RSJ International Journal of Advanced Robotics, 21(14)1617-1634, 2007

Planning and Executing Navigation Among Movable Obstacles

Mike Stilman[1] Koichi Nishiwaki[2] Satoshi Kagami[2] James J. Kuffner[1,2]

[1] *The Robotics Institute*
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, USA
{*robot, jkuffner*}*@cmu.edu*

[2] *Digital Human Research Center*
National Institute of Advanced Industrial Science and Technology
2-41-6 Aomi, Koto-ku, Tokyo, Japan 135-0064
{*k.nishiwaki, s.kagami*}*@dh.aist.go.jp*

Abstract

This paper explores autonomous locomotion, reaching, grasping and manipulation for the domain of Navigation Among Movable Obstacles (NAMO). The robot perceives and constructs a model of an environment filled with various fixed and movable obstacles, and automatically plans a navigation strategy to reach a desired goal location. The planned strategy consists of a sequence of walking and compliant manipulation operations. It is executed by the robot with online feedback. We give an overview of our NAMO system, as well as provide details of the autonomous planning, online grasping and compliant hand positioning during dynamically-stable walking. Finally, we present results of a successful implementation running on the Humanoid Robot HRP-2.

keywords: NAMO, humanoid, manipulation, motion planning, movable obstacles

1 Introduction

Search and rescue robots, nursing home assistants, housekeeping and and other service robots that are confronted with changing human environments will be required to solve the problem of Navigation Among Movable Obstacles (NAMO). The robot must reposition obstacles in order to sucessfully navigate to a goal location. In this domain, conventional navigation planners fail because the navigation task goal lies outside the accessible configuration space. Figure 1 (a-c) illustrates an example in which the robot is trapped among objects and unable to navigate to its goal. This task not only requires the robot to plan its own motion, but also to reason about how to manipulate objects in the environment to alter the reachability of its goal.

Existing motion planners enable humanoid robots to traverse complex spaces [1–3], or manipulate objects from initial to goal configurations [4–6]. However, existing systems do not allow the robot to perceive its environment and select objects for manipulation when they interfere with the robot's task. In addition to the problems caused by uncertainty in perception, the NAMO domain poses a computationally challenging high-dimensional search space that is the product of the degrees of freedom for the robot and the objects it can move [7]. In this paper we present both algorithms and experimental results for a NAMO planner that utilizes dimensionality reduction to decouple global object selection and placement from fine manipulation planning. We demonstrate that our method generates fast solutions to NAMO problems that can be used in practical real-time applications.

For the experiments presented in this paper, we have restricted the movable obstacles to tables and chairs. These objects are on casters and the action space is rigid grasp manipulation. The humanoid robot HRP-2 is presented with a $25m^2$ area that contains these objects and is given the task of navigating from its initial configuration to a user specified goal. The required navigation is impossible without first manipulating at least one unspecified object in the scene.

2 Related Work

The task of planning in a modifiable environment was introduced by Wilfong [8] and further explored by Chen [9] as a heuristic search problem. Stilman [7] and Okada [10] proposed strategies for Navigation Among Movable Obstacles in the domain of humanoid robotics. Research on the related topic of *assembly planning* has largely focused on separating a collection of parts, typically ignoring the robot/manipulator. *Rearrangement planning* includes the robot, but specifies the final configurations of all objects. [11–13] Our work assigns a single navigation task and asks the robot to choose which objects to displace and where to move them.

There is growing literature on planning for NAMO tasks. Our previous work, [7] introduced partitioning the navigational free space and planning to restore connectivity through manipulation. [10] employed a task planner to construct a graph of obstacles and select candidate objects for manipulation. [14] and [15] described planning techniques for larger classes of NAMO problems that involve interaction between objects. [16] considered similar problems in 3D manipulation by articulated robots. This paper extends [7] and presents the first implementation of autonomous NAMO on a robot platform.

We have observed that humanoids operate primarily in two dimensional navigation spaces where they are hindered by large objects, such as tables and chairs, that can be manipulated by pushing and pulling. For such actions, Harada [4] introduced the notion of Generalized Zero Moment Point (GZMP) to reflect the postural stability of a humanoid with environment contact at the hands. This method was extended in [17] to an impedance control framework for consecutive walking and pushing motions. Yoshida [5] presented an alternative form of large object manipulation by pivoting. Takubo [6] has proposed a more general walking/pushing controller that allows stable pushing during the single support phase of walking.

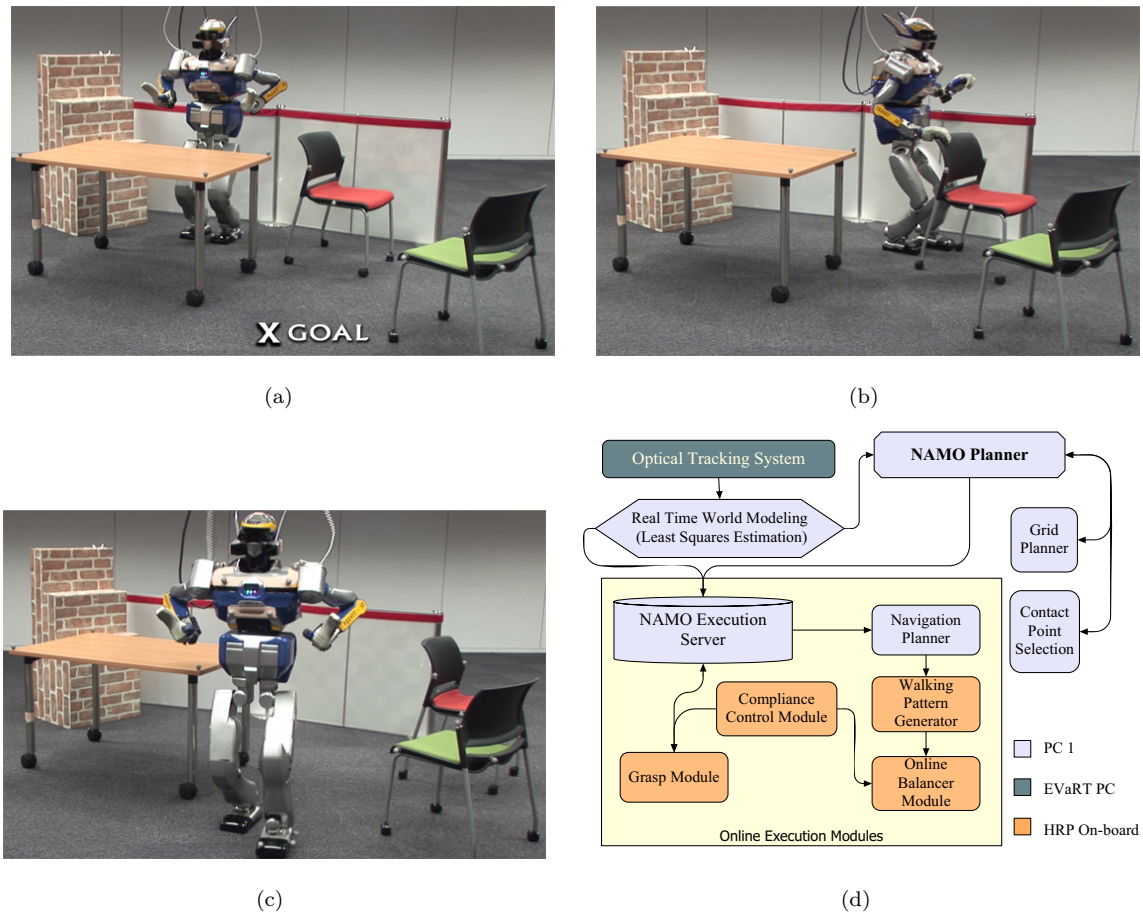


Figure 1: Still frames from the autonomous planning and execution performed by HRP-2. The robot selects the chair for motion and performs the required walking and manipulation.

Our work is complementary to these approaches. We localize world objects and construct manipulation actions based on geometric models. The focus of this paper is to address the unification of global planning and the elementary forms of fine manipulation such as robot placement, reaching, grasping and precise positioning while walking. These concepts lie between efficient high-level planning and stable low-level control. Our implementation builds upon the algorithms presented in [7] [18] [19] and [20].

3 Overview

Figure 1(d) illustrates the architecture of our complete NAMO system. EVaRT PC tracks optical markers attached to known objects using real-time external optical tracking. PC 1 acquires marker information by wireless ethernet and constructs a geometric world model. Section 4 details localization for the robot and environment objects. This section also describes the generation of contact points for planning interaction. PC 1 projects this model onto the walking surface and constructs a high level NAMO plan (Section 5).

Execution is handled by the NAMO Execution Server on PC 1. The robot iterates a sequence of three

actions: walking, grasping and manipulation. To compensate for positioning error, the server updates its world model from EVaRT prior to each action. It creates a path for each subtask and communicates the path to the HRP onboard computer. Section 6 describes how the onboard computer generates trajectories for robot joints to realize walking, grasping and position the robot's hands during walking manipulation. The onboard controller also maintains ZMP stability and compliant arm positioning during execution.

4 Perception

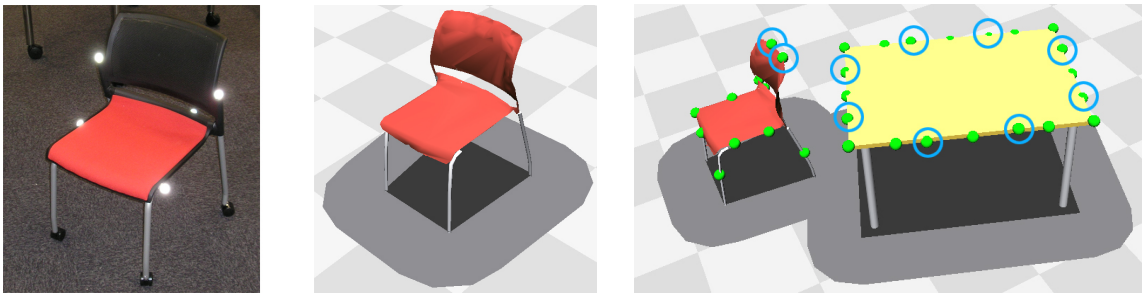
Throughout the execution of NAMO, the robot receives information from three sources: real-time external optical tracking, joint encoders and four six-axis force sensors. The force sensors at the feet and hands are discussed in Section 6. Global object pose estimates are continuously updated by means of passive marker-based optical tracking. Individual robot links are positioned by combining tracking of the robot torso with encoder readings for joint angles. We selected external optical tracking as a placeholder for future onboard perception systems in order to decouple the refinement of planning and control from environment sensing.

4.1 Object Mesh Modeling

The robot world model consists of the robot and two types of objects: movable and static. Static objects cannot be repositioned and must always be avoided. Approximate bounding box models are used to represent static objects. Movable objects may require robot manipulation and are represented internally by 3D mesh models. Our experimental setup contained two types of movable objects (chairs and tables). Prior models were constructed with the Minolta Vivid laser scanner. The resulting meshes were edited for holes and processed to minimize the overall polygon count while ensuring that at least one vertex exists in every $0.125m^3$ voxel of the mesh. This simplifies the problem of detecting object presence in any given region of space.

4.2 Model Recognition & Tracking

Precise localization of objects and model fitting was achieved using the EVa Real-Time Software (EVaRT) with the Eagle Motion Analysis optical tracking system. Each model was assigned a unique arrangement of retro-reflective markers. Under the assumption that the object is a rigid body, any six dimensional configuration of the object corresponds to a unique set of configurations for the markers. We define a *template* to be the set of x,y and z coordinates of each marker when the object's configuration matches that of the 3D mesh. EVaRT continuously tracks the locations of all the markers in a given scene. Matching the distances between markers, it partitions the markers among objects and provides our system with a set of marker locations for each object. The detected markers are rigidly transformed template markers that permit a linear relationship in the form of a transformation matrix.



(a) (b)

Figure 2: (a) Chair and 3D model used in our experiments with retro-reflective markers illuminated. (b) Automatically generated contact points. Circled points satisfy the conditions in Section IV, V. The dark and light ground regions represent planar obstacles and $\mathcal{C}$-space obstacles respectively.

Using only the visible markers we employ the orthogonal (LQ) decomposition to find the affine transformation between the template coordinates and the ones detected by EVaRT. This transform is optimal in minimizing the summed squared 3D marker error.

We refer to the collection of transformed meshes for the movable objects, static objects and the robot as the *world model*. Poses for individual robot links are found by combining joint encoder readings with the tracked position and orientation of the robot torso. The entire model is continuously updated at 30hz. Further details on our approach can be found in [21].

4.3 NAMO Configuration Space Generation

The world model is constructed in a 3D Euclidian space. However, for many navigation tasks the two dimensional sub-manifold that comprises the walking surface is sufficient. Note that larger objects such as tables and chairs that inhibit the robot's path must be pushed rather than lifted. [4, 5] Hence, their configurations are also restricted to a planar subspace. We project all objects onto the ground plane, and associate each object with the planar convex hull of the projected mesh vertices. Figure 2(a) shows a chair, its model and its projection onto the ground plane.

The resulting space does not allow interpenetration between objects. Alternative implementations could use multiple planes to allow penetration at distinct heights. Our approach is sufficient to demonstrate the feasibility of this work and provided reasonable assurance of both safety and plan validity.

The humanoid robot is represented by a vertical cylinder centered at the robot torso. A radius of $0.3m$ safely encloses torso motion. The projection of the cylinder is a circle on the ground plane. We pre-compute the navigational configuration space ($\mathcal{C}$-space) of the robot. Each $\mathcal{C}$ obstacle (O_i) is a Minkowski sum of the robot bounds with the corresponding planar object. For navigation, the robot can then be treated as a point that moves through the $\mathcal{C}$-space. The lighter ground regions around the projected obstacles in Figure 2 represent the $\mathcal{C}$ obstacles. The robot may walk on the lighter regions but its COM must not enter them.

To further decrease computation costs, the Bentley-Faust-Preparata (BFP) approximate convex hull

algorithm is used to compute $\mathcal{C}$ obstacles. [22] The resulting obstacles have significantly fewer vertices and edges while subject to only 1% error. Decreasing the number of edges reduces the cost of testing for robot collision. The error is acceptable since we can adjust the radius of robot safety bounds.

4.4 Automated Contact Point Selection

As part of NAMO planning, the robot must select locations for grasping movable objects. In our work, we have found three essential criteria for this selection:

1. *Proximity to object perimeter* - Ensure that the point is in the robot's workspace when standing next to the object.

2. *Restricted quantity* - Limit the number of possible interactions to a discrete set of grasp points to increase the efficiency of planning.

3. *Uniform dispersion* - Provide contact points for any proximal configuration of the robot.

We introduce one solution for locating such points. One can interpret the convex hull from the previous section as a representation of the object perimeter. Starting at an arbitrary vertex of this hull, our algorithm moves along the edges and places reference points at equidistant intervals. The interval is a parameter of the algorithm that we set to $0.2m$.

For each reference point, we find the closest vertex (by Euclidian distance) in the full 3D object mesh along the horizonal model axes. The selected vertices are restricted to lie in the vertical range of $[0.5m, 1.0m]$. When interpreted as contact points, we have found that these vertices satisfy the desired criteria for objects such as tables and chairs. Selected points can be seen in Figure 2(b). The robot successfully performed power grasps of our objects at the computed locations.

More complex objects may be widest outside the operating range or may have geometry that restricts the locations and orientations of valid grasps. These cases can be handled by applying a grasp planner such as GraspIt to determine valid contacts. [23] The proposed criteria can still be optimized by using proximity to the contour points as a heuristic for selecting grasps.

5 NAMO Planning

In Section 4 we described how the robot gains knowledge of its environment through both 3D and coarse planar models. Presently, our system constructs a high level NAMO plan using the planar model. Further refinement and error compensation is performed online with the more detailed 3D world model. We now discuss the high level planning and obstacle selection that occurs prior to execution.

5.1 High Level Planning

The top level planner, NAMO-Plan, is given the configurations of $\mathcal{C}$-space obstacles $W = \{q_1, q_2, ..., q_m\}$ and the position of the robot center of mass (r) (Figure 3). Conceptually, the algorithm partitions the

robot free space $\mathcal{C}_{free}$ into disjoint components or subsets $\{C_1, \ldots, C_n\}$. A free space component is a set of mutually reachable configurations. When the robot configuration r_1 is in C_i, any other configuration $r_2 \in C_i$ can be reached by a standard path planner. Locally, NAMO-PLAN searches over manipulation actions that move a single object to create a path between two disjoint free space components. Globally, the algorithm evaluates all choices for objects that connect any two neighboring free spaces. The global search finds a sequence of connections that give the robot access to the goal.

For efficient computation, we do not explicitly represent the free space components. Instead, we use a layered architecture of two planners. The high level planner searches over object motions that connect free space. Our heuristic for the high level planner is itself a simple planner, $\mathcal{P}$, implemented as a grid search. $\mathcal{P}$ identifies which objects should be manipulated first and locates points that represent disjoint regions of free space. $\mathcal{P}$ accomplishes these tasks by considering collisions as soft constraints rather than hard ones. Heuristic paths are allowed to collide with obstacles. They are ordered by a weighted sum of the number of obstacles they pass through and the Euclidian distance they cover. Each path generated by $\mathcal{P}(r, W, \ldots)$ is used to identify the first colliding object O_i that separates two sets of free points along the path. These points represent the space currently accessible to the robot, $Acc(r)$, and a disjoint component, C_j, further along the path.

After $\mathcal{P}$ selects an object, MANIP-SEARCH performs best-first search to enumerates valid, accessible grasps and potential placements, q_i', for the object. MANIP-SEARCH evaluates placements by locally searching the action space of the robot for manipulation paths in order of minimal work. The manipulation plan that displaces O_i and creates space collision-free navigation between $Acc(r)$ and C_j is returned.

We implemented NAMO-PLAN as a depth-first search to conserve space for inner calls to MANIP-SEARCH and $\mathcal{P}$. Each branch of the tree keeps track of the modified world state after a manipulation action. *AvoidList* incrementally restricts $\mathcal{P}$ from planning through pairs (O_i, C_j) in cases where O_i cannot be successfully manipulated to give the robot access to C_j. The planner backtracks locally when MANIP-SEARCH fails. It backtracks globally when all heuristic paths to the goal pass through some member of *AvoidList*.

Figure 3 details the structure of the planner in pseudo-code. Asymptotically this planner searches over all possible choices for sequences of free space components and objects that connect them. Further detail and a proof of resolution completeness for the L_1 subclass of NAMO problem is given in [7].

5.2 Action Space Details

Our planning system distinguishes large objects, such as tables, from smaller objects such as chairs. Smaller objects have less inertia and can be manipulated safely with one arm. Tables, however, have a significant impact on the robot dynamics and therefore dual arm manipulation is applied.

```
NAMO-PLAN(r, W)
 1  AvoidList ← ∅
 2  PartialPlan ← ∅
 3  while (O_i, C_j) ← P(r, W, AvoidList) ≠ NIL
 4  do
 5     if C_j = GOAL
 6        then return (PartialPlan append GOAL)
 7     (ManipPlan, r', W') ← MANIP-SEARCH(Acc(r), C_j, O_i)
 8     if ManipPlan ≠ NIL
 9        then PartialPlan append ManipPlan
10             Plan ← NAMO-PLAN(r', W')
11             if Plan ≠ NIL
12                then return (PartialPlan append Plan)
13        else  AvoidList append (O_i, C_j)
14  return NIL
```

Figure 3: Pseudocode for the global NAMO planning algorithm.

5.2.1 Smaller Object Manipulation

The robot grasps the object and can then translate itself in any of eight directions while keeping the object at a fixed angle and distance from the torso. Prior to the manipulation the robot aligns itself in the desired direction of motion. The resulting translation is always a walk forward or backward.

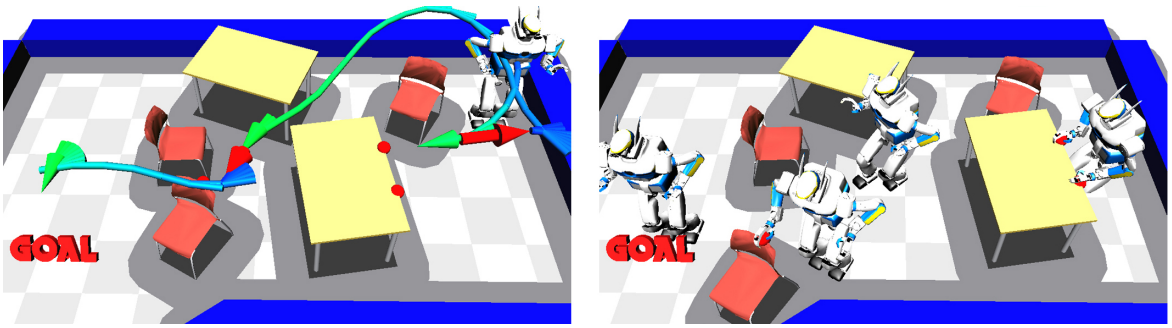
5.2.2 Larger Object Manipulation

Since the robot grasps the object with both hands we require two contact points. Generally, this implies that the robot must be aligned in a direction perpendicular to the object to gain sufficient manipulability in both arms. Consequently, the robot is permitted to translate forward and backward only in the two directions perpendicular to the line containing both contact points. In order to minimize the torques caused by object inertia, the robot is further restricted to keeping the object center of mass along the line of action.

6 Online Execution

Using the tools from the previous sections, the robot has observed the world and constructed a high-level NAMO plan for walking up to some object, moving it, and continuing to either move more objects or make its way to the goal. We now discuss how HRP-2 executed each subtask.

The execution modules interpret NAMO plans by constructing path plans, computing workspace trajectories and resolving joint trajectories. Path plans ensure collision free motion. Workspace trajectories for feet, hands and torso are optimized for stability. Finally, joint trajectories are determined by



(a) (b)

Figure 4: Simulated NAMO problem. (a) Initial state and the automatically computed NAMO plan. Arrows indicate the alternating navigation and manipulation paths. (b) The final state with traces of execution. HRP-2 pulls the table and pushes the chair prior to reaching the goal. Notice that the free space is connected.

resolved motion rate control. [24] This control method is also applied to handling minor disturbances through compliance.

6.1 Walking

Humanoid navigation consists of walking. In our domain, the robot must create a collision free path for the walk, identify foot placements and construct trajectories for the legs. The torso trajectory is optimized to fix the placement of the robot Zero Moment Point (ZMP) at the center of the respective stance foot.

6.1.1 Online Planning

The high-level NAMO planner does not explicitly yield desired walking paths for the robot. Its internal grid planner only guarantees that such a plan is possible due to the existence of free $\mathcal{C}$-space.

To create smooth walking plans, we have implemented a separate *navigation planner*. Presently, our largest concern is torso collisions, hence we first plan the torso trajectory using the $\mathcal{C}$-space world model. From any torso configuration, the robot is permitted 41 discrete actions which are tested for collisions with the $\mathcal{C}$-space obstacles.

- (1) Translate backward ($0.1m$)
- (20) Rotate in place (range of 30^o)
- (20) Translate forward ($0.2m$) & rotate (range of 30^o)

In the planar $\mathcal{C}$-space a planner could exhaustively search the space with only two actions: forward translation and rotation. Instead, we choose actions for the torso that directly translate into common footstep gaits. Placing the desired foot locations at a horizontal distance of $.09m$ from the torso along a line orthogonal to the torso trajectory yields repeatable, safe and predictable robot motion. Smoothing of the trajectory is unnecessary due to the discrete nature of humanoid walking.

The *navigation planner* is executed online prior to any walk. Due to errors in robot odometry, both the robot and the objects it moves may not be in the exact locations that are expected from previous NAMO steps. To compensate for these errors, we reacquire the world model from the optical tracker during execution and plan in the updated state.

6.1.2 Walking Execution

Our walking execution method consists of an offline Walking Pattern Generator (WPG) coupled with an online balancer as presented in [18]. Leg trajectories are calculated offline along with initial modifications to the torso trajectory according to the preview control of the ZMP balancing method [19]. Online, the torso trajectory is further modified to compensate for ZMP errors in execution as detected by the force/torque sensors at the robot's feet.

6.2 Grasping

Having walked up to an object, the robot must grasp it and then walk with it. HRP-2 reacquires the world model and determines the workspace position for its hand. It preshapes the hand to lie close to this position and then performs a final adjustment to compensate for any perception error. This process is not only functionally successful but also approximates human grasping behavior as described in [25]. Grasping is a complex problem and has been studied in [23] among others. The presented approach is simple and effective for our limited set of objects.

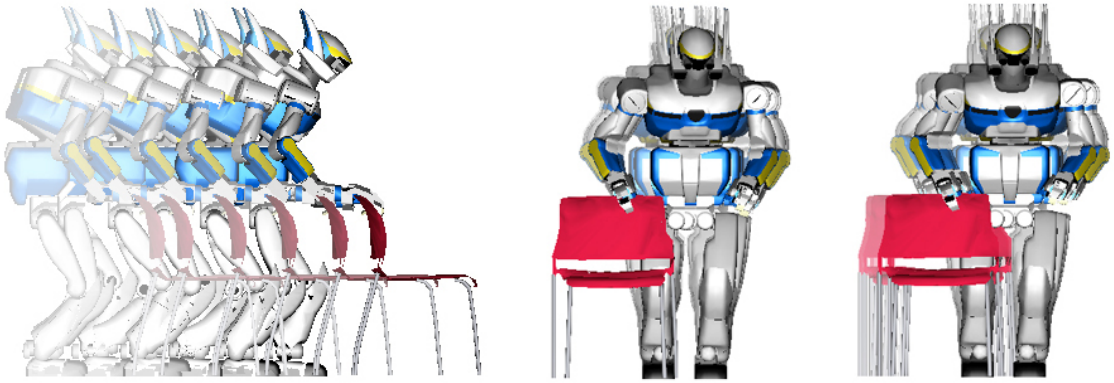
6.2.1 Preshaping and Adjustment

The initial workspace hand position for grasping is obtained by selecting a point $.05m$ above the object and $.05m$ closer to the robot (in the direction of robot motion). The Jacobian pseudo-inverse Inverse Kinematics (IK) technique [26] is used to select the arm and torso joint angles that correspond to this position. The arm trajectory is obtained by cubic spline interpolation of the initial and goal joint angles. Future work should consider planning the motion of the arm to avoid obstacle contact.

Having attained the initial grasp position for the hand, the robot moves its hand downward and then forward to close the $.05m$ gaps. At this time, the hand force sensor is used to detect contact and prevent the arm from pushing past the object surface. The two motions are generated by online application of Jacobian pseudo-inverse IK. Adjustment ensures that the robot's palm is in contact with the top surface of the object and the thumb is in contact with the outer edge. The remaining fingers are closed in a power grasp until the finger strain gauges exceed the desired threshold.

6.3 Walking and Manipulation

Walking while fixed to a large object requires additional considerations to those discussed in Section 6.1 Since the torso trajectory is modified for balance compensation, we would expect the arm and hence the object to also move as seen in Figure 5 (c). This motion leads to two significant concerns:



(a) (b) (c)

Figure 5: Orthographic projections of kinematic simulations for a dynamically stable walking gait. (a,b) are time-sequences of walks with defined arm trajectory. Notice the chair motion in (c) with no arm positioning.

- *Small safety margin for objects* - Unlike torso motion, we have no bounding cylinder when planning object motion. Introducing significant margins would prevent our planner from finding solutions in cluttered environments.

- *Impact on robot dynamics* - The torso is moved to compensate for error in the ZMP. Additional motion of an attached object without precise estimation of the dynamic impact leads to unpredictable effects on the walking gait.

Our solution is most similar to [20]. We assign workspace manipulator trajectories and iteratively compute joint angle solutions to both trajectory and balance constraints.

6.3.1 Arm Positioning During Walking

The walking pattern generator is augmented with a mode for fixed hand walking. The robot hands are assigned to fixed workspace trajectories. In our work these trajectories follow the initial torso trajectory translated to begin at the initial arm locations. This workspace path imposes that the object follow the trajectory computed by the NAMO planner. Trajectory nodes are interpolated by cubic splines. As the WPG iteratively modifies the torso trajectory, the arm angles, torso pitch and yaw are recomputed such that the hand remains fixed to the initial workspace path. This calculation is performed by applying pseudo-inverse IK to the chain of joints. Figure 5 (a,b) shows the corrected motion and Figure 6 gives trajectories for two arm angles that yield the desired compensation.

For one handed manipulation, the above computation is applied solely to the torso joints and the right arm. In two handed manipulation, the arms are solved sequentially. The right arm angles are solved first including the torso pitch joint. The left arm angles are solved in the updated torso configuration.

Future work in this direction may involve a more general prioritized framework for whole body motion such as [27]. One challenge is that all three tasks: balance control and each arm motion must be

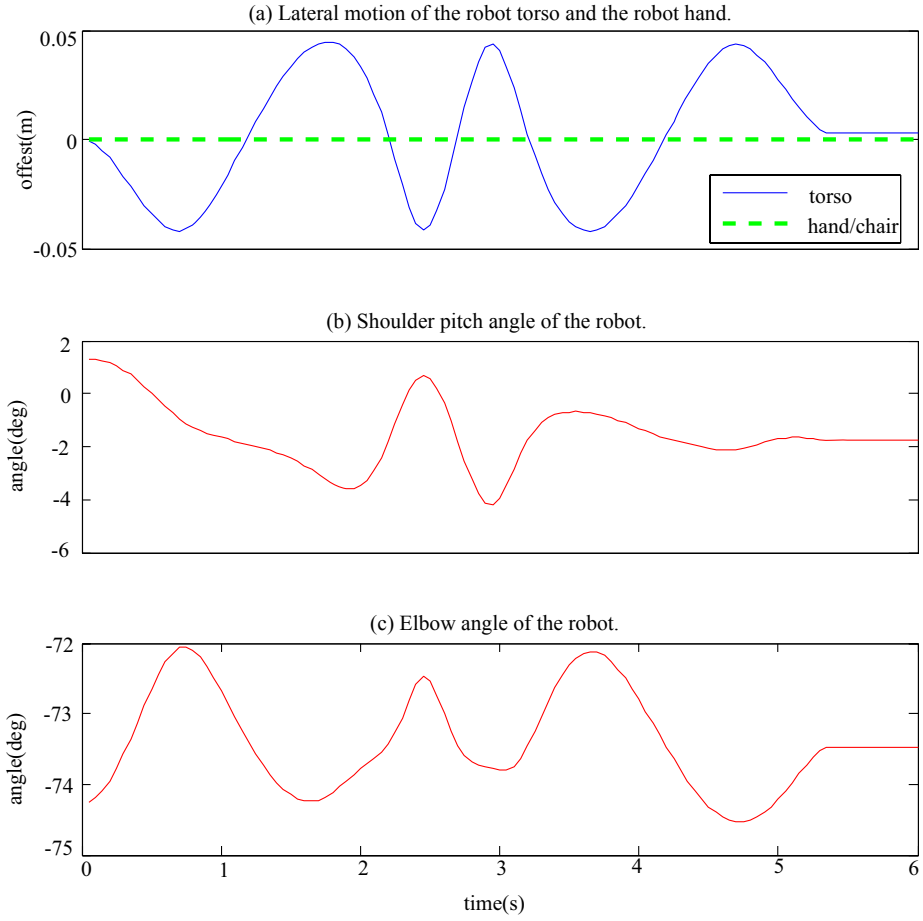


Figure 6: Traces of the planned robot motion for chair manipulation. (a) The torso moves laterally to create a dynamically stable motion according to the ZMP criterion. (b, c) Arm and chest angle trajectories are calculated to compensate for torso motion. Notice that the forward velocity is gradually increased and decreased. Also, observe the effect of drift in IK calculation.

executed successfully to ensure the safety of the robot. Lookahead planning strategies may be necessary to guarantee that the controller will not violate the task constraints.

6.3.2 Online Arm Positioning

The same process of IK calculation is repeated online when the torso position is modified by the balancer module. During execution, the torso roll and yaw are fixed to the ones decided by the WPG and only the arm angles are used to compensate the torso motion. This distinction is made to eliminate dynamic error caused by significant variations in the robot posture.

6.4 Active Compliance Control

Although fixing a workspace arm trajectory inhibits large inertial effects from the manipulated object, we must recognize that the object is still in contact with the environment. As a result, some vectors of

force applied to the object are directly reflected to the rigid HRP-2. These forces are not accounted for during pattern generation or balancing.

Compliance control is performed using six axis force sensors at the wrists and implementing a linear spring-damper at the robot hands. In the following equations, x and x_d are the current and desired position vectors respectively. k and c are gains and F is the virtual force resulting from the system.

$$F(t) = k(\mathbf{x}_d(t) - \mathbf{x}(t)) + c(\dot{\mathbf{x}}_d(t) - \dot{\mathbf{x}}(t))$$

In the case of the position controlled HRP-2, we do not directly command acceleration or velocity. Hence, we make the assumption that the commanded velocity is realized by the robot. Consequently, the system can be represented as follows:

$$\dot{\mathbf{x}}_d(t) = \frac{1}{c}(F(t) - k(\mathbf{x}_d(t) - \mathbf{x}(t)))$$

$$\mathbf{x}_d(t + \Delta t) = \mathbf{x}(t) + \Delta t \dot{\mathbf{x}}_d(t)$$

If $\mathbf{x}_d$ is initialized at the origin, we can think of this model as providing an offset for the hand position that results from a time sequence of forces experienced at the hand. This offset is added to the desired hand position prior to IK calculation during both grasping and object manipulation.

Although large damping coefficients are present for all directions of motion, the spring coefficient for the vertical direction is left small, 200 as compared to 1000 along the planar axes. (Forces are represented in N and distance in m). This is to prevent the robot from pushing down on the object, thereby compensating for much of the force reflected from the ground.

6.5 Further Considerations in Grasping

Sections 6.3 and 6.4 emphasize that subsequent to grasping, the robot is required to modify the position of its hand during walking to maintain a compliant trajectory. Since the modifications are described as workspace displacements we require IK to find joint angle solutions for these motions. However, particularly for gradient-based IK methods, it is difficult to ensure that joint limits and singularities do not interfere with the computation.

One interesting direction of research is to find *optimized grasp regions*, or workspace hand positions in the robot torso frame that provide some guaranteed bounds for the existence of IK solutions within a volume of hand motion. In our present implementation, we manually selected grasp regions to maximize the minimum distance from joint limits. Assuming knowledge of optimized grasps, the robot then takes action to ensure that the grasp occurs within the desired region. We propose two strategies for effecting this adjustment.

6.5.1 Torso Positioning

The grasp points in our work are generated by the NAMO planner. One strategy for optimizing grasp is to adjust the torso position. In other words, we add the two DOF horizontal torso position to the

Table 1: Planning and execution times.

	NAMO	Navigation	Execution
Figure 1(a) (Real)	$0.06s$	$0.09s$	$63.0s$
Figure 4 (Simulated)	$0.13s$	$0.93s$	$153.0s$

kinematic chain for grasping. This was the method applied in our work. After reaching the object and ascertaining the global position of itself and the obstacle, the robot takes two steps to adjust the torso position with respect to the contact point.

6.5.2 Micro and Macro Manipulation

The converse of our approach is to manipulate the object itself such that the final location of the contact point is within an optimized grasp region. This micro manipulation can be performed prior to the macro walking manipulation. The incentive for this is that locally, the robot can be analyzed in quasi-static balance. The planned manipulation may even be done in joint space, such as [28], to avoid IK considerations. As with torso positioning, micro manipulation prepares the world state for a successful macro interaction.

7 Results

The NAMO system described in this paper was successfully applied to a number of simulated examples such as the one presented in Figure 4 as well as the real robot control problem in Figure 1. The simulated evaluations involved all planning aspects of our work, including dynamically stable walking pattern generation for free arm and fixed arm walking gaits. We considered worlds containing up to 10 movable obstacles.

In our laboratory experiments, the robot perceived its environment and autonomously constructed a plan for reaching the goal. After turning and approaching the chair, HRP-2 successfully grasped the object, walked to displace it and then navigated to the final configuration.

Table 1 details the high-level NAMO planning time, replanning time for navigation and the total execution time for the two presented examples. Notice that the NAMO planning time is three orders of magnitude smaller than the actual time for executing the computed motion. This makes it feasible to view the NAMO system as a real-time generalization of modern path planning techniques to worlds where strictly collision-free paths are not available.

8 Discussion

In this paper we have presented the first implementation of NAMO planning on a real robot. In particular, we considered the case of humanoid robots due to their high capacity for both navigation

and dexterous manipulation. The architecture and many techniques presented in this work generalize to mobile manipulators other than HRP-2.

We view NAMO as a baseline task for environment manipulation where the robot must not only perform a given task but reason about subtasks and interact with unspecified objects. The lack of specification motivates the development of more general and robust algorithms for object interaction.

Future work should consider grasps that handle significant uncertainty as well as whole body IK methods that can be applied safely over the entire robot workspace. Optimizing grasp regions for intended object interaction is perhaps one of the most interesting and challenging problems. From the perspective of geometric planning, it will be interesting to study algorithms that offer completeness for larger classes of NAMO problems.

At present our implementation handles uncertainty in action through compliance and replanning of paths. These methods are suitable for domains with low modeling error that does not change the overall NAMO plan. In domains with greater model uncertainty the robot may need to respond to unexpected collisions as well as change its decisions about which objects to manipulate. Further integration of NAMO replanning and execution should prove to be of great interest in constructing complex autonomous behaviors by humanoid robots.

9 Acknowledgements

We thank J. Chestnutt and P. Michel for their collaboration in developing the external tracking system, and C.G. Atkeson for his support and insight. This research was partially supported by NSF grants DGE-0333420, ECS-0325383, ECS-0326095, and ANI-0224419.

REFERENCES

[1] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami. Planning biped navigation strategies in complex environments. In *2003 International Conference on Humanoid Robots*, 2003.

[2] A. Wolf, H. Ben Brown Jr., R. Casciola, A. Costa, M. Schwerin, E. Shammas, and H. Choset. A mobile hyper redundant mechanism for search and rescue task. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 2889–2895, 2003.

[3] J. Gutmann, M. Fukuchi, and M. Fujita. Real-time path planning for humanoid robot navigation. In *International Joint Conference on Artificial Intelligence*, 2005.

[4] K. Harada, S. Kajita, K. Kaneko, and H. Hirukawa. Pushing manipulation by humanoid considering two-kinds of zmps. In *IEEE Int. Conf. on Robotics and Automation*, pages 1627–1632, 2003.

[5] E. Yoshida, P. Blazevic, and V. Hugel. Pivoting manipulation of a large object. In *IEEE Int. Conf. on Robotics and Automation*, pages 1052–1057, 2005.

[6] T. Takubo, K. Inoue, and T. Arai. Pushing an object considering the hand reflect forces by humanoid robot in dynamic walking. In *IEEE Int. Conf. on Robotics and Automation*, pages 1718–1723, 2005.

[7] M. Stilman and J.J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. In *Proc. IEEE Int. Conf. on Humanoid Robotics (Humanoids'04)*, 2004.

[8] G. Wilfong. Motion panning in the presence of movable obstacles. In *Proc. ACM Symp. Computat. Geometry*, pages 279–288, 1988.

[9] P.C.Chen and Y.K.Hwang. Pracitcal path planning among movable obstacles. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 444–449, 1991.

[10] K. Okada, A. Haneda, H. Nakai, M. Inaba, and H. Inoue. Environment manipulation planner for humaonid robots using task graph that generates action sequence. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, pages 1174–1179, 2004.

[11] R. Alami, J.P. Laumond, and T. Sim'eon. Two manipulation planning algorithms. In *Workshop on the Algorithmic Foundations of Robotics*, 1994.

[12] O. Ben-Shahar and E. Rivlin. Practical pushing planning for rearrangement tasks. *IEEE Trans. on Robotics and Automation*, 14(4):549–565, 1998.

[13] J. Ota. Rearrangement of multiple movable objects. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1962–1967, 2004.

[14] M. Stilman and J. Kuffner. Planning among movable obstacles with artificial constraints. In *Workshop on the Algorithmic Foundations of Robotics*, 2006.

[15] D. Nieuwenhuisen, A.F. van der Stappen, and M.H. Overmars. An effective framework for path planning amidst movable obstacles. In *Workshop on the Algorithmic Foundations of Robotics*, 2006.

[16] M. Stilman, J. Schamburek, J. Kuffner, and T. Asfour. Manipulation planning among movable obstacles. In *IEEE Int'l Conference on Robotics and Automation (ICRA'07)*, 2007.

[17] K. Harada, S. Kajita, F. Kanehiro, K.Fujiwara, K. Kaneko, K.Yokoi, and H. Hirukawa. Real-time planning of humanoid robot's gait for force controlled manipulation. In *IEEE Int. Conf. on Robotics and Automation*, pages 616–622, 2004.

[18] K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue. Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired zmp. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2684–2689, 2002.

[19] Shuuji Kajita et. al. Biped walking pattern generation by using preview control of zero-moment point. In *IEEE Int. Conf. on Robotics and Automation*, pages 1620–1626, 2003.

[20] Y. Fukumoto, K. Nishiwaki, M. Inaba, and H. Inoue. Hand-centered whole-body motion control for a humanoid robot. In *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 1186–1191, 2004.

[21] M. Stilman, P. Michel, J. Chestnutt, K. Nishiwaki, S. Kagami, and J. Kuffner. Augmented reality for robot development and experimentation. Technical Report CMU-RI-TR-05-55, Robotics Institute, Carnegie Mellon University, November 2005.

[22] J. Bentley, G.M. Faust, and F. Preparata. Approximation algorithms for convex hulls. *Comm. of the ACM*, 25(1):64–68, 1982.

[23] Andrew T. Miller. *GraspIt: A Versatile Simulator for Robotic Grasping*. PhD thesis, Dept. of Computer Science, Columbia University, 2001.

[24] D.E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man Machine Systems*, 10:47–53, 1969.

[25] M. Jeannerod, M.A. Arbib, G. Rizzolatti, and H. Sakata. Grasping objects: the cortical mechanisms of visuomotor transformation. *Trends Neurosci.*, 18:314–320, 1995.

[26] Lorenzo Sciavicco and Bruno Siciliano. *Modeling and Control of Robot Manipulators*. 1996.

[27] O. Khatib, L. Sentis, J. Park, and J. Warren. Whole body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, pages 29–44, 2004.

[28] J.J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots under obstacle and dynamic balance constraints. In *IEEE Int'l Conf. on Robotics and Automation (ICRA'01)*, pages 692–698, 2001.