-phase an initial path is enhanced to an initial trajectory by adding default timing information respecting the dynamic and kinematic constraints. In our case the initial trajectory is composed of piecewise linear

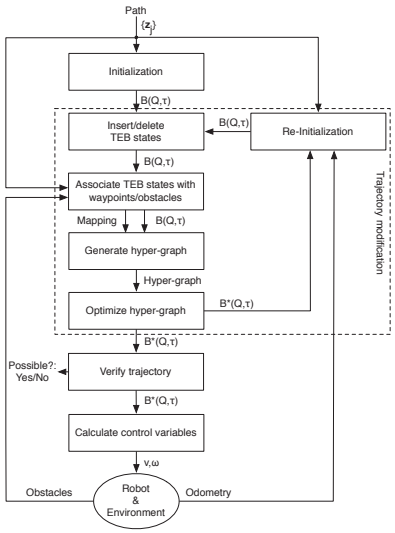


Fig. 3. Control flow of TEB-implementation

segments with a pure rotation followed by a translation. Such a path representation in terms of a polygon is commonly provided by probabilistic roadmap planners [22].

At each trajectory modification step, the algorithm dynamically adds new configurations or removes configurations in order to adjust the spatial and temporal resolution to the remaining trajectory length or planning horizon. The most recent robot perceptions of obstacles and way-points are associated with TEB states. Notice, incorporating an obstacle motion model (e.g. constant velocity) by finding the minimal spatial-temporal distance between TEB and obstacle prediction instead of using the actual pose measurement often leads to a more intuitive solution. In order to challenge our planner and to analyze how it copes with novel perceptions of obstacles at real time we consider obstacles as static in the experiments in section IV. An extension of our approach to dynamic obstacles is straightforward. The optimization problem is transformed into a hyper-graph and solved with the g2o-framework described in more detail in the following section. The g2o-framework optimizes the TEB in batch mode, therefore a new graph is generated at every iteration and initialized with the recent solution. We recommend to perform multiple iterations of the trajectory modification step during a single robot control cycle (in our case four loops including five Levenberg-Marquardt iterations each loop).

The optimized TEB is verified for the violation of hard constraints in which case the robot either stops or the motion planner is reinvoked. Upon successful verification the control variables v and ω are calculated according to the immediate next configuration in the TEB and sent to the robot as motion commands. Prior to every modification, the re-initialization-phase checks for new or modified way-points which is useful if way-points do not originate from a static map but are rather perceived as landmarks from a robocentric perspective with an on board camera or laser scanner.

III. G2O GRAPH OPTIMIZATION

g2o has been developed to solve nonlinear optimization problems with the following particular structure [19]:

$$\mathbf{F}(\mathbf{x}) = \sum_{k=\langle i,j\rangle\in\mathcal{C}} \underbrace{\mathbf{e}_k(\mathbf{x}_i,\mathbf{x}_j,\mathbf{z}_{ij})^T\mathbf{\Omega}_k\mathbf{e}_k(\mathbf{x}_i,\mathbf{x}_j,\mathbf{z}_{ij})}_{\mathbf{F}_k} \quad (6)$$

$$\mathbf{x}^* = \min_x \mathbf{F}(\mathbf{x}) \quad (7)$$

$\mathbf{x}$ denotes the parameters to be optimized, $\mathbf{z}_{ij}$ denotes the constraint between the two parameter blocks $\mathbf{x}_i$ and $\mathbf{x}_j$ and $\mathbf{\Omega}_k$ represents the information matrix of the constraint. The vector $\mathbf{e}_k(\mathbf{x}_i,\mathbf{x}_j,\mathbf{z}_{ij})$ provides the error between constraint and parameters. Notice, that (6) is the objective function typically employed in nonlinear least squares optimization.

The preparation of the TEB-problem for optimization with g2o-framework, see (4), proceeds according to (6). $\mathbf{x}$ is substituted with the TEB-tuple B. In case of scalar error terms, $\mathbf{F}_k$ simplifies to $F_k = \Omega_k e_k^2$, with the substitutions $\Omega_k = \gamma_k$ and $e_k = \sqrt{f_k}$ (using (4)). Note that for trajectory optimization the parameter $\mathbf{x}_i$ in (6) is given by the TEB-State $(\mathbf{s}_i, \Delta T_i)$ in (3).

The g2o-framework requires the definition of nodes and edges. Table I provides an overview of the nodes of the TEB. $\mathbf{b}_i$ denotes the position vector $[x_i, y_i]^T$. For each node, an increment is defined, which maps the local parametrization of the variable to its initial value. In case of incrementing the orientation, the angle is normalized to the interval $[-\pi, \pi)$ after addition of the incremental rotation to the previous orientation. The remaining variables are expressed in Euclidean coordinates, therefore simple addition suffices.

TABLE I
OVERVIEW OF THE NODES OF TEB

Variable	Symbol	Parametriz.	Increment
Position	$\mathbf{b}_i \in \mathbb{R}^2$	$(\Delta x_i\ \Delta y_i)$	$\mathbf{b}_i + \mathbf{\Delta b}_i$
Orientation	$\beta_i \in \mathbb{R}$	$\Delta\beta_i$	$normAngle(\beta_i + \Delta\beta_i)$
Time diff.	$\Delta T_i \in \mathbb{R}$	ΔT_i^*	$\Delta T_i + \Delta T_i^*$

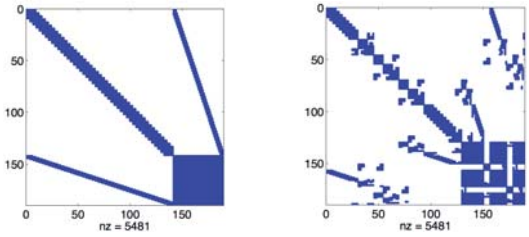
The g2o-framework requires the definition of the error function $e_k = \sqrt{f_k}$ and the weight $\Omega_k = \gamma_k$ for the configuration of each multi-edge. In the experiments presented in this paper, the Jacobian of the error function $\mathbf{e}_k$ in (6) is calculated by numerical approximation by the g2o-framework. In future work it is possible to supply derivatives in analytical form to increase the efficiency of optimization.

Equation (7) is solved with the Levenberg-Marquardt method [19]:

$$(\mathbf{H} + \lambda\mathbf{I})\mathbf{x}^* = -\mathbf{b} \quad (8)$$

$\mathbf{H} = \sum \mathbf{J}_k^T\mathbf{\Omega}_k\mathbf{J}_k$ denotes the system matrix (Hessian), λ is a damping factor which is automatically chosen by the g2o-framework. $\mathbf{x}^*$ represents the optimal TEB-states and $\mathbf{b} = \sum \mathbf{e}_k^T\mathbf{J}_k$ the error term. $\mathbf{J}_k$ denotes the Jacobian which is obtained from linearization at the current solution. An important property of the TEB is the sparseness of $\mathbf{H}$.

Fig. 4(a) illustrates an example of the TEB system matrix $\mathbf{H}$. It is sparse with only 15 percent of non-zero elements.



(a) Example system matrix of TEB

(b) TEB system-matrix after element ordering (AMD)

Fig. 4. System matrix and AMD ordering

In this example, the first 141 states correspond to the 47 configurations $\mathbf{x}_i$ whereas the final states 142-189 denote the time differences ΔT_i. These final states are related to the objective of fastest trajectory, thus this block is dense and connections grow quadratically with the dimension of the TEB.

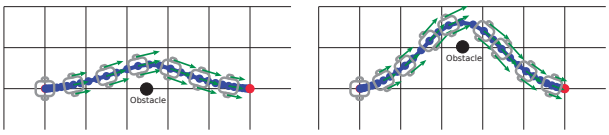
Equation (8) is solved in a numerically efficient way by means of sparse Cholesky decomposition algorithms and a-priori ordering (e.g. AMD, Fig. 4(b)) [20], [21]. The g2o-framework provides two different solvers based on Cholesky decomposition: CHOLMOD and CSparse. In first experiments, the two solvers show no significant difference in terms of the runtime behavior of the optimization. The CSparse solver seems to be slightly faster, thus the experiments in this paper are based on CSparse. It remains unclear which solver is better suited for substantially higher dimensional spaces ([19] prefers CSparse for smaller dimensions and CHOLMOD for higher ones).

IV. EXPERIMENTS AND RESULTS

Simulation and real experiments focus on a non-holonomic robot with a differential drive. The robot simulator is a virtual machine running with Intel Core i7 2x2.3GHz and 4GB RAM. Real robot experiments are performed on a Pioneer 2 with a Siemens Lifebook s6410, Core2Duo, 2.4GHz, 2GB RAM. The robot is equipped with a Hokuyo Laser Scanner.

The solutions are robust with respect to the selection of weights for the proposed objectives in the optimization problem. We set the nonholonomic constraint to 1000 and all other weights to 1.

A. Simulation experiments



(a) Snapshot 1

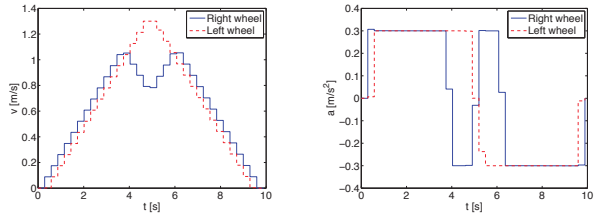
(b) Snapshot 2

Fig. 5. Obstacle avoidance (1 square = 1 m^2)

In mobile robot navigation the avoidance of collisions is an essential task. In order to accomplish obstacle avoidance for dynamic obstacles the runtime of the TEB is analyzed in an appropriate scenario. Fig. 5 shows two snapshots of

the trajectory modification with the TEB, while an obstacle deforms the original trajectory to the left from the perspective of the robot.

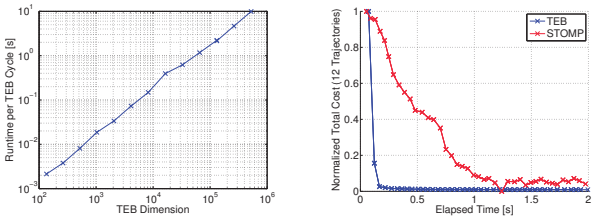
The average runtime of one single trajectory refinement cycle (see Fig. 3) in scenario of Fig. 5 is $2.1\,\text{ms} \pm 0.4\,\text{ms}$. During the entire simulation of 1000 cycles the obstacle moves back and forth towards the TEB. The computation time remains constant and is not affected by the dynamic obstacle.



(a) Velocity profile (b) Acceleration profile

Fig. 6. Velocity and acceleration profile of snapshot 2

The (green) vectors in Fig. 5 represent the translational velocity of the two wheels of the differential drive robot. The velocity and acceleration profiles of each wheel are shown in Fig. 6. The trajectory satisfies the constraints on the maximum velocity limited to $1.4\,\frac{\text{m}}{\text{s}}$ and the maximum acceleration limited to $0.3\,\frac{\text{m}}{\text{s}^2}$.



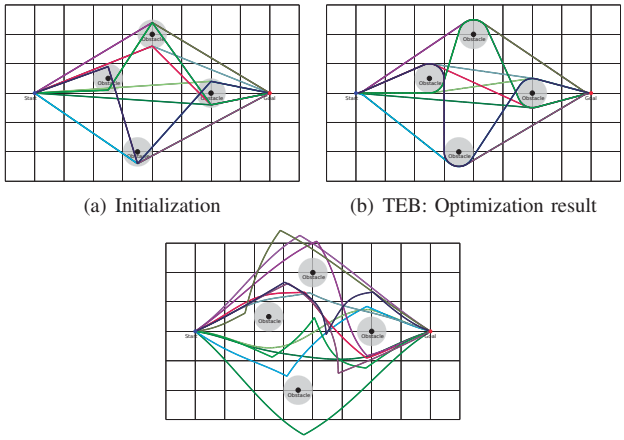
(a) (b)

Fig. 7. (a) Runtime per TEB-Cycle for increasing dimension of TEB and (b) Total cost of 12 trajectories over elapsed time (scaled to [0..1])

Another important performance aspect is the dependency of the average runtime per iteration on the dimensions of the TEB which grows linear with the number of configurations. This relationship is analyzed by increasing the number of configurations (higher density) in the very same scenario. The results are illustrated in Fig. 7(a). For more than 10000 states, corresponding to approx. 2500 configurations, at a path length of approx. 5 m the runtime exceeds the robot control cycle of approx. 20-30ms. However, in realistic applications the spatial-temporal resolution of the trajectory is significantly lower, because refinements of the initial trajectory only make sense within a look ahead distance of a few meters that equals the robots perceptual range.

A larger scenario composed of two static obstacles and four intermediate way-points is illustrated in Fig. 1(b). The trajectory satisfies all of the formulated constraints.

Fig. 8 demonstrates the power and efficiency of the proposed sparse model in combination with the g2o-framework.



(a) Initialization (b) TEB: Optimization result

(c) STOMP: Optimization result

Fig. 8. Optimization of 12 different trajectories

Twelve initial trajectories, shown in Fig. 8(a), are optimized in real-time in order to select the best candidate (e.g. with respect to optimization costs). The trajectory modification cycle (see Fig. 3) is executed in different threads on the simulation system. The two first iterations of TEB to reduce the cost of the initial trajectories to the near optimal ones shown (Fig. 8(b)) require 218 ms (see Fig. 7(b)). Every subsequent iteration with randomly moving obstacles only require $48\,\text{ms} \pm 4\,\text{ms}$. Note, that the trajectories are much longer than in the previous scenario in Fig. 5.

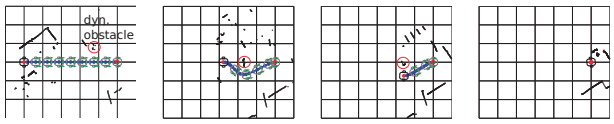
As a benchmark, the computational effort of TEB is compared with STOMP ([16]) in the same scenario (Fig. 8). STOMP is originally formulated for robotic arms, therefore the joint variables to be optimized are replaced by the planar state x and y. Notice, that STOMP does not depend on gradient information or requires differentiable objective functions. In contrast, the g2o-framework approximates gradients numerically and requires differentiable objective functions. Our STOMP implementation employs twenty random trajectory roll-outs to perform an update step.

The implementation contains the proposed acceleration matrix in combination with the obstacle cost function. Originally, STOMP aims to achieve a collision-free trajectory and not necessarily a fast one. Obviously, this is not sufficient for motion planning of mobile robots. It would be possible to extend STOMP by additional states and objectives. However, even the basic STOMP-2D implementation that only considers collision free path is outperformed by the TEB.

Figure 8(c) shows the optimization result for all 12 initial trajectories with STOMP. Each trajectory is composed of 80 2D-points which corresponds to the average number of configurations in each TEB (TEB uses dynamical resizing). The fixed number of configurations in combination with the weight of the acceleration matrix influences the task duration. For different scenarios, the weights have to be adjusted (see longer trajectories in Fig. 8(c)). With the proposed objective functions, our STOMP implementation fails to handle discontinuous initial trajectories which TEB mostly

manages. The runtime is compared to TEB in Fig. 7(b). The TEB performs the optimization significantly faster than STOMP. Note, the STOMP parameters are intuitively chosen such that they are largely comparable with TEB for the 2D case. In addition, we also implemented a 2D-version of CHOMP (without Hamilton Monte Carlo) for comparisons, but the obtained results are not robust in the above scenarios for a sufficient spatial resolution.

B. Robot experiments



(a) $t \simeq 0\,\mathrm{s}$ (b) $t \simeq 6\,\mathrm{s}$ (c) $t \simeq 11\,\mathrm{s}$ (d) $t \simeq 16\,\mathrm{s}$

Fig. 9. Avoiding a dynamic obstacle by real time adaptation of the TEB

The augmentation of elastic bands with temporal information and the use of g2o-framework allow for real-time trajectory adaptation and control of the robot. Fig. 9 shows a sequence of snapshots from a real robot experiment in which a person walks through the scene (similar to the previous simulated scenario with one obstacle). The TEB adapts the original robots trajectory ($t = 0$) in real time and avoids an imminent collision with the person during the interval $t \in [6, 12]$ by deforming the original trajectory away from the obstacle.

V. Conclusion and Further Work

This paper presents numeric aspects of the implementation of a real-time trajectory modification with TEB focusing on the implementation with the g2o-framework. The innovation of the TEB is to augment the classical elastic band with temporal information. Therefore it is possible to not only consider geometric and kinematic constraints with respect to the path but to simultaneously account for dynamic constraints of the mobile robot. g2o provides algorithms and solvers for sparse system structures. We demonstrated in this paper that the TEB exhibits such a sparse system structure and that it is therefore efficiently solved by the g2o-framework. The algorithm operates in real-time and thereby directly generates commands for the underlying robot motion controller. The method is highly flexible and is easily adapted to different robot kinematics and application requirements.

Future work is devoted to further improvements of the runtime. The first means is to provide analytical Jacobians to the optimization algorithm. The second is to re-use the a-priori ordering of the sparse TEB system-matrix from one optimization cycle to the next. The third is to dynamically adapt the resolution of the TEB both in time interval length and in degree of detail of the model according to the planning horizon. For the purpose of robot motion control, only the next few states are relevant, hence remote configurations in the far future are planned on a coarser scale.

A more fundamental change to the approach is to switch to a sparse constrained optimization framework. This renders the current formulation of constraints in terms of penalty functions obsolete.

Acknowledgment

This work has been funded by the ARTEMIS Joint Undertaking as part of the project R3-COP and from the German Federal Ministry of Education and Research (BMBF) under grant no. 01IS10004E.

References

[1] S. M. LaValle, "Planning Algorithms". Cambridge University Press, Cambridge, U.K., 2006.
[2] S. Quinlan, O. Khatib, "Elastic Bands: Connecting Path Planning and Control", in Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 802-807, 1993.
[3] S. Quinlan, "Real-time modification of collision-free paths", PhD thesis, Stanford University, 1994.
[4] M. Khatib, "Sensor-based motion control for mobile robots", Laboratoire d'Automatique et d'Analyse des Systèmes LAAS-CNRS, 1996.
[5] M. Khatib, H. Jaouni, R. Chatila, J. P. Laumond, "Dynamic Path Modification for Car-Like Nonholonomic Mobile Robots", in Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 1997.
[6] B. Graf, J. M. H. Wandosell, C. Schaeffer, "Flexible Path Planning for Nonholonomic Mobile Robots", Fraunhofer Institute Manufacturing Engineering and Automation (IPA), 2001.
[7] O. Brock, O. Khatib, "Executing Motion Plans for Robots with Many Degrees of Freedom in Dynamic Environments", in Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 1-6, 1998.
[8] F. Lamiraux, D. Bonnafous, O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots", in IEEE Transactions on Robotics, Vol. 20, No. 6, pp. 967-977, 2004.
[9] H. Kurniawati, T. Fraichard, "From path to trajectory deformation", IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS), pp. 159-164, 2007.
[10] V. Delsart, T. Fraichard, "Reactive Trajectory Deformation to Navigate Dynamic Environments", in Proc. of the Second European Robotics Symposium (EUROS), Vol. 44, pp. 233-241, 2008.
[11] B. Lau, C. Sprunk, W. Burgard, "Kinodynamic Motion Planning for Mobile Robots Using Splines", IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS), pp. 2427-2433, 2009.
[12] C. Sprunk et al. "Online Generation of Kinodynamic Trajectories for Non-Circular Omnidirectional Robots", in Proc. of the IEEE Intl. Conference on Robotics and Automation (ICRA), pp. 72-77, 2011.
[13] C. Sprunk et al., "Improved Non-linear Spline Fitting for Teaching Trajectories to Mobile Robots", in Proc. of the IEEE Intl. Conference on Robotics and Automation (ICRA), pp. 2068-2073, 2012.
[14] J. Mattingley, Y. Wang, S. Boyd, "Receding Horizon Control: Automatic Generation of High-Speed Solvers", in IEEE Control Systems Magazine, Vol. 31, No. 3, pp. 52-65, 2011.
[15] N. Ratliff et al. "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning", in IEEE Intl. Conference on Robotics and Automation (ICRA), May 2009.
[16] M. Kalakrishnan et al. "STOMP: Stochastic trajectory optimization for motion planning", in IEEE Intl. Conference on Robotics and Automation (ICRA), pp. 4569-4574, May 2011.
[17] C. Rösmann et al. "Trajectory modification considering dynamic constraints of autonomous robots", in Proceedings of the 7th German Conference on Robotics (ROBOTIK 2012). May 2012.
[18] K. Konolige, "Sparse Bundle Adjustment", in F. Labrosse et al., editors, Proc. of the British Machine Vision Conference, pages 102.1-102.11. BMVA Press, September 2010.
[19] R. Kümmerle et al., "g2o: A general framework for graph optimization", in Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA), Shanghai, China, May 2011.
[20] P. R. Amestoy, T. A. Davis, and I. S. Duff, "Algorithm 837: Amd, an approximate minimum degree ordering algorithm.", in ACM Trans. Math. Softw. vol. 30, pp. 381-388, September 2004.
[21] Y. Chen et al., "Alogithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate", in ACM Trans. Math. Softw. vol. 35, pp: 22:1-22:14, October 2008.
[22] L. E. Kavraki et al., "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", in IEEE Transactions on Robotics and Automation, Vol. 12, No.4, pp.566-580, August 1996.