Hierarchical Optimization-based Planning for High-DOF Robots

Chonhyon Park and Jia Pan and Dinesh Manocha

Abstract— **We present an optimization-based motion planning algorithm for high degree-of-freedom (DOF) robots in dynamic environments. Our formulation decomposes the high-dimensional motion planning problem into a sequence of low-dimensional sub-problems. We compute collision-free and smooth paths using optimization-based planning and trajectory perturbation for each sub-problem. The high-DOF robot is treated as a tightly coupled system, and we use constrained coordination in an incremental manner to plan its motion. We highlight the performance on robots with tens of DOFs and demonstrate performance benefits over prior methods.**

I. Introduction

Motion planning algorithms are frequently used in robotics, CAD/CAM, and bio-informatics. A key challenge in many applications is to develop fast (and almost real-time) techniques for motion planning. In this paper, we mainly deal with the problem of motion planning for high degrees-of-freedom (DOFs) robots, which include articulated robots with tens of joints. Some examples of such robots include humanoid robots (e.g., HRP-4[1] robot with 34 DOFs, Hubo II[2] with 40 DOFs) and mobile manipulators (such as Willow Garage's PR2 robot[3] with 20 DOFs), etc.

Over the last decades, many efficient techniques have been proposed that can compute collision-free paths in open spaces. These include sample-based planners, such as probabilistic roadmaps (PRM) [1] and rapidly-exploring random trees (RRT) [2]. However, it is relatively hard to give quality guarantees on the trajectories computed by sample-based planners, including torque or energy minimization, constraint handling, and generating smooth paths. For example, it is important to compute paths for mobile manipulators that can minimize energy consumption. Moreover, non-smooth and jerky paths can cause actuator damages.

There is considerable literature in robotics and motion planning on computing collision-free and smooth trajectories. Optimization-based approaches tend to pose the motion planning problem in a continuous setting and use optimization techniques to compute the trajectory [3], [4], [5], [6]. Such algorithms make it easier to generate motion trajectories that can satisfy different constraints, including collision avoidance, smoothness, and dynamics constraints, simultaneously; it achieves this by posing the constraints as appropriate cost functions. Furthermore, these approaches can be combined with replanning methods and used to compute smooth trajectories amongst dynamic obstacles [5]. However, prior applications of optimization-based motion planning algorithms have been limited to low-DOF robots. The convergence rate of the underlying numerical optimization techniques tends to decrease as the number of DOFs increases.

In this paper, we present a hierarchical optimization-based planner to compute smooth and collision-free trajectories for high-DOF robots. The underlying planner minimizes the trajectory cost function, which is formulated by the problem's constraints (collisions with the environment, the smoothness of the motion, etc.). Our formulation is based on the assumption that the optimal path lies in a lower-dimensional subspace, though the robot itself corresponds to a tightly coupled high-DOF system [7]. In particular, our approach decomposes a high-DOF robot into a hierarchal tree structure, where each node represents one component of the robot (i.e., a set of joints and the related links). Based on the decomposition, we incrementally compute the trajectory corresponding to each of the nodes that represents a sub-tree of the hierarchy, while the smoothness and collision-free constraints are taken into account in the cost function. In order to satisfy the joint constraints, the trajectories for all the components planned during the prior state are treated as constraints in the optimization formulation for the planning of the subsequent components. We highlight the performance of this model on robots with 20-40 DOFs, where we observe 1.6X to 14X speedups over prior methods.

The rest of the paper is organized as follows. In Section II, we survey related work in motion planning. We give an overview of optimization-based planning and our hierarchical representation in Section III. We present our algorithm for high-DOF robots in Section IV and highlight its performance in both static and dynamic environments in Section V.

II. Related Work

In this section, we give a brief overview of prior work on optimization-based motion planning and hierarchical motion planning.

A. Optimization-based Motion Planning

One of the widely used methods for path optimization is the 'shortcut' heuristic. Typically used as a postprocessing step, it selects adjacent pairs of configurations along a collision-free path and uses a local planner to compute a smoother path between those configurations [8], [9]. Another approach uses Voronoi diagrams to compute collision-free paths [10], and many techniques have been proposed based

Chonhyon Park, Jia Pan and Dinesh Manocha are with the Department of Computer Science, University of North Carolina at Chapel Hill. E-mail: {chpark, panj, dm}@cs.unc.edu.

[1]http://global.kawada.jp/mechatronics/hrp4.html

[2]http://hubolab.kaist.ac.kr/

[3]http://www.willowgarage.com/pages/pr2/overview

on numerical optimization. [11] pioneered the use of potential field for real-time obstacle avoidance [12], [13] and extended these approaches by modeling the trajectory as elastic bands or elastic strips and using gradient-based methods to compute minimum-energy paths. All these methods require a collision-free path as an initial trajectory approximation for the optimization algorithm. Some recent approaches, such as [3], [4], [14] and [5], directly encode the collision-free constraints and use a numerical solver to compute a trajectory that satisfies all the constraints. Some of these techniques explicitly compute the gradient [3], [14] while others do not [4], [5].

B. Hierarchical Motion Planning

The hierarchical mechanism decomposes a higher-dimensional planning problem into several lower-dimensional planning problems. This is based on the divide-and-conquer paradigm and can substantially reduce the complexity of the planning problem [15]. The incompleteness of the resulting planning algorithms can be improved by greedy techniques based on back-tracing [16]. The hierarchical mechanism has been used to improve the performance for articulated robots [15] or for multi-robot systems [17]. Different coordination schemes [18] and [19] have been proposed to guarantee that decomposed components have consistent motions. Simple decomposition into the lower- and upper-body has been used to plan the motion for human-like robots [20]; a more detailed decomposition is used to accelerate whole-body planning for high-DOF robots using sampling-based planners [21], [22]. Recently, hierarchical mechanisms have also been used to accelerate Markov Decision Process [23] and task planning [24], [25].

III. Overview

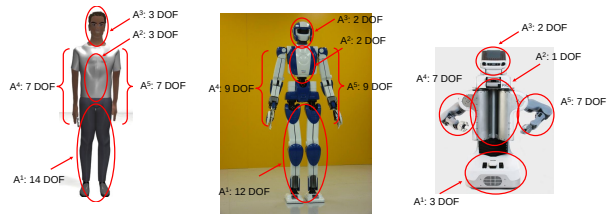
Our hierarchical planner is developed from the optimization-based planner ITOMP. In this section we first briefly describe the ITOMP algorithm and then give an overview of our hierarchical decomposition scheme.

A. ITOMP: Optimization-based Planning

ITOMP [5] is a replanning algorithm that uses optimization techniques to compute a collision-free path. In order to handle dynamic obstacles and perform real-time planning, ITOMP uses a replanning scheme. First, the trajectory of the moving obstacles is estimated over a short time horizon using simple estimation techniques. Next, a conservative bound on the position of the moving obstacles is computed over a local time interval. Finally, a trajectory that connects the initial and goal configurations of the robot is calculated by solving an optimization problem that avoids collisions with the obstacles while satisfying smoothness and torque constraints. As previous methods [3], [4] do, ITOMP assumes that the solution trajectory is discretized into N waypoints equally spaced in time. Each waypoint is denoted as a configuration $\mathbf{q}_i$, $i = 1, ..., N$. The overall optimization problem is formalized as follows:

$$\min_{\mathbf{q}_1,...,\mathbf{q}_N} \sum_{k=1}^{N} (c_s(\mathbf{q}_k) + c_d(\mathbf{q}_k) + c_o(\mathbf{q}_k)) + \frac{1}{2}\|\mathbf{AQ}\|^2, \quad (1)$$

where the three cost terms $c_s(\cdot)$, $c_d(\cdot)$, and $c_o(\cdot)$ represent the static obstacle cost, the dynamic obstacle cost, and the problem-specific additional constraints, respectively. $\|\mathbf{AQ}\|^2$ represents the smoothness cost, which is computed by the sum of squared accelerations along the trajectory using the same matrix $\mathbf{A}$ proposed by [3] and $\mathbf{Q} = [\mathbf{q}_1^T, ..., \mathbf{q}_N^T]^T$. The solution to the optimization problem in (1) corresponds to a local or global optimal trajectory of the robot.



(a) A human-like model (34 DOFs) (b) HRP-4 (34 DOFs) (c) PR-2 (20 DOFs)

Fig. 1: An example of hierarchical decomposition for various robots. These hierarchical decompositions are used to divide a high-dimensional problem into a sequence of low-dimensional problems.

B. Assumptions and Notations

A configuration of a robot $\mathbf{q}$ is determined by all the actuated joints of the robot, as well as the position and orientation of the robot in the coordinate frame. The high-DOF robot is hierarchically decomposed into n different components $\{A^1, A^2, ..., A^n\}$. Accordingly, the configuration $\mathbf{q}$ can also be represented as the concatenation of the configuration $\mathbf{q}^i$ for each body component: i.e., $\mathbf{q} = [(\mathbf{q}^1)^T, (\mathbf{q}^2)^T, ..., (\mathbf{q}^n)^T]$, where $\mathbf{q}^i$ corresponds to the configuration of A^i. Moreover, $\mathbf{q}^i$ is determined by all A^i's actuated joints, including the joint through which A^i is connected to its parent component. For the root component A^1, additional unactuated DOFs can be added to the system to specify the position and orientation of the coordinate frame associated with the system. We denote the trajectory for a robot as $M(t)$, which is a discretized trajectory composed of N waypoints in the configuration space: $M(t) = \{\mathbf{q}_1, ..., \mathbf{q}_N\}$. The trajectory for each component A^i is represented as $M^i(t)$, which also contains N waypoints, i.e., $M^i(t) = \{\mathbf{q}_1^i, ..., \mathbf{q}_N^i\}$. We use $\bar{\mathbf{q}}_k^i$ to denote the k-th waypoint corresponding to component A^i and all its previous components, i.e., $\bar{\mathbf{q}}_k^i = [(\mathbf{q}_k^1)^T, ..., (\mathbf{q}_k^{i-1})^T, (\mathbf{q}_k^i)^T]^T$. Similarly, $\bar{M}^i(t)$ corresponds to the trajectory of $\bar{\mathbf{q}}^i$.

C. Hierarchial Optimization-based Planning

Fig. 1 shows a natural decomposition scheme for different robots that is based on the robots' functional components. The high-DOF system is divided into several parts: a lower body (including legs and pelvis for human-like model, or a 3-DOF base for the PR2 robot), a torso, a head, a left arm and a right arm. The components are ordered according to

the hierarchy, taking into account the physical volume of each component.

We can incrementally plan the trajectory of a high-DOF robot based on this decomposition. First, we compute a trajectory $M^1(t)$ for the root component A^1. Then we fix the trajectory for A^1 and compute a trajectory for its child component A^2 by considering A^1 as a moving obstacle in the optimization formulation for A^2. However, there may be no feasible trajectory for A^2 because A^1 blocks it as an obstacle. In such cases, we first slightly modify the trajectory of A^1 based on workspace heuristics and search whether it is possible to compute a collision-free trajectory for A^2. If such local trajectory refinement does not result in a feasible solution, we perform back-tracing; we merge A^1 and A^2 into a larger component $A^{1,2}$ and then try to compute a collision-free path for this larger component using optimization-based planning. After the trajectory for A^2 is computed, we extend the approach in an incremental manner to compute a collision-free path for A^3, now treating A^1 and A^2 as moving obstacles. This process is repeated for all n components, and a trajectory for the overall robot is computed.

For ITOMP, the hierarchical planning is implemented by decomposing (1) into n optimization problems, one for each component A^j:

$$\min_{\mathbf{q}_1^j,\ldots,\mathbf{q}_N^j} \sum_{k=1}^{N} (c_s(\bar{\mathbf{q}}_k^j) + c_d(\bar{\mathbf{q}}_k^j) + c_o(\bar{\mathbf{q}}_k^j)) + \frac{1}{2}\|\bar{\mathbf{A}}^j \bar{\mathbf{Q}}^j\|^2, \quad (2)$$

where we compute the optimal waypoints $\mathbf{q}_k^j$ for components A^j while fixing the waypoints $\mathbf{q}_k^i$ for all the previous components $A^{1\leq i\leq j-1}$. $\bar{\mathbf{A}}^j$ is the smoothness matrix; it is similar to $\mathbf{A}$ in Equation (1), but it is resized to the length of $\bar{\mathbf{q}}_k^i$. $\bar{\mathbf{Q}}^j$ is defined as $\bar{\mathbf{Q}}^j = [(\bar{\mathbf{q}}_1^j)^T, ..., (\bar{\mathbf{q}}_N^j)^T]^T$.

IV. Algorithm

In this section, we present our hierarchical optimization-based planning algorithm. We first explain how the hierarchical decomposition is integrated into the replanning framework. Next we present our trajectory optimization and local refinement methods, which use the incremental coordination algorithm. A detailed theoretical analysis of our planning algorithm can be found in [26].

A. Replanning with Multi-stage Planning

Our algorithm traverses the entire hierarchy of the robot $\{A^1, A^2, ..., A^n\}$ sequentially in a breadth-first order using n planning stages. Stage i computes the trajectory for A^i and improves the trajectories of $\{A^1, ..., A^{i-1}\}$, which were computed in prior stages.

The overall planning algorithm is shown in Fig. 2. During planning stage i, the planning algorithm computes $\{M_{i-1}^1(t), M_{i-1}^2(t), ...M_{i-1}^{i-1}(t)\}$, which correspond to the trajectories for $\{A^1, A^2, ..., A^{i-1}\}$ computed during the previous stage $i-1$. We use $M_j^i(t)$ to denote the improved trajectory, $M^i(t)$, after stage j. According to our notation, we denote $\{M_j^1(t), M_j^2(t), ...M_j^i(t)\}$ as $\bar{M}_j^i(t)$. With the input

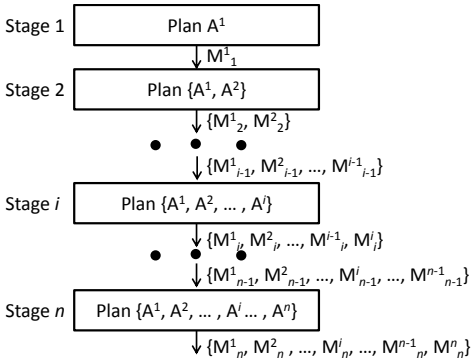
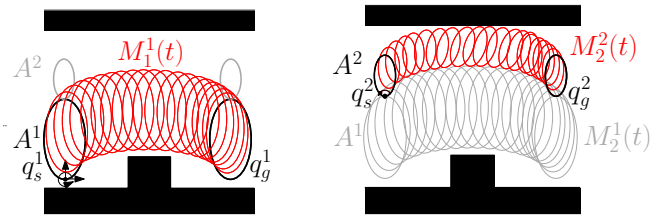


Fig. 2: The multi-stage planning of a hierarchical model. The robot is represented using a hierarchy of components $\{A^1, A^2, ..., A^n\}$ as shown in Fig. 1. In stage 1, the algorithm computes trajectory M_1^1 for component A^1. In the next stage, the algorithm improves the trajectory M_1^1 into M_2^1 while computing M_2^2 for A^2. After n stages, the planner computes the trajectory for the entire component, $\{M_n^1, M_n^2, ..., M_n^n\}$.



(a) Planning of trajectory $M_1^1(t)$ for A^1 in stage 1.

(b) Planning of trajectory $M_2^2(t)$ for A^2 in stage 2.

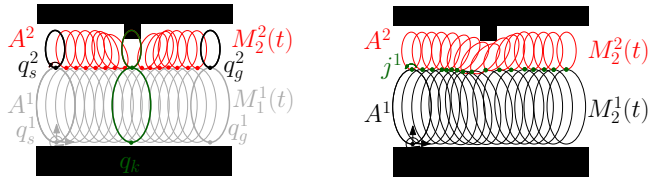
Fig. 3: Incremental trajectory planning. The robot model consists of $\{A^1$ (3 DOFs), A^2 (1 DOF)$\}$. (a) During stage 1, the algorithm computes trajectory $M_1^1(t)$ for A^1 while avoiding collisions between A^1 and the obstacle shown in the black region. (b) During stage 2 of the planning algorithm, the trajectory $M_2^2(t)$ for A^2 is computed while A^1 is assumed to move along the trajectory $M_1^1(t)$.

$\bar{M}_{i-1}^{i-1}(t)$, planning stage i computes $\bar{M}_i^i(t)$. The algorithm searches the configuration space of A^i to compute the trajectory using numeric optimization. During the computation of $M_i^i(t)$, the algorithm may perform local refinement of $\bar{M}_{i-1}^{i-1}(t)$ to compute $\bar{M}_i^{i-1}(t)$.

B. Trajectory Optimization using Constrained Coordination

In order to perform optimization, the problem is formalized using (1). Similarly to the previous work [3], [4], [5], we model the cost of static obstacles using a signed Euclidean Distance Transform (EDT) and model the cost of dynamic obstacles using geometric collision checking.

We use the incremental coordination approach [17], [21] in our planning algorithm. This approach computes the trajectory for a new component while using prior trajectories as constraints. During each planning stage, the algorithm computes the trajectory for a subset of the robot components in order to compute the whole-body motion trajectory incrementally. In Fig. 3, the 2D robot has two components, A^1 and A^2. Each component has only one link. A^1 has 3 DOFs corresponding to the position and orientation of the robot in 2D space. A^2 has 1 DOF corresponding to the angle that connects A^1 and A^2. Therefore, the configuration vectors $\mathbf{q}_k^1$ and $\mathbf{q}_k^2$ have dimensions 3 and 1, respectively. The trajectory



(a) There is no collision-free configuration $\mathbf{q}_k^2$ for A^2 at time k if A^1 moves along $M_1^1(t)$.

(b) With local refinement, the planner finds feasible solutions $M_2^1(t)$ and $M_2^1(t)$.

Fig. 4: Planning with local refinement. By adjusting the configuration of the joint j^1 connecting A^1 and A^2, we can move A^1 away from the obstacle and leave more space for A^2 to pass through. As a result, the planner can compute a collision-free solution $\{M_2^1(t), M_2^2(t)\}$.

$M(t)$ is a sequence of N configurations at discretized time steps. During planning stage 1, the algorithm computes the trajectory M_1^1 for A^1, which connects the start configuration $\mathbf{q}_s^1$ of A^1 with its goal configuration $\mathbf{q}_g^1$. During planning stage 2, the trajectory $M_2^2(t)$ for A^2 is computed, while A^1 is assumed to move along the trajectory $M_1^1(t)$.

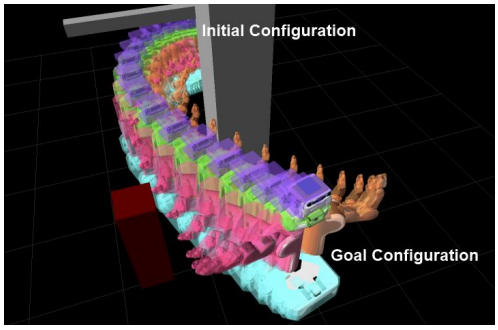
During a planning stage, our algorithm computes a minimum-cost trajectory using (2). With this equation, the planning stage j can compute a trajectory $M_j^j(t)$ for A^j which is smooth and does not collide with obstacles. The cost term $\frac{1}{2}\|\bar{\mathbf{A}}^j\bar{\mathbf{Q}}^j\|^2$ computes the smoothness cost. For the collision cost $c_s(\bar{\mathbf{q}}_k^j)$ and $c_d(\bar{\mathbf{q}}_k^j)$, the poses of all the links in A^j are evaluated based on forward kinematics according to $\bar{\mathbf{q}}_k^j$.

C. Local Refinement in Trajectory Optimization

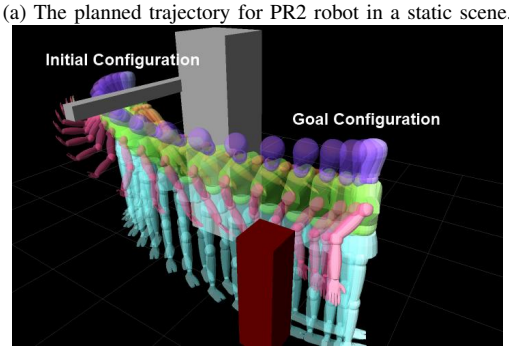
In this section we present the local refinement scheme used as part of trajectory optimization. It improves the trajectories of the robot components which were computed during the previous stages. In our incremental planning algorithm, the trajectory of a robot component is computed using an optimization formulation such that the trajectories of prior components are constrained to lie on the paths computed during previous stages. However, the optimization-based planner may fail to find a solution that satisfies all the constraints. Fig. 4a shows such an example for a simple 2D robot which consists of two components A^1 and A^2. The trajectory $M_1^1(t)$ for A^1, which is computed during planning stage 1, is collision-free. However, when computing a solution for A^2, A^1 is constrained to move along $M_1^1(t)$. This may result in no feasible solution for A^2 which avoids collisions with the environment. The back-tracing approach, which replans the trajectory with merged component $A^{1,2}$, can find a solution in such case, but it is expensive for higher-dimensional problems. As shown in Fig. 4b, we refine the trajectory $M_1^1(t)$ by adjusting the configuration of the joint connecting A^1 and A^2, then move A^1 away from the obstacles by a displacement $\mathbf{r}$.

V. Results

In this section, we highlight the performance of our hierarchical planning algorithm in static and dynamic environments. We have implemented our algorithm in a simulator



(a) The planned trajectory for PR2 robot in a static scene.



(b) The planned trajectory for a human-like model.

Fig. 5: Hierarchical planning of robots in a static environment. The planned trajectory for different components is marked using different colors.

with both a human-like robot model and Willow Garage's PR2 robot model. We decompose each model into five components each (shown in Fig. 1). For the PR2 robot, we compute the trajectory for all the 20 DOFs, which are shown in Fig. 1c. The human-like model has 34 DOFs, which are shown in Fig. 1a. In practice, human-like robots move from their base position using walking, which can be efficiently computed using motion generators [27], [28] rather than pure motion planning algorithms. Therefore, we compute the 3 DOFs trajectory of the lower body component A^1 using our motion planning algorithm; after that we use the motion generator to generate the walking motion, which follows the computed 3 DOFs trajectory of A^1. This reduces the DOFs for motion planning computations from 34 to 23.

We highlight all the results of motion planning in different environments in Table I. We compute the motions of PR2 and the human-like robot in two static environments and another environment with dynamic obstacles. We compute the motion trajectory using our hierarchical planning algorithm, and compare its performance with the motion trajectory computed using ITOMP. We measure the number of iterations in the optimization routines and the planning time to find the first collision-free solution. We also evaluate the quality of the computed trajectory by evaluating the cost functions and success rate of the optimization-based planner. The results are shown in Table I and correspond to the averages and standard deviations of 100 trials for each scenario. In most cases, hierarchical planning outperforms ITOMP.

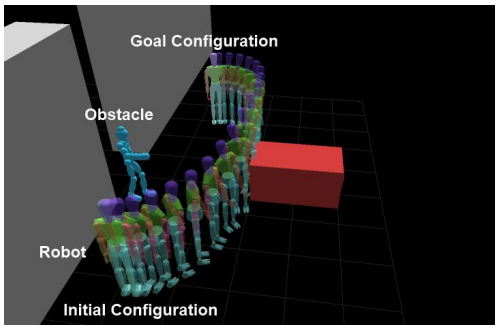
Fig. 5 shows our first benchmark in a static environment. The environment has several static obstacles, which prevent

		ITOMP				Hierarchical Planning			
		Iterations	Planning Time(s)	Cost	Success Rate	Iterations	Planning Time(s)	Cost	Success Rate
		Mean (Std. Dev.)				Mean (Std. Dev.)			
Human -like Robot	Static Environment 1	418.25 (344.90)	20.93 (16.24)	0.032 (0.011)	1.00	84.74 (18.00)	2.81 (0.50)	0.036 (0.000)	1.00
	Static Environment 2	461.26 (539.66)	30.78 (35.63)	0.017 (0.000)	1.00	54.02 (15.62)	2.21 (0.53)	0.025 (0.000)	1.00
	Dynamic Environment	13.99 (2.30)	1.71 (0.17)	0.058 (0.000)	0.89	18.89 (3.35)	1.33 (0.12)	0.101 (0.000)	0.95
PR2	Static Environment 1	102.06 (33.11)	8.20 (2.35)	0.033 (0.000)	1.00	90.75 (22.53)	5.11 (1.09)	0.032 (0.000)	1.00
	Static Environment 2	167.26 (239.65)	16.00 (22.42)	0.033 (0.000)	1.00	104.13 (73.08)	6.09 (4.11)	0.032 (0.000)	1.00
	Dynamic Environment	8.81 (3.90)	1.54 (0.42)	0.051 (0.000)	0.96	16.51 (12.12)	1.66 (0.66)	0.051 (0.004)	0.99

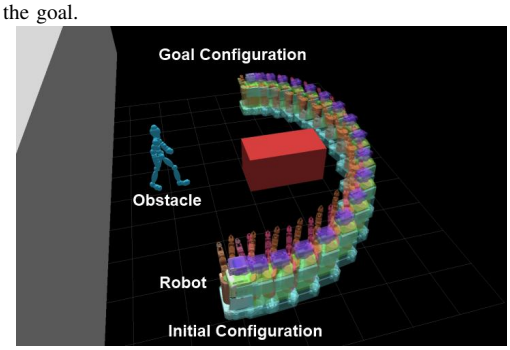
TABLE I: The performance of the hierarchical planning algorithm is compared to the ITOMP algorithm. We compute collision-free trajectories in static and dynamic environments. We measure the number of iterations in numerical optimization, planning time to find the first collision-free solution, trajectory cost, and the success rate of our planner. In the static scenes hierarchical planning results in up to 14X speedup over ITOMP algorithm. The trajectory costs for both algorithms are small (less than 0.1), which means the quality of the solution with the hierarchical planner is close to the trajectory computed by ITOMP.

	Stage	Iterations	Planning Time	Cost	Back -tracing
PR2	A^1 (3 DOFs)	63.28	3.13	0.019	0/100
	A^2 (1 DOFs)	1.00	0.12	0.000	0/100
	A^3 (2 DOFs)	1.00	0.12	0.000	0/100
	A^4 (7 DOFs)	9.26	0.60	0.008	0/100
	A^5 (7 DOFs)	36.17	1.96	0.005	1/100
	Overall Planning	90.75	5.11	0.032	1/100
Human -like Robot	A^1 (3 DOFs)	7.33	0.25	0.009	0/100
	A^2 (3 DOFs)	15.18	0.53	0.009	0/100
	A^3 (3 DOFs)	24.10	0.65	0.000	0/100
	A^4 (7 DOFs)	18.81	0.69	0.012	0/100
	A^5 (7 DOFs)	19.32	0.69	0.005	0/100
	Overall Planning	84.74	2.81	0.036	0/100

TABLE II: The planning results obtained from a static environment. We show the number of iterations, planning time to find the first collision-free solution, trajectory costs, and the number of back-tracings of each stage of our hierarchical planning algorithm.



(a) Due to the human-like obstacle, the robot cannot compute a collision-free path to the goal.



(b) The robot takes a long way around to avoid potential collision with the human-like obstacle.

Fig. 6: Planning in a dynamic environment. We use a human-like obstacle which follows a path generated from motion-capture data.

the initial trajectory from being collision-free. Using hierarchical planning, we incrementally compute the trajectory of the robot from components A^1 to A^5. In Table II, we show the timings and the trajectory costs of each stage of the hierarchical planning. PR2's base has a large physical volume, which makes it hard to find a collision-free solution for this environment. As a result, most of the planning time in this scenario is spent in the stage corresponding to A^1. The computed trajectories of A^1 have no collisions with A^2 and A^3 and therefore require only a single iteration of the optimization algorithm for all trials. In the two other stages, which compute trajectories for the arm components, A^3 and A^4, the planner run tens of iterations to compute an improved trajectory to ensure that A^3 and A^4 have no collisions. In the decomposition of the human-like model, each of the stages takes similar time and none of them dominates the overall computation. This demonstrates that the decomposition used for this model divides the high-DOF planning problem into almost equal-sized low-dimensional sub-problems, which results in an overall performance improvement as compared to high-DOF planning. We observe that the speedup due to hierarchical planning is about 7X for the human-like model and about 1.6X for the PR2 model. In both cases, the trajectory cost corresponding to the optimization function with our hierarchical algorithm is close to the trajectory cost calculated by ITOMP. This implies that the trajectories computed by both these algorithms are quite close. Our second benchmark has a more challenging environment. In this environment, the hierarchical planning observes 14X speedup for the human-like model, and 2.6X for the PR2 model.

We also evaluated the performance of our algorithm in a dynamic scene (Fig. 6). We use a human-like obstacle

which follows a path from motion-captured data. The path of the obstacles is designed to interrupt the path of the robot during execution. We set the replanning time step interval as 3 second; the planner fails if it cannot find a collision-free trajectory within that time interval. In such dynamic scenes, the planner tends to improve trajectory computation during a given time step, but not for the overall duration. As a result, it is more important to measure the success rate of each planner instead of the overall planning time or the number of iterations. With the same replanning time step interval, our hierarchical planner has a higher success rate in dynamic environments than ITOMP.

VI. Conclusions and Future Work

We present an optimization-based motion planning algorithm for high DOF robots. Our algorithm decomposes the high-dimensional motion planning problem into a sequence of low-dimensional sub-problems and computes the solution for each sub-problem in an incremental manner. We use constrained coordination and local refinement to incrementally compute the motion. We highlight the performance on 20 DOF PR-2 robot and a human-like robot with 34 DOFs, along with use of walking generator. In static environments, our algorithm observes up to 14X speedup while still generating smooth trajectories. In dynamic environments, we show that the algorithm can increase the success rate of the planning.

There are many avenues for future work. It would be interesting to extend our approach to compute whole-body trajectory computation for high DOF human-like models. We would also like to evaluate its performance in more dynamic environments and ensure that the resulting motion of human-like robots is dynamically stable.

VII. Acknowledgments

This research is supported in part by ARO Contract W911NF-10-1-0506, NSF awards 0917040, 0904990, 1000579 and 1117127, and Willow Garage.

References

[1] L. Kavraki, P. Svestka, J. C. Latombe, and M. 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.

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

[3] N. Ratliff, M. Zucker, J. A. D. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proceedings of International Conference on Robotics and Automation*, 2009, pp. 489–494.

[4] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[5] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proceedings of International Conference on Automated Planning and Scheduling*, 2012.

[6] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *International Journal of Robotics Research*, 2012.

[7] P. Vernaza and D. D. Lee, "Learning dimensional descent for optimal motion planning in high-dimensional spaces," in *Proceedings of AAAI Conference on Artificial Intelligence*, 2011, pp. 1126–1132.

[8] P. Chen and Y. Hwang, "Sandros: a dynamic graph search algorithm for motion planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 390–403, jun 1998.

[9] J. Pan, L. Zhang, and D. Manocha, "Collision-free and curvature-continuous path smoothing in cluttered environments," in *Proceedings of Robotics: Science and Systems*, 2011.

[10] M. Garber and M. C. Lin, "Constraint-based motion planning using voronoi diagrams," in *Algorithmic Foundations of Robotics V*, ser. Springer Tracts in Advanced Robotics, 2004, vol. 7, pp. 541–558.

[11] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal on Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[12] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *International Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[13] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1993, pp. 802–807 vol.2.

[14] A. Dragan, N. Ratliff, and S. Srinivasa, "Manipulation planning with goal sets using constrained trajectory optimization," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2011, pp. 4582–4588.

[15] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: a framework for real-time motion planning in high-dimensional configuration spaces," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2001, pp. 1469–1474.

[16] R. Alami, F. Robert, F. Ingrand, and S. Suzuki, "Multi-robot cooperation through incremental plan-merging," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3. IEEE, 1995, pp. 2573–2579.

[17] P. Isto and M. Saha, "A slicing connection strategy for constructing PRMs in high-dimensional C-spaces," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2006, pp. 1249–1254.

[18] M. Erdmann and T. Lozano-Pérez, "On multiple moving objects," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1986, pp. 1419–1424.

[19] M. Saha and P. Isto, "Multi-robot motion planning by incremental coordination," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 5960–5963.

[20] G. Arechavaleta, C. Esteves, and J.-P. Laumond, "Planning fine motions for a digital factotum," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2004, pp. 822–827.

[21] L. Zhang, J. Pan, and D. Manocha, "Motion planning of human-like robots using constrained coordination," in *IEEE-RAS International Conference on Humanoid Robots*, 2009, pp. 188–195.

[22] J. Pan, L. Zhang, M. C. Lin, and D. Manocha, "A hybrid approach for simulating human motion in constrained environments," *Computer Animation and Virual Worlds*, vol. 21, pp. 137–149, 2010.

[23] J. L. Barry, L. P. Kaelbling, and T. Lozano-Pérez, "DetH*: Approximate hierarchical solution of large markov decision processes," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2011, pp. 1928–1935.

[24] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2011, pp. 1470–1477.

[25] ——, "Pre-image backchaining in belief space for mobile manipulation," in *Proceedings of International Symposium on Robotics Research*, 2011.

[26] C. Park, J. Pan, and D. Manocha, "Analysis of hierarchical optimization-based planning," Department of Computer Science, University of North Carolina at Chapel Hill, Tech. Rep., 2013.

[27] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie, "Planning walking patterns for a biped robot," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 3, pp. 280–289, 2001.

[28] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1620–1626.