In *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*
Santa Monica, CA, Nov. 2004

Towards Single-Arm Reaching for Humanoids in Dynamic Environments

Evan Drumwright, Marcelo Kallmann, Maja Matarić

USC Robotics Research Labs
Interaction Lab
University of Southern California
941 West 37th Place, Henry Salvatori Building, Room 230
Los Angeles, California 90089-0781, USA
{*drumwrig|kallmann|mataric*}*@usc.edu*

The problem of humanoid agents and robots reaching to arbitrary targets in environments with static and dynamic obstacles has not yet been investigated in detail. Typical approaches include using randomized motion planning or constructing uninformed trajectories (e.g., linear interpolation between initial and target positions and orientations of the hand) in operational space, in hopes that inter-link and agent-environment collisions do not occur. In this paper, we test the most popular algorithms for motion generation for single-arm reaching in environments with randomly placed obstacles of random sizes. Additionally, we attempt to formalize the concept of *motor primitives*, and test a motor primitive implementation in the same experiments. We conduct an analysis of the efficacy of the algorithms for reaching in static environments, and discuss the extensibility of the algorithms towards reaching in dynamic environments.

Keywords: reaching; motion planning; trajectory formation; motor primitives

1. Introduction

Humanoid robots and agents are intended to inhabit our surroundings and thus must be able to perform the quintessential humanoid activity of reaching in dynamic, obstacle-ridden environments. Sampling-based algorithms have recently enjoyed success in planning motions for agents with few degrees-of-freedom (DOF) in dynamic environments and relatively many DOF in static environments. No research has effectively addressed the problem of motion planning on agents with many DOF, such as humanoids, in dynamic environments. In this paper, we test several kinds of motion generating algorithms, including *motor primitives*[1,2] and rapidly-exploring random trees (RRT)[3], on single-arm reaching tasks in environments with randomly-placed obstacles, in the hope of determining which algorithms might be most successful in environments with dynamic obstacles.

A formal statement of the problem that we are attempting to solve follows. *Given an initial posture of the robot or agent and the configuration of obstacles in the environment, which algorithm is generally most successful at generating joint-space trajectories that result in the hand reaching a target position and orientation?*

Success is determined not only by achieving the goal task, but also by avoiding self-collisions and collisions with obstacles in the environment.

The problem as stated assumes only static obstacles, which, alone, portray the task incompletely. However, it is hoped that the full problem (i.e., including dynamic obstacles) shares enough properties with the problem discussed in this paper that the results discovered here are applicable to both. Additionally, this paper examines the extensibility of the dominant algorithms for generating joint-angle trajectories, all of which operate intrinsically in kinematic spaces, to the problem of reaching in dynamic environments, which must naturally take robot and obstacle dynamics into account.

This paper does not attempt to compare running times between the various algorithms. Not only is such a comparison very difficult given the probabilistic nature of the RRT algorithm, but such comparisons have little meaning without definitive lower bounds on the running times of the various algorithms. For example, all but the RRT algorithm rely on matrix inversion, for which a tight lower bound has yet to be found. It may also be possible to adjust the subcomponents of the algorithms to improve running times, thereby invalidating any such comparison.

One other caveat must be mentioned. The algorithms examined in this paper do not attempt to utilize locomotion to perform reaching. While it can be posited that reaching is best performed by simultaneously accounting for locomotion within the algorithms, we have chosen to decouple the problems for the sake of simplicity. For this reason, all reaching targets are to local (i.e., no locomotion required) targets.

An outline of the paper follows. Section 2 discusses relevant background research, encompassing planning algorithms and motor primitives. Section 3 attempts to formalize the concept of a motor primitive. Section 4 details the experimental setup. Section 5 lists the results from the experiments. Section 6 discusses the results that were obtained.

2. Background

Humanoid reaching and grasping has recently begun receiving attention from the computer graphics community. Kuffner[4] began using probabilistic motion planners with kinematically simulated humanoids to reach in static environments; however, reaching is just one component in his general system for generating human-like motion. Kallmann et al.[5] use a collection of techniques, including RRTs, locomotion, and roadmap restructuring for more robust and naturally appearing reaching; however, their methods are still restricted to static environments.

2.1. *Combinatorial motion planning*

Lindemann and LaValle[6] note that, while obstacles in the workspace are well-defined, invalid configurations in C-space are more difficult to represent. Canny's roadmap algorithm[7] was able to solve the motion-planning problem for static environments in running time exponential in the degrees-of-freedom of the robot.

Lindemann and Lavalle[6] note that this algorithm employs techniques which are very difficult to implement correctly. Worse, the running time of this algorithm grows very rapidly with the number of geometric primitives used to represent the robot and the obstacles. Combinatorial motion planning algorithms are useful in highly limited cases, where the number of DOF of the robot is quite small and the geometry of the robot and the environment are both very simple. Sharir[8] discusses worst-case complexity bounds on motion planning algorithms in more detail. Generally, combinatorial algorithms are intractable for reaching applications with humanoid robots, and are not tested in this paper for that reason.

2.2. *Sampling-based motion planning*

Sampling-based methods are general algorithms applicable to control problems of arbitrary dimension. They can be divided into two main categories: multi-query and single-query. Multi-query methods build a Probabilistic Roadmap (PRM)[9] that can be used for several different queries in a single static environment. The basic procedure consists of randomly sampling the configuration space, creating nodes when samples are valid, and connecting pairs of nodes each time the connection is tested to be valid and the nodes are considered to be close enough. Several variations to the basic PRM approach have been proposed[10,11,12]. A good overview and comparison is given by Geraerts and Overmars[13].

Single-query methods are typically used when the environment is dynamic. Roadmaps are built specifically for each query, but trees are used instead of graphs to increase efficiency. The Rapidly-exploring Random Tree (RRT)[14,15] is a popular single-query method. The basic premise behind it is to expand nodes of the tree toward random samples until the goal configuration is reached. Another effective method is based on Expansive Spaces Trees[16], where nodes in low-density locations in the configuration space are locally expanded. An efficient bi-directional version[17] incorporating lazy collision detection[12] is also available.

In the experiments presented in this paper, we evaluate the use of a RRT[14,15] single-query planner for deriving collision-free reaching motions. The RRT planner is currently considered to be one of the most effective methods for planning in static environments.

2.3. *Motion planning in dynamic environments*

Motion planning in dynamic environments must handle the added constraints of time, velocity, and control dynamics. Additionally, the act of planning may itself cause solutions to be lost, if search is not pursued in the correct order. A seminal result by Reif and Sharir[18] proved, that for the simplest case of planning, the motion of a point-robot moving with unbounded velocity among moving obstacles, motion planning is NP-hard. If the velocity of the robot is bounded, the problem is PSPACE-hard. The velocities of humanoid robots and simulated agents are uneasily bounded due to dynamics; the velocity bounds in one posture are not necessarily

equal to the velocity bounds in another. Finally, this discussion assumes that the obstacle trajectories are known in advance, a considerable requirement.

Current planners for dynamic environments scale very poorly; robots are limited to around ten DOF. Both LaValle and Kuffner[19] and Hsu et al.[20] have explored "randomized kinodynamic planning", which plans by feeding random inputs into the controls of a robot and integrating its equations of motion. While these methods can produce valid plans (i.e., a plan cannot be generated that the robot is physically incapable of executing), they complicate the planning process with the addition of the state space and control space.

If robot dynamics can be ignored, dynamic roadmaps can be constructed that are updated only as the environment changes, as researched by Kallmann and Matarić [21] and Leven and Hutchinson [22]. In particular, Kallmann and Matarić conducted extensive experiments pitting the dynamically recomputed roadmaps versus RRTs. They found that the dynamic computation was not much faster, especially for 3D-situated robots, than complete recomputation using RRTs. The current running time of RRTs makes that situation intractable.

2.4. *Motor primitives*

Suppose that there is a language, in the mathematical sense of the word, that describes human movement. Such a language could be used both to generate and recognize human-like movement. The constructs in this language can then be labeled *motor primitives*. Researchers from the fields of neuroscience, biology, robotics, computer graphics, and machine vision are actively pursuing the search for motor primitives, also known as *basis behaviors*, *convergent force fields*, *movemes*, and *motion graphs*. The initial inspiration for the motor primitive hypothesis came from Johannson's experiments[23] with moving lights attached to joints of the human body. The experiments showed that humans are able to recognize human movement from very little data, namely video that shows only the moving lights. Furthremore, the neuroscience community has discovered mirror neurons in monkeys[24] and humans[25,26], which fire both when the animal is performing an action and seeing that action performed, coupling action perception and generation. Additionally, Giszter et al.[27] were able to generate naturally appearing movements in spinalized frogs via microstimulation applied to the spinal cords. By directing the application of the stimulation, different movements were produced. The authors noted that the movements, which acted as convergent force fields, could form building blocks for more complex behaviors.

A complete survey of various versions of motor primitives is beyond the scope of this paper. Instead, the use of motor primitives within the field of robotics is discussed. Roboticists have experimented with many representations of motor primitives, in both kinematic and motor command spaces. Hodgins and Wooten[28] developed sets of primitive controllers and finite-state machines for switching between them in order to control simulated robots performing complex athletic tasks.

Matarić et al.[29] discuss the effectiveness of three dynamic control systems for a simulated humanoid performing a dancing task. Faloutsos et al.[30] use sets of controllers, each more complex than Hodgins and Wooten's, with a priority-based arbitration scheme and pre and post-conditions to animate virtual stuntmen.

For control in kinematic spaces, Fod et al.[31] and Jenkins and Matarić[2] cluster motion-capture data into "behaviors"; the original motion can be reconstituted or novel motion can be formed via interpolation between clusters. Drumwright and Matarić[1] use a motion-interpolation system on manually choreographed movements. Ijspeert et al.[?] utilize nonlinear dynamical systems for trajectory formation that is used to drive the movements of a humanoid robot.

3. Motor primitives

The previous section discussed the background of motor primitives, but did not attempt to unify the representations used by various researchers. Such a quest appears challenging, given the extreme differences mong various kinematic space and motor command space representations that have been employed. Therefore, this paper attempts to unify the various kinematic representations such that our primitive implementation is indicative of the level of performance achievable by motor primitives for *reaching*. The kinematic representation is chosen over the motor command representation because the former allows for workspace independent planning [32] and for agent-independent planning.

A redefinition of motor primitive for the purposes of this paper is as follows: *A motor primitive is a mechanism for generating joint-space or operational space trajectories bounded by "polynomial time", given the current state and kinematic goal(s) of the robot and the state of the environment.* Polynomial time is designated as $O(i^x j^y)$, where i is the number of DOF of the agent, j is the number of obstacles in the environment, and x and y are some (hopefully small) constants. A primitive is capable of generating an uncountably infinite number of trajectories (i.e., it is a member of Cantor's infinity class $\aleph_2$ of all curved shapes).

A primitive can be formally written as:

$$\pi(s, g) \rightarrow f(t)$$

where $\pi(.)$ is the primitive, s is the concatenated (i.e., environment and robot) state vector, g is a vector of kinematic goals, and $f(t)$ is the trajectory output.

A primitive can also be implemented as a dynamical system, as in Schaal's Dynamic Motor Primitives[32]. In that case, the functionally equivalent formal definition becomes:

$$\dot{f} = \psi(s, g)$$

Trajectories can then be formed by integrating f. $\psi(.)$ is a function that determines a gradient given a current state and goal.

The above definition is necessarily general. It must account for past and future research in motor primitives. It also theoretically empowers primitives (even a single primitive) to produce the entire class of human movement. Some existing mechanisms, such as the spline-based trajectory formation algorithm, are already included in the definition. Other methods, such as the combinatorial motion planning algorithms, exist outside of this class by virtue of computational complexity. Sampling-based planners are not included in the motor primitive class, even when a finite number of samples are used, due to the potentially unbounded time required to obtain a sample.

4. Experiments

Recall that we are attempting to determine which algorithm is most successful at generating collision-free reaches from an initial posture to a target position and orientation. Our experiments consisted of 16,504 randomly generated environments, each consisting of between 0 and 25 cubes of randomly generated size and position. Given enough cubes of varying size, it is possible to approximate any environment, leading to elimination of *a priori* knowledge from the experiments. For each environment, an initial posture of the humanoid was randomly generated, and a target position and orientation for the hand were randomly generated as well. The initial posture consisted of randomly sampled values (within joint limits) for both arms (7 DOF), the lower back (3 DOF), and the neck (3 DOF). Only 10 DOF of the agent were active in the experiments - 7 DOF in the arm and 3 DOF in the lower back. All random values in the experiments were drawn from uniform distributions and generated by a pseudo-random number generator.

The four algorithms discussed below were each tested in every environment. Success or failure and manner of failure (if any) were recorded. Every algorithm attempted to generate a joint-angle trajectory from the initial posture to any posture with the hand at the target position and orientation. Small deviations from the target were deemed acceptable– the norm of the positional error was as high as 2.54cm, and the orientation error up to 10 degrees– but collisions of the humanoid with itself or external objects resulted in failure.

An analysis of the performance of each algorithm was made after the experiments were compleated. It is desirable to determine whether certain algorithms perform better in chosen environments. We chose to perform the analysis in this manner, rather than selecting a benchmark battery of tests in advance; our intent was to prevent the introduction of possible experimenter bias on test similarity. Such bias might hide subtle trends in an algorithm's performance, and it could admit *a priori* knowledge into the study.

4.1. *Algorithms*

4.1.1. *Linear operational-space trajectories with Jacobian pseudo-inverse kinematics*

The baseline algorithm used in our experiments is based upon the pseudo-inverse Jacobian inverse kinematics (IK) method. It relies upon a linearly interpolated path between the initial and goal positions and orientations of the simulated humanoid's hand. The linear trajectory combined with the IK method results in an algorithm that blindly attempts to achieve its goal, regardless of the environment. This algorithm improves upon standard resolved-motion rate control by using a regularization method for singularity avoidance described by Baerlocher[33]. Note that this algorithm is a member of the motor primitive class.

4.1.2. *Baseline algorithm with obstacle avoidance integrated into inverse kinematics*

We utilize the redundancy in the manipulator to avoid obstacles through the Jacobian pseudo-inverse with a homogeneous solution. Maciejewski and Klein's algorithm[34] is the chosen implementation. This algorithm also uses a linearly interpolated path between the initial and goal positions and orientations of the humanoid's hand. The method does not plan its way around obstacles as much as reactively attempt to avoid them. The primary downfall to this algorithm is the number of parameters that must be tuned (five) in order to achieve good balance between target convergence and obstacle avoidance. Such problems are inherent to multiple-task Jacobian pseudo-inverse formulations, as noted by Baerlocher[33]. Note that this algorithm is also a member of the motor primitive class.

4.1.3. *Motor primitive vocabulary of common reaches*

The Verbs and Adverbs system, developed by Rose et al.[35], was used to implement a vocabulary of common human-like reaches. This system interpolates between exemplar reaching trajectories in Cartesian space in order to generate reaches to novel locations. The motion database consisted of 784 example motions to and from eight locations around the humanoid, with motions perturbed as necessary to avoid collision with a single small obstacle placed at 27 different locations. The $\binom{8}{2} = 28$ combinations of source and target reaches, in concert with the 28 possible obstacle positions (the scenario corresponding to absence of the obstacle is included), result in the 784-exemplar database. The vocabulary was chosen in an attempt to mimic common human reaches (in terms of initial and final positions and obstacle avoidance), and in so doing, to test their effectiveness. We utilized this system by simulating performance of each of the 28 primitives to the target location; if no collision was detected, the primitive was successful. In this way, the primitive operates blindly as in the previous two algorithms: no planning is performed.

The input space for the interpolator is 6-dimensional, consisting of two concate-

nated 3-dimensional vectors. The first vector represents the initial hand position of the robot in Cartesian space; the second vector represents the final hand position. The output space is 4-dimensional; it is composed of trajectories of the hand position (time is the fourth dimension). The rotational trajectory of the hand is produced by linear interpolation between the initial and target orientations of the hand. Finally, joint-space trajectories can be produced by running an inverse-kinematics algorithm on the combined hand position/orientation trajectories.

We did not attempt to use a primitive parameterization that incorporated the orientation of the hand. Such a parameterization would have allowed interpolation to occur in joint-space rather than Cartesian-space. Cartesian-space interpolation requires resolution into joint-angles via inverse kinematics, and is capable of generating "natural" motion only with great difficulty. In contrast, joint-space interpolation generates motion representative of the exemplar motions; if the exemplars appear natural, then the generated motions will also. Despite this disadvantage with regard to generating natural motion, interpolation in Cartesian-space allows the use of an obstacle-avoidance IK algorithm, increasing the solubility of primitives.

The Verbs and Adverbs system was chosen primarily for its intuitive interpolation results. Other interpolation methods, such as Shepard's[36], failed to produce motions that were highly similar to the exemplar motions, even when very close in the input space. Research also exists[37] that demonstrates how to determine the appropriate input-space parameters to produce a given output (i.e., invert the interpolation mechanism) for Verbs and Adverbs. Such results are necessary to determine the 6-dimensional input-space parameters that generate a trajectory from an initial Cartesian position to a final Cartesian position.

4.1.4. *Joint-space RRT Planner*

The joint-space rapidly-exploring randomly-tree planner is implemented as specified by LaValle[15]. The planner is given an obstacle-free, final posture as its goal; this posture causes the agent's hand to be in the correct position and orientation. The posture obviates the need for inverse kinematics to resolve a suitable configuration from the hand target. In this way, the RRT planner is given an advantage that it would not have solving a problem *in situ*.

4.1.5. *Hybrid algorithms*

Simple hybrid algorithms can be constructed by splicing together one or more of the standard methods. The hybrid is successful if any of the composing methods is successful. We tested hybrids of all possible combinations of the four algorithms just discussed.

4.1.6. *Choosing sampling rates and maximum number of samples*

The algorithms that utilize Jacobian-based IK must transform an operational space trajectory into a joint-space trajectory. A higher sampling rate along this trajectory translates into better performance for the inverse kinematics algorithm at the expense of greater computation: the algorithm's running time scales linearly with the sampling rate. The RRT-based methods also operate better with more samples but experience even greater computational expense; the running time of the RRT algorithm is of quadratic complexity in the number of samples (due to the nearest neighbor search). Additionally, while the running time of the inverse kinematics algorithm can be reasonably predicted based upon the sampling rate, no such prediction can be made for the RRT-based methods. This failure stems from the variation in the time required to obtain a sample; the time required is proportional to the clutter in the environment. This issue makes determining an appropriate number of samples for RRTs difficult.

The inverse kinematics-based algorithms utilized 10,000 samples from the desired trajectory. Using more samples would have resulted in slightly better convergence for the algorithms but would have increased the experimental time dramatically. This slowdown is due to the motor primitives using the relatively slow obstacle avoidance IK algorithm. The RRT-based methods were allowed as many as 3,000 samples, and seemed to benefit very little from adding more. The median number of samples needed in using RRTs for a successful path, when one was found, was 35.

4.2. *Humanoid agent*

The humanoid agent used in the experiments was created from models publicly available from The Princeton Shape Retrieval and Analysis Group[38]. It uses 45 Euler joints for a total of 135 degrees-of-freedom. Each arm is composed of nine DOF, each hip and leg constitutes nine DOF, and the spinal column accounts for fifteen DOF (of which nine DOF lie in the neck). The fingers compose the remaining DOF; the fingers resemble that of a human in complexity and collectively account for 42 DOF per hand. Joint limits approximating those of humans are enforced. Using joint limits, some Euler joints are downgraded to revolute joints without violating generality in the software; for example, the three DOF elbow is reduced to one DOF. The agent stands 1.55 meters tall, and is depicted in Figure 1. Note that the algorithms tested controlled only the right arm and one Euler joint in the spine. Additionally, two DOF in the arm were disabled to better simulate a human arm, resulting in ten effective DOF. The humanoid is simulated using only kinematics; dynamics are not taken into account.

5. Results

The results from the experiment are detailed in Tables 1 and 2. It is apparent that the standard RRT planner performs far better than the other tested algorithms.

The tables also indicate that motor primitives are a powerful method; the small vocabulary of reaching primitives clearly outperformed the standard method of linear hand trajectories and Jacobian pseudo-inverse IK. Table 2 shows that the hybrid algorithms are able to increase performance over their constituent algorithms only slightly.

Figures 1, 2, and 3 demonstrate successes with the algorithms on several environments. Each pair of images depicts the start and end of the motion generated by an individual algorithm. Note that images from the linear hand trajectory with obstacle-avoidance inverse kinematics are not presented; these images highly resemble those generated by the linear hand trajectory with least-squares inverse kinematics algorithm.

Table 1. Results for four primary algorithms over 16,504 trials. Path failure indicates failure of RRT to find a path. Convergence failure indicates failure of IK-based algorithms due to IK convergence. Collision failure indicates failure of IK-based algorithms due to collision. Reaching vocabulary does not include failure results because each primitive in the vocabulary can be considered a separate algorithm.

Algorithm	Success	Path failure	Convergence failure	Collision failure
RRT	77.0%	23.0%	N/A	N/A
Linear hand traj + LS[1] IK	21.7%	N/A	13.8%	64.5%
Linear hand traj + OA[2] IK	21.7%	N/A	13.8%	64.5%
Reaching vocabulary	30.4%	N/A	N/A	N/A

[1] Least-squares
[2] Obstacle-avoidance

6. Discussion

6.1. *Performance expectations*

Our reaching database implementation of motor primitives is obviously not the best method for the single-arm reaching task. Its blind, brute-force attempt of all primitives in the vocabulary might work well in spaces with available knowledge, but is not extremely effective in environments without *a priori* knowledge. Compared with the RRT planner, which is intricately aware of its environment as it expands through the configuration space with a Voronoi bias [39], the reaching database is a poor substitute. However, the comparison between the primitive-based methods indicates that adding more primitives does boost performance. Whether performance can be increased to an acceptable level without adding too many primitives is a question for future research.

Table 2. Results for hypothetical hybrid algorithms over 16,504 trials. Overlap indicates the amount of solution overlap, defined as percentage of trials for which more than one of the algorithms in a hybrid is able to solve.

RRT	LS IK[1]	OA IK[2]	Reaching vocab	Success	Overlap
			•	30.4%	N/A
		•		21.7%	N/A
		•	•	32.1%	62.7%
	•			30.4%	N/A
	•		•	32.1%	62.6
	•	•		21.7%	99.7%
	•	•	•	32.1%	67.9%
•				77.0%	N/A
•			•	83.0%	29.5%
•		•		81.0%	21.8%
•		•	•	83.4%	35.6%
•	•			81.0%	21.8%
•	•		•	83.4%	35.0%
•	•	•		81.0%	26.8%
•	•	•	•	83.4%	35.5%

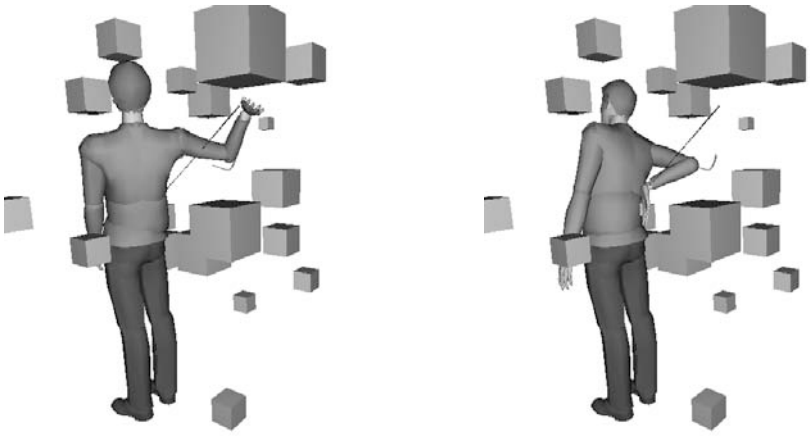
[1]Linear hand trajectory + least-squares IK
[2]Linear hand trajectory + obstacle-avoidance IK

6.2. *Improving performance*

It is desirable to know how much performance can be improved asymptotically with the various algorithms. For example, while there is currently no such analysis available for RRTs, Kavraki et al.[40] have provided a proof relating the failure of a probabilistic roadmap planner (a cognate to the RRT) to the number of samples and the ratio of free volume of configuration space to total volume of configuration space. The proof shows that the probability of the planner failing decays exponentially with the number of random samples employed, assuming that a solution exists.

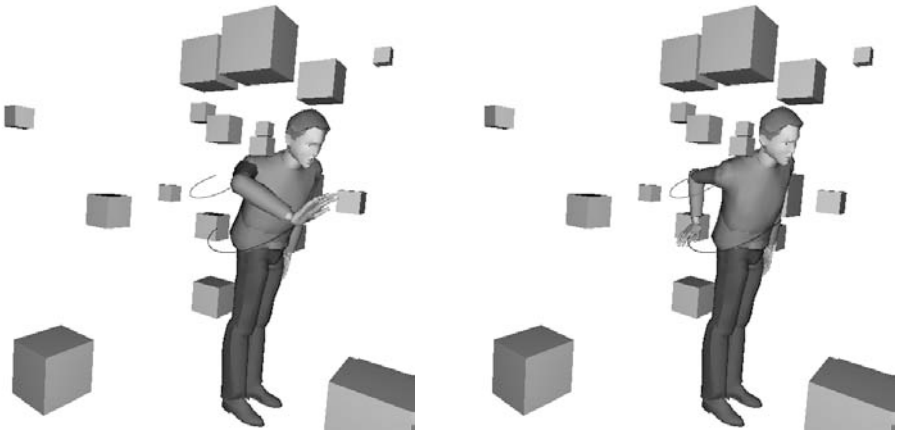
Two of the methods used in this paper rely upon the obstacle avoidance inverse kinematics algorithm. However, this algorithm performs only marginally better than the standard least-squares algorithm. This unexpected result is a product of the algorithm's extensive parameter-tuning requirements; it is very difficult to find a good (much less optimal) set of constants. A better set of parameters should increase performance, albeit probably only slightly.

It is unclear where the limit on performance for reaching with motor primitives lies. Selecting a bigger and/or better vocabulary than is used in this comparison could improve performance dramatically. This performance increase is illustrated in our results by the database vocabulary of 28 primitives outperforming the linear trajectory algorithms, which, as noted are also motor primitives. Alternatively, a different interpolation system might provide better performance. These are issues



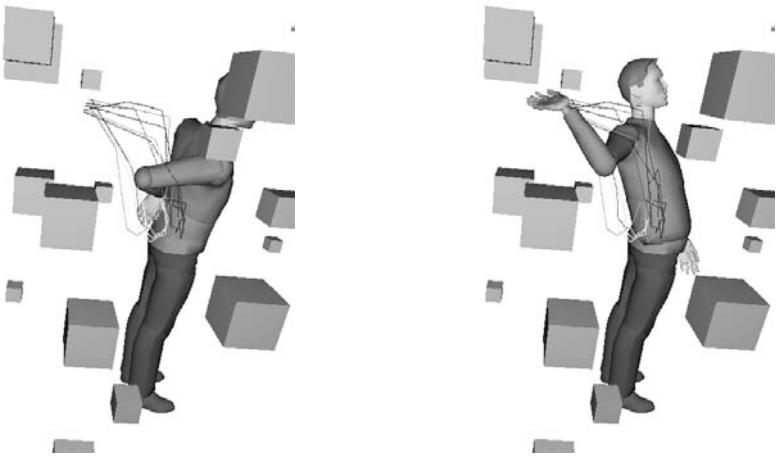
(a) Motion start (b) Motion end

Fig. 1. Start and end of a motion generated by a linear hand trajectory with least-squares inverse kinematics. Paths to be taken by the hand and elbow are drawn.



(a) Motion start (b) Motion end

Fig. 2. Start and end of a motion generated by RRT. Paths to be taken by the hand and elbow are drawn.



(a) Motion start (b) Motion end

Fig. 3. Start of motion generated by the primitive reaching vocabulary. Paths taken by the hand of successful primitives are shown. Note that only one of these paths corresponds to the end motion.

for future research.

6.3. *Extensibility to dynamic environments*

As previously stated, none of the algorithms presented in this paper projects immediately to the problem of humanoid reaching in dynamic environments. The extension of the algorithms of this problem is now discussed.

Kinodynamic algorithms[19,20] are an extension of the standard RRT algorithm to dynamic environments. These methods are limited to low DOF (i.e., around ten), making them ill-suited for most humanoid tasks. Indeed, standard RRTs are analogous to classical AI algorithms for searching in static environments: they both experience high performance in deterministic, static environments, but seem to have difficulty scaling to dynamic environments. One potential problem for kinodynamic algorithms in the context of the humanoid reaching task is the potentially unbounded time required to obtain a sample. Research into kinodynamic algorithms is still nascent; further investigation is needed to determine the efficacy of these algorithms for humanoid reaching.

Motor primitives seem to be a potentially feasible choice for reaching in dynamic environments, perhaps given some *a priori* knowledge about the environment. While a vocabulary of primitives may not always provide a solution (even when one exists), the running time of primitives is more predictable than that of sampling-based planners. Primitive running time is dependent upon number of robot DOF and

obstacles in the environment, while sampling-based planners are dependent upon these factors as well as the amount of "clutter" in the environment. However, a means to improve motor primitive efficiency must be found or computational power has to be significantly increased if primitives are to be used for this type of reaching. Together with the advent of *a priori* knowledge into the reaching task, lower running times could make motor primitives the most attractive choice for single-arm reaching in dynamic environments. However, these issues must be investigated further.

7. Conclusion

We have presented a comparison of the dominant methods for motion planning in the context of humanoid reaching in static environments. The rapidly-exploring random tree (RRT) method proved to be the most successful method. Hybrid methods and extension of the algorithms towards dynamic environments were discussed. Future work will incorporate dynamic obstacles and *a priori* knowledge towards obtaining effective reaching performance for humanoid robots in dynamic environments.

Acknowledgements

This work was funded by the Defense Advance Research Projects Agency (DARPA) grants MARS DABT63-99-1-0015 and DARPA MARS 2020 5-39509-A, and the Office of Naval Research (ONR) MURI grant N0001-01-1-035. The PQP[41] software package was used for distance calculation and collision detection. LAPACK[42] was used for matrix inversion and f2c[43] was used to access LAPACK from C++.

References

1. E. Drumwright and M. Matarić, "Generating and recognizing free-space movements in humanoid robots," in *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, October 2003, pp. 1672–1678.
2. O. C. Jenkins and M. J. Matarić, "Performance-derived behavior vocabularies: Data-driven acquisition of skills from motion," *Intl. Journal of Humanoid Robotics (to appear)*, October 2004.
3. S. LaValle, "Planning algorithms (in preparation)."
4. J. James J. Kuffner, "Autonomous agents for real-time animation," Ph.D. dissertation, Stanford University, December 1999.
5. M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann, "Planning collision-free reaching motions for interactive object manipulation and grasping," in *Proc. of Eurographics*, Grenada, Spain, 2003, pp. 313–322.
6. S. R. Lindemann and S. M. LaValle, "Current issues in sampling-based motion planning," in *Proc. Eight Intl. Symp. on Robotics Research (to appear)*, P. Dario and R. Chatila, Eds. Berlin: Springer-Verlag, 2004.
7. J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.
8. M. Sharir, "Handbook of discrete and computational geometry," J. E. Goodman and J. O'Rourke, Eds. Boca Raton, Florida: CRC Press, 1997, ch. Algorithmic Motion Planning, pp. 733–754.

9. L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
10. K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plakue, and L. E. Kavraki, "Multiple query probabilistic roadmap planning using single query planning primitives," in *Proc. of the Intl. Conf. on Intelligent Robots and Systems (IROS)*, Las Vegas, USA, 2003, pp. 656–661.
11. T. Simeon, J. P. Laumond, and C. Nissoux, "Visibility based probabilistic roadmaps for motion planning," *Advanced Robotics Journal*, vol. 14, no. 6, 2000.
12. R. Bohlin and L. Kavraki, "Path planning using lazy prm," in *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, San Francisco, USA, 2000, pp. 521–528.
13. R. Geraerts and M. Overmars, "A comparative study of probabilistic roadmap planners," in *Proc. of the Intl. Workshop on Algorithmic Foundations of Robotics (WAFR)*, Nice, France, 2002, pp. 443–57.
14. J. J. J. Kuffner and S. M. L. Valle, "Rrt-connect: an efficient approach to single-query path planning," in *Proc. of the Intl. Conf. on Robotics and Automation (ICRA)*, San Francisco, USA, 2000, pp. 995–1001.
15. S. M. LaValle, "Rapidly-exploring random trees: a new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. TR 98-11, October 1998.
16. D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Intl. Journal of Computational Geometry and Applications*, vol. 9, no. 4, pp. 495–512, 1999.
17. G. Sanchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Robotics Research: The Tenth Intl. Symp.*, R. Jarvis and A. Zelinsky, Eds. Springer, 2003, pp. 403–417.
18. J. H. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," in *IEEE Symposium on Foundations of Computer Science*, 1995, pp. 144–154.
19. S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *Intl. Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.
20. D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Intl. Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.
21. M. Kallmann and M. Matarić, "Motion planning using dynamic roadmaps," in *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA)*, New Orleans, LA, April 2004, pp. 4399–4404.
22. P. Leven and S. Hutchinson, "Toward real-time path planning in changing environments," in *Workshop on the Algorithmic Foundations of Robotics*, 2000, pp. 393–406.
23. G. Johannson, "Visual perception of biological motion and a model for its analysis," *Perception and Psychophysics*, vol. 14, pp. 201–211, 1973.
24. G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi, "Premotor cortex and the recognition of motor actions," *Cognitive Brain Research*, vol. 3, pp. 131–141, 1996.
25. L. Fadiga, L. Fogassi, G. Pavesi, and G. Rizzolatti, "Motor facilitation during action observation," *Journal of Neurophysiology*, vol. 73, pp. 2608–2611, 1995.
26. G. Rizzolatti, L. Fadiga, M. Matelli, V. Bettinardi, E. Paulesu, D. Perani, and F. Fazie, "Localization of grasp representations in human by pet: 1. observation versus execution," *Experimental Brain Research*, vol. 111, pp. 246–252, 1996.
27. S. F. Giszter, F. A. Mussa-Ivaldi, and E. Bizzi, "Convergent force fields organized in the frog's spinal cord," *The Journal of Neuroscience*, vol. 13, no. 2, pp. 467–491, February 1993.
28. J. Hodgins and V. Wooten, "Animating human athletes," in *Robotics Research: The*

Eighth Int. Symposium, Y. Shirai and S. Hirose, Eds. Berlin: Springer-Verlag, 1998, pp. 356–367.
29. M. Matarić, V. Zordan, and M. Williamson, "Making complex articulated agents dance: an analysis of control methods drawn from robotics, animation, and biology," *Autonomous Agents and Multiagent Systems*, vol. 2, no. 1, pp. 23–44, March 1999.
30. P. Faloutsos, M. van de Panne, and D. Terzopoulos, "The virtual stuntman: dynamic characters with a repertoire of motor skills," *Computers and Graphics*, vol. 25, no. 6, pp. 933–953, December 2001.
31. A. Fod, M. Matarić, and O. Jenkins, "Automated derivation of primitives for movement classification," *Autonomous Robots*, vol. 12, no. 1, pp. 39–54, January 2002.
32. S. Schaal, "Dynamic movement primitives- a framework for motor control in humanoids and humanoid robots," in *Proc. of the Intl. Symp. on Adaptive Motion of Animals and Machines*, Kyoto, Japan, March 2003.
33. P. Baerlocher, "Inverse kinematics techniques for the interactive posture control of articulated figures," Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, 2001.
34. A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *Intl. Journal of Robotics Research*, vol. 4, no. 3, pp. 109–117, 1985.
35. C. Rose, B. Bodenheimer, and M. Cohen, "Verbs and adverbs: Multidimensional motion interpolation using radial basis functions," *IEEE Computer Graphics and Applications*, vol. 18, no. 5, pp. 32–40, 1998.
36. D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the ACM national conference*. ACM Press, 1968, pp. 517–524.
37. C. Rose, P. Sloan, and M. Cohen, "Artist-directed inverse-kinematics using radial basis function interpolation," *Eurographics*, vol. 20, no. 3, pp. 239–250, 2001.
38. Princeton Shape Retrieval and Analysis Group, "3D Model Search Engine," http://shape.cs.princeton.edu/search.html.
39. S. LaValle and J. James J. Kuffner, "Rapidly exploring random trees: progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B. Donald, K. Lynch, and D. Rus, Eds. Wellesley, MA: AK Peters, 2001, pp. 293–308.
40. L. Kavraki, M. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 1, pp. 166–171, 1998.
41. UNC Research Group on Modeling, Physically-Based Simulation and Applications, "PQP - A Proximity Query Package," http://www.cs.unc.edu/ geom/SSV/.
42. The LAPACK project, "LAPACK - Linear Algebra PACKage," http://www.netlib.org/lapack.
43. The f2c project, "A Fortran to C translator," http://www.netlib.org/f2c.