Robot Catching

Marcia Riley[1,2] and Christopher Atkeson[1,2]

[1] ATR HIP and ISD,
ATR International, Japan
[2] College of Computing, Georgia Institute of Technology,
Atlanta, GA, USA
{mriley, cga}@cc.gatech.edu
http://www.cc.gatech.edu/~mriley
http://www.cc.gatech.edu/fac/Chris.Atkeson

Abstract. Our focus is on creating interesting, realistic behaviors for humanoid robots and virtual characters. Among the methods we use to create human-like movements are a template-based approach, where example movements from humans are stored for later use, and a dynamical systems approach where new movements are generated from motion primitives as they are needed. Here we discuss a dynamical systems approach for implementing ball catching for a 30-degree-of-freedom humanoid robot. In our catching task, we generate ball-hand impact predictions and human-like motion trajectories to move the hand to the impact position. We also discuss what is necessary to use a template-based approach to generate behaviors.

1 Introduction

Generating human-like motion for virtual characters and humanoid robots facilitates our goal of building more engaging machine behaviors and human-machine interactions. Evidence and experience show that agents with human-like traits provide an effective way to interact with a person [13, 23]. To create these behaviors, we borrow from results in human motor control and computational neuroscience.

Synthesizing motion for humanoid robots is difficult because of the large number of degrees of freedom. Additionally, we encounter challenges not found when generating motion for a virtual human because we must obey the physics of the real world. Here we present work on a ball catching task for the humanoid robot shown in Figure 1 (www.sarcos.com). We take a dynamical systems approach and generate new movements as they are needed. We also discuss the merits of a template or library-based approach in creating humanoid motion.

1.1 Why Catching?

Playing catch is a simple, safe, and fun way for a human to interact with a humanoid robot. Interactive tasks are particularly interesting because they engage a person in a behavior with a robot, and require real time solutions from the robot to participate in that behavior. A humanoid robot with its increased complexity (our robot has 30 degrees of freedom, Figure 2) challenges us to find satisfactory and human-like movements. Because it is humanoid, the robot lends itself to solutions inspired by theories of human movement, and we exploit this knowledge of human movement in our catching task solution.

1.2 Related Work

We have explored several types of catching with the humanoid robot, including three ball juggling (Figure 3) [21]. Mason reviews related work in robot juggling and dynamic manipulation (www.juggling.org/help/misc/robots.html).

There are many forms of catching. This paper focuses on catching as an inelastic collision between a robot surface and a flying object [8, 21]. We use a baseball glove to catch in this work, while catching using funnels is described in [16, 17]. Several groups have implemented catching in Kendama, a ball in cup task, including Miura's students at the University of Tokyo, Sakaguchi and Miyazaki [18, 19], and Kawato's group at ATR [11]. We have also implemented open loop paddle juggling on this robot, where a ball is batted (an elastic collision) rather than caught, and the characteristics of the hit trajectory and paddle stabilize the pattern [20].

We do not match the ball velocity [6] as we primarily catch balls moving downwards rather than horizontally and the acceleration and range of robot movement is limited so it is difficult to get up to speed and slow down in

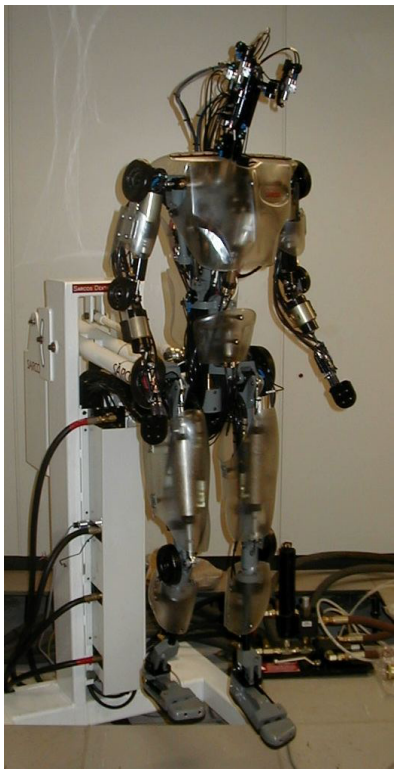

Fig. 1. The humanoid robot in our laboratory

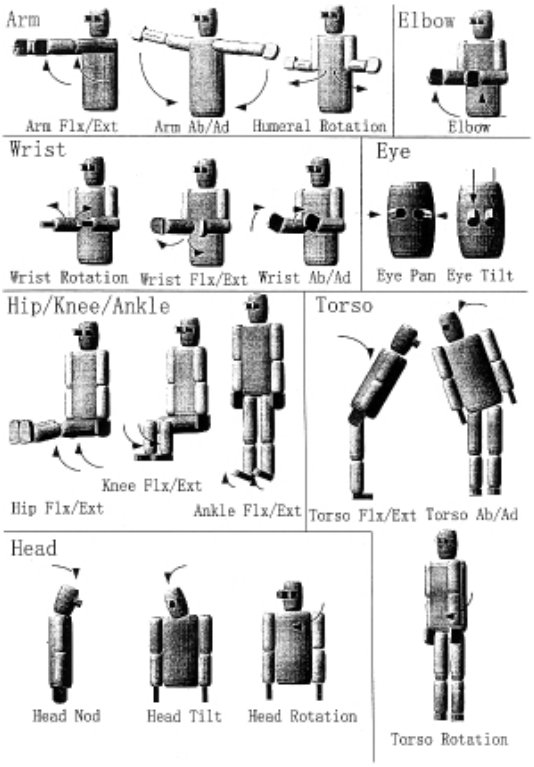

Fig. 2. The robot's degrees of freedom

the movement range available. The robot does not have fingers, so it cannot grasp the ball (diwww.epfl.ch/lami/robots/jongleur.html) [6].

There is a large literature on the tracking of flying objects, and we describe the reasoning behind our tracking algorithm in a later section. The use of a state observer for a ball continuously in flight with intermittent impacts is described in [15]. Since the ball we are tracking is only in flight for a short time and often cannot be seen when it is in the thrower's hand, observer and Kalman filter based approaches did not work well for us.

There are many possible trajectories to get from where the robot is at the start of the task to the catch point. We explore a dynamical systems approach. We plan to explore a template-based approach where we use a library of motion capture trajectories and select the most appropriate trajectory for the current goal, and an optimal control strategy approach similar to [6] or [8]. [2] and [3] use a feedback or policy-based approach to generate a trajectory and match the ball velocity. [10] describes approaches to guide missiles to intercept targets, and a robot intercept trajectory generator adapted from missile algorithms. We note that the bang-bang or bang-coast-bang trajectories used in many of these papers are not smooth enough for our humanoid robot, and the discontinuity in acceleration and force excites too much vibration in our robot.

Objects other than balls can be caught, and catching can be generalized to any process that reduces the volume of state space a system or part of a system can occupy. Nakanishi, Fukuda, and Koditschek describe a brachiating robot that catches “branches” as it swings through the “jungle” [12]. Work on swing up can be viewed as a form of dynamic catch (www.cc.gatech.edu/grads/b/Gary.N.Boone/acrobot/acrobot.html) [1, 24].

2 Implementation

To successfully catch a ball the robot must be able to detect the ball, determine when it is in flight, track and predict the ball's trajectory, and plan and execute a movement to intercept the ball. When estimating the ball trajectory, we use an iterative prediction algorithm which adjusts the prediction according to new sensory input. Additionally, we use an online motion trajectory generator, which changes the intercept trajectory according to the revised impact prediction information.

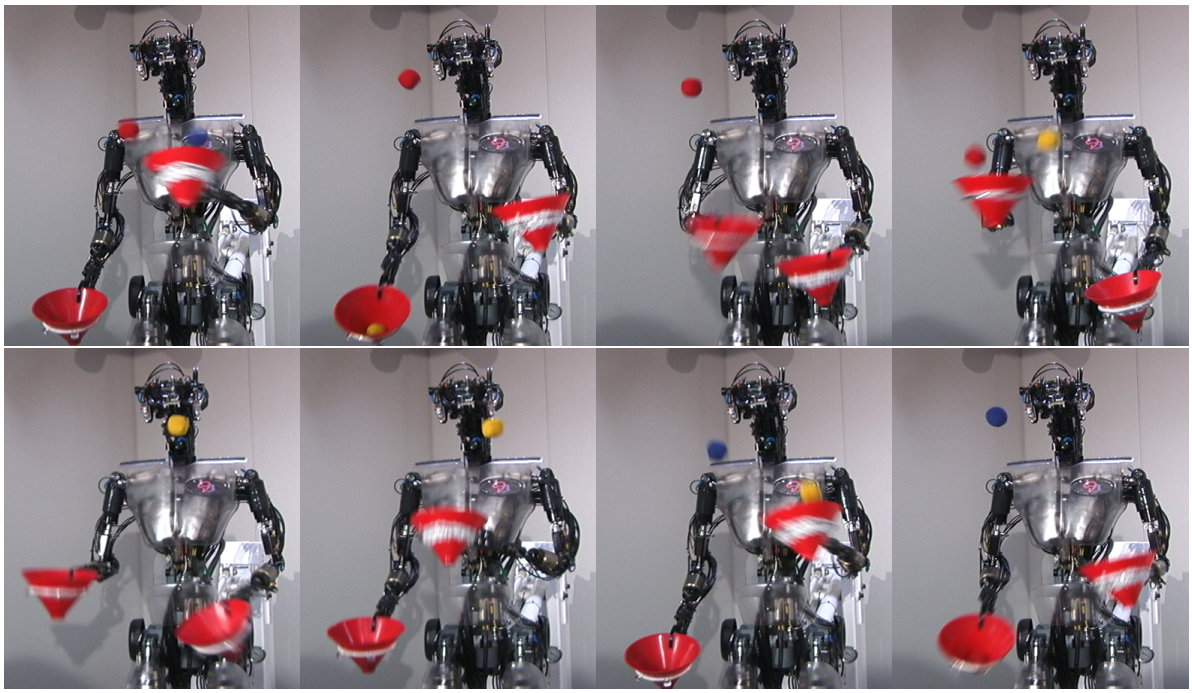

Fig. 3. The humanoid robot juggling 3 balls, using kitchen funnels for hands.

The QuickMag (OKK Inc., Japan) stereo color vision system is used to detect the ball and track its flight. It supplies the three dimensional centroids of pixels of a particular color range 60 times a second. To detect when the ball is in flight, the thrower's hand is also tracked, so that the ball position relative to the human's hand position can be determined. To the robot, the ball is in flight when the distance between the ball and hand surpasses a pre-defined threshold distance.

Once the ball is in flight, the robot starts collecting ball positions (x_i, y_i, z_i) to use in predicting the time and place of ball-glove impact. (The robot is wearing a child's baseball glove on its left hand to make the catch.) Optimally, enough frames of data should be available at the first prediction to provide a fairly good solution. A good solution is one in which subsequent predictions will result in relatively small changes to the original solution in order to avoid large changes in direction for the robot movement.

The current prediction algorithm assumes that the z-component of the ball trajectory is a parabola, thus we ignore the effects of drag. We can solve the following system of equations to calculate the parameters of the

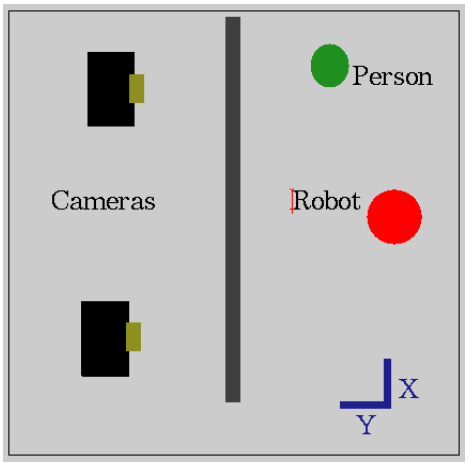

Fig. 4. An overhead view of the catching setup. Vision cameras are mounted on the ceiling.

parabola

$$z_i = at_i^2 + bt_i + c, i = 1, \ldots, n, \tag{1}$$

where $n \geq 5$ is the number of measurements. The resulting normal system of equations is given by

$$\begin{bmatrix} \sum_{i=1}^{n} 1 & \sum_{i=1}^{n} t_i & \sum_{i=1}^{n} t_i^2 \\ \sum_{i=1}^{n} t_i & \sum_{i=1}^{n} t_i^2 & \sum_{i=1}^{n} t_i^3 \\ \sum_{i=1}^{n} t_i^2 & \sum_{i=1}^{n} t_i^3 & \sum_{i=1}^{n} t_i^4 + \lambda \end{bmatrix} \begin{bmatrix} c \\ b \\ a \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} z_i \\ \sum_{i=1}^{n} z_i t_i \\ \sum_{i=1}^{n} z_i t_i^2 - \lambda \frac{1}{2} g \end{bmatrix} \tag{2}$$

where the additional parameters in the system (λ and - $\lambda \frac{1}{2} g$) account for gravity.

For a desired pre-determined catch height, the intercept time is estimated using the calculated parabola. Then the corresponding x and y coordinates for this catch time are estimated assuming that the x and y components of the trajectory are linear. With each new frame of vision data, this prediction is recalculated incorporating all available frames of vision data.

Predictions are currently begun after 5 frames are collected from the vision system once the ball is in flight. The process continues until the catch is made, or another end condition is triggered, such as the ball being lower than the impact height (a miss), or the task exceeding a maximum time limit.

Once we have a target position, we generate trajectories to move the hand to the target. We prefer a trajectory that is efficient and human-like. Thus, we require a desired trajectory that generates smooth motion where the desired velocity and acceleration at the movement beginning and end are zero. Since we are intercepting a ball that is primarily falling downwards, and catching with an inelastic collision, we did not attempt to match the ball velocity at the catch. We decided to catching movements as needed from motion primitives. Here we experiment with programmable pattern generators (PPGs) [22], whose design is based on hypotheses about how human movement is generated. PPGs produce trajectories similar to what is observed in humans. For a point-to-point movement, for example, they produce smooth desired trajectories with approximately bell-shaped velocity profiles. In catching, this point-to-point movement is from the initial hand position to the predicted ball-hand impact position. A first order differential equation is augmented by nonlinear terms to build the smooth movement. The input is a desired Cartesian target and an average velocity. The output is a smooth trajectory plan. Additionally, trajectories planned with PPGs can be modified on-line as new sensor information becomes available. More details are described in [22]. An example trajectory generated from PPGs for one of the robot's catches is shown in Figure 5.

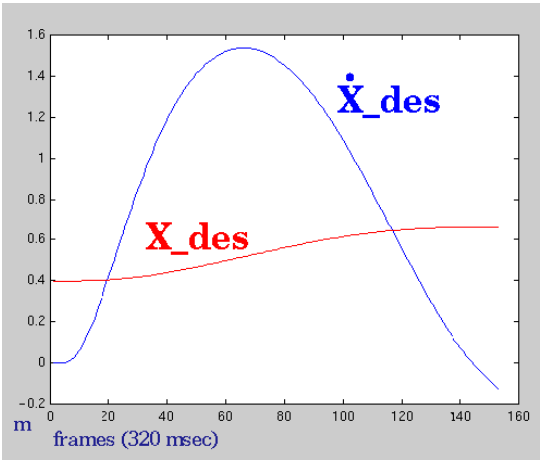

Fig. 5. A smooth hand trajectory X_{des} and its velocity profile $\dot{X_{des}}$

Since the ball position and the predicted impact position, and thus the desired end effector position, are given in Cartesian space, we must solve the inverse kinematics problem to arrive at a desired body configuration described by joint angles that will place the end effector in the target Cartesian location. To accomplish this we use the solution described in [26]. This allows us to convert quickly from Cartesian to joint space.

Using the inverse kinematics solver and Cartesian trajectory planner, we can attain the desired final position incrementally as follows. For our current impact prediction, the trajectory specifies a partial movement toward the goal, supplying both desired position and desired velocity. The inverse kinematics solver is then used to follow in joint space the Cartesian path specified by the trajectory. The inverse kinematics solution is calculated 480 times a second (the robot's task servo rate). As new ball position data becomes available, it can be used to modify the Cartesian target used to plan the movement. In this way, the robot is always moving toward the current goal in small steps, and can alter the motion trajectory as the impact prediction coordinates change.

3 Results and Discussion

Hand velocities needed by the robot to catch the throws typically vary between 0.160 meters/second and 1.2 meters/second. Distances from the initial hand position to the final position usually range between approximately 0.07 meters (almost thrown into the glove) and over 1 meter. The robot fits a parabolic trajectory accurately at heights varying from .32 meters to .4 meters. (The usual starting height for the robot's left hand was .27 meters, with the origin of the coordinate system located near the pelvis of the robot.) The catching motion is completed in roughly 350 to 850 msec.

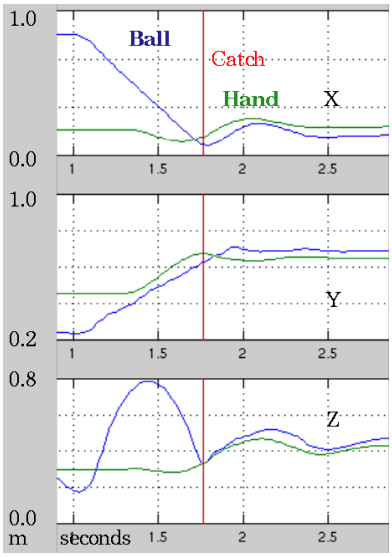

Fig. 6. A successful catch marked by the vertical line. Hand and ball trajectories intersect closely enough to make the catch.

Reasons for missed catches vary. The most common misses are due to the ball being thrown outside the robot's workspace, and the ball bumping into but not staying in the glove. (Currently we have no fingers to close the glove once the ball is inside.) Additionally the position data provided by the vision system can be noisy. We discuss one possible improvement to noise below. Finally, sometimes a catch may fail because the desired trajectories were not performed accurately enough. To improve this, more precise trajectory following based on machine learning could be used. Despite these problems, the robot catches reliably. Figure 6 shows a ball and robot hand trajectory for a successful catch.

Figure 7 shows an estimation of the ball position provided by the vision system. Noise is evident especially in the y coordinate, which is the depth coordinate with respect to the vision system, and thus the most difficult to accurately estimate. To improve the estimation and minimize the noise, we are currently testing adding Kalman filtering [9] to the prediction algorithm. The difficulty is that the assumptions about the type of errors/noise in the sensors and dynamics used to design the Kalman filter do not match the errors/noise in our implementation, leading to poor catch location prediction during the startup transient when the ball is thrown. If the initial gains for measurement mismatch are low, it takes too long for the filter to match what the ball is actually doing. If the initial gains are too high, the start up transient is short, but the tracking is noisy. It is difficult to choose filter design parameters to get good transient performance. The strength of the prediction algorithm we are currently using is that we don't have to model sensor or process errors/noise, and we get accurate predictions after only a small

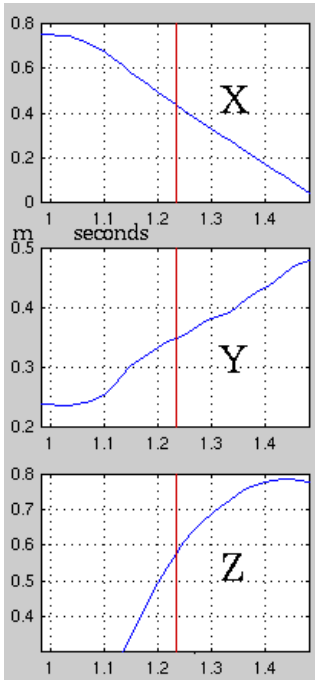

Fig. 7. A measured ball trajectory. Note the noise in the Y component of the position data.

number of vision frames. A blend of the current prediction method with Kalman filtering may be a better solution if predictions with Kalman filtering prove to be more accurate after the startup transient. The current algorithm could be used for early predictions, and a Kalman filter used for later predictions.

Besides changes to account for noise, learning could be included in the catching task. Currently, the task allows the user to set the following parameters: the desired hand distance threshold to signal flight of the ball, the ball impact height for the prediction algorithm, and the minimum number of vision frames necessary to begin predicting the ball-glove impact time and place for the given height. These parameters can instead be decided by the robot using decision rules or optimizing some criteria relating to the desired behavior. For example, if the robot desires more time to catch the ball, it may decide to lower the target impact height.

Studies such as [7] may yield useful ideas for humanoid catching. Like [7] we are also analyzing human motion capture catching data. Our aim is to determine human strategies that may improve the robot's behavior. One possible strategy is minimizing the relative velocities of the hand and ball as a catch is being made, instead of simply heading to the goal position. Another strategy is coming up under the ball to ensure that the follow through helps keep the ball in the glove.

We wish to improve the naturalness of the catching movement. People normally move their head as they track an object's flight. Here, although we are using external cameras as the robot's eyes, we will implement simple head movements corresponding to the ball movement to give the feeling that the robot is watching the ball's trajectory with the head-mounted cameras. In the future, we will implement the catching task using vision data from these head-mounted cameras.

In addition to the head, the entire body movement during a catch should look natural. In this work, we have observed two broad types of robot catches: inside catches, where the robot reaches toward its body, and outside, where it extends its arm away from its body to catch the ball. The inside catch often requires pulling the arm back while rotating around the elbow, while the outside catch requires extension of the arm from the shoulder and elbow. At present, one default posture for all the robot's joints is used to resolve redundancies in the inverse kinematics solver. To ensure appropriate inverse kinematic solutions over the entire body for a given type of catch, a preferred posture corresponding to the predicted location of the catch can be given to the inverse kinematics solver, thus biasing the solution toward more appropriate postures.

4 Conclusion

In creating interesting behaviors for the humanoid robot in our laboratory we consider approaches including generating new movements and drawing from a library of human examples. As was discussed earlier, generating new

Fig. 8. A frame of motion showing the end of a catching sequence.

movements requires a choice of movement primitives and some choices about the characteristics of the desired trajectories.

When using a template-based approach, we must address the issue of how to adapt human behavior to humanoids. Synthesizing motion for models of different kinematic and dynamic structures has been an interesting topic for many researchers [4, 5]. Motion generalization realized automatically or semi-automatically is valuable for computer animation and the gaming industry, where reuse of motion on multiple characters is advantageous. This approach can also be used to collect a large library of movements, to be replayed later. Many video games rely on this type of approach.

When changing from the human to the humanoid we must decide what features of the movement to preserve. How do we identify these features? How can we parameterize the motion, and describe it to the robot or link it with some criteria which will produce the desired effect? We have dealt with some of these issues in our dancing motion synthesis [14] [27], but there is still much to be explored.

Once we have created the behaviors, we need a way to evaluate them. In the future we will work on evaluation and on coaching, where the desired features are requested at a high level and then mapped onto low level primitives to accomplish the motion.

Being able to faithfully reproduce human-like movements for virtual characters and humanoid robots from human data helps create engaging human-machine interactions. Humanoid robots especially provide a rich and challenging platform for implementing such behaviors. In this work, we discuss our methods for implementing ball catching for a humanoid robot which rely on a dynamical systems approach to generate the movements. These movements are based on motion primitives derived from studies of human motion where the target position is given by an iterative impact prediction algorithm. We suggest various strategies to improve the robustness of these methods, and also discuss template-based approaches for behavior generation.

Videos showing our humanoid robot catching can be viewed at http://www.cc.gatech.edu/∼mriley.

Acknowledgments: Support for both investigators was provided by the ATR Human Information Processing Research Laboratories, ATR Information Sciences Division, and by National Science Foundation Award IIS-9711770.

References

1. C. G. Atkeson and S. Schaal, “Robot Learning From Demonstration”, *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)*, Edited by Douglas H. Fisher, Jr. pp. 12-20, Morgan Kaufmann, San Francisco, CA, 1997.
2. M. Buehler, D.E. Koditschek, P.J. Kindlmann, Planning and control of robotic juggling and catching tasks. *International Journal of Robotics Research*, vol.13, no.2, pages 101-108, April 1994.
3. R. R. Burridge, A. A. Rizzi, and D.E. Koditschek, Toward a Dynamical Pick and Place, *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 95*, 2:292–297, 1995.
4. M. Gleicher, Retargeting motion to new characters. In *Computer Graphics, Proc. SIGGRAPH '98*, pages 33-42, July 1998.
5. J. K. Hodgins and N.S. Pollard, Adapting Simulated Behaviors For New Characters. In *Computer Graphics, Proc. SIGGRAPH 97*, Los Angeles, CA, 1997.
6. B. Hove and J.-J.E. Slotine, Experiments in robotic catching. In *Proc. IEEE American Control Conference*, pages 380-385, Boston, MA, June 1991.
7. S. Kajikawa, M. Saito, K. Ohba, H. Inooka, Analysis of human arm movement for catching a moving object. In *Proc. IEEE Conference on Systems, Man, and Cybernetics*, pages 698-703, Tokyo, Japan, October 1999.

8. K. M. Lynch and M. T. Mason, Dynamic Nonprehensile Manipulation: Controlability, Planning, and Experiments, *International Journal of Robotics Research*, 18(1):64–92, 1999.
9. P. Maybeck, *Stochastic Models, Estimation, and, Control, Volume 1*, Academic Press, 1979.
10. M. Mehrandezh, N. M. Sela, R. G. Fenton, and B. Banhabib, Robotic Interception of Moving Objects Using an Augmented Ideal Proportional Navigation Guidance Technique, *IEEE Transactions on Systems, Man, and Cybernetics* 30(3):238–250, 2000.
11. H. Miyamoto, Schaal S, Gandolfo F, Gomi H, Koike Y, Osu R, Nakano E, Wada Y, Kawato M, A Kendama learning robot based on bi-directional theory, *Neural Networks* 9;1281–1302, 1996.
12. J. Nakanishi, T. Fukuda, and D. E. Koditschek, A Brachiating Robot Contoller, *IEEE Transactions on Robotics and Automation*, 16(2):109–123, 2000.
13. C. Nass, J. Steuer, and E. Tauber, Computers are Social Actors. In *Proc. CHI '94*, pages 72-78, April 1994.
14. M. Riley, A.Ude, C.G. Atkeson, Methods for Motion Generation and Interaction with a Humanoid Robot: Case Studies of Dancing and Catching, In *Proc. AAAI and CMU Workshop on Interactive Robotics and Entertainment 2000*, Pittsburgh, Pennsylvania, April 2000.
15. A. A. Rizzi and D. E. Koditschek, An Active Visual Estimator For Dextrous Manipulation, *IEEE Transactions on Robotics and Automation*, 12(5): 697–713, 1996.
16. T. Sakaguchi, Y. Masutani, and F. Miyazaki, A Study On Juggling Task, *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 91*, 1418–1423, 1991.
17. T. Sakaguchi, M. Fujita, H. Watanabe, and F. Miyazaki, Motion Planning and Control for a Robot Performer, *IEEE International Conference on Robotics and Automation*, 925–931, 1993.
18. T. Sakaguchi and F. Miyazaki. Dynamic manipulation of ball-in-cup game. *IEEE International Conference on Robotics and Automation*, 2941–2948, 1994.
19. S. Sato, T. Sakaguchi, Y. Masutani, and F. Miyazaki, Mastering a task with interaction between robot and its environment (kendama task). *J. Advanced Automation Technology* 6(1):34–41, 1994.
20. S. Schaal and C. G. Atkeson, Open Loop Stable Control Strategies For Robot Juggling, *IEEE International Conference on Robotics and Automation*, 3:913–918, 1993.
21. S. Schaal, C. G. Atkeson, and S. Botros, What Should Be Learned? *Seventh Yale Workshop on Adaptive and Learning Systems*, 199–204.
22. S. Schaal and D. Sternad, Programmable pattern generators. International Conference on Computational Intelligence in Neuroscience (ICCIN'98), pages 48-51. Research Triangle Park, NC, October 1998.
23. A. Sloman and M. Croucher, Why robots will have emotions. In *Proc. 7th International Joint Conference on Artificial Intelligence*, Vancouver, 1981.
24. M. W. Spong, Swing Up Control of the Acrobot, *IEEE International Conference on Robotics and Automation*, 2356–2361, 1994.
25. K. Takenaka, Dynamical Control of Manipulator with Vision (Cup and Ball Game Demonstrated by Robot), *The Transactions of the Japan Society of Mechanical Engineers* C:50 1984.
26. Tevatia, G., and Schaal, S., Inverse kinematics for humanoid robots. In *Proc. IEEE International Conference on Robotics and Automation*, San Francisco, CA, April, 2000.
27. A. Ude, C. G. Atkeson, M. Riley, Planning of joint trajectories for humanoid robots using B-spline wavelets. In *IEEE Conference on Robotics and Automation*, San Francisco, CA, April 2000.