Proceedings of the First
Artificial Intelligence and Interactive
Digital Entertainment Conference
(AIIDE-2005)

Scalable Solutions for Interactive Virtual Humans That Can Manipulate Objects *

Marcelo Kallmann
University of Southern California
Institute for Creative Technologies
13274 Fiji Way, Marina del Rey, CA 90292
kallmann@ict.usc.edu

Abstract

This paper presents scalable solutions for achieving virtual humans able to manipulate objects in interactive virtual environments. The scalability trades computational time with the ability of addressing increasingly difficult constraints. In time-critical environments, arm motions are computed in few milliseconds using fast analytical Inverse Kinematics. For other types of applications where collision-free motions are required, a randomized motion planner capable of generating motions of average complexity in about a second of computation time is employed. The steps required for defining and computing different types of manipulations are described in this paper.

Introduction

Virtual humans able to manipulate objects in interactive virtual environments have direct applications in several domains, such as: Design, Ergonomics, Virtual Reality and Computer Games.

Besides being tedious and time-consuming, it is not possible to pre-design motions when objects to manipulate can arbitrarily change position in the scene. Alternatively, manipulations can be synthesized by computing arm motions exactly reaching given hand target locations in the virtual environment (see Fig. 1). In its simplest form, this type of problem involves solving the Inverse Kinematics (IK) (Watt & Watt 1992) of the character's arm.

For object manipulations such as pressing buttons, opening drawers or grasping, additional issues must be addressed: collisions, closing fingers, synchronization with moving objects, etc. Among these, the most complex issue is the generation of collision-free motions in arbitrary environments. This is a motion planning problem and is mainly addressed in the robotics literature (Latombe 1990).

*The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

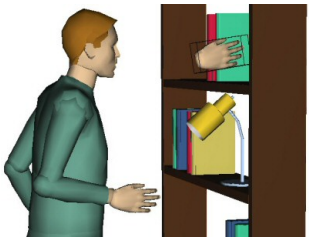

Figure 1: A hand target attached to a book.

This paper presents new techniques based on fast analytical IK (Tolani & Badler 1996) and on randomized on-line motion planning (Kuffner & LaValle 2000) for solving several types of object manipulations in a scalable fashion.

For time critical applications motions are solely computed using an analytical IK solver with extensions for addressing: spine motions, anatomical articulation range limits, and for avoiding collisions. These extensions can be selected according to specific needs and the computation can be performed in the range of few milliseconds.

When collision-free motions among arbitrary obstacles are required, motion planning is employed and the computation time is increased to about one second for planning motions of average complexity. Even if restrictive in many cases, such computational time may be acceptable in several applications involving one or few characters, as for instance in the case of virtual humans demonstrating complex machines and procedures (Rickel & Johnson 1999).

A scalable approach is therefore sought for addressing such different requirements in an unified framework.

Related Work

Only few animation frameworks have focused on virtual humans manipulating objects in interactive virtual environments. While grasping has been the primary interest (Geib, Levison, & Moore 1994), manipulations such as opening drawers and pressing buttons were also addressed (Kallmann 2004). The main idea behind these works is to annotate the environment with interaction information

such as sequences of grasps (Aydin & Nakajima 1999; Rijpkema & Girard 1991) for each object to be manipulated.

Motions are mainly synthesized by Inverse Kinematics (IK) (Watt & Watt 1992). Jacobian-based numerical IK solvers can address arbitrary kinematic chains and several extensions are available, e.g. for minimizing multiple objective functions (Baerlocher & Boulic 1998). For simple linkages (like arms and legs) analytical methods exist and are much more efficient. In this paper, the method proposed by (Tolani & Badler 1996) is used and extended for taking into account joint limits, spine motions and collision avoidance.

IK alone is not capable of generating collision-free motions, which always involve the use of some searching procedure. Searching can be either performed on-line for each problem or reduced to a graph search by pre-computing *roadmaps* (Kavraki *et al.* 1996) (Geraerts & Overmars 2002). However roadmaps are well suited only to static environments, even if alternatives exist (Kallmann *et al.* 2003; Kallmann & Matarić 2004).

On-line planners, also known as single-query methods, explore the search space specifically for each query. When exploration is limited to the workspace (Bandi & Thalmann 2000; Liu & Badler 2003; Yamane, Kuffner, & Hodgins 2004), the planner searches for a sequence of suitable hand locations and relies on an IK algorithm to compute the postures passing by them. The final valid motion is obtained trough interpolation of the postures.

Another approach is to search in the configuration space (Koga *et al.* 1994; Kuffner & Latombe 2000), which is defined by the degrees of freedom (DOFs) of the linkage. Simpler algorithms (not requiring IK during the search) addressing the entire solution space are achieved. However, as the search space grows exponentially with its dimension, simplifying control layers are required (Kallmann *et al.* 2003) for generating whole body motions.

This paper addresses these issues by including spine motions and leg flexion only during the determination of goal postures with IK. Collision-free arm motions are computed with an implementation of the *Rapidly-Exploring Random Tree* (RRT) (LaValle 1998) planner in its bidirectional version (Kuffner & LaValle 2000). The planner search is limited to the configuration space of a 7-DOF arm (only 1 dimension higher than the workspace). In addition, a new sampling routine is proposed for generating more pleasant arm postures. As a result, more human-like motions can be achieved and simple queries can be solved very efficiently due to the RRT greedy behavior.

Paper Overview

Let $h^l = (p, q)$ be a *hand location* specification where: $p \in \mathbb{R}^3$ is the position in global coordinates of the wrist joint, and $q \in \mathbb{S}^3$ is the orientation in global coordinates of the same joint, in quaternion representation (Watt & Watt 1992).

Given a desired hand location h^l to be reached, the problem of computing an arm motion bringing the character's hand to h^l is first addressed using an Inverse Kinematics (IK) solver, with extensions for addressing: joint limits, leg flexion, spine motion, and collision avoidance.

The IK collision avoidance only corrects the elbow position during a straight line hand trajectory. A bidirectional RRT planner is used for obtaining collision-free motions.

Let now h^s be a *hand shape* specified by the joint values of the finger articulations of one hand of the character. Note that by symmetry it is possible to apply any given h^s to either the right or left hand.

A complete *hand target* $h^t = (h^l, h^s)$ is specified with both a hand location and shape. Hand targets are annotated in the environment and indicate how to reach, grasp or manipulate objects (see Fig. 1).

After presenting the used IK and motion planner in the following sections, alternatives for using these methods in different types of manipulations are given. The example of a virtual human relocating books in a bookshelf is presented and computation times are given.

Inverse Kinematics and Extensions

The used analytical IK solver (Tolani & Badler 1996) can be applied to either the arms or legs of a character. For each arm or leg, there are 7 degrees of freedom (DOFs) to be determined for reaching a given goal hand target with 6 DOFs. The missing DOF is modeled as the *orbit angle* (also called as the *swivel angle*).

Given a desired hand location h^l and orbit angle α, the IK solver is then capable of providing the 7 DOF arm pose reaching h^l. The orbit angle is defined as zero when the elbow is in its lowest possible position, and as α grows, the elbow gets higher by rotating outwards. Fig. 2 left shows an arm posture with orbit angle of 20 degrees.

It is clear that not all given α will lead to anatomically feasible arm postures. A method that automatically determines α values respecting joint limits and avoiding collisions is given in this section.

Joint Limits Bounding anatomically correct postures starts with imposing joint range limits to the articulations of the skeleton. For anatomical articulations with a 3-DOF rotation R, e.g. the shoulder, using maximum and minimum Euler angles is not meaningful. The simplest solution is to use the swing-and-twist decomposition (Grassia 1998) :

$$R = R^{twist} R^{swing},$$

$$\text{where } R^{twist} = R_z(\theta), \text{ and } R^{swing} = [S_x \quad S_y \quad 0]$$

The swing rotation is parameterized by the above axis-angle, having $\|R^{swing}\|$ as rotation angle, and R^{swing} as rotation axis, which always lies in the x-y plane of the local frame. The axial rotation (or twist) that follows occurs around the z-axis of the same local frame.

The one singularity of the parameterization is reached when the swing vector has norm π. Consequently, the singularity is easily avoided by choosing an appropriate zero posture (i.e., when $S_x = S_y = 0$). For the shoulder joint for instance, the reference (zero) posture has the arm outstretched.

The swing-and-twist decomposition is used to model the shoulder (3 DOFs). The elbow has flexion and twist rotations defined with two Euler angles (2 DOFs), and the wrist has a swing rotation (2 DOFs) parameterized exactly as the described swing-and-twist, however considering the twist rotation θ always 0.

Such representation allows the use of spherical polygons (Korein 1985) for restricting the swing motion of the shoulder and wrist. Spherical polygons can be manually edited for defining a precise bounding curve. However a simpler, more efficient, and still acceptable solution for bounding swing limits is followed based on spherical ellipses. In this case, a swing rotation can be checked for validity simply by replacing the axis-angle parameters into the ellipse's equation. For precise details, see (Baerlocher 2001). The twist and flexion rotations of the remaining DOFs are correctly limited by minimum and maximum angles.

Collisions *Collision geometries* are attached to the joints of the character as needed for the purpose of collision detection. These are rigid models which in general are simpler than the models used for visualization of the character.

The VCollide package (Gottschalk, Lin, & Manocha 1996) is employed for querying if body parts self-intersect or intersect with the environment. At initialization phase, the character is required to be in a guaranteed valid posture. Several collisions will be reported between pair of models which are adjacent. These pairs are detected and disabled in all future queries. For testing a new character pose, the transformation matrices of the collision geometries are updated accordingly, and a new query reports if any collisions are found.

Orbit Angle Determination For maximum efficiency, the analytical IK solver proposed in (Tolani & Badler 1996) was rewritten for giving joint values directly in the considered representation based on swing-and-twist parameterizations. Still, the orbit angle α has to be determined.

The solution proposed here is to start with a given low angle, e.g. of 20 degrees. Then, the posture given by the IK solver is checked for validity in terms of joint limits and collisions. If the posture is not valid, a quick iterative method is performed for verifying if other α values can lead to a valid posture.

At each iteration, α is incremented and decremented by Δ, and the two new postures given by the IK solver are checked for validity. If a valid posture is found the process successfully stops. Otherwise, if given minimum and maximum α values are reached, failure is returned. Better performance is achieved in a greedy fashion, i.e. when the increment Δ increases during the iterations.

As the search range is small this simple process is very efficient. By setting a range from -15 to 130 degrees, and correctly adjusting the increments, the whole process can be limited to few collision tests. Note that both joint limits and collisions are addressed in an unified way. Fig. 2 shows an example of an emergent joint coupling automatically achieved, and Fig. 3 shows a self-collision avoided.

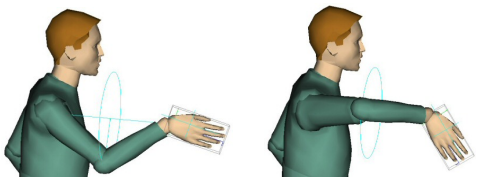

Figure 2: Starting from its orbit angle of 20 degrees (left), as the wrist rotates downwards, the elbow rotates to a higher position (right), ensuring the wrist respects its joint limits.

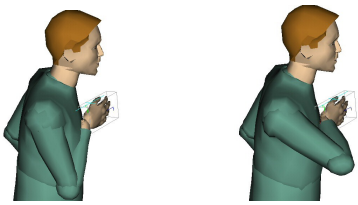

Figure 3: If collisions are detected (left) the orbit angle is updated and a valid posture is eventually found (right).

Spine Motion and Leg Flexion Before applying the IK solver for reaching a given hand location h^l, simple checks can be used in order to determine if a spine motion or leg flexion should be applied.

For example, more natural postures are achieved when the joints of the spine are simply rotated towards h^l: if h^l is close to the torso, a very small rotation is applied, and if h^l is distant, a larger rotation is used. If h^l is too low and not reachable, leg flexion might turn h^l into a reachable location. Leg flexion can be simply implemented by lowering the skeleton root while using IK to maintain the feet fixed in their original position. Fine tunning is inevitable for achieving the intended overall behavior, and designers can define the parameters in order to create custom behaviors. As an example, Fig. 1 shows the character with a standard spine posture while in Fig. 4 the spine is slightly bent.

Collision-Free Motions

Let s denote the 7-DOF arm configuration plus any additional DOFs in case spine motion and/or leg flexion are considered by the IK solver. In any case, configuration s is referred to as an arm configuration.

Let h^l be the location to reach with a collision-free motion. Let s_{init} be the arm configuration of the character in its current, initial posture. Let s_{goal} be the arm configuration given by the IK solver for reaching h^l. Note that, even if the IK

solver has a built-in collision avoidance mechanism, a failure can be reported and in this case h^l is considered to not be reachable.

Once the IK solver can successfully determine s_{goal}, two search trees T_1 and T_2 are initialized having s_{init} and s_{goal} respectively as root nodes. Algorithm 1 can then be called and it presents an implementation of the bidirection RRT planner (Kuffner & LaValle 2000).

Algorithm 1 PLANPATH (T_1, T_2)

```
1: while elapsed time ≤ maximum allowed time do
2:    s_sample ← SAMPLEPOSTURE().
3:    s_1 ← closest node to s_sample in T_1.
4:    s_2 ← closest node to s_sample in T_2.
5:    if INTERPOLATIONVALID ( s_1, s_2 ) then
6:       return MAKEPATH (root(T_1), s_1, s_2, root(T_2)).
7:    end if
8:    s_exp ← NODEEXPANSION (s_1, s_sample, ε).
9:    if s_exp ≠ null and INTERPOLATIONVALID (s_exp, s_2)
      then
10:      return MAKEPATH (root(T_1), s_exp, s_2, root(T_2)).
11:   end if
12:   Swap T_1 and T_2.
13: end while
14: return failure.
```

Line 2 of algorithm 1 requires a configuration sampling routine. This routine guides the whole search and therefore is of main importance. The basic process samples random valid joint values inside their range limits, which are defined either by maximum and minimum angle values or by a spherical ellipse in case of swing rotations. A sample is only retained if it implies no collisions, otherwise another sample is determined until a collision-free one is found.

Randomly sampling valid postures has the effect of biasing the search towards the free spaces. Although large volumes of free space are located at the sides of the character, realistic manipulations are mainly carried out in front of the character.

A simple correction technique consists of highly biasing the elbow flexion to almost full flexion. This has the effect of avoiding solutions with the arm outstretched, resulting in more natural motions. As we perform a bidirectional search, it also contributes to decomposing the manipulation in natural two phases: bringing the arm closer to the body and then extending it towards the goal. The biasing method starts sampling the elbow flexion DOF with values in the interval between 100% and 90% of flexion, and as the number of iterations grow, the sampling interval gets larger until reaching the joint limits.

The wrist swing rotation is also highly biased to a smaller range than its valid range, resulting in more pleasant postures as this joint is only secondarily used for avoiding obstacles. Finally, when spine DOFs and/or root translation DOFs (for knee flexion) are used, they are sampled with very small variation, only to give an overall human-like movement, and therefore these DOFs do not augment the dimension of the considered search space.

Lines 3 and 4 of algorithm 1 require searching for the closest configuration in each tree. A linear search suffices as the trees are not expected to grow much. The used distance metric takes the maximum of the Euclidian distances between the position of corresponding joints, posed at each configuration.

Lines 5 and 9 check if the interpolation between two configurations is valid. This can be done by performing several discrete collision checks along the interpolation between the two configurations. However more efficient continuous methods are available (Schwarzer, Saha, & Latombe 2002).

The expanded node s_{exp} in line 8 uses s_{sample} as growing direction and is determined as follows:

$$s_{exp} = interp(s_1, s_{sample}, t), \text{ where}$$
$$t = \varepsilon / d,\ d = dist(s_1, s_{sample})$$

Null is returned in case the expansion is not valid, i.e. if the interpolation between s_1 and s_{exp} is not valid. Otherwise s_{exp} is linked to s_1, making the tree to grow by one node and one edge. Factor ε gives the incremental step taken during the search. Large steps make trees to grow quickly but with more difficulty to capture the free configuration space around obstacles. Too small values generate roadmaps with too many nodes, slowing down the tree expansion.

Whenever a valid connection between T_1 and T_2 occurs, a path is computed and returned as a valid solution. Routine MAKEPATH(s_1, s_2, s_3, s_4) (lines 6 and 10) connects the tree branch leading s_1 to s_2 with the tree branch joining s_3 with s_4 by the interpolation between s_2 and s_3.

A final post-processing step for smoothing the path is required. The simplest approach is to pass few random linearization iterations. Each linearization consists of selecting two random configurations s_1 and s_2 (not necessarily nodes) along the solution path and replacing the sub path between s_1 and s_2 by the straight interpolation between them, if the replacement is a valid path (Schwarzer, Saha, & Latombe 2002). Fig. 4 shows the graph of a search and the solution path, before smoothing.

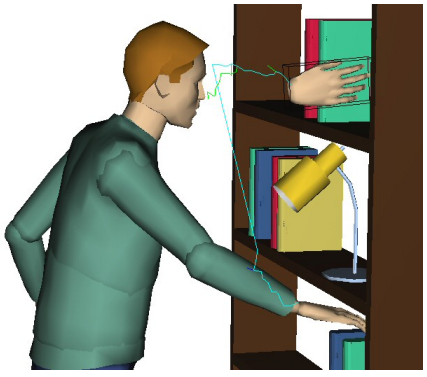

Figure 4: The polygonal line joining the character's hand to the target hand represents the trajectory of the wrist joint for a valid collision-free motion.

Scalable Specification of Manipulations

Consider now the problem of moving the character's hand from its current location and shape $h^t(0) = \{h^l(0), h^s(0)\}$ to a given hand target $h^t(1) = \{h^l(1), h^s(1)\}$.

Neuroscience research provides several computational models (2/3 power law, Fitts' law, etc) (Schaal 2002) that can help synthesizing realistic arm motions. The simplest model states that in point-to-point reaching movements, the hand path approximates a straight line, and the tangential velocity resembles a symmetrical bell-shape. Such guidelines are followed here.

Let $h^t(t) = \{h^l(t), h^s(t)\}, t \in [0, 1]$ be the interpolated hand target. Linear interpolation is used for the wrist position and spherical linear interpolation is used for the wrist orientation. The rotations of the finger joints defining the hand shapes are interpolated according to their parameterization. Finally, for each t, the IK solver is used for deriving the arm configuration $s(t)$ reaching $h^l(t)$.

When collision-free motions are required, the motion planner is used for generating a motion as a sequence of arm configurations. Such motion can be re-parameterized and can have added finger movements from the interpolation between the initial and goal hand shapes. The result is a complete reaching motion $s(t)$, $t \in [0, 1]$, where the target $h^t(1)$ is reached with configuration $s(1)$.

For both the planned and the IK-based motion, when parameter t varies in $[0, 1]$ with constant velocity, the wrist joint will also move with constant velocity in the workspace. The simple cubic spline variable change $t = -2h^3 + 3h^2$ will introduce a bell-shape velocity profile when h varies with constant velocity in $[0, 1]$.

It is now possible to choose between three approaches for computing arm motions:

- IK-based motion without performing the collision avoidance tests requires less than 1 millisecond of computation time and does not depend on the complexity of the scene.
- IK-based motion with the collision avoidance tests requires from 1 to 30 milliseconds, depending on how much search was performed.
- In scenarios with average complexity, the motion planner requires computation times from 20 milliseconds to 2 seconds. The performance depends on the complexity of the solution, which is related to how cluttered is the environment. The book relocation example shown in Fig. 5 required 1.3 seconds. These times do not include the path smoothing step, that can take from 0.5 to 1.5 seconds depending on the desired quality. Such experiments were conducted in a 3GHz pentium, with collision detection handling 12K triangles.

Several types of manipulations can be specified by using the presented techniques:

- Grasping can be modeled trough a sequence of hand targets. A simple sequence consists of an initial arm movement towards a pre-target, followed by a slower motion towards the final target with final hand shape. Similar sequences are described in the neuroscience literature (Schaal 2002). Optional extensions such as finger-object collision tests for perfect finger closure and multiple grasping styles per object can be added as needed.
- Object relocation is treated exactly as a reaching problem: after attaching the grasped object to the wrist joint, the object is considered making part of the hand and a new motion towards the desired object placement can be computed (see Fig. 5).
- Pushing and pulling can be performed with the IK solver. Consider the case of opening a drawer. After the drawer handle is grasped, a translation motion is applied to the drawer and the hand follows the drawer's motion, while the arm posture is updated with the IK solver.

Conclusions

This paper presents new techniques based on analytical Inverse Kinematics and on randomized on-line motion planning for synthesizing object manipulations.

Besides the overall system, the main contributions are: a new orbit angle search mechanism for the analytical IK solver that achieves joint coupling and collision avoidance, and new sampling strategies for achieving more natural whole-body postures with motion planners. The techniques have straightforward implementation.

As computers get faster, randomized motion planners might significantly improve the motion autonomy of characters in interactive applications.

References

Aydin, Y., and Nakajima, M. 1999. Database guided computer animation of human grasping using forward and inverse kinematics. *Computer & Graphics* 23:145–154.

Baerlocher, P., and Boulic, R. 1998. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *Proceedings of IEEE IROS'98*, 323–329.

Baerlocher, P. 2001. *Inverse Kinematics Techniques for the Interactive Posture Control of Articulated Figures*. Ph.D. Dissertation, Swiss Federal Institute of Technology, EPFL. Thesis number 2383.

Bandi, S., and Thalmann, D. 2000. Path finding for human motion in virtual environments. *Computational Geometry: Theory and Applications* 15(1-3):103–127.

Geib, C.; Levison, L.; and Moore, M. B. 1994. Sodajack: An architecture for agents that search and manipulate objects. Technical Report MS-CIS-94-16, University of Pennsylvania, Department of Computer and Information Science.

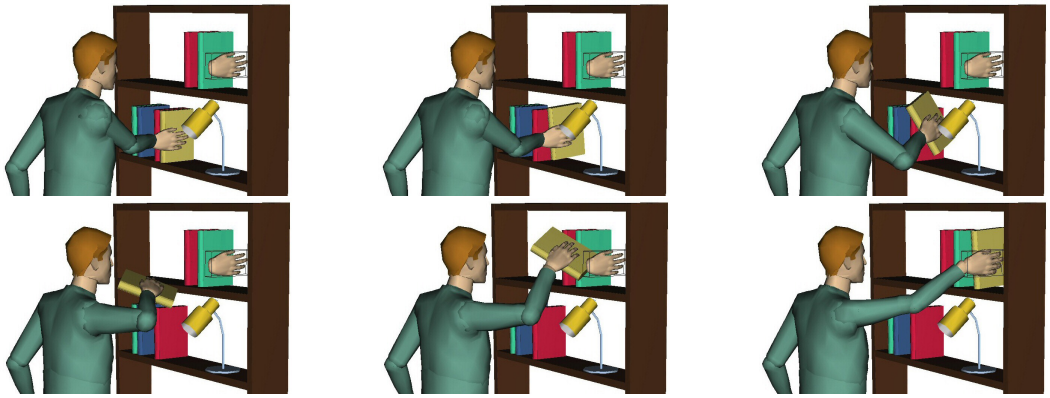

Figure 5: Book relocation example.

Geraerts, R., and Overmars, M. 2002. A comparative study of probabilistic roadmap planners. In *Proceedings of the Int'l Workshop on Algorithmic Foundations of Robotics (WAFR'02)*.

Gottschalk, S.; Lin, M. C.; and Manocha, D. 1996. Obbtree: A hierarchical structure for rapid interference detection. *Computer Graphics SIGGRAPH'96* 30(Annual Conference Series):171–180.

Grassia, S. 1998. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools* 3(3):29–48.

Kallmann, M., and Matarić, M. 2004. Motion planning using dynamic roadmaps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'04)*, 4399–4404.

Kallmann, M.; Aubel, A.; Abaci, T.; and Thalmann, D. 2003. Planning collision-free reaching motions for interactive object manipulation and grasping. *Computer graphics Forum (Proceedings of Eurographics'03)* 22(3):313–322.

Kallmann, M. 2004. Interaction with 3-d objects. In Magnenat-Thalmann, N., and Thalmann, D., eds., *Handbook of Virtual Humans*. John Wiley & Sons, first edition. 303–322.

Kavraki, L.; Svestka, P.; Latombe, J.-C.; and Overmars, M. 1996. Probabilistic roadmaps for fast path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12:566–580.

Koga, Y.; Kondo, K.; Kuffner, J. J.; and Latombe, J.-C. 1994. Planning motions with intentions. In *Proceedings of SIGGRAPH'94*, 395–408. ACM Press.

Korein, J. U. 1985. *A Geometric Investigation of Reach*. The MIT Press.

Kuffner, J. J., and Latombe, J.-C. 2000. Interactive manipulation planning for animated characters. In *Proceedings of Pacific Graphics'00*. poster paper.

Kuffner, J. J., and LaValle, S. M. 2000. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings of IEEE Int'l Conference on Robotics and Automation (ICRA'00)*.

Latombe, J.-C. 1990. *Robot Motion Planning*. Kluwer Academic Publisher.

LaValle, S. 1998. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, Computer Science Department.

Liu, Y., and Badler, N. I. 2003. Real-time reach planning for animated characters using hardware acceleration. In *Proceedings of Computer Animation and Social Agents (CASA'03)*, 86–93.

Rickel, J., and Johnson, L. 1999. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence* 13(4-5):343–382.

Rijpkema, H., and Girard, M. 1991. Computer animation of knowledge-based human grasping. In *Proceedings of SIGGRAPH'91*, 339–348.

Schaal, S. 2002. Arm and hand movement control. In Arbib, M., ed., *The handbook of brain theory and neural networks*. The MIT Press, second edition. 110–113.

Schwarzer, F.; Saha, M.; and Latombe, J.-C. 2002. Exact collision checking of robot paths. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics (WAFR'02)*.

Tolani, D., and Badler, N. 1996. Real-time inverse kinematics of the human arm. *Presence* 5(4):393–401.

Watt, A., and Watt, M. 1992. *Advanced Animation and Rendering Techniques*. ACM Press.

Yamane, K.; Kuffner, J. J.; and Hodgins, J. K. 2004. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics (Proceedings of SIGGRAPH'04)* 23(3):532–539.