Motion Planning for Robotic Manipulators with Independent Wrist Joints

Kalin Gochev† Venkatraman Narayanan‡ Benjamin Cohen† Alla Safonova† Maxim Likhachev‡

***Abstract*— Advanced modern humanoid robots often have complex manipulators with a large number of degrees of freedom. Thus, motion planning for such manipulators is a very computationally challenging problem. However, often robotic manipulators allow the wrist degrees of freedom to be controlled independently from the configuration of the rest of the arm. In this paper we show how to split the high dimensional planning problem into two lower-dimensional sub-problems—planning for the main arm joints and planning for the wrist joints, without losing guarantees on completeness. This approach is an extension of our previously developed framework for planning with adaptive dimensionality. Experimentally, we show that this approach is very effective in speeding up planning for robotic arms on Willow Garage's PR2 platform. We compare our algorithm with several popular alternative approaches for performing motion planning for robotic arms. The results we observe illustrate that our algorithm provides a good balance between planning time, planning success rate, path consistency, and path quality.**

Keywords: Motion Planning, Manipulation Planning, Planning Algorithms, Heuristic Search

I. Introduction

With the increased availability of humanoid robots, motion planning for robotic manipulators is becoming a very common planning problem. Robots need to be able to quickly plan trajectories for their manipulators in order to safely interact with their surroundings. However, advanced humanoid robots, such as Honda's Asimo, Willow Garage's PR2, and CMU's HERB, often have complex manipulators with a large number of degrees of freedom. This high dimensionality of the robotic manipulator makes the motion planning problem computationally challenging.

Often robotic manipulators allow the wrist degrees of freedom to be controlled independently from the configuration of the rest of the arm. This is true of the arms on the PR2, Barrett arm, KUKA arm, and other popular robotic arms. This property allows us to decouple planning for the main arm joints from planning for the wrist joints, thus splitting the planning problem into two lower-dimensional sub-problems. Considering only the main arm joints gives us a state-space of lower dimensionality, which makes this approach well suited for our previously developed framework for planning with adaptive dimensionality [1], [2]. The algorithm for planning with adaptive dimensionality searches through a lower dimensional state-space and it only considers the additional degrees of freedom of the wrist in the areas of the state-space that *require* reasoning about the end-effector orientation.

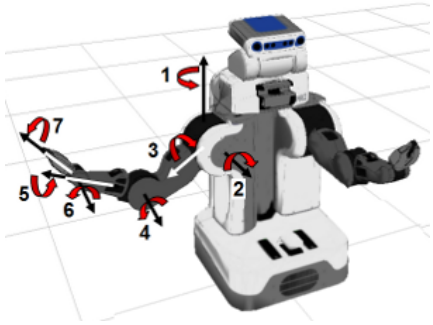

Fig. 1. The degrees of freedom of the arm of a PR2 robot: (1) shoulder pan, (2) shoulder lift, (3) shoulder roll, (4) elbow flex, (5) forearm roll, (6) wrist flex, (7) wrist roll.

In this work we develop an extension to our framework for planning with adaptive dimensionality, which exploits the specific properties of planning for robotic manipulators with independent wrist joints. In particular, it exploits the fact that the wrist can be controlled independently from the rest of the arm to split the planning problem into two much easier sub-problems, while maintaining strong theoretical guarantees, such as completeness and bounds on solution cost sub-optimality. We apply this algorithm to 7-DoF motion planning for a robotic arm on Willow Garage's PR2 platform. We compare our algorithm with several popular alternative approaches for performing motion planning for robotic arms. The results we observe illustrate that our algorithm provides a good balance between planning time, planning success rate, and path quality.

II. Related Work

Sampling-based motion planners [3], [4], [5] have become extremely popular in recent years. They have been shown to solve impressive high-dimensional motion planning problems, while being simple, fast, and easy to implement. These methods have also been extended to support motion constraints through rejection sampling [6].

Our approach to motion planning differs from the sampling-based methods in several important aspects. First, sampling-based motion planners are mainly concerned with finding any feasible path, rather than minimizing the cost of a solution. The notable exception is the RRT* algorithm [7], which is one of the algorithms that we compare our approach

This research was sponsored by ONR grant N00014-09-1-1052, DARPA CSSG program D11AP00275 and the Army Research Laboratory Cooperative Agreement Number W911NF-10-2-0016.

† Department of Computer and Information Science, University of Pennsylvania

‡ Robotics Institute, Carnegie Mellon University

against experimentally. These approaches sacrifice cost minimization in order to gain very fast planning speeds. As such, they may often produce solutions of unpredictable length involving highly sub-optimal or jerky motions that may be hard for the manipulator to follow. To compensate for the lack of solution cost minimization, sampling-based methods rely on various smoothing techniques to improve the quality of the computed trajectory. While often helpful, smoothing may fail in cluttered environments. Second, search-based planners produce more consistent solutions between planning episodes with similar start and goal configurations.

Several motion planning algorithms have been developed that also try to minimize the cost of solutions through optimization techniques [8], [9]. The Covariant Hamiltonian Optimization and Motion Planning (CHOMP) algorithm [8] works by creating a naive initial trajectory from start to goal, and then uses a method similar to gradient descent to try to minimize the cost function. The use of gradient descent, however, makes the approach vulnerable to local minima in the cost function. The Stochastic Trajectory Optimization for Motion Planning (STOMP) algorithm [9], on the other hand, relies on generating noisy trajectories to explore the space around a naive initial trajectory. It then iteratively combines these trajectories to produce an updated trajectory with lower cost. A cost function based on a combination of obstacle and smoothness cost is optimized in each iteration. The stochastic nature of the approach makes it less vulnerable to local minima in the cost function.

A variety of techniques have been developed in order to improve planning times in high-dimensional configuration spaces. Many approaches involve a two layer planning scheme with a low-dimensional global planner that provides input to a high-dimensional local planner. The local planners operate on a much smaller subset of the environment, and thus, they can afford to incorporate more dimensions into the planning process while meeting planning time constraints [10], [11], [12]. However, these approaches can result in highly sub-optimal trajectories and even trajectories that are infeasible due to mismatches in the assumptions made by the global and local planners. Another way to reduce the dimensionality of planning problems is through the use of hierarchical planners and state abstraction, such as the approaches taken in [13], [14]. Computing very accurate heuristics is another way of improving the performance of searches through high-dimensional state-spaces [15]. The heuristics are often derived by solving a relaxed lower-dimensional representation of the original planning problem. Planning with adaptive dimensionality differs from the approaches above in several aspects. It does not explicitly split the planning in two levels, but rather mixes low-dimensional and high-dimensional states within a single planning process. Planning with adaptive dimensionality can also make effective use of a very accurate heuristic, but it does not rely on the heuristic alone to improve performance. Thus, it is more robust to handling local minima in the heuristic function.

III. Motion Planning for Manipulators with Independent Wrist Joints

A. Problem Definition

Following the approach taken in [16], we are assuming that the planning problem is represented by a discretized finite state-space S and a set of transitions $T = \{(X_i, X_j) | X_i, X_j \in S\}$. Every state X is defined by a state-vector of discretized joint angle values for all the joints of the manipulator. Each transition, or motion primitive, $(X_i, X_j) \in T$ represents to a feasible transition between the corresponding joint angle values of the two states, and is associated with a positive cost $c(X_i, X_j)$. Thus, we have an edge-weighted graph G with a vertex set S and edge set T. We will use the notation $\pi_G(X_i, X_j)$ to denote a path from state X_i to state X_j in G, and its cost will be denoted by $c(\pi_G(X_i, X_j))$. We will use $\pi^*_G(X_i, X_j)$ to denote a least-cost path. The goal of the planner is to find a least-cost path in G from a given start state X_S to a goal state X_G. Alternatively, given a desired sub-optimality bound $\epsilon \geq 1$, the goal of the planner is to find a path $\pi^\epsilon_G(X_S, X_G)$, such that $c(\pi^\epsilon_G(X_S, X_G)) \leq \epsilon \cdot c(\pi^*_G(X_S, X_G))$.

B. Algorithm

In this work we consider the motion planning problem for robotic manipulators whose joints can be controlled independently from the configuration of the rest of the arm. Our approach subdivides the original high-dimensional planning problem into two lower-dimensional problems—planning for the main arm joints, and planning for the wrist joints. Our high-dimensional state-space S^{hd} is defined by the full arm configuration and a set of high-dimensional transitions T^{hd}, giving us a high-dimensional edge-weighted graph G^{hd} encoding the original search problem. We consider a lower-dimensional state-space S^{ld}, which only includes the main arm joints, disregarding the degrees of freedom of the wrist, and a lower-dimensional transition set T^{ld}. S^{ld} and T^{ld} give us a low-dimensional edge-weighted graph G^{ld}. We use our previously developed framework for planning with adaptive dimensionality to iteratively construct and search a hybrid graph G^{ad}, which contains both high-dimensional and low-dimensional states. Initially G^{ad} is identical to G^{ld} (Alg. 1 Line 1). As the search progresses, regions of high-dimensional states and transitions are added to G^{ad} only in areas where high-dimensional planning is necessary to maintain guarantees on completeness and bounds on solution cost sub-optimality. High-dimensional regions are always introduced in G^{ad} around the start and goal states, X_S and X_G (Alg. 1 Lines 2,3). Just like the original algorithm for planning with adaptive dimensionality, each iteration of our algorithm consists of two phases: planning phase and tracking phase.

In the planning phase, the current instance of G^{ad} is searched for a path $\pi_{G^{ad}}(X_S, X_G)$, which is of cost no greater than ϵ_{plan} times the optimal path cost from start to goal in G^{ad}. Any graph search algorithm that provides a bound on path cost sub-optimality can be used to compute

$\pi_{G^{ad}}$. In this work we use an incremental weighted A^* algorithm as in [17], which re-uses search efforts between algorithm iterations to speed up the planning. The planning phase, in effect, solves the first sub-problem of our high-dimensional planning problem, by providing a trajectory for the main arm angles only.

The tracking phase then needs to solve the second sub-problem—planning for the wrist—and provide a feasible, collision-free fully high-dimensional trajectory for the manipulator from start to goal. In this work we propose two approaches for extending and speeding up the tracking phase of the algorithm, which are based on the fact that the wrist degrees of freedom can be controlled independently of the rest of the arm.

C. Extended Tracking Phase

Building on the properties of motion planning for manipulators with independent wrist joints, we can dramatically speed up the tracking phase. In particular, we introduce two very fast ways to try to convert the path generated by the adaptive planning phase into a feasible fully high-dimensional trajectory that satisfies the desired cost sub-optimality bound. The feasibility and correctness of these approaches rely on the property that the wrist joint can be controlled independently from the rest of the arm. Thus, the tracking phase now consists of 3 steps.

1) Interpolation: Consider an adaptive path $\pi_{ad} = S_1, S_2, ...S_n$ computed by the adaptive planning phase of the current iteration of the algorithm. As G^{ad} is a hybrid graph consisting of both low- and high-dimensional states, π_{ad} is a path that also consists of both low- and high-dimensional states. Also S_1 and S_n are the start X_S and goal X_G states, respectively, and are always high-dimensional states. Without loss of generality, let $\pi_{ld} = S_i, S_{i+1}, ..., S_{i+k}$ be a segment of π_{ad} containing only low-dimensional states, such that the state S_{i-1} preceding the segment, and the state S_{i+k+1} following the segment are high-dimensional states. Thus, we know the desired wrist joint coordinates at the beginning and at the end of π_{ld}, but we do not have information about the wrist joint coordinates throughout π_{ld}. We can interpolate between the two desired wrist joint coordinates to compute wrist joint coordinates for each of the low-D states on the segment π_{ld}. If we use such interpolation for every low-D segment along the adaptive path π_{ad}, we can convert the adaptive path to a fully high-dimensional path π_{interp} (Alg. 1, Line 12). The use of interpolation is feasible only if the degrees of freedom of the wrist can be controlled independently from the other joint angles in the arm, otherwise the wrist trajectory generated by interpolation might not be feasible for execution by the manipulator. If π_{interp} is collision-free and satisfies the joint limit constraints, and moreover, its cost satisfies the sub-optimality bound criteria $c(\pi_{interp}) \leq \epsilon_{track} \cdot c(\pi_{ad})$, then π_{interp} is a valid high-dimensional path that satisfies the desired sub-optimality bound $c(\pi_{interp}) \leq \epsilon_{plan} \cdot \epsilon_{track} \cdot \pi^*_{G^{hd}}$. Thus, we can stop planning and return π_{interp} as a valid plan. If the interpolation step fails to produce a feasible collision-free path, the tracking phase proceeds to step 2 (planning for the wrist joint, described below). If π_{interp} is invalid, the locations where it violates system constraints, such as collisions with the environment or joint limits, are used as potential locations for introducing new high-dimensional regions into G^{ad}. Since interpolation is very fast, this additional step does not add any significant computational burden per iteration. Moreover, in open environments with few obstacles, this approach is very effective in quickly producing a valid high-dimensional path.

Algorithm 1 Planning with adaptive dimensionality for independent wrist joints

```
1: G^ad = G^ld
2: Add-HD-Region(G^ad, X_S)
3: Add-HD-Region(G^ad, X_G)
4: loop
5:                                          ▷ Adaptive Planning Phase
6:     search G^ad for a path π_ad(X_S, X_G)
7:     if π_ad(X_S, X_G) is not found then
8:         return no path from X_S to X_G exists
9:     end if
10:                                                ▷ Tracking Phase
11:                                                ▷ 1. Interpolation
12:    π_interp = ComputeInterpolatedPath(π_ad)
13:    if interp. success and c(π_interp) ≤ ε_track · c(π_ad) then
14:        return π_interp
15:    end if
16:                                    ▷ 2. Planning for the wrist joints
17:    π_hd = PlanForWristJoints(π_ad)
18:    if wrist plan success and c(π_hd) ≤ ε_track · c(π_ad) then
19:        return π_hd
20:    end if
21:                                    ▷ 3. High-dimensional tracking
22:    construct a tunnel τ around π_ad(X_S, X_G)
23:    search τ for a path π_τ(X_S, X_G)
24:    if π_τ(X_S, X_G) is not found then
25:        find state(s) X_r where to insert new HD region(s)
26:        Add-or-Grow-HD-Region(G^ad, X_r)
27:    else if c(π_τ(X_S, X_G)) > ε_track · c(π_ad(X_S, X_G)) then
28:        find state(s) X_r where to insert new HD region(s)
29:        Add-or-Grow-HD-Region(G^ad, X_r)
30:    else
31:        return π_τ(X_S, X_G)
32:    end if
33: end loop
```

2) Planning for the wrist joint: The fact that the degrees of freedom controlling the wrist are independent from the configuration of the rest of the arm allows us to treat the wrist separately. Thus, the second step of the tracking phase is effectively a search through the wrist configurations over the adaptive path $\pi_{ad} = S_1, S_2, ...S_n$ computed by the planning phase of the current iteration (Fig. 2, Alg. 1 Line 17). This step gets executed only if the interpolation in step 1, described above, fails to produce a valid path. Each state in this state-space S^w is defined by a state vector $(wrist, i)$, where $wrist$ is a vector of the wrist joint coordinates, and $i = 1...n$ is a path index. In addition, each such state $X = (wrist, k)$ corresponds to a high-dimensional state $X' = (S^{arm}_k, wrist)$, where S^{arm}_k is the state vector of the main arm joint coordinates of the k-th state in π_{ad} (S_k), and

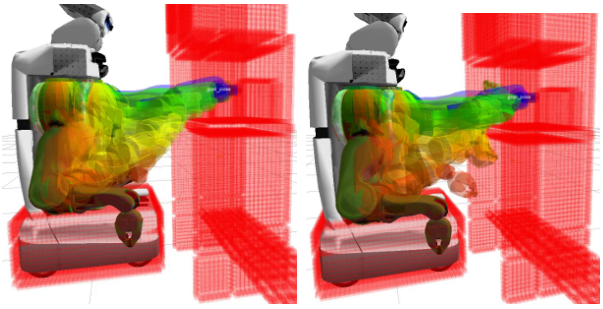

Fig. 2. Trajectory computed for the 4 main arm angles during the adaptive planning phase (left) and the resulting 7-DoF trajectory after successful planning for the wrist in the tracking phase (right).

$wrist$ is the vector of wrist joint coordinates of the state X. Thus, the state X augments the adaptive state S_k with information about the wrist joint coordinates to produce a fully-defined high-dimensional state X' for the entire arm. Let us denote this mapping by $\Omega_{\pi_{ad}} : S^w \rightarrow S^{hd}$. We will omit the subscript π_{ad} if it is understood. Note that $\Omega_{\pi_{ad}}$ disregards any wrist orientation information for the high-dimensional states in π_{ad} computed during the adaptive planning phase.

The start state of this search is then $W_S = (S_1^{wrist}, 1)$, where S_1^{wrist} is the vector of wrist joint coordinates of S_1, and the goal state is $W_G = (S_n^{wrist}, n)$, where S_n^{wrist} is the vector of wrist joint coordinates of S_n. Note that S_1 and S_n are the start state X_S and the goal state X_G, respectively, and thus are always high-dimensional, so S_1^{wrist} and S_n^{wrist} are defined. Also, $\Omega(W_S) = X_S$ and $\Omega(W_G) = X_G$.

We allow the following transitions T^w within this state-space:

- We allow the path index to increase by 1, while the wrist joint coordinates remain the same: $(wrist, i) \Rightarrow (wrist, i{+}1)$ (if $i{+}1 \leq n$). This corresponds to moving the arm along the computed path without changing the wrist angles.
- We allow the path index to remain the same, but the wrist joint coordinates to change by using a set of feasible transitions for the wrist: $(wrist_1, i) \Rightarrow (wrist_2, i)$. This corresponds to changing the wrist angles only, without changing the configuration of the main arm joints.
- We allow the path index to increase by 1 and also the wrist joint coordinates to change by using a set of feasible transitions for the wrist: $(wrist_1, i) \Rightarrow (wrist_2, i+1)$ (if $(i+1 \leq n)$. This corresponds to changing the wrist angles while moving along the computed path.

Such transitions are only feasible if the degrees of freedom of the wrist can be controlled independently from the other joint angles in the arm. The cost of each transition

$$X = (wrist_x, i) \Rightarrow Y = (wrist_y, j)$$

is equal to the cost between the two corresponding high-dimensional states $X' = \Omega(X) = (S_i^{arm}, wrist_x)$ and $Y' = \Omega(Y) = (S_j^{arm}, wrist_y)$. We also perform high-dimensional collision-checking on the transition $X' \Rightarrow Y'$ and invalid transitions are discarded by the search.

If we find a path π_{S^w} from start to goal through this graph $G^w = (S^w, T^w)$, we can convert it to a complete high-dimensional path π_{hd} by using the mapping Ω (Alg. 1, Line 17). Then π_{hd} is a valid path from the start arm configuration X_S to the goal arm configuration X_G. If its cost satisfies the sub-optimality bound criteria $c(\pi_{hd}) \leq \epsilon_{track} \cdot c(\pi_{ad})$, then π_{hd} is a valid high-dimensional path that satisfies the desired sub-optimality bound $c(\pi_{hd}) \leq \epsilon_{plan} \cdot \epsilon_{track} \cdot \pi^*_{G^{hd}}$. Thus, we can stop planning and return π_{hd} as a valid path. If π_{S^w} does not exist or $c(\pi_{hd}) > \epsilon_{track} \cdot c(\pi_{ad})$, we proceed to step 3 of the tracking phase—high-dimensional tracking. If the search fails, the location of the state with the highest path index value expanded during the search is used as a potential location for introducing a new high-dimensional region into G^{ad}, as it indicates the location farthest along π_{ad} the search was able to reach before it failed, and that location might require high-dimensional planning.

The search through S^w is very constrained and low-dimensional. As such, it usually completes very quickly and incurs only a minor computational burden on the tracking phase. Moreover, our results suggest that it is extremely effective in computing feasible high-dimensional paths even in cluttered environments.

Tracking steps 1 and 2 solve the second sub-problem of our original high-dimensional planning problem, augmenting the solution of the first sub-problem with valid coordinates for the wrist joint angles to produce a valid feasible trajectory for the full arm. If steps 1 and 2 fail to produce a valid high-dimensional path from start to goal, we revert to the default method for tracking used by planning with adaptive dimensionality.

3) High-dimensional tracking: High-dimensional tracking (Alg. 1, Lines 22-32) is described in detail in [1] and [2]. The method involves constructing a high-dimensional tunnel τ (subgraph of G^{hd}) around the path computed by the adaptive planning phase π_{ad}. Then a high-dimensional search from start to goal is performed confined to τ—transitions that lead to states outside τ are discarded. If a path from start to goal π_τ is found and its cost satisfies $c(\pi_\tau) \leq \epsilon_{track} \cdot c(\pi_{ad})$, then π_τ is returned as a valid path from start to goal. If π_τ does not exist or its cost is too large, new high-dimensional regions are introduced into G^{ad} and the adaptive planner proceeds to the next iteration.

D. Theoretical Properties

The extended algorithm for planning with adaptive dimensionality preserves the theoretical properties of the original algorithm. If the high-dimensional state-space S^{hd} is finite, the algorithm is complete with respect to the underlying graph G^{hd} encoding the search problem and is guaranteed to terminate. If a path π is found by the algorithm, then π satisfies

$$c(\pi) \leq \epsilon_{plan} \cdot \epsilon_{track} \cdot \pi^*_{G^{hd}}(X_S, X_G)$$

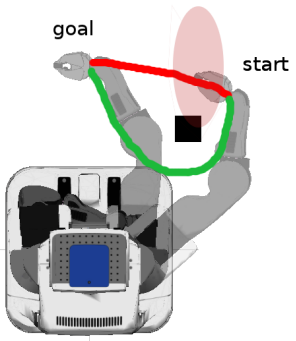

Fig. 3. PR2 planning an arm motion around a thin tall obstacle. Black box: tall obstacle, red: heuristic shortest path, green: feasible end-effector path, shaded region: heuristic function local minimum.

where ϵ_{plan} is the sub-optimality bound used in the search through the hybrid state-space G^{ad} in the adaptive planning phase of the algorithm. In other words, the cost of a path returned by the algorithm is bounded by $\epsilon_{plan} \cdot \epsilon_{track}$ times the cost of an optimal path from X_S to X_G in the high-dimensional graph G^{hd}.

The theoretical properties of the algorithm guarantee that on a finite state-space the algorithm will find a path from start to goal if one exists, and that the cost of the returned path will be within a user-specified sub-optimality bound of the optimal path cost. This makes the approach suitable for applications where the minimization of a cost function, such as energy consumption or time to reach the goal, is an important concern. Moreover, the deterministic nature of the algorithm makes it very consistent and predictable–similar start/goal configurations yield similar solutions. Consistency and predictability are usually desirable properties when robots are working around humans.

IV. Implementation

A. State-spaces

To validate our algorithm, we consider the problem of motion planning for a 7-DoF robotic arm on Willow Garage's PR2 platform. The full arm configuration on the PR2 is defined by its seven joint angles: shoulder pan, shoulder lift, upper arm roll, elbow flex, forearm roll, wrist flex, and wrist roll (Fig. 1). Thus, we have a 7-DoF high-dimensional state-space S^{hd}. Low-dimensional states, on the other hand, are defined by only 4 angles: shoulder pan, shoulder lift, upper arm roll, and elbow flex, disregarding the 3 degrees of freedom controlling the wrist. In our implementation, all angles are uniformly discretized with 3° resolution within their respective joint limit intervals. Full-arm (and grasped object, if any) collision checking against the environment is performed on 7D states. A more relaxed collision checking is performed on 4D states–only the upper arm and the forearm links are collision-checked against the environment. Since 4D states do not contain information about the end-effector orientation, it is not possible to perform gripper and grasped object collision checking.

Planning for the wrist joint over an adaptive path (step 2 of the tracking phase) is done in a 4-DoF state-space S^w defined by 4D state vectors (forearm roll, wrist flex, wrist roll, path index).

In [1], 7D/3D planning with adaptive dimensionality was used to perform motion planning for the arm of a PR2 robot, where 3D states represented the end-effector position in (x, y, z). In contrast, in this work we choose to do the planning with adaptive dimensionality in 7D/4D. Firstly, this allows us to speed up the tracking phase, as described in section III-C. Second, the four dimensions we select for the low-dimensional states determine the positions and orientations of the two largest links of the arm–the upper arm and the forearm. This allows for much more accurate collision checking for low-D states, as the positions of the two largest links of the arm are known for low-D states. Thus, the adaptive planning phase produces trajectories that are more likely to be tracked successfully. For example, when using 3D end-effector (x, y, z) low-D states, the planning phase will produce a low-D end-effector path similar to the one shown in red in Fig. 3, which will be impossible for the tracking phase to follow and a new high-D region will be introduced behind the obstacle. By using 4D main arm joint angles as low-D states, on the other hand, the planning phase search will produce a low-D path similar to the one shown in green in Fig. 3, which will be more likely to be tracked successfully without additional iterations being necessary.

A desired 6-DoF cartesian pose is used to define the goal. Note that, due to the redundancy in the manipulator, a 6-DoF cartesian pose corresponds to multiple goal states in the 7-DoF state-space of the arm.

B. Transitions and Cost Function

We use motion primitives, similarly to [16], to construct the transitions we use in our graph. Thus, each transition represents a feasible path from one state to another. The graph is constructed dynamically as the graph search progresses, as the size of the state-space is prohibitively large to pre-compute the entire graph. Similar to [16], each of our high-dimensional transitions is a 7-DoF vector of joint velocities for each of the joints in the arm, and a 4-DoF vector for low-dimensional states. We use a very simple set of fixed transitions, allowing only one joint angle to change with each transition. For each joint, we have 2 short transitions allowing ± 1 unit of discretization change in the joint angle value, and 2 long transitions allowing ± 2 units of discretization change in the joint angle value. Thus, we have a total of 28 possible transitions for each high-dimensional state, and 16 possible transitions for low-dimensional states. Similarly, when planning for the wrist joint over an adaptive path (step 2 of the tracking phase) we use transitions allowing ± 1 unit of discretization change in the joint angle value for each of the three wrist joints. These transitions were selected for the sake of simplicity. However, more complex transitions that operate on several joints simultaneously can be used by the planner.

We take the same approach as in [18] for computing dynamic transitions. For any high-dimensional state S whose end-effector position is within a fixed distance threshold of the goal position, we try to compute a dynamic transition using inverse kinematics. The inverse kinematics solver is seeded with the joint angles $angles_S$ of the state S and is asked to compute joint angles $angles_{IK}$ that satisfy the goal position and orientation of the end-effector (i.e. the 6-DoF cartesian goal pose). If the kinematics solver is able to compute joint angles $angles_{IK}$ satisfying the goal constraints, and the interpolated trajectory from $angles_S$ to $angles_{IK}$ is collision-free and obeys joint limit constraints, then this trajectory (from $angles_S$ to $angles_{IK}$) is used as a transition from S to the goal state defined by $angles_{IK}$.

For any high-dimensional state S whose end-effector position matches exactly the goal position, we use an analytical solver to compute the values for the forearm roll, wrist flex, and wrist roll angles, that would satisfy the goal orientation constraints, while maintaining the same values for the other 4 joint angles [18]. If the transition from S to the desired values for forearm roll, wrist flex, and wrist roll is collision-free and obeys joint limit constraints, it is used as a transition from S to the goal state.

C. Heuristic

The heuristic function in a graph search serves to improve the efficiency of the search by guiding it in promising directions. A common way to compute a heuristic function is to use a simplified lower-dimensional version of the planning problem. In order to be informative, the heuristic needs to capture environment properties, such as obstacles. This is especially true for cluttered environments. To compute the heuristic function, we discretize the environment into 3D voxels and we use a 3D Dijkstra's search accounting for obstacles to find the least cost paths for the end-effector from every voxel to the goal voxel (corresponding to the (x, y, z) position of the cartesian goal pose). We use a highly optimized implementation of 3D Dijkstra's search, which is able to very quickly compute the heuristic. This heuristic is very helpful in guiding the search around the obstacles in the environment and towards the cartesian goal position. Figure 3 shows an example where the 3D Dijkstra's search heuristic has a pronounced local minimum (shaded area behind the black obstacle). Our approach is quite robust with respect to such local minima as these local minima are overcome by expanding states in the much smaller 4D state-space.

V. Experimental Setup

We used Willow Garage's PR2 robot as the testing platform to evaluate the performance of our new approach for arm motion planning with adaptive dimensionality. Most of our experiments were performed in an open-source simulator called Gazebo. The robot model used in simulation is a fairly accurate representation of the PR2. We also ran several experiments on the PR2 itself.

Algorithm	Planning Time (s)				Successful
	mean	std dev	min	max	Plans
7D/4D Adaptive	0.93	0.70	0.03	8.47	87.36%
4D ARA*	0.12	0.16	0.01	1.27	71.51%
7D ARA*	2.96	2.00	0.01	9.95	52.96%
OMPL PRM	0.33	2.13	0.01	9.43	83.80%
OMPL RRT-Connect	0.03	0.03	0.01	0.39	86.62%
OMPL RRT*	0.36	1.32	0.01	9.73	86.42%

TABLE I

Planning time and success rate comparison between arm planners on 524 planning scenarios in simulation. Results for all sampling-based (OMPL) planners are averaged over 10 planning trials on each scenario.

A. Simulation

To measure the performance of the algorithm, we used 524 planning scenarios through various environments. The difficulty level of the environments varied from obstacle-free to highly cluttered. Some examples of environments used in our simulations are shown in Fig. 4—various tables, shelves, bookcases, and cuboid obstacles of random sizes and locations. The difficulty level of the 524 planning scenarios varied based on the environment used in the scenario, and the particular start and goal configurations. In some scenarios the path from start to goal was fairly trivial, where in others, highly complex maneuvering was necessary to reach the goal. We compared our 7D/4D adaptive planner to a number of popular planners available from the Open Motion Planning Library (OMPL) [19]—PRM planner, RRT-Connect planner, and RRT* planner. We also compared against a 4D ARA* planner that only considers the wrist orientation near the goal position, and a 7D ARA* planner that plans in all 7-DoF. Each planner was given a 10-second planning limit to produce a path for each of the 524 environments. If a planner failed to produce a path within the allowed time limit, the scenario was reported as failure. Due to their randomized nature, the sampling-based OMPL planners were given 10 planning trials on each of the scenarios and the observed results were averaged.

B. Physical PR2

We also developed a framework for the use of the 7D/4D adaptive planner on a real PR2. The framework accepts and serves planning requests to a desired 6-DoF cartesian pose for the end-effector either programmatically, or through the use of a GUI. We observed quick responsiveness from the planner when asked to produce paths through typical household environments. An example scenario of the robot reaching into a refrigerator, grasping an object, and safely retrieving the object from the fridge is shown in the video accompanying this work. Since grasp selection is outside the scope of this work, a teleoperator selected a suitable grasp pose.

VI. Results

As seen in Table I, the adaptive planner was not able to match the planning times of the sampling-based OMPL

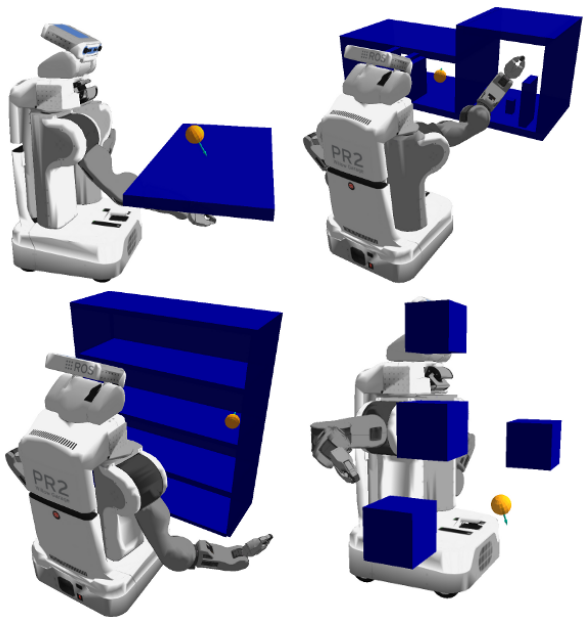

Fig. 4. Examples of environments used in simulation: table-top, shelf, bookcase, cuboid obstacles

Algorithm	Distance Traveled (m)					
	Wrist		Gripper Tip		Elbow	
	mean	std dev	mean	std dev	mean	std dev
7D/4D Adaptive	1.30	0.70	1.84	0.72	0.64	0.36
4D ARA*	1.64	0.91	1.98	1.22	0.79	0.41
7D ARA*	1.44	0.80	1.86	1.20	0.71	0.39
OMPL PRM	1.75	0.91	2.23	1.12	1.10	0.58
OMPL RRT-Connect	1.56	0.79	1.93	0.93	1.01	0.52
OMPL RRT*	1.53	0.77	1.91	0.92	0.98	0.50

TABLE II

PATH QUALITY COMPARISON BETWEEN VARIOUS ARM PLANNERS: THE AVERAGE DISTANCE TRAVELED BY THE WRIST, GRIPPER TIP, AND ELBOW FOR THE TRAJECTORIES COMPUTED BY EACH PLANNER ON 524 PLANNING SCENARIOS IN SIMULATION. THE SAME SMOOTHING WAS PERFOMED ON THE TRAJECTORIES FROM ALL PLANNERS. RESULTS FOR ALL SAMPLING-BASED (OMPL) PLANNERS ARE AVERAGED OVER 10 PLANNING TRIALS ON EACH SCENARIO.

Tracking Step	% of successful tracking phases	Avg. Time (s)
1. Interpolation	49.55%	0.001
2. 4D Orientation Planning	44.59%	0.244
3. HD Tracking	5.86%	1.431

TABLE III

PERFORMANCE AND SUCCESS RATE OF EACH OF THE THREE STEPS OF THE TRACKING PHASE OF THE 7D/4D ADAPTIVE PLANNER. OVER 94% OF SUCCESSFUL TRACKING PHASES ARE COMPLETED BY THE MUCH FASTER INTERPOLATION OR 4D ORIENTATION PLANNING STEPS. THE MORE COMPUTATIONALLY EXPENSIVE HD TRACKING IS PERFORMED IN LESS THAN 6% OF THE TRACKING PHASES.

planners. However, the achieved average planning time is still quite satisfactory. The 7D ARA* planner demonstrated the worst performance with highest average planning time and only solving just over half of the scenarios. The 4D ARA* planner was able to achieve planning times similar to the OMPL planners, however as it considers the end-effector orientation only in a small region around the goal, it is unable to solve planning problems that require the end-effector orientation to change far from the goal position, which explains the relatively low success rate, especially in cluttered environments. The 7D/4D adaptive planner had the highest success rate and was able to solve some of the toughest scenarios within the allowed time limit.

More specifically, the sampling-based methods performed best on the more open scenarios with fewer obstacles, where a feasible path was easy to identify with only a few samples. Our approach was also able to solve such problems quickly, however, the planning times were 2-4 times slower (but still within 1.5 seconds). The benefits of our algorithm were most obvious on the more cluttered scenarios, some of which exhibited narrow solution spaces, which were challenging for the sampling-based methods. The performance of our approach does not suffer in such scenarios and it was able to solve those scenarios quickly. The scenarios that our approach exhibited its worst performance were situations for which the 3D Dijkstra heuristic for the end-effector was leading the search in an unfeasible direction or it exhibited pronounced local minima. This occurred most often on the environments with random cuboid obstacles (Fig. 4 bottom right). For some scenarios the heuristic was "pulling" the end-effector to the far side of a cuboid obstacle, similar to the example shown in Fig. 3. Significantly larger number of state expansions were necessary to overcome the heuristic local minimum leading to higher planning times.

Table II illustrates the average quality of the paths generated by each of the 6 planners. For each computed trajectory, we kept track of the distance traveled by three key points on the arm—the wrist, the elbow, and the gripper tip. As seen in the table, the 7D/4D planner produced the shortest trajectories on average. The OMPL planners had significantly higher distances traveled by the elbow, even after trajectory smoothing. This suggests that many of the trajectories computed by the OMPL planners had unnecessary elbow motions.

Table III shows the benefits of the main contribution of this work. It illustrates the number of successful tracking phases completed by each of the three tracking steps as a percentage of all tracking phases performed, and the average time of each tracking step. Nearly half of all tracking phases were solved by simple interpolation, which took a negligible amount of time. The 4-DoF end-effector orientation tracking was successful in nearly 45% of the tracking phases performed, and was much quicker than performing high-dimensional tracking. The more computationally expensive HD tracking had to be performed only in less than 6% of all tracking phases. Thus, the two proposed novel ways of performing tracking in the context of planning with adaptive dimensionality for robotic arms with independent wrist joints prove to be very effective and significantly improve the performance of the algorithm.

VII. Conclusion

In this work we presented an extension of our algorithm for planning with adaptive dimensionality for robotic manipulators with independent wrist joints. Our results show that exploiting the fact that the manipulator's wrist can be controlled independently from the rest of the arm significantly improves the performance of the tracking phase of the algorithm. Although our planner does not match the efficiency of sampling-based methods, our experimental results, coupled with strong guarantees on completeness and bounds on solution cost sub-optimality, show that it provides a good compromise between planning time, planning success rate, and solution quality.

References

[1] K. Gochev, B. Cohen, J. Butzke, A. Safonova, and M. Likhachev, "Path planning with adaptive dimensionality," in *Proceedings of the Symposium on Combinatorial Search (SoCS)*, 2011.

[2] K. Gochev, A. Safonova, and M. Likhachev, "Planning with adaptive dimensionality for mobile manipulation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[3] L. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000, pp. 995–1001.

[5] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, IEEE Press. IEEE Press, April 2000, pp. 521–528.

[6] I. A. Sucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic Foundation of Robotics VIII (Proceedings of Workshop on the Algorithmic Foundations of Robotics)*, vol. 57, 2009, pp. 449–464. [Online]. Available: http://www.springerlink.com/content/gm47pt40p0740125/

[7] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[8] N. Ratliff, M. Zucker, J. A. D. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.

[9] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *International Conference on Robotics and Automation*, 2011.

[10] S. Thrun *et al.*, "Map learning and high-speed navigation in RHINO," in *AI-based Mobile Robots: Case Studies of Successful Robot Systems*, D. Kortenkamp, R. Bonasso, and R. Murphy, Eds. Cambridge, MA: MIT Press, 1998.

[11] R. Philippsen and R. Siegwart, "Smooth and efficient obstacle avoidance for a tour guide robot," in *ICRA*, 2003, pp. 446–451.

[12] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999, pp. 341–346.

[13] A. Botea, M. Müller, and J. Schaeffer, "Near Optimal Hierarchical Path-Finding," *Journal of Game Development*, vol. 1, no. 1, pp. 7–28, 2004.

[14] V. Bulitko, N. Sturtevant, J. Lu, and T. Yau, "Graph abstraction in real-time heuristic search," *Journal of Artificial Intelligence Research (JAIR)*, vol. 30, pp. 51–100, 2007.

[15] R. Knepper and A. Kelly, "High performance state lattice planning using heuristic look-up tables," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 3375–3380.

[16] B. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2902–2908.

[17] K. Gochev, A. Safonova, and M. Likhachev, "Incremental planning with adaptive dimensionality," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2013.

[18] B. J. Cohen, G. Subramania, S. Chitta, and M. Likhachev, "Planning for manipulation with adaptive motion primitives." in *ICRA*. IEEE, 2011, pp. 5478–5485.

[19] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.