Abstract—We present a novel optimization-based motion planning algorithm for high degree-of-freedom (DOF) robots. Our approach combines a multi-contact dynamics formulation with optimization-based planning to compute collision-free, smooth and dynamically stable trajectories. Our formulation is general and can handle multiple simultaneous contacts and non-planar surfaces. We highlight the performance of our algorithm in simulated environments on human-like robots with tens of DOFs.

I. INTRODUCTION

Over the last few years, robots with complex shapes and a high number of controllable joints have been increasingly used for various applications. These include highly articulated bipedal humanoid robots (e.g., HRP-4\textsuperscript{1} robot with 34 DOFs, and Hubo II\textsuperscript{2}, with 40 DOFs). This increased complexity of the robots results in two major challenges for motion planning: (1) the high number of degrees-of-freedom (DOFs) increases the dimensionality of both the configuration and the search spaces, thereby increasing the cost of path computation; and (2) only a subset of possible motion are dynamically stable due to the robot’s kinematics and shape. As a result, it is a major challenge to efficiently compute a collision-free trajectory for the robot that can satisfy all stability and smoothness constraints.

There is considerable work on motion planning for high-DOF robots. At a broad level, the previous work can be classified into sample-based planners and optimization-based planners. Sample-based planners, which are based on probabilistic roadmaps [1] or rapidly-exploring random trees [2], are relatively easy to implement and can compute collision-free paths in high dimensions. Recent progress on sample-based planner makes it possible to handle some constraints such as path smoothness and dynamic stability [3], [4], [5]. However, it is relatively difficult to guarantee the quality of the trajectories computed by sample-based planners. On the other hand, optimization-based planners compute a trajectory using a continuous formulation of the problem [6], [7], [8], [9]. Different constraints can be formulated as part of the optimization function for trajectory computation, including path-smoothness constraints and collision checking with static and dynamic obstacles.

One key task for motion planning is to ensure that the robot’s motion resulting from the planned trajectory satisfies the stability constraints. For example, the computed posture should be statically stable, and the projection of the center of mass of the robot should lie inside the foot-supported polygon; similarly, the zero moment point (ZMP) should lie inside the support polygon [10] for dynamic stability on a flat plane. Some of the current algorithms plan a trajectory that satisfies the ZMP constraint and derive the appropriate hip or torso motion to compute a trajectory. However, the resulting hip or torso motion is often jerky because it has to compensate for the lower-limb motions [11]. In the general case, a robot is dynamically stable when the forces and torques acting on the robot maintain an equilibrium; Newton-Euler equations can be used to compute those forces and torques [12]. Since the contacts between the robot and the obstacles exert forces on the robot, we need to compute the appropriate forces (including their duration) from the contacts as part of overall motion planning.

In this paper, we present an efficient optimization-based planning algorithm to compute dynamically stable, smooth, and collision-free robot motions for high-DOF robots. Our planner minimizes the trajectory cost function, which is composed of cost functions for dynamic stability, calculation of a collision-free path, and path smoothness. We also optimize the durations of the contacts along with the state of the robot, which allows our algorithm to compute a stable motion with multiple contacts. In order to accelerate the computation, we use a hierarchical decomposition of a high-DOF robot and compute the trajectory of low-DOF components in a bottom-up manner. Based on the decomposition, we incrementally compute the trajectory for each node in the hierarchy. Moreover, the trajectories of 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 on non-planar surfaces with multiple contacts. Moreover, we observe considerable speedups over prior optimization-based algorithms.

The rest of the paper is organized as follows. In Section II, we survey related work in motion planning and compare our approach with prior methods. We give an overview of the background algorithms in Section III. In Section IV, we present our planning algorithm, based on dynamic stability constraints. Finally, we highlight our algorithm’s performance in simulated environments in Section V.
II. RELATED WORK

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

A. Optimization-based Motion Planning

The simplest trajectory smoothing algorithms use the shortcut method to smoothen the motion trajectory. These algorithms are used as a postprocess on computed collision-free trajectories; they optimize adjacent pairs of configurations along the computed trajectory, using local planning, to compute smooth paths [13], [14].

Many techniques based on numerical optimization have been proposed in the literature [15]. Khatib proposed the use of potential fields for real-time obstacle avoidance [16]. This approach is extended using elastic strips [17] and elastic bands [18] to compute minimum-energy paths using gradient-descent methods. These methods use a collision-free path as an initial trajectory approximation for the optimization algorithm.

Some recent approaches, such as [6], [7], [19] and [8], directly encode constraints (e.g. the requirement that the path be collision-free and smooth) into the cost functions, then use a numerical solver to compute a trajectory that satisfies all the constraints; these approaches do not require the collision-free initial trajectory, as is the case with prior numerical techniques. Some of these techniques explicitly compute the gradient [6], [19], while others do not [7], [8].

B. Motion Stability Constraints

Ensuring that the computed motion is stable is an important criterion in motion planning for high-DOF robots. There is considerable work on the walking motion of bipedal robots [11]; proper, stable walking motion is essential for humanoid robots. In order to handle motion dynamics, the stability constraint is formulated to maintain the equilibrium among the forces and torques acting on the robot: inertia, Coriolis, gravity, ground-reaction forces from contacts, etc. In this section, we give an overview of the previous motion planning approaches that achieve dynamic stability in their computed motions and compare our algorithm with them.

The zero moment point (ZMP)-based methods compute ZMP in the support polygon based on the position and orientation of the robot as a function of the joints of the robot, as well as by the position and orientation of the robot in the workspace. We denote the trajectory for a robot as a function \( M(t) \) for \( t \in [0, T] \), where we assume that we know the length of the trajectory \( T \). \( M(t) \) is a discretized trajectory composed of \( N + 2 \) waypoint configurations: \( M(t) = \{q_I, q_1, ..., q_N, q_G\} \), where \( q_k \) is a trajectory waypoint at time \( \frac{k}{N+1} T \). \( q_I \) and \( q_G \) represent the given initial and goal configurations, respectively. Given \( q_I \) and \( q_G \), ITOMP generates the initial trajectory by discretizing the configurations along a curve that connect \( q_I \) and \( q_G \) into \( N + 1 \) segments equally spaced in time.

ITOMP computes a smooth trajectory \( M(t) \) that connects the initial and goal configurations of the robot by solving an optimization problem. The optimization algorithm avoids collisions with obstacles using conservative bounds while simultaneously satisfying other constraints. ITOMP optimizes the positions of internal waypoints \( \{q_1, ..., q_N\} \) by optimizing the following cost function:

\[
\min_{q_1, ..., q_N} \sum_{k=1}^{N} (C_{Obs}(q_k) + C_{Spec}(q_k) + \|q_{k-1} - 2q_k + q_{k+1}\|^2),
\]

(1)
where the cost terms \( C_{Obs}(\cdot) \) and \( C_{Spec}(\cdot) \) represent the obstacle cost and the problem-specific additional constraints, respectively. \( \sum_{k=1}^{N} \| q_{k-1} - 2q_k + q_{k+1} \|^2 \) represents the smoothness of the entire trajectory, which is computed by the sum of squared accelerations along the trajectory.

The optimization problem in (1) computes a local or global optimal trajectory of the robot configuration waypoints \( q_k \). In order to improve robot’s responsiveness and safety in dynamic environments, ITOMP uses a replanning approach to compute collision-free paths amongst dynamic obstacles. Using estimations of the trajectories of moving obstacles over a short time horizon, ITOMP computes conservative bounds on the position of the obstacles over a short time interval.

### B. Contact-Invariant Optimization

In order to compute a physically correct, stable motion, the intermittent contacts between the robot and the environment during the motion trajectory should be planned. For example, a simple walking motion for a humanoid robot requires planning both when a foot is on the ground and when it is not in contact with the ground, and this computation must be performed for each foot. Some earlier approaches [20] use pre-defined positions for footsteps to simplify the problem, but this works only in limited cases where the footsteps are uniform and symmetric.

We use the Contact-Invariant Optimization (CIO) approach. In this formulation, the robot has several potential contact points (e.g. feet or hands), that can make contacts with the obstacles in the environment. It is assumed that both the robot links and obstacles are rigid, and that each contact point has dry friction. In optimization-based planning, additional contact-related variables for the potential contact points need to be optimized along with the trajectory waypoints to determine when the corresponding contacts exist in the computed trajectory.

The CIO approach introduces contact phases. Instead of defining the contact-related variables as a trajectory with \( N \) waypoint values, we can approximate the trajectory with fewer \( P \) values, where \( P \) is the number of contact phases and \( P < N \). The trajectory of contact-related variables is defined as \( a^l = \{ a^l_1, ..., a^l_P \} \) for \( l \)-th potential contact point, and a map \( \rho(k) = p \) is used to retrieve the corresponding contact variable \( a^l_p \) for a waypoint \( q_k \). This approach assumes that the contacts are invariant in a contact phase. It reduces the number of variables, \( a^l_p \), that are used during the optimization algorithm. A large value of \( a^l_p \) implies that the contact \( l \) must be active during the phase \( p \); for a small \( a^l_p \), the contact \( l \) can be ignored.

For a waypoint \( q_k \), the CIO approach computes the stability cost by using two sub-cost functions:

\[
C_{Stability}(q_k) = C_{Physics}(q_k) + C_{Contact}(q_k).
\]

\( C_{Physics}(\cdot) \) represents the cost due to the violation of the balance, and \( C_{Contact}(\cdot) \) represents the cost of the violation of contacts. The contact invariant cost \( C_{Contact}(\cdot) \) is defined as

\[
C_{Contact}(q_k) = \sum_{l=1}^{L} \sum_{k=1}^{N} a^l_{\rho(k)} (\| \mathbf{e}^l_k(q_k) \|^2 + \| \mathbf{\dot{e}}^l_k(q_k) \|^2),
\]

where \( L \) is the total number of potential contact points, and \( \mathbf{e}^l_k \) and \( \mathbf{\dot{e}}^l_k \) are the contact-violation vector and the velocity of the \( l \)-th contact point at a waypoint \( q_k \), respectively. \( \mathbf{e}^l_k \) is a 4D vector that concatenates the 3D position and normal angle differences between the \( l \)-th contact point and the nearest point on an obstacle. Therefore, \( \mathbf{e}^l_k \) represents the misalignment between the \( l \)-th potential contact point on the robot and the nearest point on the environment in both position and orientation. If \( a^l_{\rho(k)} \) is large, the misalignment of the \( l \)-th contact makes the cost function very high, while the misalignment of small \( a^l_{\rho(k)} \) does not result in significant cost. \( \mathbf{\dot{e}}^l_k \) for a large value of \( a^l_{\rho(k)} \) corresponds to slip of the contact point.

The cost of \( C_{Contact}(q_k) \) corresponds to the global minimum when all \( a_{\rho(k)} \) are zero. However, these cases are prevented by the second cost term \( C_{Physics}(q_k) \), which represents the cost that penalizes for the violation of the equilibrium of forces and torques. If the contact variables \( a_{\rho(k)} \) have small values, it increases the cost of \( C_{Physics}(q_k) \) as described in Section IV-C.

### IV. Motion Planning with Dynamic Stability

In this section, we present the details of our approach that computes a collision-free, smooth trajectory that maintains the robot’s dynamic stability. We first present the trajectory optimization function. Next, we introduce the underlying physics-based formulation of the cost computation and describe the overall algorithm.

#### A. Optimization with Stability Cost

Based on the ITOMP cost function (1), our planning algorithm uses CIO to compute a dynamically stable trajectory for robots. Based on CIO, our new optimization formulation is:

\[
\min_{a_1, ..., a_N} \sum_{k=1}^{N} (C_{Obs}(q_k) + C_{Stability}(q_k, a_{\rho(k)})) + \| q_{k-1} - 2q_k + q_{k+1} \|^2,
\]

where \( 1 \leq \rho(k) \leq P \), and \( a_k = [a^l_1, ..., a^l_P] \), the vector of contact variables of \( L \) potential contact points for phase \( i \). In our objective function (4), \( C_{Stability}(q_k) \) is the stability cost for the waypoint \( q_k \), which is defined in Equation (2). Though \([28]\) uses a simplified physics model to make animated characters move naturally, we compute \( C_{Physics}(q_k) \) accurately based on Newton-Euler equations.

#### B. Dynamic Stability Computation

A key issue in our formulation is computation of physics-violation cost \( C_{Physics}(q_k) \) for maintaining dynamic stability (as shown in Equation (2)). We first describe our physics-based formulation. Fig. 1(a) illustrates a high-DOF humanoid robot, which makes contacts with the ground plane
using its feet. Let \( \Sigma_R \) be the global coordinate frame, \( J \) be the number of links in the robot, and \( c_1, \ldots, c_L \) be the positions of \( L \) contact points. There are several wrenches (forces and torques) exerted on the robot. The robot is dynamically stable when all wrenches on the robot constitute an equilibrium [12].

1) Contact wrench : The sum of contact wrenches \( \mathbf{w}_c \) applied to the robot from contact points \( c_l \) with respect to \( \Sigma_R \) is given by

\[
\mathbf{w}_c = \sum_{l=1}^{L} \mathbf{w}_l^c = \sum_{l=1}^{L} \left[ \mathbf{f}_l + \mathbf{r}_l \times \mathbf{f}_l \right],
\]

where \( \mathbf{f}_l \) is the contact force of \( c_l \) and \( \mathbf{r}_l \) is the position vector of \( c_l \) in the frame \( \Sigma_R \). Coulomb’s friction law stipulates that \( \mathbf{f}_l \) should be constrained in its friction cone \( F_l \) to avoid any slipping motion. This constraint can be formulated as:

\[
f_{l1}^2 + f_{l2}^2 + f_{l3}^2 \leq \mu f_{ln}^2,
\]

where \( \begin{bmatrix} f_{ln} & f_{lt} & f_{lo} \end{bmatrix}^T \) corresponds to \( \mathbf{f}_l \), with respect to the frame of \( c_l \), which is defined by the axes of the contact normal \( \mathbf{n}_l \), \( \mathbf{t}_l \) and \( \mathbf{o}_l \) that satisfy \( \mathbf{n}_l \times \mathbf{t}_l = \mathbf{o}_l \).

Our formulation of \( \mathbf{w}_c \) considers the contact normal and the friction coefficient. This makes it general enough for uneven ground surface, unlike prior approaches based on ZMP.

2) Gravity wrench : The gravity wrench \( \mathbf{w}_g \) is

\[
\mathbf{w}_g = \begin{bmatrix} \mathbf{Mg} \\ \mathbf{p}_g \times \mathbf{Mg} \end{bmatrix},
\]

where \( \mathbf{p}_g \) is the center of mass (CoM) of the robot. \( \mathbf{p}_g \) can be computed by \( \sum_{j=1}^{L} m_j \mathbf{p}_j / \sum_{j=1}^{L} m_j \), where \( m_j \) and \( \mathbf{p}_j \) are the mass and the position of \( j \)-th link of the robot in \( \Sigma_R \), respectively. Here \( \mathbf{g} \) is \( \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T \).

3) Inertia wrench : The inertia wrench \( \mathbf{w}_i \) can be written as

\[
\mathbf{w}_i = \begin{bmatrix} \mathbf{M} \dot{\mathbf{p}}_g \\ \mathbf{p}_g \times \mathbf{M} \dot{\mathbf{p}}_g - \mathbf{L} \end{bmatrix},
\]

where \( \mathbf{L} \) is the angular momentum of the robot with respect to \( \mathbf{p}_g \) is defined as

\[
\mathbf{L} = \sum_{j=1}^{J} \left[ m_j (\mathbf{p}_j - \mathbf{p}_g) \times \dot{\mathbf{p}}_j + I_j \omega_j \right],
\]

where \( I_j \) and \( \omega_j \) are the inertia tensor and the angular velocity of the \( j \)-th link of the robot, respectively.

The robot is dynamically stable when it satisfies

\[
\mathbf{w}_c + \mathbf{w}_g + \mathbf{w}_i = \mathbf{0}.
\]

C. Computation of Physics Violation Cost

Next we describe the computation of the physics-violation cost \( C_{Physics}(\mathbf{q}_k) \) in (2). First we formulate the combination of contact forces, which can be defined as:

\[
\mathbf{f} = [\mathbf{f}_1^T, \ldots, \mathbf{f}_L^T]^T.
\]

Equation (5) can be represented as \( \mathbf{w}_c = \mathbf{B} \mathbf{f} \), where \( \mathbf{B} \) is the corresponding \( 6 \times 3L \) matrix. Using this formulation, we solve an inverse dynamics computation problem, which computes \( \mathbf{f} \) such that it satisfies the Coulomb friction constraints of Equation (6):

\[
\mathbf{f} = \arg \min_{\mathbf{f}} (\| \mathbf{B}^* \mathbf{f} + \mathbf{w}_g + \mathbf{w}_i + \mathbf{f}^T \mathbf{R}^* \mathbf{f}^* ).
\]

The Coulomb friction constraint is usually converted to an inequality constraints, using a pyramid to approximate a friction cone \( F_i \) (shown in Fig. 1(b)). The constraint for \( \mathbf{f}_i \) is reduced to

\[
- \mu f_{in} \leq f_{lt} \leq \mu f_{in}
\]

where \( \begin{bmatrix} f_{ln} & f_{lt} & f_{lo} \end{bmatrix} \) is the actual optimal force that satisfies (10), when \( - \mu f_{in} \leq f_{lt} \leq \mu f_{in} \).

In (12), we add the contact variable penalty term \( \mathbf{f}^T \mathbf{R}^* \mathbf{f}^* \) as it is used in [28]. It increases the difference between \( \mathbf{f} \) from (12) and the actual optimal force that satisfies (10), when contact variable \( \mathbf{a}^l \) is small for a large contact force \( \mathbf{f} \).

The matrix \( \mathbf{R} \) is a \( 3L \times 3L \) diagonal matrix, and its diagonal elements correspond to

\[
R_{jj} = \frac{k_0}{(a^l_{(k)})^2 + k_1},
\]

where \( 3L - 2 < j < 3L \). \( k_0 \) and \( k_1 \) are constants that control the weight of the penalty cost.

The quadratic programming (QP) problem (12) can be solved using a QP solver; the result value of \( \mathbf{f} \) is used to compute the \( C_{Physics}(\mathbf{q}_k) \), which is evaluated as

\[
C_{Physics}(\mathbf{q}_k) = \| \mathbf{B}(\mathbf{q}_k) \mathbf{f} + \mathbf{w}_g(\mathbf{q}_k) + \mathbf{w}_i(\mathbf{q}_k) \|.
\]
V. RESULTS

In this section, we highlight the performance of our planning algorithm in simulated environments. We have implemented our algorithm in a simulator with a human-like robot model which has 34 DOFs.

We highlight the results for motion planning in different environments for the robot in Table I. We compute the trajectories of the robot in two environments, where the robot must move by walking from the initial configuration to the goal configuration. We evaluate the performance in two scenarios, where the robot needs to make contacts using its hand with the environments. We measure three components to evaluate the performance: the number of iterations in the optimization routines; the planning time to find the first collision-free and stable solution; and the smoothness of the trajectory. The results, shown in Table I, are the averages and standard deviations of 100 trials for each scenario. Videos of these and other benchmark experiments can be found at http://gamma.cs.unc.edu/ITOMP/.

We use hierarchical planning in our benchmarks to improve the planning performance. Hierarchical planning is a divide-and-conquer approach that decomposes a high-dimensional planning problem into several lower-dimensional problems [30], [31]. We decompose a robot into 5 different components: a lower body, which includes legs and pelvis; a torso; a head; a left arm; and a right arm, then incrementally plan the trajectory of the robot using this decomposition. In Fig. 2-5, different robot components used in hierarchical planning are marked with different colors.

Parameter values used our experiments are: \( N \) (Number of waypoints) = 100, \( P \) (Number of contact phases) = 5, \( k_0 \), \( k_1 \) (Contact variable penalty terms) = 0.01, 0.001, \( r \) (local displacement vector) = 0.1, \( T \) (length of the motion)=5.

Our first benchmark is planning a trajectory on an uneven terrain. The height of the terrain varies such that the ZMP-based methods may not be able to compute a dynamically stable solution. The planners with stability constraints compute the contact points between the robot’s feet and the terrain, and place the robot feet on these points. This generates a walking motion towards the goal configuration while satisfying the stability requirements. Fig. 2 shows the trajectory computed by the dynamic stability constraint.

In our second benchmark, the environment consists of several obstacles that the robot needs to avoid. We place an obstacle on the ground that the robot cannot go around, forcing it to pass over the obstacle. The trajectory computed with the stability constraint is shown in Fig. 3. In the computed trajectory, the robot does not collide with the obstacles and passes over the obstacles on the ground.

In the next two benchmarks, we test our algorithm with scenarios where the robot makes additional contacts with its right hand, while satisfying the stability constraints. The robot exerts force on the objects in the environment to perform manipulation tasks. In the third benchmark, the robot pushes a door (Fig. 4) to reach a goal. The robot pulls a drawer (Fig. 5) to move it to the desired position in the last benchmark.

Fig. 6 highlights the timing breakdown of an iteration of the trajectory computation. The percentage of time spent in stability cost computation takes 23.2% of the total computation time.

**TABLE I: Planning results for different benchmarks on a single CPU core.**

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th># Contacts</th>
<th>Iterations</th>
<th>Planning Time(s)</th>
<th>Trajectory Smoothness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steps (Fig. 2)</td>
<td>34</td>
<td>140.27</td>
<td>17.467</td>
<td>10.315</td>
</tr>
<tr>
<td>Obstacles (Fig. 3)</td>
<td>34</td>
<td>68.11</td>
<td>10.213</td>
<td>5.626</td>
</tr>
<tr>
<td>Door (Fig. 4)</td>
<td>3</td>
<td>35.64</td>
<td>4.404</td>
<td>4.419</td>
</tr>
<tr>
<td>Drawer (Fig. 5)</td>
<td>3</td>
<td>73.954</td>
<td>13.054</td>
<td>0.579</td>
</tr>
</tbody>
</table>

**TABLE II: This table compares the feature of our motion planning with dynamic stability algorithm with other approaches.**

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Collision-aware</th>
<th>Dynamic Stability</th>
<th>Uneven Terrain</th>
<th>Smooth Motion</th>
<th>Obstacle Movement of cOm</th>
<th>Physically Correct Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stability Computation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>in Wrench Space [21]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transform from Statically Balanced Path [5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contact-Sensitive Optimization [26]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct Contact Force Optimization [27]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our Approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comparisons:** Our algorithm combines the CIO approach and the wrench-space stability computation, integrating them into a hierarchical optimization framework. Our approach can compute smooth, physically-correct motions while efficiently computing the motions and reactions resulting from various contacts. Our approach is more than an order of magnitude faster than the other planning algorithms described above ([15], [28], [27]). At the same time, other planners with close to real-time performance either do not perform obstacle-aware motion planning or do not provide similar guarantees on dynamic stability. Table II shows a summary of the capabilities of the different algorithms.
VI. CONCLUSIONS, LIMITATIONS AND FUTURE WORK

We present a fast, dynamically stable, optimization-based motion planning algorithm for high-DOF robots. We use contact variables to compute dynamically stable motions. The stability of the motion is computed in a wrench space, and we compute the friction force that creates an equilibrium between the forces exerted on the robot. Our formulation of contacts is general and can handle multiple contacts simultaneously. We highlight the performance of our algorithm using a human-like robot with 34 DOFs.

There are some limitations to our approach. Currently, our approach (like the previous ITOMP [8] and CIO [28] approaches) assumes a fixed length of motion. This length of motion could be optimized in the future. Also, our formulation uses discretized waypoints on the continuous trajectory and the computation is only performed on the waypoints. However, the error due to the small interval is small and can be easily corrected with a real-time control approaches [11], [22]. For a feasible trajectory computed by optimization-based planner, a controller can be used to provide a feedback according to the measured executed trajectory.

There are many avenues for future work. Recently, we have extended our approach on more complex benchmarks where multiple robots and obstacles exist [32]. Furthermore, we would like to investigate other optimization techniques to improve optimization performance.

VII. ACKNOWLEDGMENTS

This research is supported in part by ARO Contract W911NF-10-1-0506, NSF awards 1000579, 1117127 and 1305286, and a grant from Sandia Labs.

REFERENCES
