Smooth and Dynamically Stable Navigation of Multiple Human-Like Robots

Chonhyon Park and Dinesh Manocha

University of North Carolina, Chapel Hill, NC 27599, USA,
{chpark,dm}@cs.unc.edu
http://gamma.cs.unc.edu/MultiRobot/ (Videos included)

Abstract. We present a novel algorithm for smooth and collision-free navigation for multiple human-like robots. Our approach combines reciprocal collision avoidance with kinematic and dynamic stability constraints to compute a non-oscillatory trajectory for each high-DOF robot. We use a multi-level optimization algorithm that combines acceleration-velocity obstacles with trajectory optimization. We highlight our algorithm's performance in different environments containing multiple human-like robots with tens of DOFs.

1 Introduction

Human-like robots are frequently used in robotics, computer animation, computer-aided design and related applications. As part of the recent DARPA challenge, attention to autonomous planning for humanoid robots has increased; this has stimulated considerable interest in efficient planning algorithms for high-DOF robots. Multiple human-like characters are commonly used in computer animation, and it is important to automatically compute their motion. Digital models of humans or mannequins are frequently used in assembly and virtual prototyping applications for design, assembly, and maintenance (for example, evacuation planning for a building or an airplane).

Human-like robots are, however, a challenge for planning. They have high degrees-of-freedom (DOF), which increases the complexity of their configuration and search spaces. The planning for these robots must also satisfy their kinematic and dynamic stability constraints: the computed posture for each human-like robot should be statically stable, and the forces and torques acting on each robot should maintain an equilibrium for dynamic stability. Some additional challenges arise in environments with multiple robots. The total number of DOF of the system increases linearly with the number of robots in the environment. Furthermore, it is important to compute smooth, non-oscillatory trajectories for these robots. Finally, the resulting environments may be non-planar (e.g. stairs), and it is difficult to navigate these complex environments while maintaining the stability and smoothness constraints. At the same time, most motion planning algorithms for multi-robot systems are restricted to low-DOF robots or do not take into account dynamics and stability constraints for human-like robots.

There is extensive work on motion planning for high-DOF robots as well as collision-free navigation of low-DOF multiple robots. The simplest algorithms for single-robot planning are based on sampling-based algorithms [15, 18] and can be extended to take into account kinematic and dynamic constraints [28, 4, 7, 10], There is recent resurgence in use of optimization-based techniques [24, 14, 21] as they can generate collision-free and smooth trajectories. Most optimization-based planners are designed for a single robot planning scenario in an environment composed of static and dynamic obstacles.

Main Results: In this paper, we address the problem of efficient navigation of multiple high-DOF human-like robots. Our approach can generate non-oscillatory, collision-free trajectories for each robot while accounting for kinematics, dynamics, and smoothness constraints. We use a multi-level optimization based algorithm to compute these trajectories. In the first level, we compute collision-free trajectories for each robot using *acceleration-velocity obstacles*. Our formulation takes into account the kinematic constraints of each human-like robot and reduces the computation to linear programming. The resulting trajectories are then used in the second phase to compute smooth motion for each DOF or joint of the human-like robot. We optimize the trajectory to compute a physically correct, dynamically stable motion for each robot (e.g. a walking motion) that takes into account all the contacts between the robot and the environment. The overall formulation is efficient and conservative. If the optimization algorithm computes the trajectories, they are are guaranteed to satisfy the constraints: smoothness, dynamic stability, and collision avoidance. However, it is possible (e.g. narrow passages) that the optimization algorithm may not find a global minima that would satisfy all the constraints.

We have evaluated our algorithm in different environments, using from 2 to 8 human-like robots with 34 DOFs each. We use a hierarchical decomposition scheme to improve the performance of the high-DOF planning for each robot.

The rest of the paper is organized as follows. In Section 2, we survey related work in optimization-based planning for high-DOF robots and in planning for multiple robots. In Section 3, we describe the two-level optimization algorithm that computes a trajectory for each robot. Section 4 analyzes our algorithm and provides guarantees on the resulting trajectories. Finally, we highlight our algorithm's performance in different scenarios in Section 5.

2 Related Work

In this section, we give a brief overview of prior work in optimization-based and multi-robot planning.

2.1 Optimization-based Motion Planning for High-DOF Robots

Optimization-based planners compute a trajectory using a continuous planning formulation. The optimization function can have various constraints formulated into trajectory computation, including path smoothness and collision-avoidance

constraints. Khatib proposed the use of potential fields for real-time obstacle avoidance [16]. This approach is extended using elastic strips [8] and elastic bands [23] to compute minimum-energy paths using gradient-descent methods. Some recent approaches, such as [24, 14] and [21], directly encode constraints into the optimization cost functions, then use a numerical solver to compute a trajectory.

Some optimization-based planning approaches take into account the stability of the motion, which is an important criterion in motion planning for high-DOF human-like robots. These include techniques based on inverse pendulum [13] or the zero moment point [12], but these approaches are limited to planar ground (i.e. flat surfaces). Recently, many optimization-based approaches approaches have integrated stability constraints directly into trajectory optimization [26, 20, 22, 9]. Mordatch et al. [20] use a contact-invariant optimization formulation, along with a simplified physics model, to generate various motions for animated characters. Posa et al. [22] directly optimize the contact forces, along with the state of the robot and the user input. Dai and Tedrake [9] formulate the uncertainty of the terrain into the optimization formulation.

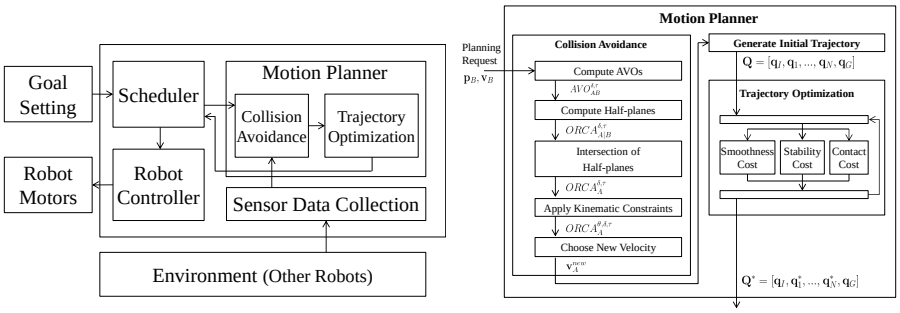
2.2 Multi-Robot Collision Avoidance

Algorithms that plan for multiple robots can be classified into either centralized or decoupled algorithms. The centralized planners [19, 25, 29] treat multiple robots as a single robot with the combined DOFs of all component robots and apply single-robot planning algorithms to the combination. Because the complexity of the centralized planning grows as the number of robots increases, the centralized planners are limited to environments with only a few low-DOF robots. On the other hand, the decoupled planners compute robot trajectories in a distributed manner, which allows them to plan for a large number of robots [1, 17].

Velocity obstacles [11] is one of the most widely-used approaches for motion planning in dynamic environments. The basic velocity-obstacle method was extended by Berg et al., who offered a method using reciprocal velocity obstacles [5] that is especially useful for avoiding collisions between robots; it is fast and generates oscillation-free motion among the robots. Some variants of reciprocal velocity obstacles also take into account maximum acceleration limits or smoothness constraints [6, 2]. However, these approaches all assume that the robots have simple disc-like or spherical shapes.

3 Planning Algorithm

In this section, we give an overview and the details of our two-level motion planning algorithm for multiple high-DOF human-like robots. Our optimization-based algorithm is decomposed into two levels. The first level computes collision-free trajectories for multiple robots based on kinematic constraints. The second level optimizes the individual trajectories with smoothness and stability constraints.



(a) The overall planning algorithm (b) Two-level planner

Fig. 1: (a) An overview of our planning algorithm. The scheduler module sends a planning request to the planner during each time step. The computed trajectory is sent to the robot controller. (b) The motion planner is decomposed into two levels: collision avoidance and trajectory optimization and various stages of each level are shown in the figure. The collision avoidance module computes a velocity that avoids collisions with other robots, then generates an initial trajectory for each robot that is then used for trajectory optimization.

3.1 Overview

Our planning algorithm assumes that each of the multiple robots computes its own trajectory in a decoupled manner without any explicit communication between the robots. We assume that each planner has a full representation of the environment and of the position and velocity of the other robots in the environment. Our current formulation therefore doesn't account for any uncertainty in the environment or the position and velocity of other robots.

Fig. 1(a) gives an overview of the algorithm for each robot. The planning approach consists of multiple modules: scheduler, sensor data collection, and robot controller. When a goal position (new or initial) for each robot is set, the scheduler sends a planning request to the motion planner. The planning request has a planning time limit Δt. The sensor module updates the environmental information, including the positions and velocities of other robots; based on this environmental information, the motion planner then computes a trajectory for the next execution step.

The motion planner is decomposed into two levels. For each robot A, the first level computes a collision-avoiding velocity $\mathbf{v}_A^{new}$ that ensures that A does not collide with other robots during that interval. In the computation of the collision-avoiding velocity, we model each robot A as a 2D disk, which can be defined using a point $\mathbf{p}_A = (x_A, y_A)$ and a radius r_A that can cover the actual robot. We use the 2D position of the root link of the model hierarchy, which usually corresponds to waist or pelvis link of a human-like robot, as $\mathbf{p}_A$ and denote it as the root of the robot A. The computed velocity $\mathbf{v}_A^{new}$ is constrained

by the kinematic constraints of the given human-like robot model, and these constraints depend on the orientation of the robot θ_A. We denote this velocity bound computed by the kinematic constraints for a human-like robot as $H(\theta_A)$. $\mathbf{v}_A^{new}$ is used to generate a collision-free initial trajectory for the second level, which then computes a trajectory for the robot using trajectory optimization. The second level takes into account the robot model's smoothness and dynamic stability constraints.

3.2 Collision Avoidance

Our collision avoidance computation algorithm is based on *acceleration-velocity obstacles* (AVO) [6]. AVO is a set of velocities at which the robot would collide with obstacles (including other robots) if the robot velocity is in AVO. AVO can be defined in 2D space for two disc-shaped robots A and B. First we denote an open disc of radius r centered at $\mathbf{p}$ as

$$D(\mathbf{p}, r) = \{\mathbf{q} | \|\mathbf{q} - \mathbf{p}\| < r\}. \tag{1}$$

Using (1), AVO for robot A respect to B is defined as the set of all relative velocities $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$ such that:

$$AVO_{AB}^{\delta,\tau} = \bigcup_{t \in [0,\tau]} D\left(\frac{\delta(e^{-t/\delta} - 1)\mathbf{v}_{AB} - \mathbf{p}_{AB}}{t + \delta(e^{-t/\delta} - 1)}, \frac{r_{AB}}{t + \delta(e^{-t/\delta} - 1)}\right), \tag{2}$$

where $\mathbf{p}_{AB} = \mathbf{p}_A - \mathbf{p}_B$ is the relative position, $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$ is the relative velocity, $r_{AB} = r_A + r_B$ is the sum of robot radii, τ is the time horizon, and δ is an acceleration control parameter [6]. The definition implies that if the robot A chooses a new velocity $\mathbf{v}'_A$ which pushes $\mathbf{v}_{AB}$ outside of $AVO_{AB}^{\delta,\tau}$, the robots will not collide before time τ while A and B have the same acceleration control parameter δ. The set of collision-avoiding velocities $CA_{A|B}^{\delta,\tau}$ for A with respect to B is defined as

$$CA_{A|B}^{\delta,\tau}(\mathbf{v}_B) = \{\mathbf{v} + \mathbf{v}_B | \mathbf{v} \notin AVO_{AB}^{\delta,\tau}\}. \tag{3}$$

In a multi-robot planning environment with more than two robots, the computed velocity of the collision avoidance should be outside the AVOs of all other robots. We use the *optimal reciprocal collision avoidance* (ORCA) algorithm [5] to compute these velocities. Rather than computing the exact set of collision-avoiding velocities $CA_{A|B}^{\delta,\tau}$, the ORCA algorithm defines the permitted velocities for A respect to B as a half-plane:

$$ORCA_{A|B}^{\delta,\tau}(\mathbf{v}_B) = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}_A^{opt} + \frac{1}{2}\mathbf{u})) \cdot \mathbf{n} \geq 0\}, \tag{4}$$

where $\mathbf{v}_A^{opt}$ is *optimization velocity* (the velocity that the robot would have chosen if there are had been no obstacles), $\mathbf{u} = (\arg\min_{\mathbf{v} \in \partial AVO_{A|B}^{\delta,\tau}} \|\mathbf{v} - (\mathbf{v}_A^{opt} -$

$\mathbf{v}_B^{opt})\|) - (\mathbf{v}_A^{opt} - \mathbf{v}_B^{opt})$ and $\mathbf{n}$ is the outward normal of the boundary of $AVO_{A|B}^{\delta,\tau}$ at $(\mathbf{v}_A^{opt} - \mathbf{v}_B^{opt}) + \mathbf{u}$. The computed velocities eliminate oscillatory motion or trajectory without any explicit communications between the robots [5]. The optimal velocity $ORCA_A^{\delta,\tau}$ for A is computed as the intersection of the half-planes,

$$ORCA_A^{\delta,\tau} = \bigcap_{B \neq A} ORCA_{A|B}^{\delta,\tau}(\mathbf{v}_B). \tag{5}$$

Our algorithm also applies the kinematic constraints of the human-like robot (Section 3.3),

$$ORCA_A^{\theta,\delta,\tau} = H(\theta) \cap ORCA_A^{\delta,\tau}. \tag{6}$$

From the set of velocities in $ORCA_A^{\theta,\delta,\tau}$, our algorithm computes the velocity that is closest to $\mathbf{v}_A^{opt}$; it then computes the goal position of the current planning step $\mathbf{p}_A^{goal}$:

$$\mathbf{v}_A^{new} = \underset{\mathbf{v} \in ORCA_A^{\theta,\delta,\tau}}{\arg\min} \|\mathbf{v} - \mathbf{v}_A^{opt}\|, \tag{7}$$

$$\mathbf{p}_A^{goal} = \mathbf{p}_A + \mathbf{v}_A^{new} \Delta t. \tag{8}$$

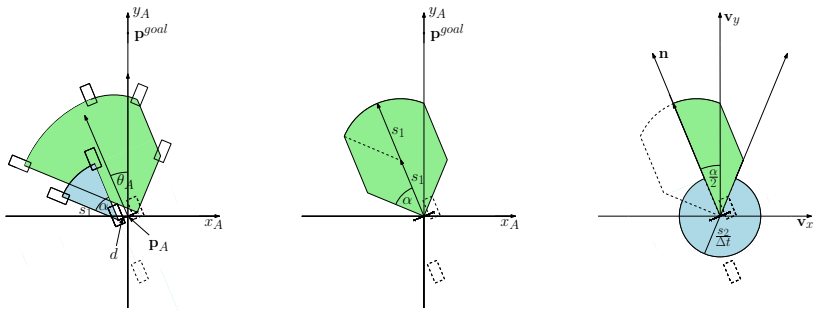
3.3 Kinematic Constraints for Human-like Robots

The computation of AVO (2) requires the radius of the robot. We use the *personal space* defined by a radius r, rather than the exact physical extent of the robot, as part of our collision avoidance computation. *Personal space* is a psychological concept corresponds to the empty region between two nearby persons that reflects each person's comfort level and allows enough space for them to swing their limbs. In the context of multi-robot planning, this *Personal space* is used to provide each robot enough space to move its arms and legs.

The basic AVO algorithm takes into account kinematic constraints in terms of maximum velocity and acceleration limits. But the prior algorithm is designed for 2D disc like robots, and is therefore indifferent to the robot's orientation when computing velocities [6]. For human-like robots, kinematic constraints are highly dependent on the robot's orientation. Our approach therefore uses orientation-dependent velocity constraints since we are working with human-like robots. We denote these orientation-dependent velocity constraints as $H(\theta)$, and add these constraints to the computed set of collision-avoiding velocities.

A locomotion of a human-like robot A is modeled by a trajectory of the robot root states, $(\mathbf{p}_A, \mathbf{v}_A, \theta_A)$. The human-like robots move by taking individual footsteps, which correspond to contacts between robot feet and the ground. Each new footstep's generation is constrained by the last footstep [27]. In order to formulate this constraint, we assume the human-like robot moves based on *forward walking*, and that the robot's root orientation is the same as the orientation of one of the feet that moved in the last footstep [3]. The constraint of the permitted new position of a left foot can be formulated as

$$\{(x - d\sin\theta + s_1\cos\theta', y - d\cos\theta + s_1\sin\theta') | \theta' \in [\theta, \theta + \alpha]\}, \tag{9}$$



(a) Permitted left foot position (blue) and right foot position (green).

(b) Permitted root position (x_A, y_A) for two steps (green).

(c) Forward walking bound (blue) and side-stepping bound (green).

Fig. 2: Formulating the velocity bound of the kinematic constraints $H(\theta)$ for a human-like robot. (a) The permitted left foot position and right foot position for forward walking, when the last step is taken by the right foot (shown in dotted lines). In a *forward walking*, a single left step only can turn the robot orientation to the left; with only two steps, the robot can orient itself to the left and right sides, within a range of $[\theta - \alpha, \theta + \alpha]$ (α is the maximum z-axis rotational angle of the left foot). (b) Permitted robot root position for two steps, which is a symmetric shape corresponding to the robot orientation θ_A. (c) The velocity bound $H(\theta)$ for a human-like robot. The robot maintains its orientation except when it is facing towards the goal; it then orients itself toward the goal.

where d is the distance between the robot root and the left foot, s_1 is the bound of *forward walking* (i.e., the length of a stride), and α is the maximum z-axis rotational angle of the left foot. The constrained region is shown in Fig. 2(a) in the blue color. The position of the previous footstep is shown with the dotted line. The robot can only change its orientation to one single side within a single footstep; if we plan only a single footstep during the planning interval, it therefore invalidates most of the collision avoiding velocities. Rather than using a single footstep, we compute two consecutive steps within our planning time interval. In this case, the permitted position of the second right footstep is anything within the range of orientations in $[\theta - \alpha, \theta + \alpha]$ (green region shown in Fig. 2(a)), and the robot root position has a large symmetric bound corresponding to the robot orientation θ_A, as shown in Fig. 2(b). The velocity bound depending on the robot's orientation corresponds to the bound on the position of robot's root, and can be computed from the two constants, stride distance s and time step Δt. However, the *forward walking* assumption means that the robot cannot choose to move to the side or back. Robots can, however, move a distance s_2 in an arbitrary direction using two footsteps without changing their orientation; this is useful when the robot, in order to avoid collisions with other robots, must move in a direction very different from its current orientation. Therefore,

we formulate the velocity constraint $H(\theta)$ shown in Fig. 2(c) using both *forward walking* and side-stepping, depending on the angle between robot's current orientation and its orientation towards the goal position $\mathbf{p}^{goal} = (x^{goal}, y^{goal})$. We use the *forward walking* bound when $\theta \in \tan^{-1}(\mathbf{p}^{goal} - \mathbf{p}_A) \pm \alpha/2$; otherwise we use the side-stepping bound $D(0, s_2/\Delta t)$. The bound on the angle α allows the robot to change its orientation towards its desired orientation in one planning step, and perform side-stepping or back-stepping motion when the robot is not heading towards $\mathbf{p}^{goal}$ because of other nearby robots. The velocity bound $H(\theta)$ is formulated as the union of the side-stepping and the *forward walking* bounds,

$$D(0, \frac{s_2}{\Delta t}) \bigcup \Bigg(D(0, 2\frac{s_1}{\Delta t}) \cap \Big\{ \mathbf{v} \Big| \left\| \tan^{-1}(\mathbf{v}) - \theta^{goal} \right\| < \frac{\alpha}{2} \Big\} \\ \cap \Big\{ \mathbf{v} | \mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n} < \frac{s_1}{\Delta t} \sin\alpha \Big\} \Bigg), \tag{10}$$

where $\theta^{goal} = \tan^{-1}(\mathbf{p}^{goal} - \mathbf{p}_A)$, and $\mathbf{n} = (\cos\theta, \sin\theta)$.

3.4 Trajectory Optimization

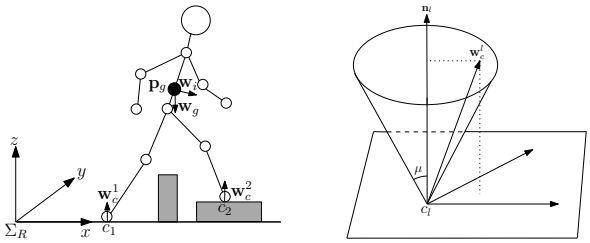
The second level of our motion planner performs trajectory optimization based on all the DOFs of the robot. The initial trajectory $\mathbf{Q}$ is initialized using the result of the computation of the collision-avoiding velocity. The trajectory of the root position is initialized as a cubic polynomial curve by Hermite interpolation using the position and the velocity of the goal position of the prior planning step and the new collision-avoiding velocity and position as the end-point constraints. Then $\mathbf{Q}$ is optimized using all DOFs of the robot. We use the ITOMP optimization-based framework [21] for the trajectory optimization. The trajectory $\mathbf{Q}$ is first discretized into $N + 2$ waypoints, $\{\mathbf{q}_I, \mathbf{q}_1, ..., \mathbf{q}_N, \mathbf{q}_G\}$, where N is the number of internal waypoints. Each waypoint $\mathbf{q}_i$ is a robot configuration of all actuated joints and the position of the robot root. The start configuration $\mathbf{q}_I$ is set to the current robot configuration, and the goal configuration $\mathbf{q}_G$ is set with the goal position of the robot root $\mathbf{p}_A^{goal}$. The internal waypoints $\{\mathbf{q}_1, ..., \mathbf{q}_N\}$ are initialized using an interpolation to generate a smooth trajectory. Using the internal waypoints $\{\mathbf{q}_1, ..., \mathbf{q}_N\}$ as the optimization variables, our planner optimizes the following cost function to compute the optimal trajectory:

$$\mathbf{Q}^* = \arg\min_{\mathbf{q}_1, ..., \mathbf{q}_N} \sum_{k=1}^{N} (C(\mathbf{q}_k) + \|\mathbf{q}_{k-1} - 2\mathbf{q}_k + \mathbf{q}_{k+1}\|^2), \tag{11}$$

where the term $C(\mathbf{q}_k)$ represents the cost function for the waypoint $\mathbf{q}_k$, and the second term $\|\mathbf{q}_{k-1} - 2\mathbf{q}_k + \mathbf{q}_{k+1}\|^2$ represents the smoothness at waypoint $\mathbf{q}_k$. The waypoint smoothness is computed based on the finite-difference accelerations on the joint trajectories.

3.5 Dynamic Stability and Contacts Generation

It is important that the trajectories of articulated human-like robots with high-DOFs are dynamically stable and that the robot is able to maintain its balance.



(a) Forces exerted on the robot.

(b) Friction cone.

Fig. 3: A human-like robot makes contacts c_1 and c_2 with the ground plane. The gravity force $\mathbf{w}_g$ and the inertia force $\mathbf{w}_i$ are applied to the robot. The contact forces $\mathbf{w}_c^1$ and $\mathbf{w}_c^2$ can have values in their friction cone. The robot is stable when $\mathbf{w}_c^1 + \mathbf{w}_c^2 + \mathbf{w}_g + \mathbf{w}_i = \mathbf{0}$.

As shown in Fig. 3(a), a robot configuration $\mathbf{q}_k$ is stable when the sum of the all internal and external forces exerted on the robot is zero [30]. This zero sum implies that the contacts between the robot and the rest of the environment (e.g. the ground) need to be planned accordingly, positioning the robot so that the resultant forces maintain that equilibrium.

In our formulation, the waypoint cost function $C(\mathbf{q}_k)$ includes the costs for the stability constraints that account for force equilibrium and contacts generation for a waypoint $\mathbf{q}_k$. These costs can be expressed as

$$C(\mathbf{q}_k) = C_{Stability}(\mathbf{q}_k) + C_{Contact}(\mathbf{q}_k). \tag{12}$$

$C_{Stability}(\mathbf{q}_k)$ represents the cost from the violation of the equilibrium of forces, and it is defined as

$$C_{Stability}(\mathbf{q}_k) = \min_{\mathbf{w}_c^1,\ldots,\mathbf{w}_c^L} \| \sum_{l=1}^{L} \mathbf{w}_c^l + \mathbf{w}_g(\mathbf{q}_k) + \mathbf{w}_i(\mathbf{q}_k) \|, \tag{13}$$

where L is the total number of contact points, and $\mathbf{w}_g(\mathbf{q}_k)$ and $\mathbf{w}_i(\mathbf{q}_k)$ are the gravity and inertia forces at waypoint $\mathbf{q}_k$, respectively. $\mathbf{w}_c^l$ is the contact reaction force of l-th contact point, which is constrained by Coulomb's friction law. $\mathbf{w}_c^l$ is its friction cone defined by a friction coefficient μ to avoid slipping (Fig. 3(b)). In other words, $\mathbf{w}_c^l$ should satisfy

$$\|\mathbf{w}_c^l - (\mathbf{n}^l \cdot \mathbf{w}_c^l)\mathbf{n}^l\| \leq \mu(\mathbf{n}^l \cdot \mathbf{w}_c^l), \tag{14}$$

where $\mathbf{n}^l$ is the contact normal of l-th contact.

$C_{Contact}(\mathbf{q}_k)$ represents the penalty cost from the invalid contacts generation: The reaction forces from contact points affect the stability cost computation of (13). The magnitudes of the contact forces can be directly optimized in the trajectory optimization [22], but we use instead an indirect approach used in *Contact-Invariant Optimization*, that assigns scalar variables ρ for contact

points, then computes the appropriate contact forces $\mathbf{w}_c$ and the penalty cost $C_{Contact}(\mathbf{q}_k)$ [20]. The cost is defined as

$$C_{Contact}(\mathbf{q}_k) = \sum_{l=1}^{L} \rho_k^l (\|\mathbf{e}_k^l(\mathbf{q}_k)\|^2 + \|\dot{\mathbf{c}}_k^l(\mathbf{q}_k)\|^2), \tag{15}$$

where L is the total number of potential contact points and $\dot{\mathbf{c}}_k^l$ is the velocity of the l-th contact point $\mathbf{c}_k^l$. $\mathbf{e}_k^l$ represents the distance from $\mathbf{c}_k^l$ to the nearest point on the obstacles. Therefore, the cost function becomes high when the contact point is not on the environment or is sliding. ρ_k^l is a scalar variable that represents whether the l-th contact is active in the waypoint $\mathbf{q}_k$. The contact cost is ignored when ρ_k^l is 0, which implies that the corresponding contact point $\mathbf{c}_k^l$ is inactive at waypoint $\mathbf{q}_k$.

4 Mathematical Guarantees

In this section, we give the mathematical guarantees of our planning algorithm's suitability for high-DOF human-like robots; including smoothness of the computed trajectory and local collision avoidance among multiple robots.

4.1 Smoothness of the Computed Trajectory

The ORCA algorithm generates a continuous trajectory of velocities, and that this continuous velocity trajectory guarantees the smoothness of the robot trajectory [5]. In our approach, the velocities computed by ORCA in the first computation level are used to generate the initial trajectory $\mathbf{Q}$, which used as input for the trajectory optimization in the second-level computation. Therefore, it is necessary to show that the trajectory after the optimization is smooth, and that the partial trajectories computed in multiple planning steps keep the continuity between next and previous trajectories at their endpoints.

Theorem 1. *Given a small duration δt between adjacent waypoints on a trajectory that is computed from our planning algorithm, the velocity of the trajectory is continuous, i.e., the waypoints on the trajectory satisfy $\dot{\mathbf{q}}_i \approx \dot{\mathbf{q}}_{i+1}$ for all i, where $\approx$ denotes 'arbitrarily close to' as $\delta t \to 0$.*

Proof. Our planning algorithm computes partial trajectories of length Δt in multiple planning steps using ORCA-based collision avoidance and the trajectory optimization. As described in Section 3.4, the waypoints $\mathbf{q}_i$ on the trajectory are evenly spaced by δt and evaluated on polynomial curves of joints $\mathbf{P}(t)$, i.e., $\mathbf{q}_i = \mathbf{P}(i \cdot \delta t)$. Since $\dot{\mathbf{P}}(t) \approx \dot{\mathbf{P}}(t + \delta t)$ holds for any polynomial curves, $\dot{\mathbf{q}}_i = \dot{\mathbf{P}}(i \cdot \delta t) \approx \dot{\mathbf{P}}(i \cdot \delta t + \delta t) = \dot{\mathbf{q}}_{i+1}$ for the initial trajectories. It is guaranteed in [24] that the trajectory optimization using covariant gradient updates keeps the smoothness of the initial trajectory.

In the each planning step, the planner uses the position, velocity, and acceleration of the goal waypoint of the last planning step as the endpoint constraint

when performing the initial trajectory computation of the partial trajectory. It ensures that any pair of two adjacent waypoints $(\mathbf{q}_i, \mathbf{q}_{i+1})$ on the entire trajectory is a subset of a single planning step trajectory, where $\dot{\mathbf{q}}_i \approx \dot{\mathbf{q}}_{i+1}$ holds.

4.2 Local Collision Avoidance

It is given by [5] that the ORCA algorithm can guarantee that the computed trajectory for robot A is collision-free for time τ if $ORCA_A^{\delta,\tau}$ is not empty; choosing $\mathbf{v}_A^{opt}$ carefully ensures that it is never empty. We can claim that this holds true for our collision avoidance computation even with the kinematic constraints introduced by high-DOF human-like robots.

Theorem 2. *Given a time step δt, the computed trajectory for a robot A does not collide with trajectories of any other robot B for $B \neq A$, if $ORCA_A^{\delta,\tau}$ is not empty.*

Proof. In Fig. 2(c), $H(\theta)$ covers the velocities that $\|v\| \leq s/\Delta t$, regardless of the orientation θ. It implies $ORCA_A^{\theta,\delta,\tau}$ can be non-empty if $ORCA_A^{\delta,\tau}$ is not empty. If the collision avoidance algorithm chooses a collision-avoiding velocity $\mathbf{v}_A^{new}$ in $ORCA_A^{\theta,\delta,\tau}$, that means that the personal space of robot A, $D(\mathbf{p}_A, r_A)$ will not intersect with the personal spaces of other robots during time interval τ. The robot A is always completely contained by $D(\mathbf{p}_A, r_A)$, so the robot A does not intersect with other robots if $\Delta t \leq \tau$.

5 Experimental Results

In this section, we describe the implementation of our multi-robot planning algorithm and present the results in different scenarios. We implemented our algorithm for simulated robots using a human-like robot model which has 34 DOFs. The robot model is 2.3m tall, and we set the variables for kinematic constraints and replanning: radius of the personal space $r = 1.0$m, stride $s = 0.5$m, maximum foot z-axis rotational angle $\alpha = \pi/2$, the number of internal waypoints in the trajectory optimization $N = 100$, planning step size $\Delta t = 2$ sec., the velocity obstacle time horizon $\tau = 10$ sec., and the acceleration control parameter $\delta = 4$ sec. The planning computation for each robot was performed using separate threads. As it is shown in Fig. 1, each planner first computes a collision avoiding velocity $\mathbf{v}_A^{new}$ and corresponding initial trajectory, then optimizes the trajectory, which has two footsteps. We decompose the robot model and plan the trajectories of the decomposed robot components in a hierarchical manner to improve the performance of the planning. Timing results were taken on a PC equipped with an Intel i7-2600 8-core CPU 3.4GHz.

We test our approach in several benchmark scenarios to demonstrate the collision avoidance behavior and dynamically stable motions. We highlight the results for planning in different benchmarks in Table 1.

- *Position Exchange* (Fig. 4(a)) : Two robots exchange their positions by passing each other.

Benchmark	Number of Robots (DOFs)	Trajectory Length (s)	Collision Avoidance Time (ms)	Trajectory Optimization Time (ms)	Features
Position Exchange	2(68)	40s	0.007ms	617ms	Collision avoidance on a non-planar ground
Dynamic Obstacles	8(272)	48s	0.023ms	476ms	Real-time dynamic obstacle handling
Circle	8(272)	76s	0.030ms	670ms	Kinematic constraints (w/ Side-stepping)
Circle w/o Side-stepping	8(272)	96s	0.031ms	656ms	Kinematic constraints (w/o Side-stepping)
Narrow Passage	8(272)	100s	0.045ms	1108ms	Hierarchical planning for narrow passage

Table 1: Planning results for different benchmarks. We show the number of robots; the trajectory length that corresponds to the total time that the robots took to reach their goals; the average computation times for the collision avoidance and the trajectory optimization for each planning step. Videos of these benchmarks can be found at http://gamma.cs.unc.edu/MultiRobot/.

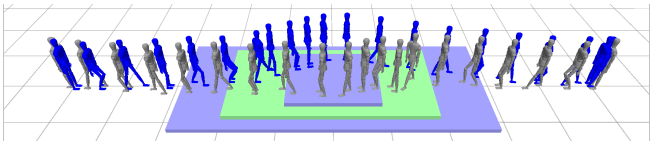
- *Dynamic Obstacles* (Fig. 4(b)) : The benchmark has moving obstacles, and 8 robots have to cross obstacle's path to navigate to their goals.
- *Circle* (Fig. 4(c), 5) : We initialize 8 robots on a circle. Each robot moves through the center of the circle to the goal position opposite its initial position.
- *Narrow Passage* (Fig. 4(d)) : Static obstacles make narrow passages, which is like a building entrance. 8 robots move through the narrow passage.

Videos of these and other benchmark experiments can be found at http://gamma.cs.unc.edu/MultiRobot/.

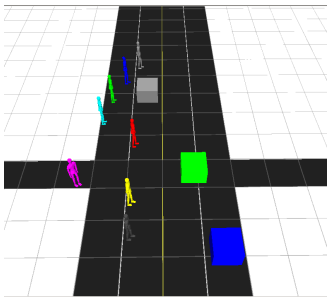
Position Exchange scenario is used as a benchmark for many ORCA-based approaches [5, 6]. In this benchmark, two robots are initialized to exchange their positions by passing each other. They move directly toward their goals at beginning, but when the robots notice that a collision will happen within τ, they change their directions to avoid the collision. Furthermore, we consider an uneven group with steps. Our planner compute the walking motion on uneven ground using the contact and stability constraints (12).

Dynamic Obstacles benchmark has three dynamic obstacles that move using constant velocities, and are not reactive to the robots. Robots know the velocities and positions of the obstacles, and move while avoiding collisions with the dynamic obstacles. This benchmark shows that our approach can naturally deal with the presence of obstacles that do not adapt its motion to the other robots, using human-like robots with forward walking and side-stepping motions.

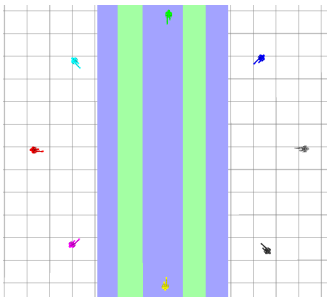
Our third benchmark is *Circle*, where the robots are placed along the circumference of a circle and their corresponding goal are at the anti-podal positions. The ground is not planar, but the computed trajectories are smooth and dynamically stable, with no oscillations or collisions. We also computed robot



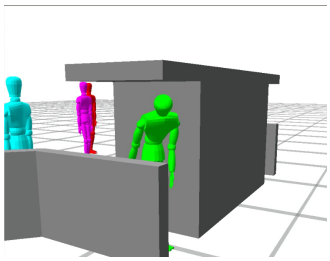
(a) *Position Exchange* benchmark



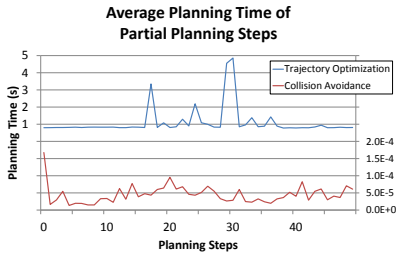
(b) *Dynamic Obstacles* benchmark



(c) *Circle* benchmark



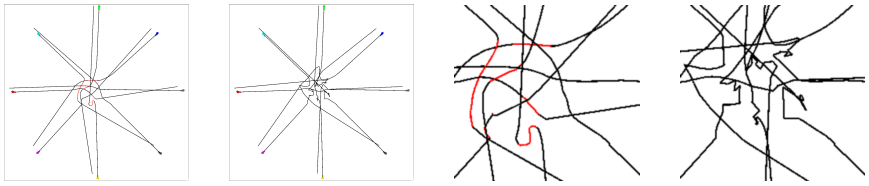
(d) *Narrow Passage* benchmark



(e) Planning time of each planning step in *Narrow Passage* benchmark.

Fig. 4: (a)(b)(c)(d) Benchmarks used in our paper. (e) Plot of the the planning time of the collision avoidance and the trajectory optimization along the trajectory for a robot.

trajectories of *Circle* benchmark with restricting the robot motion only to forward walking, and show the comparison of the computed trajectories in Fig. 5. In the trajectories with side-stepping (Fig. 5(a)(c)), the red segments show side-stepping motions. It shows that robots use the side-stepping to move around other robots. For example, the yellow robot moves using back steps to avoid colliding with the magenta robot. The trajectories are collision-free and smooth. However, if we restrict the robot motion only to forward walking (Fig. 5(b)(d)), the trajectories are neither collision-free nor smooth (See Theorem 2). Furthermore, the time that the robots reach their goal of the forward walking only motions is 96 sec., which is greater than 76 sec. of motions using both forward walking and side-stepping.



(a) Trace of robots with side-stepping. (b) Trace of robots with side-stepping. (c) Zoomed center of (a). (d) Zoomed center of (c).

Fig. 5: Comparison of computed trajectories of *Circle* benchmark with and without side-stepping. Black segments show the forward walking motions, and the red segments show side-stepping motions.

Finally, we highlight some narrow passages due to static obstacles in the *Narrow Passage* benchmark. In this benchmark, the width of the passage is shorter than the personal space radius $r = 1.0$, and we use a smaller radius for collision avoidance computation. Moreover, there are obstacles at a height that is the same as that of the robot. In order to handle the obstacles, we add the collision cost in trajectory optimization. Fig. 4(d) shows that the robots move their arms and heads to avoid collisions with the obstacles in the computed trajectories. In Fig. 4(e), we show the planning time of the collision avoidance and the trajectory optimization for each planning step for a robot. It shows that the collision avoidance computation takes less than 0.01ms, during the entire trajectory. Most of the time is spent in trajectory optimization.

6 Conclusions, Limitations and Future Work

In this paper we have proposed a motion planning algorithm for multiple high-DOF human-like robots. We model the kinematic constraints of a human-like robot and apply these constraints in computing collision avoidance. We have combined this collision avoidance formulation with trajectory optimization in the entire planning framework using the result of the collision avoidance computation to generate the initial trajectory for the trajectory optimization. The trajectory optimization step uses constraints for contacts and stability; these constraints ensure that the computed motion is dynamically feasible for the given high-DOF human-like robot. Therefore, our planning algorithm computes trajectories for multiple robots, which are guaranteed to be smooth, to avoid collisions with other robots, to obey the kinematic constraints of human-like robots, and to remain dynamically stable. We validate our algorithm in several benchmark scenarios where multiple robots move on uneven ground without collisions.

There are some limitations to our approach. The guarantees of the collision avoidance algorithm on the collision-free initial trajectory holds for a limited set of optimal velocities $\mathbf{v}_A^{opt}$. The collision avoidance performs better when using the current velocity as the optimal velocity, but this breaks the guarantees.

There are many avenues for future work. Our approach expects reciprocality from other robots. However, there are scenarios in which full reciprocality is not expected, such as environments have multiple robots and multiple humans. One possible future direction for research is the creation of a planning algorithm for use in environments containing both robots and humans. Furthermore, our approach can be combined with real human behavior [3] to generate virtual human motions. Moreover, we would like to investigate better modeling of the kinematic constraints of human-like robots $H(\theta)$ for high-speed motions.

7 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

1. Abe, Y., Yoshiki, M.: Collision avoidance method for multiple autonomous mobile agents by implicit cooperation. In: Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on. vol. 3, pp. 1207–1212. IEEE (2001)
2. Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., Siegwart, R.: Optimal reciprocal collision avoidance for multiple non-holonomic robots. Springer (2013)
3. Arechavaleta, G., Laumond, J.P., Hicheur, H., Berthoz, A.: An optimality principle governing human walking. Robotics, IEEE Transactions on 24(1), 5–14 (2008)
4. Berenson, D., Srinivasa, S., Kuffner, J.: Task space regions a framework for pose-constrained manipulation planning. The International Journal of Robotics Research 30(12), 1435–1460 (2011)
5. van den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics research, pp. 3–19. Springer (2011)
6. van den Berg, J., Snape, J., Guy, S.J., Manocha, D.: Reciprocal collision avoidance with acceleration-velocity obstacles. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on. pp. 3475–3482. IEEE (2011)
7. Bouyarmane, K., Kheddar, A.: Humanoid robot locomotion and manipulation step planning. Advanced Robotics 26(10), 1099–1126 (2012)
8. Brock, O., Khatib, O.: Elastic strips: A framework for motion generation in human environments. International Journal of Robotics Research 21(12), 1031–1052 (2002)
9. Dai, H., Tedrake, R.: Optimizing robust limit cycles for legged locomotion on unknown terrain. In: IEEE Conference on Decision and Control. pp. 1207–1213 (2012)
10. Dalibard, S., El Khoury, A., Lamiraux, F., Nakhaei, A., Taıx, M., Laumond, J.P.: Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach. The International Journal of Robotics Research (2013)
11. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. International Journal of Robotics Research 17(7), 760–772 (1998)
12. Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., Tanie, K.: Planning walking patterns for a biped robot. IEEE Transactions on Robotics and Automation 17(3), 280–289 (2001)

13. Kajita, S., Tani, K.: Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. In: Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on. pp. 1405–1411. IEEE (1991)
14. Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., Schaal, S.: STOMP: Stochastic trajectory optimization for motion planning. In: Proceedings of IEEE International Conference on Robotics and Automation. pp. 4569–4574 (2011)
15. Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
16. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. International Journal on Robotics Research 5(1), 90–98 (1986)
17. Kluge, B., Prassler, E.: Reflective navigation: Individual behaviors and group behaviors. In: IEEE International Conference on Robotics and Automation. pp. 4172–4177 (2004)
18. Kuffner, J., LaValle, S.: RRT-connect: An efficient approach to single-query path planning. In: Proceedings of IEEE International Conference on Robotics and Automation. pp. 995 – 1001 (2000)
19. LaValle, S.M., Hutchinson, S.A.: Optimal motion planning for multiple robots having independent goals. Robotics and Automation, IEEE Transactions on 14(6), 912–925 (1998)
20. Mordatch, I., Todorov, E., Popović, Z.: Discovery of complex behaviors through contact-invariant optimization. ACM Transactions on Graphics (TOG) 31(4), 43 (2012)
21. Park, C., Pan, J., Manocha, D.: ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. In: Proceedings of International Conference on Automated Planning and Scheduling (2012)
22. Posa, M., Tedrake, R.: Direct trajectory optimization of rigid body dynamical systems through contact. In: Algorithmic Foundations of Robotics X, pp. 527–542. Springer (2013)
23. Quinlan, S., Khatib, O.: Elastic bands: connecting path planning and control. In: Proceedings of IEEE International Conference on Robotics and Automation. pp. 802–807 vol.2 (1993)
24. Ratliff, N., Zucker, M., Bagnell, J.A.D., Srinivasa, S.: CHOMP: Gradient optimization techniques for efficient motion planning. In: Proceedings of International Conference on Robotics and Automation. pp. 489–494 (2009)
25. Sanchez, G., Latombe, J.C.: Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In: Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. vol. 2, pp. 2112–2119. IEEE (2002)
26. Schultz, G., Mombaur, K.: Modeling and optimal control of human-like running. Mechatronics, IEEE/ASME Transactions on 15(5), 783–792 (2010)
27. Singh, S., Kapadia, M., Reinman, G., Faloutsos, P.: Footstep navigation for dynamic crowds. Computer Animation and Virtual Worlds 22(2-3), 151–158 (2011)
28. Stilman, M.: Global manipulation planning in robot joint space with task constraints. Robotics, IEEE Transactions on 26(3), 576–584 (2010)
29. Švestka, P., Overmars, M.H.: Coordinated path planning for multiple robots. Robotics and autonomous systems 23(3), 125–152 (1998)
30. Trinkle, J.C., Pang, J.S., Sudarsky, S., Lo, G.: On dynamic multi-rigid-body contact problems with coulomb friction. ZAMM-Journal of Applied Mathematics and Mechanics 77(4), 267–279 (1997)