Procedings of 13th International Conference on Intelligent Autonomous Systems (IAS), Padova, Italy, July 2014.

Continuous Motion Planning for Service Robots with Multiresolution in Time

Ricarda Steffens, Matthias Nieuwenhuisen, and Sven Behnke

Autonomous Intelligent Systems Group, Institute for Computer Science VI,
University of Bonn, Germany
ricarda.steffens@gmail.com, nieuwenh@ais.uni-bonn.de,
behnke@cs.uni-bonn.de

Abstract. We present an approach to continuous motion planning with multiresolution in time. Our approach is based on stochastic trajectory optimization for motion planning (STOMP) and designed to decrease the optimization time in order to enable frequent replanning. Since service robots operate in environments with dynamic obstacles, it is likely that planned trajectories become invalid over time. Thus, it is not necessary to provide trajectories with a uniform high resolution. Our multiresolutional approach implicitly considers the uncertainty of the future by providing a trajectory with a gradually coarser schedule, which is refined trough replanning. In addition to employing temporal multiresolution, we speed up trajectory optimization by initializing replanning with the previous plan. The proposed multiresolution STOMP is evaluated in simulation in comparison to the original STOMP implementation. Our experiments show that multiresolution STOMP reduces the planning time and, hence, is able to avoid dynamic obstacles.

Keywords: motion planning, service robots, multiresolution

1 Introduction

Service robots directly work next to human users in household or working environments while performing their tasks such as, e.g., fetching objects. In order to assure safety during robot movement, motion planning has to be employed to provide collision-free and feasible trajectories. Since environments shared with humans are dynamic, the planning has to consider not only static obstacles and possible self-collisions, but also dynamic obstacles. Due to the motion of dynamic obstacles as well as the movement velocities of the robot itself, the planning process needs to be fast and, similarly, frequent replanning is required to rapidly adapt to changes in the environment. Apart from this, the planned motions should not only be feasible but also smooth, because otherwise, the movement of the robot becomes non-predictable for human users and violations of joint limits may occur. In spite of motion planning being computationally expensive, it is necessary to have a motion planner which is capable of meeting these requirements.

In this paper, we present an approach to continuous motion planning based on the motion planning algorithm STOMP by Kalakrishnan et al. [1]. The algorithm employs stochastic trajectory optimization and, unlike other sampling-based motion planning algorithms, the resulting trajectory is already smooth and does not need further filtering. Although it converges slightly slower compared to gradient-based optimization methods, it is able to overcome local minima because of its stochastic approach. Overall, it is reasonable to use STOMP as a starting point for employing fast and efficient motion planning. Since frequent replanning depends on a fast execution of the planning process, the runtime of the STOMP algorithm has to be decreased. We utilize the uncertainty of the future state of the dynamic environment and employ multiresolution in time to reduce the computational complexity of the calculations made for planning. Moreover, the trajectory used for initializing the planner can be extracted from previous plans, which accelerates replanning. By not only including the robot arm but also additional joints into the planning process, it is also possible to increase the probability of finding feasible trajectories. We demonstrate our approach by employing it on our domestic service robot Cosero and evaluating it in simulation.

After discussing work related to ours, we give a brief system overview in Sec. 3. In Sec. 4, we detail our approach to continuous motion planning. We then present the experiments conducted for evaluation and their results in Sec. 5.

2 Related Work

Previous work on motion planning is extensive, as planned motion is an essential ability required by every type of robot. The research platform PR2 by Willow Garage, for example, employs a whole planning pipeline integrated into their middleware ROS [2]. A number of interchangeable motion planning algorithms in this pipeline are taken from the Open Motion Planning Library (OMPL) [3], including Kinematic Planning by Interior-Exterior Cell Exploration (KPIECE) [4]. While the basic functionality for motion planning is provided, it is, however, inevitable to modify the components in order to enable features such as frequent replanning.

In addition to those sampling-based motion planning algorithms, the Covariant Hamiltonian Optimization and Motion Planning (CHOMP) is available, which is a gradient optimization algorithm presented by Ratliff et al. [5]. It uses trajectory samples, which initially can include collisions, and performs a covariant gradient descent by means of a differentiable cost function to find an already smooth and collision-free trajectory.

A planning algorithm based on CHOMP is the Stochastic Trajectory Optimization for Motion Planning (STOMP) by Kalakrishnan et al. [1]. STOMP combines the advantages of CHOMP with a stochastic approach. In contrast to CHOMP, it is no longer required to use cost functions for which gradients are available, while the performance of STOMP stays comparable. This allows to include costs with regard to, for instance, general constraints or motor torques.

Since STOMP is designed and implemented to only being executed once in a while, the algorithm is not able to consider dynamic obstacles, unless it is actively called consecutively to generate a new plan from scratch.

Another algorithm derived from CHOMP is ITOMP, an incremental trajectory optimization algorithm for real-time replanning in dynamic environments [6]. In order to consider dynamic obstacles, conservative bounds around them are computed by predicting their velocity and future position. Since fixed timings for the trajectory waypoints are employed and replanning is done within a time budget, generated trajectories may not always be collision-free.

Other interesting approaches include elastic strips by Brock and Khatib [7] and the motion planning employed for the robot Rollin' Justin by the German Aerospace Center (DLR) [8]. Both make use of whole-body motion planning and reactive control by means of potential fields, while elastic strips uses the latter to conform the planned trajectories. Since reactive behavior does not take the near future into account, failures occur with both elastic strips and the DLR motion planning and therefore, complete global replanning becomes necessary.

Integrating a multiresolutional approach into planning with regard to the environment representation or with respect to time is not new. In our prior work both approaches of local multiresolution path planning were proposed for mobile robots [9]. In both cases, different resolutions are nested hierarchically, leading the precision to decrease gradually.

In terms of motion planning for end-effectors, He et al. also follow the idea of a multiresolutional planning by proposing a multigrid CHOMP [10]. In contrast to our approach from [9], all of the time steps are first computed in a low resolution and are subsequently upsampled by adding intermediate points between time steps from the coarser resolution, thus leading to a uniformly spaced trajectory. They also utilize the fact, that the optimization algorithm converges faster, if an already near-optimal trajectory is given as initialization.

Their idea of a multiresolutional trajectory does not appeal to us, because they have to plan multiple times, yet they end up with uniform spaced timesteps in a high resolution. Moreover, frequent replanning has to be executed in order to consider dynamic obstacles. In contrast, utilizing our previous work provides a finished plan, which is refined during replanning.

In this paper we therefore adopt our previous approach for multiresolution in time in combination with frequent replanning while reusing previously planned trajectories as an initialization to speed up convergence.

3 System Overview

Our domestic service robot Cosero [11] was built following an anthropomorphic design approach. It consists of a mobile base with an omni-directional drive, a movable torso attached to a vertical linear actuator, two anthropomorphic arms each equipped with a gripper, and a communication head (see Fig. 1). The vertical torso motion allows Cosero to manipulate objects on elevated surfaces, but also to grasp objects from the ground. In addition, its trunk can be rotated

Fig. 1. Domestic service robot Cosero during the RoboCup 2013 competition in Eindhoven.

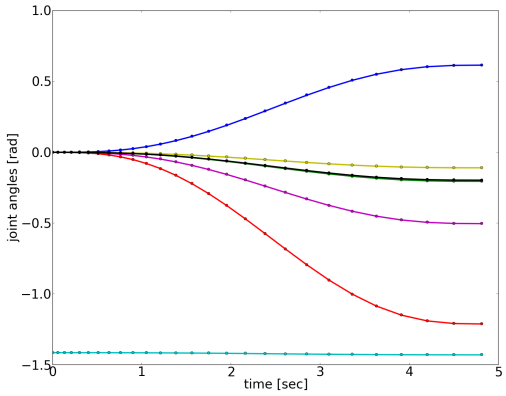


Fig. 2. A set of initial trajectories with multiresolutional spacing for the seven joints in the robot arm.

around the vertical, which yields a larger workspace and concurrently improves the interaction with a user. For perceiving the environment, Cosero is equipped with four laser-range finders and a Microsoft Kinect RGB-D camera in its head. The robot possesses a total of 32 joints, which are driven by Robotis Dynamixel actuators and are revolute, except for one vertical prismatic joint. Each of its anthropomorphic arms has 7 DOF and an attached end effector with 2 DOF utilizing two parallel fingers on each opposing side. Each arm is able to handle a maximum payload of 1.5 kg.

In prior work—mobile bin picking [12]—we employed ROS' planning pipeline for motion planning. After generating a feasible grasp for one out of a pile of objects in a bin, motion planning is employed for moving the end effector over the box and subsequently, to a pre-grasp pose. On that account, motion planning is performed by means of lazy bi-directional KPIECE [4], which is included in OMPL. Finally, the planned motion is executed by the robot and the desired object is grasped. In this application the robot plans once for each call and subsequently executes the planned trajectory without renewed collision checking.

4 Motion Planning

In order to realize frequent replanning, it is crucial to accelerate the execution speed of the motion planning. One approach for reducing the number of calculations and thus speeding up runtime is multiresolution.

4.1 STOMP

Unlike sampling-based motion planners, STOMP defines the given motion planning problem as an optimization problem. Accordingly, the goal is to find a

trajectory, which minimizes the costs calculated by a predefined cost function. As an input STOMP gets a start and a goal configuration $x_0, x_N \in \mathbb{R}^J$, with J being the number of joints. The output of the algorithm is one trajectory vector $\theta \in \mathbb{R}^N$ per joint, discretized into N waypoints. Besides a cost function STOMP has to be initialized with a joint trajectory, e.g., an interpolation between start and goal configuration. The optimization problem STOMP solves iteratively is defined by

$$\min_{\tilde{\theta}} \mathbb{E}\left[\sum_{i=1}^{N} q(\tilde{\theta}_i) + \frac{1}{2}\tilde{\theta}^\top \mathbf{R}\tilde{\theta}\right]$$

with $\tilde{\theta} = \mathcal{N}(\theta, \Sigma)$ being a noisy joint parameter vector with mean θ and covariance Σ. $q(\tilde{\theta}_i)$ is a predefined cost function calculating the costs for each state in $\tilde{\theta}$. $\theta^\top \mathbf{R}\theta$ describes the sum of squared accelerations along the trajectory with $\mathbf{R}$ being a matrix representing control costs. STOMP now attempts to solve the defined optimization problem by means of a stochastic optimization method. As a result it is possible to minimize the predefined cost function $q(\tilde{\theta})$, even though it might not be differentiable and therefore, no gradients are available for this function.

4.2 Multiresolution in Time

Whereas environments with only static obstacles are constant throughout time, dynamic environments change regularly due to dynamic obstacles as well as the motion of the robot itself. Consequently, the future state of a dynamic environment is uncertain, as obstacles might appear or move suddenly. The multiresolutional approach takes advantage of this property by implicitly taking the uncertain future into account. It is assumed that while a planned trajectory is executed, the future waypoints might become invalid. Hence, it is not necessary to have a trajectory with a high uniform resolution throughout time. In order to reach a goal configuration, it suffices to plan the initial segments of a trajectory with high temporal resolution and to decrease temporal resolution with distance from the starting time.

As this results in a smaller number of time steps, less calculations have to be conducted with a multiresolutional spacing, thus leading to a computational advantage compared to the uniformly discretization of the trajectory length. Certainly, the planned multiresolutional trajectory is not as precise and smooth as the uniformly spaced one and, hence, it would be difficult to execute the entire trajectory directly. This is not done, however, but the multiresolutional trajectory is refined gradually by means of frequent replanning.

By default, the duration of a trajectory is uniformly discretized into a number of timesteps. When utilizing multiresolution, the time intervals differ in their length with regard to the duration of the whole trajectory. With an increasing duration, the resolution of the time intervals decreases. As a result, the lengths of the time intervals increase by means of a pre-defined function determining the growth of duration between timesteps.

In contrast to the original STOMP implementation which uses a fixed trajectory duration of 5 s, we calculate the duration and therefore the number of waypoints online for each new planning problem. Since the start and goal configurations are given, the angular distance between the configurations can be calculated for each joint. By means of the maximum speed of the built in Dynamixel actuators without load and the weight of the construction parts an educated guess of 0.3 rad/s regarding the average velocity of the actuators can be made. As a result, the duration of a new trajectory can be calculated by

$$d = \frac{max_{j \in J}(|x_{j,N} - x_{j,0}|)}{v} + \Delta, \tag{1}$$

where J is the number of joints, $x_{j,0}$ and $x_{j,N}$ are the start and goal configurations for a specific joint and v is the average velocity of the actuators. Due to the fact, that the goal configuration cannot be reached by means of a linear interpolation when obstacles are blocking the way, an extra time padding Δ is added to the calculated duration.

In order to determine a multiresolutional trajectory, a minimum trajectory discretization as well as a function for determining the growth of the time intervals is needed. Although this function can be chosen arbitrarily, the length of the time intervals should not become too large in order to avoid an insufficient number of waypoints for collision avoidance. To determine the size of the time intervals δt, we use a linear function defined by

$$\delta t(x) = 0.01t + r, \tag{2}$$

with r being the minimum resolution for discretization, which we chose as $0.05\,s$.

When initializing a new trajectory, the intermediate points between the start and goal configuration are filled by means of Bézier splines. An example multiresolutional trajectory for all of the 7 joints of the robot arm is shown in Fig. 2.

As the smoothness of the trajectory is measured by the acceleration along the trajectory, differentiation matrices are pre-defined in the original implementation for the purpose of calculating the first, second and also third order of derivatives. When applying the multiresolutional approach each waypoint in the trajectory has a different, multiresolutional spacing to its neighbors. Thus, the finite differencing filters have to be computed for each point separately with regard to the multiresolutional waypoint spacing. In order to get the derivatives for each waypoint, the algorithm for finite differencing by Fornberg is employed [13]. Finally, the fourth order of accuracy of each of the orders of derivative is used to build the multiresolutional differentiation matrices. Since the trajectories are newly created for each planning problem, this computation also has to be executed for each planning problem.

4.3 Frequent Replanning

STOMP is a motion planning algorithm, which performs local optimization and therefore, primarily finds locally optimal trajectories as opposed to finding global

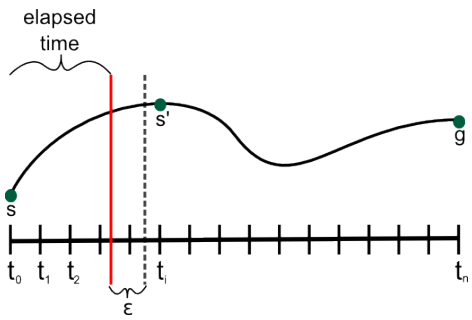


Fig. 3. The idea of moving the start index when replanning. s and g are the start and goal configurations, s' is the new start index at time t_i and ϵ represents the mean time needed for replanning.

solutions. Thus, its performance changes depending on the initial trajectory supplied for optimization. As the sampled noise used for optimization lies within a fixed interval, the impact of the exploration steps in each iteration is limited. Consequently, the further away an initial trajectory is from the locally optimal trajectory, the more iterations are needed to converge to it. The original implementation of STOMP always uses a new trajectory, which is created by Bézier splines interpolation, as an initialization for the optimization process. This works good if no obstacle is in the vicinity of the robot, but with an obstacle near it, the optimization process has to start from scratch when replanning.

We assume that planning takes place frequently and also consecutively, as the robot might directly repeat tasks or replan while executing a task. Meanwhile, static and even dynamic obstacles in the workspace of the robot most likely stay in the same position or area. Hence, the previously planned trajectory is very probably similar or near to the locally optimal trajectory and therefore, reusing the previous plan leads to a quicker convergence of the optimizer, as the old trajectory only has to be refined.

Combining frequent replanning with reusing previously planned trajectories as an input for optimization leads to two different cases:

I) When replanning is executed with a different goal, the trajectory is newly created and initialized without reusing the previously planned trajectory.

II) Using the former trajectory as an input for planning to the same goal results in the problem of choosing a new start configuration concerning time. Since the robot starts executing the planned trajectory right after it was published, the new starting point needs to be ahead of the current time. In order to find the new starting point, the elapsed time since the prior trajectory was published is determined and an additional value ϵ is added to this duration as a time padding for of the mean time needed for finishing one planning cycle. This guarantees a smooth transition between the old trajectory being in execution and the newly optimized part. Eventually, the complete duration is increased to the next existing index as shown in Fig. 3. When using multiresolutional trajectories, not

only the start index must be moved, but the entire trajectory needs to be resampled to maintain the local multiresolution property. As illustrated in Fig. 4, the gradual increase of resolution leads to a refinement of the plan.

4.4 Adding Joints

It is expedient to include additional joints into the planning process, as more degrees of freedom might lead to better solutions or even finding a solution at all, when planning only with the arm and endeffector fails. In case of our robot Cosero, the torso yaw joint and the prismatic torso elevator joint can be added, which increases the workspace of the robot drastically.

Adding supplementary joints to the planning with regard to the STOMP implementation is not difficult and can be realized by means of configuration files. However, appending joints might lead to side effects within the planner with respect to the kinematic chain of the robot.

Although many different possibilities for avoiding collisions are gained, the extra joints and therefore additional dimensions in c-space, increases the calculations needed for collision detection, inverse and forward kinematics and the optimization process itself. Consequently, the optimization slows down but due to the acceleration by the multiresolutional approach, constant replanning as well as incorporating additional joints is possible without any problems.

5 Evaluation

For evaluation, our multiresolutional STOMP implementation is compared to the original STOMP with regard to overall runtime, success in avoiding static and dynamic obstacles, the number of needed iterations and length of trajectory. In addition, LBKPIECE is used as a reference data for finding feasible grasp positions in a shelf experiment. The experiments are executed in simulation within the ROS environment on a desktop computer with an Intel Core i7 940 quadcore CPU with 2.93 GHz and 24 GB RAM.

In order to evaluate the approach of using a previously planned trajectory as an initialization of the optimization process, we compare our multiresolutional STOMP implementation with this feature enabled with our STOMP utilizing the standard initialization used by the original STOMP. In the experiment depicted in Fig. 5, first an obstacle is placed outside of the vicinity of the robot. Then the dynamic obstacle is slowly approaching the robot and subsequently, after a

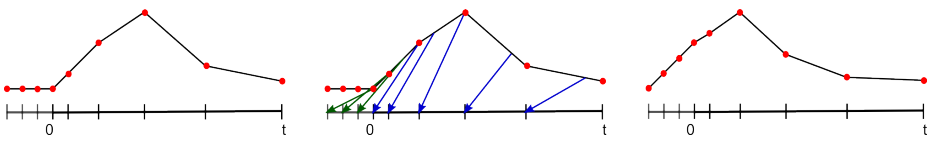
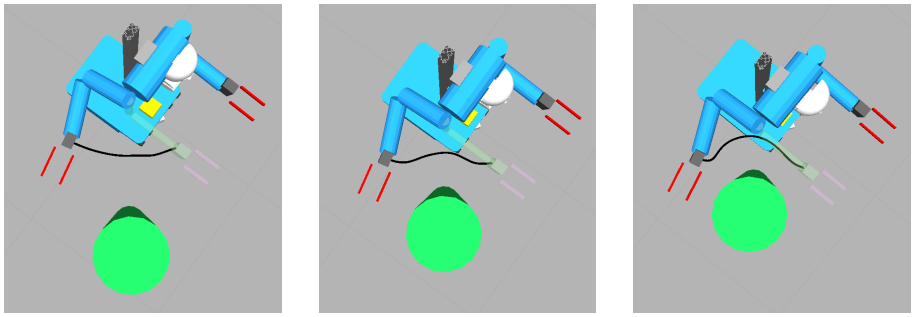


Fig. 4. Moving a multiresolutional trajectory for two timesteps requieres resampling.



(a) (b) (c)

Fig. 5. The avoidance of a dynamic obstacle by means of frequent replanning. The obstacle moves towards the robot and the plan is optimized with respect to the obstacle.

Table 1. Comparison of our multiresolutional STOMP with and without using previously planned trajectories as an initialization while avoiding a dynamic obstacle. The minimum number of iterations is set to 19 for this experiment.

	standard initialization	initialization with old trajectories
Success	100%	100%
No. of waypoints	27	27
Duration	4.55 s	4.55 s
Iterations	22.44	19.04
Overall runtime [s]	0.23 $\pm$0.036	0.21 $\pm$0.025

short time of adjusting to the obstacle, leaves its vicinity again. This is done manually once within 500 planning cycles.

The results of the experiment are shown in Tab. 1 as well as Fig. 6. Overall, both versions of our implementation find feasible and collision-free trajectories in each planning cycle. Nevertheless, both, the average number of iterations as well as the average overall runtime of our planner version without the initialization with recent trajectories, are higher than the respective values of our STOMP with initialization. While the number of iterations of our multiresolutional STOMP version without initialization is high most of the time, in which the dynamic obstacle was moved to and away from the robot, the other planner version again only shows single peaks for every time a larger movement was executed. This indicates, that our multiresolutional planner using the standard initialization not only needs many iterations for newly adapting to an occurring obstacle in each planning cycle, but also needs more iterations to return to the plan without an obstacle present. The experiment demonstrates, that the frequent replanning in order to avoid dynamic obstacles is successful, especially when combined with the reuse of previously planned trajectories as an initialization to the optimization of the STOMP algorithm.

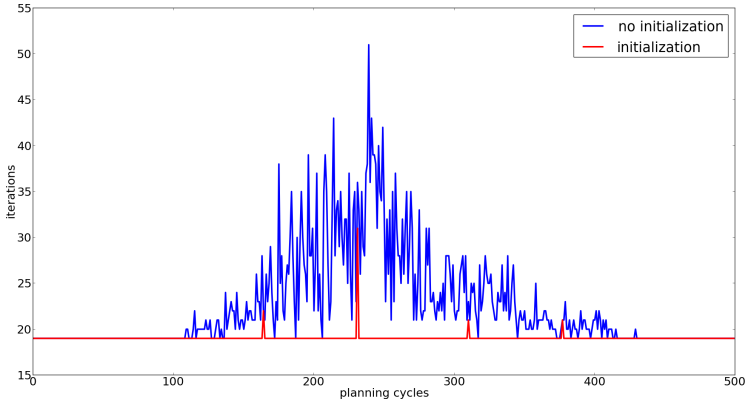


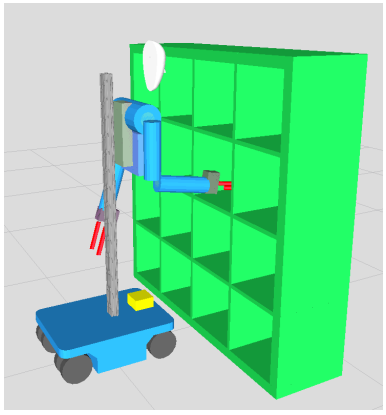
Fig. 6. The number of iterations until convergence with and without initializing with the recent trajectory. The moving of the obstacle was executed manually. Hence, the planning cycles with obstacles moved in and out of the vicinity differ slightly.

Table 2. Comparison of LBKPIECE, original STOMP and our multiresolutional STOMP when planning within the shelf. The success represents the rate for finding feasible plans.

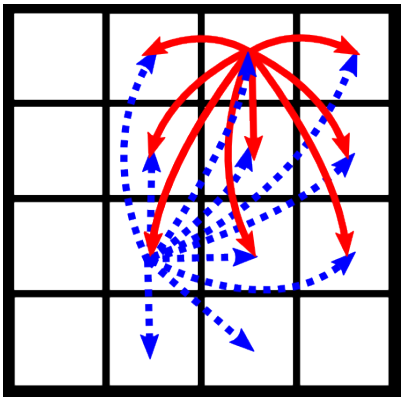
	KPIECE	original STOMP	multiresolutional STOMP
Success	96.94%	72.22%	86.96%
Overall runtime $[s]$	N/A	1.71 ± 0.73	0.81 ± 0.32

In another experiment both, the original STOMP implementation and ours, are compared in a shelf scenario (cf. Fig. 7a). For this purpose, we built a standard shelf with measurements of $39 \times 149 \times 149$ cm and 16 cases in simulation and placed the robot in front of it aligned to the center of the shelf. In a first step, we then utilized LBKPIECE with an extended terminating condition of 120 s allowed planning time in order to get reference data regarding the inverse kinematic solutions of the robot as well as the existence of a feasible path. The planning is executed for each combination of start and goal cases, if a valid inverse kinematics solution was found for both configurations. Results for two example start cases are depicted in Fig. 7b. Subsequently, the planning is conducted with both STOMP implementations.

The results of the experiment are shown in Tab. 2. Overall, our multiresolutional STOMP implementation has a higher success rate than the original STOMP implementation while needing less overall runtime. Although the multiresolutional trajectory has to be refined in terms of frequent replanning, the replanning is executed during the movement of the endeffector. Moreover, as shown in Tab. 1 employing the initialization of previously found trajectories accelerates the overall runtime when replanning, which especially benefits more difficult scenarios. As a result, our multiresolutional STOMP is able to successfully plan obstacle free trajectories within a reasonable runtime.



(a) The model of Cosero standing centered in front of a shelf.



(b) Two examples for found plans by the LBKPIECE algorithm from two different start cases within the shelf.

Fig. 7. The setup of the shelf experiment in simulation is shown in Fig. 7a and two example results of planning with LBKPIECE are depicted in Fig. 7b.

Table 3. Comparison of the lengths of a uniform and a refined multiresolutional trajectory provided for one example planning problem including an obstacle.

Joints	uniform $[rad]$	multiresolutional $[rad]$
Shoulder pitch	0.92	0.75
Shoulder roll	0.18	0.17
Shoulder yaw	1.34	1.32
Elbow pitch	0.41	0.35
Wrist yaw	0.53	0.53
Wrist pitch	0.20	0.19
Wrist roll	0.25	0.16
$\sum$	3.83	3.47

In order to compare the quality of multiresolutional solutions, an example planning problem including a simple obstacle was used to calculate the length of the resulting trajectories of our STOMP implementation with uniformly spaced timesteps and multiresolutional time intervals. Since the multiresolutional trajectory is refined with every new replanning, only the waypoints in high resolution are used for the calculation. The result is shown in Tab. 3. This example demonstrates, that the quality of multiresolutional trajectories is comparable to the quality of uniform trajectories.

6 Conclusions

In this paper, we presented an approach for continuous motion planning based on the motion planning algorithm STOMP. In order to enable frequent replanning

and additional joints for planning, multiresolution in time is used to decrease the runtime of the planning process. Moreover, a good guess for an initial trajectory is used to accelerate the optimization process. We evaluated the proposed approach in experiments in terms of overall runtime, success in avoiding static and dynamic obstacles, and trajectory length. Our experiments demonstrated that the proposed changes to the original STOMP method are beneficial.

One possibility for future work would be to extend the upper body motion planning presented in this paper to a whole-body motion planning by combining it with the navigation of the robot base. Moreover, a sequential planning for both arms could be integrated in terms of consecutively determining the goal configurations of both arms by means of inverse kinematics and subsequently solving the motion planning problem with STOMP.

References

1. Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., Schaal, S.: Stomp: Stochastic trajectory optimization for motion planning. In: Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA). (2011)
2. Chitta, S., Jones, E.G., Ciocarlie, M.T., Hsiao, K.: Mobile manipulation in unstructured environments: Perception, planning, and execution. IEEE Robotics & Automation Magazine **19**(2) (2012) 58–71
3. Şucan, I.A., Moll, M., Kavraki, L.E.: The Open Motion Planning Library. IEEE Robotics & Automation Magazine **19**(4) (2012) 72–82
4. Şucan, I.A., Kavraki, L.E.: Kinodynamic motion planning by interior-exterior cell exploration. In: Algorithmic Foundation of Robotics VIII, Int. Workshop on the Algorithmic Foundations of Robotics (WAFR). Volume 57 of Springer Tracts in Advanced Robotics., Springer (2008) 449–464
5. Ratliff, N.D., Zucker, M., Bagnell, J.A., Srinivasa, S.: Chomp: Gradient optimization techniques for efficient motion planning. In: Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA). (2009)
6. Park, C., Pan, J., Manocha, D.: Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments. In: ICAPS, AAAI (2012)
7. Brock, O., Khatib, O.: Elastic strips: A framework for motion generation in human environments. The Int. Journal of Robotic Research **21**(12) (2002) 1031–1052
8. Dietrich, A., Wimböck, T., Albu-Schäffer, A., Hirzinger, G.: Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom. IEEE Robotics & Automation Magazine **19**(2) (2012) 20–33
9. Behnke, S.: Local multiresolution path planning. In: RoboCup 2003: Robot Soccer World Cup VII. Volume 3020 of LNCS., Springer (2004) 332–343
10. He, K., Martin, E., Zucker, M.: Multigrid CHOMP with local smoothing. In: Proc. of 13th IEEE-RAS Int. Conference on Humanoid Robots (Humanoids). (2013)
11. Stückler, J., Behnke, S.: Integrating indoor mobility, object manipulation, and intuitive interaction for domestic service tasks. In: Proceedings of 9th IEEE-RAS International Conference on Humanoid Robots (Humanoids). 506–513
12. Nieuwenhuisen, M., Droeschel, D., Holz, D., Stückler, J., Berner, A., Li, J., Klein, R., Behnke, S.: Mobile bin picking with an anthropomorphic service robot. In: Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA). (2013)
13. Fornberg, B.: Generation of finite difference formulas on arbitrarily spaced grids. Mathematics of computation **51**(184) (1988) 699–706