Task Space Motion Planning Using Reactive Control

Matthias Behnisch, Robert Haschke and Michael Gienger

Abstract— **In this paper we present an approach to reduce the effort for planning robot motions by shifting the planning problem to a high-level representation. We combine classical sampling-based random tree planning with a reactive controller connecting sampling points with nontrivial trajectories, utilizing redundant DOFs to locally avoid obstacles. While the reactive planner operates locally on a short time scale, the complementary sampling-based method is able to find globally feasible solutions due to its larger preview horizon. Additionally, planning is done in a low-dimensional task space instead of the high-dimensional joint space. Comparing the average planning time and number of tree extensions for several scenarios and planning methods, we demonstrate that this hybrid planning approach is capable of solving a large fraction of planning queries while saving considerable planning time.**

I. INTRODUCTION

The idea of generating movements by reacting to the environment was addressed by potential field methods since the early days of motion planning [1], emphasizing real-time path generation and obstacle avoidance. Recent work on humanoid robot control picks up the idea of incorporating environmental reactive behavior into the domain of trajectory generation within redundant whole body motion control frameworks [2]–[4]. However, these potential field methods suffer from the problem of operating in a local scope only, unable to find global solutions in the presence of local minima due to complex environments. This weakness originally led to the development of planning algorithms that globally search the configuration space, like complete combinatorial planning or probabilistic complete sampling-based planning. For an overview of motion planning techniques see [5], [6].

Sampling-based planning methods are particularly successful in solving complicated problems in high-dimensional configuration spaces, even under presence of non-holonomic or kinodynamic constraints. Nevertheless, the computing time of those methods scales exponentially with the number of degrees of freedom. Modern humanoid robot platforms possess a large number of joints, typically resulting in a highly redundant kinematic structure. To keep the computing time bounded and thus meet the real-time requirements of robots operating in dynamic human environments, more efficient search methods or pruning techniques are needed. Several heuristics were developed to reduce the search space and to handle redundancy.

The work in [7] specifies goals in the lower-dimensional workspace and a heuristic, estimating the workspace distances of the robot to the goal and to obstacles, is used to bias the configuration space search of a Rapidly Exploring Random Tree (RRT) [8]. JT-RRT builds upon the same principle, but imposes a stronger goal bias by randomly performing direct movements towards the goal, employing the Jacobian transpose as a approximate mapping between end-effector poses and configuration space postures [9]. In [10] (BiSpace) two search trees are used: one in configuration space starting at the current posture and another one in the workspace originating from the workspace target. The search of the RRT based algorithm is focused on connecting both trees, again utilizing a workspace metric.

M. Behnisch and R. Haschke are with the Research Institute for Cognition and Robotics, Bielefeld University, Bielefeld, Germany {mbehnisc,rhaschke}@cor-lab.uni-bielefeld.de

M. Gienger is with the Honda Research Institute Europe, Offenbach, Germany michael.gienger@honda-ri.de

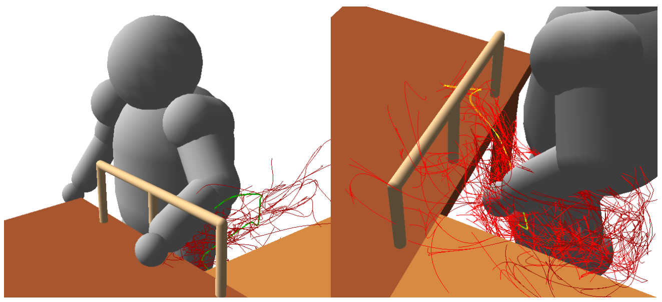

Fig. 1. Sampling-based planning in the task space with the Honda humanoid research robot. The solution trajectory is shown in green/yellow.

While the aforementioned planners add different workspace heuristics to a configuration space search, it is also possible to perform the entire search in the workspace, or more generally in the task space [11]. Here a Jacobian pseudo-inverse feedback controller generates configuration space trajectories for a highly redundant robot arm based on a low dimensional task specification. It is shown that searching the space of task positions with a modified RRT algorithm can lead to an enormous performance improvement in contrast to high-dimensional configuration space search. However not all possible solutions can be found, since the set of exploreable trajectories is restricted.

The technique in [12] reduces the search effort by decomposing the problem into two subproblems. First subproblem is to find a workspace tunnel of free space, that is assumed to contain the swept workspace volume of the robot following a solution path. The second subproblem is to find a collision-free path inside this tunnel, where a redundant control scheme is employed, utilizing distance based potential functions to repel the robots links from obstacles and to attract the end-effector through the tunnel. The Elastic Strips method [13] takes the same approach of refining a predetermined workspace path to obtain the final configuration space trajectory. Since the refinement of a given path fails in situations where local minima or topological changes occur, Elastic Roadmaps [14] add a global search method. A sampling-based planning approach is taken to create a roadmap in workspace and individual nodes are allowed to move in response to obstacles while the connections between them are constantly updated. However, since the roadmap is computed in the workspace, there is no guarantee that a motion between nodes is possible in all cases.

In the present paper we follow the approach of searching the task space directly with a sampling-based random tree planner, while a feedback control scheme is used to exploit the redundant space in order to reactively avoid obstacles by computing distance based potential functions. Section II gives an overview of the concept of hybrid planning, followed by a description of the planning algorithm and the reactive whole body motion control framework. Section III outlines the simulation experiments conducted to compare the performance, whose results are presented in section IV, followed by a discussion in section V and conclusions in section VI.

II. HYBRID PLANNING

The motion planning approach presented in this paper builds on the concept of shifting the planning problem to a high-level representation. The task of the planner is reduced to specify sub-goals for a reactive control framework, able to avoid obstacles autonomously. This way, our method is hybrid in the sense of dividing the search into a global component, based on a sampling-based planning algorithm, and a local component, which exploits the robot's redundancy to avoid obstacles with distance based potential functions.

The main idea of sampling-based planning is to abandon an explicit representation of the state space and instead only use an implicit representation of sub-paths between randomly sampled positions (e.g. see [5], [6]). How these samples are created and how the search for a solution path is organized depends on the algorithm used, while the underlying process of generating motions between sampled positions is handled by a local planner. The local planner is responsible to ensure that the sub-paths are valid movements and collision free. It can be shown that sampling-based planning is probabilistically complete, i.e. in the limit of an infinite number of samples, the probability to find a solution – if there is one – converges to one, provided the sampling is done in a uniform random fashion.

Figure 2 illustrates how hybrid planning differs from classic sampling-based planning. The left side shows the classic approach, where sampling and local planning is directly done in the configuration space of the robot. In our hybrid planning method, the simple local planner, generating straight-line motions between nodes, is replaced by a more powerful planner that builds upon the reactive control framework. This is shown on the right side of figure 2. Sampling is now reduced to the lower-dimensional task space and it is the

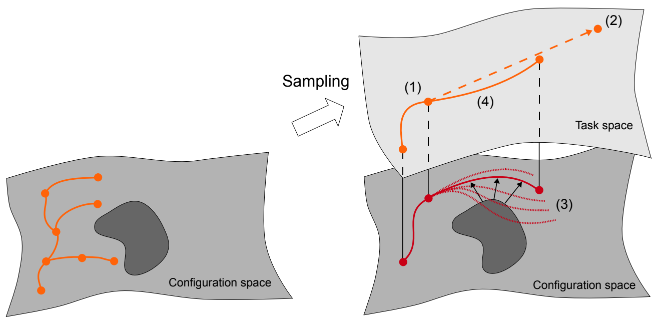

Fig. 2. LEFT: Sampling-based planning in the configuration space. RIGHT: Hybrid planning reduces sampling to the task space. Starting at some task space and corresponding configuration space position (1), a sub-trajectory towards a target (2) in task space is created. The reactive local planner adjusts the configuration space trajectory based on a distance based potential function (3) without changing the task space trajectory (4).

duty of the reactive local planner to generate a corresponding trajectory in the configuration space.

To get a better understanding on how the reactive local planner creates motions, it is useful to examine some properties of the underlying control framework. It is based on the well known principle of resolved motion rate control (e.g. see [15]), as a way to control robotic manipulators by specifying end-effector velocities. A differential kinematic mapping between joint and end-effector velocities, or more generally task velocities, is computed. For redundant manipulators, this mapping is not unique, that is a given task velocity can be realized by a continuum of joint velocities. For this case the mapping can be decomposed into the space of realizable task space movements and the space of redundant joint movements, containing self-motions of the manipulator that do not affect task execution. While the movement in the task space is determined by the target position, the movement in the redundant space can be arbitrarily chosen.

Hybrid planning now proceeds by sampling a task space target position as an input to the control system. While the task space trajectory is fully determined, the trajectory in the redundant space can be chosen in order to avoid obstacles. For this a potential function, encoding the distance to obstacles, is minimized by gradient descent. If this potential function is properly defined, it can prevent local collisions sufficiently well to enable the sampling-based planner to find global solutions in the lower-dimensional task space on a coarser spatial scale.

While this hybrid planning approach can solve many planning problems, its success is dependent on the ability of the control framework of finding suitable configuration space trajectories. This can not be guaranteed, since the redundant motion is determined with local search only. Thus the probabilistically completeness of sampling-based planning is only applicable to the task space search. If a configuration can not be created with any task space position, it is not considered during the search, thus hybrid planning is not complete concerning the space of all possible configurations.

Another limitation is that the search can only be performed in a directed manner. Although the iteration of the control system could be reversed in principle, it has to be initialized with a starting configuration. If the desired posture at the goal is not known, this information is missing.

Because the task space state does not represent the whole motion in the configuration space, storing the configuration space position together with the task space position during the search is advantageous. Else the recreation of the corresponding configuration space position would need a complete traversal of the search tree, beginning at the starting point.

To summarize, the key features of our hybrid planning method are as follows:

- The space that is searched by sampling is reduced to the lower-dimensional task space.
- The ambiguity of redundant manipulators is resolved during planning, thus it is sufficient to specify a single task space goal instead of a whole set of possible joint postures at the goal.
- By adding a local obstacle avoidance method in the redundant space, the set of collision free trajectories that can be produced is enlarged.
- Planning and reactive obstacle avoidance occur simultaneously in a one-shot fashion, no preprocessing is necessary.
- Planning is done in the space of control inputs, hence resulting movements can be directly executed with the controller in the same way as they are planned. No postprocessing is necessary to modify a plan to be executable on a controller that moves real hardware.
- The task space used is not restricted to be the cartesian workspace with sampling-based planning. Instead every input to a control system can be used as task space, even if this input is of high dimension and exhibits non-holonomic or kinodynamic constraints.

A. Planning Algorithm

The global search component in our hybrid planning approach is based on the Expansive Space Tree (EST) algorithm [16], a probabilistic single-query sampling-based random tree planner. A key point that distinguishes the EST algorithm from others is how tree nodes are selected for extension and in which direction the extension is performed. While the Rapidly Exploring Random Tree (RRT) [8] first determines an exploration direction and then picks the closest tree node to expand towards this direction, the EST algorithm reverses this process by first selecting a node and than choosing the direction independently. This way EST does not depend on state space metrics to determine the exploration direction, thus omitting the challenging problem of designing a suitable metric [17]. Also EST is specifically designed for problems that involve kinodynamic constraints. If local motions are generated in terms of a control system, the tree extension proceeds by sampling in the space of control inputs and by integrating the control system forward. This way constraints are satisfied by construction. Guided Expansive Space Trees [18] and Utility-guided Random Trees [19] further refine the concept of EST, in order to exploit as much information as possible during the search process and to avoid inefficient exploration behavior. Since we want to focus on the impact of redundant obstacle avoidance here, the plain EST algorithm is used.

Algorithm 1 Task space EST

```
$T \leftarrow \emptyset$
$x \leftarrow x_{start}$
while $d(x_{goal}, x) > goalregion$ do
    $(x_{cur}, \dot{x}_{cur}, q_{cur}) \leftarrow SELECT(T)$
    if $random([0..1]) \leq goalbias$ then
        $x_{target} \leftarrow x_{goal}$
    else
        $x_{target} \leftarrow SAMPLE(x_{cur}, \sigma)$

    $(x, \dot{x}, q, t_{valid}) \leftarrow EXTEND(\dot{x}_{cur}, q_{cur}, x_{target}, t_{max})$
    if $t_{min} \leq t_{valid} \leq t_{max}$ then
        $T \leftarrow ADD(T, x, \dot{x}, q)$
```

Algorithm 2 Configuration space EST with task space bias

```
$T \leftarrow \emptyset$
$x \leftarrow x_{start}$
while $d(x_{goal}, x) > goalregion$ do
    $q_{cur} \leftarrow SELECT(T)$
    if $random([0..1]) \leq goalbias$ then
        $x_{target} \leftarrow x_{goal}$
        $(q, t_{valid}) \leftarrow EXTEND(q_{cur}, x_{target}, t_{max})$
        if $t_{min} \leq t_{valid} \leq t_{max}$ then
            $T \leftarrow ADD(T, q)$
    else
        $q_{target} \leftarrow SAMPLE(q_{tree}, \sigma)$
        if  $q \leftarrow CONNECT(q_{cur}, q_{target})$ then
            $T \leftarrow ADD(T, q)$
```

Algorithm 1 shows the EST algorithm adapted for planning in the task space. The selection of tree nodes remains the same. As the first step in each iteration, $SELECT$ picks a node from the tree for expansion. A weight is assigned to each node that determines the probability of choosing the node. It is defined in terms of how densely the neighborhood of the node is sampled and is higher for nodes with few neighbors, thus steering the exploration towards less explored regions of the space.

A common method to focus the growth of the tree is to do a direct attempt to reach the goal with a certain probability $goalbias$. Without goal biasing, the function $SAMPLE$ creates a new target sample from a Gaussian distribution, centered at the selected node and using a fixed standard deviation σ.

Given the desired target position, the next step differs for planning in the task space. The local planner is called with the $EXTEND$ function with the objective of creating a movement towards a given task space target position x_{target} with maximum duration t_{max}, while simultaneously checking for collisions and returning the largest possible collision free duration t_{valid}. The task space target position

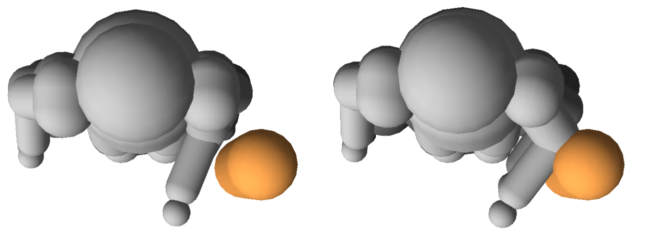

Fig. 3. Top view of a simulation of the Honda humanoid research robot doing a forward movement with one hand in presence of an obstacle, using reactive obstacle avoidance (left) and without obstacle avoidance (right).

is treated as the input to the underlying control system and by integrating the system forward in time, trajectories are created. When the iteration of the control system is done, $EXTEND$ returns not only the last task space position reached, but also the task space velocity at this point and the corresponding configuration space position. Thus more data has to be stored in the tree, appended by the ADD function, if the valid duration of the trajectory is between a minimum and maximum value. The loop is repeated, until the distance towards the goal gets smaller than a threshold.

As mentioned before, pure task space planning only considers a subset of possible motions and is not able to find solutions to all problems that are solvable. To get an impression of the fraction of solutions that can be found by task space planning, we compare task space planning against a planner that searches the configuration space, shown as algorithm 2. It follows the methodology of JT-RRT [9] by interleaving configuration space search with direct connection attempts in task space. In the goal biasing case the algorithm proceeds similar as task space EST, with the difference that the extension is always done towards the goal. Instead of utilizing the Jacobian transpose like JT-RRT, the $EXTEND$ function, i.e. our control system, is used here. The Jacobian transpose represents task space movements that decrease the distance to the target in every iteration, but not in an optimal way of taking the best direction possible to get closer to the target. Our control system does create an optimal movement towards the target by using the Jacobian pseudo-inverse and thus results in a stronger, more directed bias towards the goal. If no goal biasing is done, a classic configuration space search is carried out following the EST algorithm. The $CONNECT$ function performs direct connections in the configuration space.

B. Reactive Control

The local planner of the planning algorithm, is based upon the whole body motion control framework for the Honda humanoid research robot [2]. The framework solves the inverse kinematics problem for the upper body of the robot, following the resolved motion rate control principle, with the redundancy resolved with the gradient projection method, i.e. it generates joint velocities given a desired task position while utilizing the robots redundancy to optimize secondary motion objectives. Secondary motions objectives employed within this framework are joint limit avoidance [2], self-collision avoidance [4] and external collision avoidance [3]. Here all three criteria are used, following the formulation in [2], [3].

Tasks are encoded with task space attractor points as a compact movement representation. Possible task modalities are the left or right hand position $x \in \mathbb{R}^3$, as well as the so called hand-attitude description. Instead of specifying the complete orientation of the hand, the position of the grasp axis is given to which the hand aligns. The orientation around that axis is kept unconstrained. Thus the task attractor x representing the combined position and orientation lies on the 5 dimensional manifold $\mathbb{R}^3 \times \mathbb{S}^2$ embedded in $\mathbb{R}^6$.

Once a target task position x is set, an attractor point is continuously moved on a linear trajectory towards the target, with a simple second-order attractor dynamic generating suitable task velocities $\dot{x}$. This approach leads to a smooth bell shaped velocity profile with a limited acceleration. Due to this process, our task space model exhibits second-order differential constraints, i.e. the allowable accelerations at each point are restricted.

Now, with the task Jacobian $J(q) \in \mathbb{R}^{m\times n}$ relating joint velocities $\dot{q} \in \mathbb{R}^n$ to task velocities $\dot{x} \in \mathbb{R}^m$, the inverse problem of computing $\dot{q}$ given $\dot{x}$ can be solved with the gradient projection method (e.g. see [15]) as

$$\dot{q} = J^{\#}\dot{x} - NW^{-1}\left(\alpha\frac{\partial H}{\partial q}\right)^T . \qquad (1)$$

$J^{\#}$ denotes the weighted generalized pseudo-inverse $J^{\#} = W^{-1}J^T(JW^{-1}J^T)^{-1}$ and N the null space projection matrix $N = (1 - J^{\#}J)$. The robots redundancy is exploited by projection of a joint space gradient into the Jacobian null space that minimizes a cost function H by gradient descent with step width α. Since joint-limit avoidance and collision-avoidance motion criteria are incorporated into the framework, the cost function consists of two terms:

$$\frac{\partial H}{\partial q} = \frac{\partial H_{limit}}{\partial q} + \frac{\partial H_{coll}}{\partial q} \qquad (2)$$

The joint limit avoidance cost function H_{limit} penalizes deviations of individual joints q_i from a desired reference position $q_{0,i}$, normalized with respect to the range of the joint [15]:

$$H_{limit} = \sum_{i=1}^{n}\left(\frac{q_i - q_{0,i}}{q_{max,i} - q_{min,i}}\right)^2 \qquad (3)$$

The collision avoidance cost function H_{coll} penalizes both collisions of the robot with itself and collisions with external objects. To this end, pairs of closest points between robot segments and between robot segments and external objects are computed. The costs for each individual pair p_i with associated distance d_{p_i} are given by [3]

$$g_{p_i} = \begin{cases} s(d_{p_i} - d_B)^2 & 0 \le d_{p_i} \le d_B \\ 0 & d_{p_i} > d_B \end{cases} \qquad (4)$$

The overall cost function H_{coll}, incorporating all distances, is defined as the sum of individual costs over the set of all closest point pairs $\{p_i | i = 1..P\}$:

$$H_{coll} = \sum_{i=1}^{P} g_{p_i} \qquad (5)$$

The collision avoidance is only activated if the distance is below the threshold d_B. The parameter s determines the slope of the cost function g_{p_i}.

III. SIMULATION

The proposed hybrid planning method was tested in a simulation framework, combining the whole body motion control for the Honda humanoid research robot, the Vortex engine [20] for collision checking and OOPSMP [21], [22], a library implementing various sampling-based planning algorithms. Here a EST variant was used that assigns weights to nodes based on the number of outgoing edges, in order to estimate the level of exploration in a neighborhood around the nodes, as outlined in section II-A. Two setups were used, as shown in figure 4. In the first, the task is to reach towards a fixed position, with several obstacles placed in front of the robot that are randomly displaced vertically. In the second setup, multiple goals on a table have to be reached. For each setup and target position four distinct planners were run: task space EST (algorithm 1) with reactive and non-reactive control and configuration space EST with task space bias (algorithm 2), also with both reactive and non-reactive control. The minimum and maximum duration of extension paths were set to $t_{min} = 0.1$ and $t_{max} = 0.4$ and sampling was done with $\sigma = 0.5$ in the task space and $\sigma = 1.5$ in the configuration space. In the table setup a moderate goal bias of 0.25 was used and in the random obstacle setup a stronger value of 0.4. Task space targets comprise either only the 3D position of the left hand or the position and orientation (5D), while the configuration space involves 9 joints in a kinematic chain from the hip to the hand-tip. All simulations were done on workstations with Intel Core2 CPUs (3 GHz) and 4GB RAM. The overall runtime and number of tree extensions were averaged over 30 runs for each target and in the random obstacle placement task in addition over 100 different displacements.

IV. RESULTS

Figure 5 shows the simulation results for both setups, separated into the different variants of the planning algorithms and into the two task spaces used. An important value is the number of successfully solved positions, visible in the first column. Here the success rate of direct movement attempts without planning is also shown. The remaining statistics are computed for the successful runs only, again separated into two subsets. For a fair comparison, the first subset contains the planning queries that can be solved by all planners (filtered). The second subset contains queries that, in the task space, are only solved with the help of reactive obstacle avoidance (obst. avoid.).

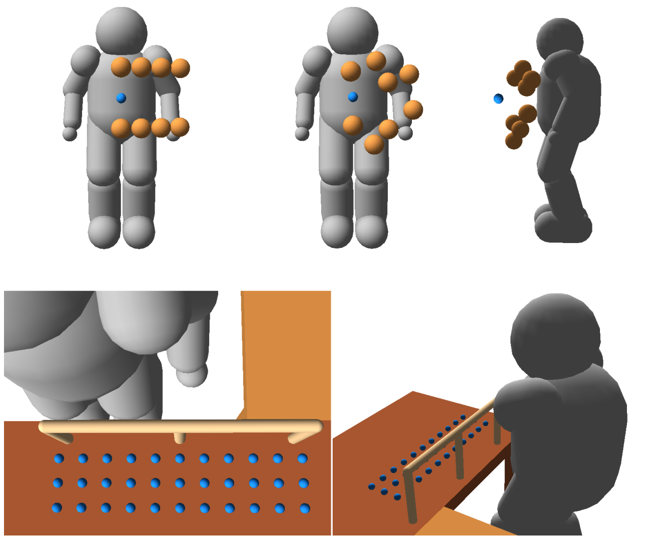
Fig. 4. The two setups used for simulation. Top row: Obstacles are placed in front of the robot and randomly displaced. Goal is to reach the position marked in blue with the left hand. Bottom row: Goal is to reach each of the marked positions with the left hand.

If we examine the average runtime of planning in the configuration and task space, there is an advantage in favor of task space planning visible. The number of tree-extensions shows the same qualitative result, even though the relative differences are greater. Comparing the average runtime for the cases with and without reactive obstacle avoidance, the most significant difference occurs if the task space is the 5-dimensional position and orientation space. For the 3-dimensional position space there is no clear difference visible.

Figure 6 shows the average runtime at every individual target position for the second setup. Here the results of figure 5 are subdivided on the spatial scale. The difference between configuration and task space planning, especially with reactive obstacle avoidance, tends to be greater on the left side of the table, more distant to reach for the left hand. Regarding reactive obstacle avoidance, the observation that the performance gain is larger for the 5-dimensional task space can be reproduced here.

Besides the average runtime, it is important to look at the percentage of solved runs. In both setups, configuration space planning is more successful in solving queries than task space planning. However the difference is quite small, what is especially apparent looking at the different table positions in figure 6. Here the spatial distribution of solved runs can be seen. Positions in the middle can not be reached at all. For the reachable positions, configuration space search solves a few targets more than reactive task space search and task space search without reactive obstacle avoidance finds the least solutions.

V. DISCUSSION

Our proposition of saving planning time by searching the task space instead of the configuration space holds for these two setups. The reduced dimensionality – from

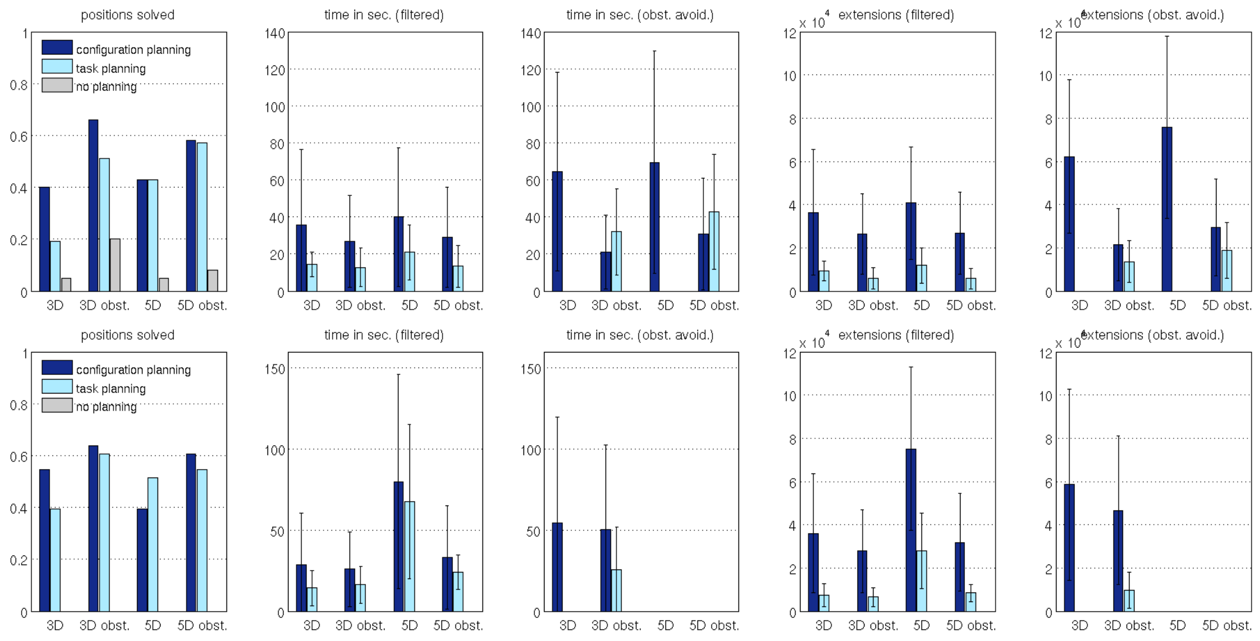

Fig. 5. Results for randomly displaced obstacles (top) and table scene (bottom), showing (from left to right) the fraction of queries solved, the average runtime for the subset of queries that are solved by all planners (filtered) and for the subset that is only solved with reactive obstacle avoidance (obst. avoid.). Further the average number of tree-extensions for the same subsets is shown. Dark blue bars show results for configuration space planning, light blue bars results for task space planning and grey bars results if no planning is done. The bars are grouped according to the task space dimension (3D or 5D) either with (obst.) or without obstacle avoidance.

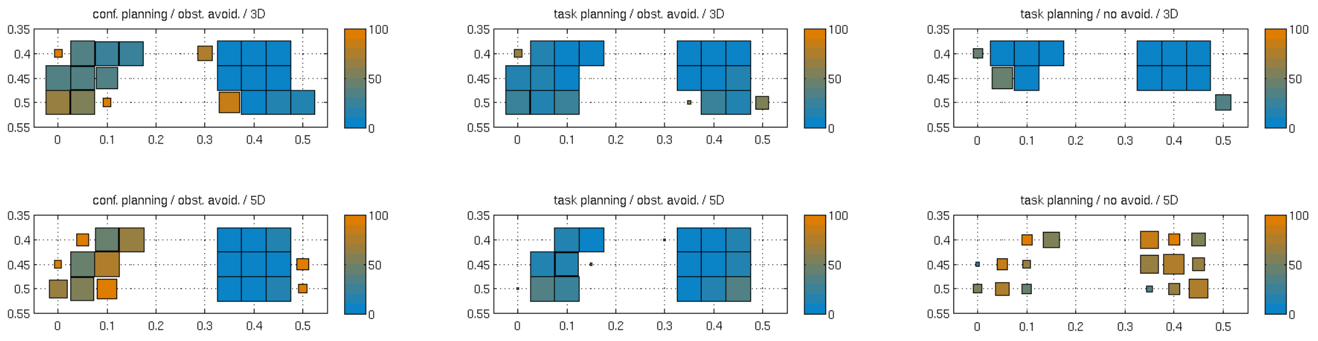

Fig. 6. Average solution time for all table positions, for planning the position (top, 3D) and position with orientation (bottom, 5D) of the hand. Three planners are shown here: Configuration space EST with task space bias using obstacle avoidance, task space EST with obstacle avoidance and task space EST without obstacle avoidance (from left to right). The size of the boxes depict the fraction of solved planning runs.

9 (configuration) to 5 and 3 (task) dimensions – can be exploited to speedup the search, although some overhead is added with the control framework involved. Every tree extension comes with the cost of computing the Jacobian pseudo-inverse in every iteration of the control system and for the case of reactive obstacle avoidance, a significant amount of time is spend computing closest distances.

Notably, the averaged runtimes show a large variance over all planning runs. One explanation can be ascribed to the random nature of the search process but also, figure 6 suggests that the runtimes vary across different target positions too. Some show an equally low or high runtime, while at some positions a greater difference can be seen for different planners. This can be interpreted as the difficulty to reach certain targets, and under this premise we argue that planning with reactive obstacle avoidance eases planning substantially for many difficult target positions.

In theory the configuration space search is able to find all possible solutions, while the task space search is limited to a subset. This is confirmed by the simulation, but only for some difficult situations. The incorporation of reactive obstacle avoidance enables task space search to solve nearly as many queries as configuration space search. Thus if this limitation can be neglected, task space planning is advantageous by speeding up the planning process.

Due to the limited computation time, solutions are not always found in time. This explains the observation in figure 6 that with increasing average runtime, the fraction of solved runs decreases. Also the difference between configuration space search with and without reactive obstacle avoidance

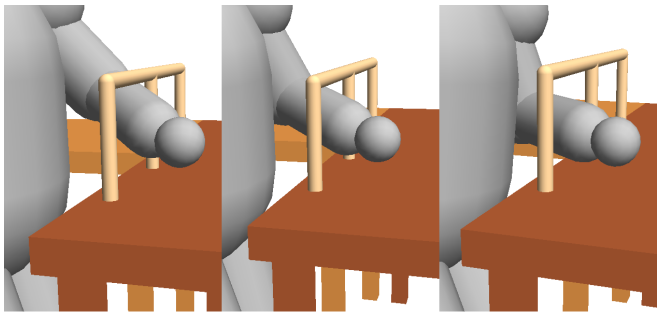

Fig. 7. Approaching a 5D target position (right) with non-reactive (left) and reactive control (middle). The posture with reactive control (middle) already lies closer to the target position. Thus reactive control simplifies task space planning here.

can be explained by this circumstance, because the search takes longer for the latter case and thus was prematurely terminated more often.

An interesting counter intuitive result is the fact that reactive obstacle avoidance has a bigger effect in the case of the 5 dimensional task space, since in that case the redundant space that can be used for avoidance motions is more limited than in the case of a 3 dimensional task space. An intuition to why this happens is shown in figure 7. On the left side, the arm posture without reactive behavior is shown, and in the middle the same posture with reactive behavior. Compared to the target position on the right side, with the grasp-axis pointing upwards, the reactive posture is already closer to the goal. Thus reactive obstacle avoidance is more effective here because the goal incidentally lies close to the posture generated by the reactive behavior.

VI. CONCLUSIONS

A combination of classical sampling-based random tree planning with a reactive control framework that is able to exploit redundancy to avoid obstacles is presented in this paper. The effort of planning motions is reduced by shifting the search to a lower-dimensional task space. Although the subset of planning queries that can be solved is limited, hybrid task space planning can solve a large fraction of queries. The use of redundant obstacle avoidance enhances the applicability of the approach by enlarging the number of queries that can be solved. This is shown by a simulation study with the whole body motion control framework for the Honda humanoid research robot and an adaption of the EST sampling-based planning algorithm for planning in the task space. We compare task space planning against an algorithm for configuration space planning that utilizes the control framework to bias the search in the task space. For cases where pure task space planning fails, it is feasible to switch towards planning in the configuration space, if a strong bias in the task space is imposed. Building upon these results, the development of a planner with the ability of interleaved task space and configuration space planning, depending on the difficulty of the planning problem, might be an interesting subject for further research.

ACKNOWLEDGMENTS

Matthias Behnisch gratefully acknowledges the financial support from Honda Research Institute Europe for the project "Sampling-based planning for goal directed robot control".

REFERENCES

[1] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, vol. 5, no. 1, p. 90, 1986.
[2] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2005.
[3] M. Toussaint, M. Gienger, and C. Goerick, "Optimization of sequential attractor-based movement for compact movement representation," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2007.
[4] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick, "Real-time self collision avoidance with whole body motion control for humanoid robots," in *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, 2007.
[5] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.
[6] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
[7] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006.
[8] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Computer Science Dept, Iowa State University, Tech. Rep. TR*, pp. 98–11, 1998.
[9] M. V. Weghe, D. Ferguson, and S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2007.
[10] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner, "Bispace planning: Concurrent multi-space exploration," in *Proceedings of Robotics: Science and Systems IV*, 2008.
[11] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space voronoi bias," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.
[12] O. Brock and L. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2001.
[13] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *The International Journal of Robotics Research*, vol. 21, no. 12, p. 1031, 2002.
[14] Y. Yang and O. Brock, "Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments," in *Proceedings of Robotics: Science and Systems*, 2006.
[15] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley Publishing Co., Inc., 1991.
[16] D. Hsu, R. Kindel, J. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, p. 233, 2002.
[17] S. LaValle and J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions: the Fourth Workshop on the Algorithmic Foundations of Robotics*, 2001, p. 293.
[18] J. Phillips, N. Bedrossian, and L. Kavraki, "Guided expansive spaces trees: a search strategy for motion-and cost-constrained state spaces," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.
[19] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007, pp. 3307–3312.
[20] CM Labs Vortex. [Online]. Available: http://www.vxsim.com/
[21] E. Plaku, K. Bekris, and L. Kavraki, "Oops for motion planning: An online, open-source, programming system," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.
[22] OOPSMP. [Online]. Available: http://www.kavrakilab.rice.edu/