

# Sensor-Based Online Trajectory Generation for Smoothly Grasping Moving Objects.

M. Zhang and M. Buehler

Centre for Intelligent Machines  
McGill University, Department of Mechanical Engineering  
zhangmei@cim.mcgill.edu      buehler@cim.mcgill.edu

## Abstract

We present a new approach to online trajectory planning for target tracking, dynamic grasping and catching. The robot executes a geometric controller – it simply evaluates a nonlinear function which maps the currently measured position and velocity of the object to be grasped into a current desired robot pose. If the robot tracks these setpoints, it is guaranteed to match the object's velocity and acceleration on a specified grasp surface. We develop a geometric controller which specifies the full 6dof position and orientation of the robot's end effector. A planar simulation demonstrates that this paradigm performs favorably when compared with the traditional planning approach. Since it does not depend on future object measurements, no object model is needed for trajectory prediction. Without trajectory prediction, the computational effort is drastically reduced, allowing for higher controller speed and tracking feedback gains. At the same time this approach provides a framework for general sensor based control of robotic tasks.

## 1. Introduction

We define dynamic grasping as the approaching of a moving object and final velocity and acceleration matching at a desired grasp location. It is a fundamental task that precedes many manipulation operations and is by itself the main constituent of a large class of robotic tasks. In manufacturing, robots pick up objects from conveyor belts and J-hooks, and will in the near future have to grasp objects with à priori unknown trajectories handed to them from humans or other robots. In space, this problem occurs when capturing satellites or when containing and docking space assemblies.

These tasks belong to a larger class of tasks which require "dynamical dexterity" and which are one of the key challenges in robotics. Dynamic manipulation, catching, juggling, walking, running, brachiating are all confronting this challenge. Despite the apparent ease with which humans accomplish them, they still represent a myriad of difficult research challenges. We must integrate sensing, filtering, sensor fusion, planning, modelling, control, all under hard real time con-

straints, to succeed.

One of the first accounts of dynamic grasping in 2D was given by Andersson [1]. He described and implemented an approach where rolling ping pong balls on a slanted surface were caught. A straight line was fitted to the ball trajectory which was intersected with an "intercept line" to determine the catch point. This catch point was selected to be reachable by the robot in advance of the ball. Velocity or acceleration matching were not attempted. Lin, Zeman and Patel [10] proposed a heuristic procedure which is divided into two parts. A coarse tuning algorithm first drives the end-effector into the neighborhood of the object in near-minimum time and a fine tuning algorithm then provides precise matching of the object's trajectory. Houshangi [8] addressed explicitly the time delay introduced by image processing in visual feedback controlled robots by predicting the object trajectory via an auto-regressive discrete-time model. A planner adapted online to changes in the target position. Hove and Slotine [9] implemented spatial robot catching based on a trajectory matching algorithm that combines an observer with a varying-strength filter, an error estimator, and an initial motion algorithm. They applied this approach successfully to catching balls with the four degree-of-freedom WAM cable driven arm. Allen et al. [6] grasp a moving train with a PUMA under vision guidance based on straight line planning in cartesian space between the robot's end effector and the target. Gosselin's work on global inverse robot kinematics [7] includes a path planning scheme to catch moving objects: The last object state measurements are employed to predict its path as a parabola, assuming a free falling object. The closest point along this trajectory to the robot's base is then selected as the catching point. Quintic polynomials are then planned in joint space between the current and the catch configuration. The most recent version of the classical sense-plan-replan paradigm is found in [5] where an object moving in the plane is caught based on a predicted path assuming constant acceleration. The problem of blending trajectory segments in real time subject to sensory inputs and tracking constraints is solved by Lloyd and Hayward [11].

The paper is outlined as follows. In Section 2 we pose the smooth grasping problem. We then review the classical "replanning" approach in Section 3 and illustrate the basic idea of the geometric control approach with a simple example in Section 4. The new geometric controller for smooth grasping is derived in Section 5 and then compared to the replanning approach in Section 6.

## 2. Problem Formulation

Let

$$\mathbf{r}_d = \begin{bmatrix} r_{d,x} \\ r_{d,y} \\ r_{d,z} \end{bmatrix}, \mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \in \mathbb{R}^3$$

represent the desired and actual robot end effector position and the actual object position, respectively. The end point position error vector is

$$\mathbf{e}_b = \mathbf{r}_d - \mathbf{b}$$

and the tracking error

$$\mathbf{e}_r = \mathbf{r}_d - \mathbf{r}.$$

We translate the temporal specification for making a grasp contact into a geometric one by assuming that the contact time  $t_c$  is the time when the object traverses the horizontal "grasp surface"

$$b_z(t_c) = 0. \quad (1)$$

Depending on the task and the robot's workspace, other surfaces can be selected.

A **smooth grasp** of a moving object is defined as a match between the position, velocity and acceleration of the object and the robot's end effector on the grasp surface, i.e.

**Condition 1:**

$$\mathbf{e}_b(t_c) = \dot{\mathbf{e}}_b(t_c) = \ddot{\mathbf{e}}_b(t_c) = \mathbf{0}. \quad (2)$$

In addition, in order to avoid pre-grasp collisions between robot and object, we require that

**Condition 2:**

$$r_{d,z}(t) < b_z(t) \quad \forall t < t_c. \quad (3)$$

Just like in the related literature, the tracking problem is not directly addressed here. However, in order to facilitate tracking, the initial desired and actual robot positions, velocities and accelerations at  $t_0$  should match,

**Condition 3:**

$$\mathbf{e}_r(t_0) = \dot{\mathbf{e}}_r(t_0) = \ddot{\mathbf{e}}_r(t_0) = \mathbf{0}. \quad (4)$$

Since we are focussing on the online trajectory generation problem, we also make the following assumptions:

1. The inverse kinematics, possible singularities and interference with the workspace boundaries and other obstacles are dealt with elsewhere. There are several solutions available, for example [7].
2. A motion tracking system provides the object's cartesian position and associated filtering software to derive the velocities. Such a system based on real time stereo vision to provide the inputs to a geometric controller for robot juggling has been successfully implemented in [12].
3. The robot end effector is equipped with a gripper which can be closed when the smooth grasp conditions are satisfied.

## 3. Review of the Replanning Approach

In case the object's trajectory is known exactly a priori, its intersection with the desired contact surface, and the time of contact,  $t_c$ , can be computed. Now a quintic polynomial of time can be constructed which satisfies initial and final position, velocity and acceleration conditions in task or joint space and achieves a smooth grasp.

The problem with this approach is that in practice, the exact future object trajectory is never available accurately enough for smooth grasping purposes. Therefore the robot trajectory has to be continuously replanned based on the latest sensory information. This in effect leads to a highly inefficient implementation, since only the initial small portion of the planned trajectory is used. This problem is exacerbated if no information about the object's dynamics is available to the robot trajectory planner. In this case, typically a straight line (constant velocity) or parabola (constant acceleration) is assumed.

## 4. Geometric Controller Background

The motivation for this paper originates from earlier work on robot juggling [3] where the groundwork for geometric controllers was laid and where they were applied successfully to juggling one and two balls in the plane [2]. This concept was extended further by Rizzi and Koditschek [12] to spatial juggling of two balls with a three degree-of-freedom direct drive robot, guided by a real time stereo vision system.

It turns out that a simplified version of the geometric controller for juggling can accommodate a velocity and acceleration matching contact with a moving object at a desired position. To illustrate the basic idea, consider the trivial 1 dof prismatic robot in Figure 1 with vertical coordinate  $r$  and an object moving along a line, with height  $b$ .

A geometric controller for the desired robot height  $r_d$  to smoothly grasp the object would be

$$r_d = \kappa \cdot \text{atan}(b/\kappa), \quad (5)$$

where  $\kappa > 0$  is a user selectable gain. For purposes of improved tracking, the desired robot velocity is available to the PD controller as well:

$$\dot{r}_d = \kappa \frac{\dot{b}/\kappa}{1 + (b/\kappa)^2}. \quad (6)$$

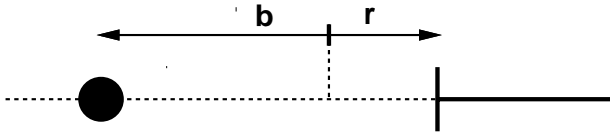


Figure 1. 1 dof illustration

Assuming perfect tracking,  $r(t) = r_d(t)$  robot-object contacts can only occur when  $r = b = 0$ , and at that point a velocity matching contact is guaranteed,

$$\dot{r}_c = \dot{r}(r = b = 0) = \kappa \frac{\dot{b}/\kappa}{1 + (b/\kappa)^2} = \dot{b}_c.$$

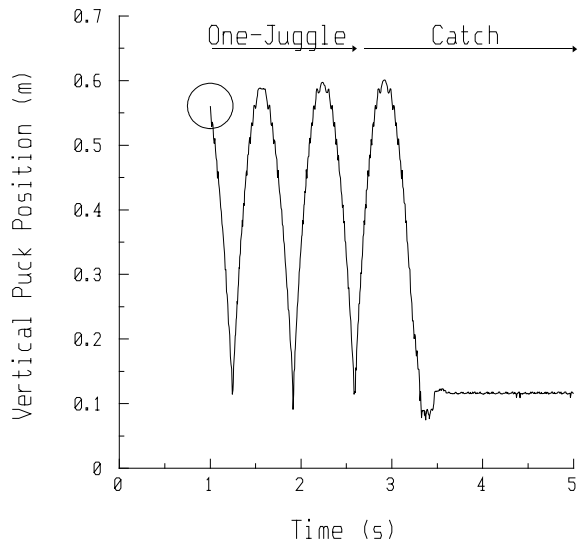


Figure 2. Juggle followed by a catch (5), from [4].

The object's acceleration is matched as well:

$$\begin{aligned} \ddot{r}_c = \ddot{r}(r = b = 0) &= \frac{\ddot{b}}{1 + (b/\kappa)^2} - \frac{2(b/\kappa^2)\dot{b}^2}{(1 + (b/\kappa)^2)^2} \\ &= \ddot{b}_c \end{aligned}$$

A successful experimental implementation of this geometric controller, taken from [4] is shown in Figure 2 where it is preceded by a vertical one-juggle, controlled by a similar algorithm.

The controller is called geometric since it contains no dynamics itself, it is simply a map from current object phase space to a desired current robot state. All the planning to ensure smooth grasping is done off line, when designing the controller. Notice that

this approach completely eliminates the online trajectory planning in the traditional sense. The task is encoded in (5) and is achieved by tracking the set-points computed by the same equation at every instant. No assumptions were made about the object's trajectory, and the algorithm will succeed as long as tracking is achieved. In the next section we will generalize this basic controller (5) for dynamic grasping in six degrees-of-freedom.

## 5. A New Geometric Controller

The simple geometric catch controller (5) has two shortcomings. First, it needs to be extended for spatial grasping. Second, at  $t = 0$  there might be a large error  $e_r$  between the commanded and the actual robot position, which leads to large transient torque requirements and might ultimately prevent a sufficiently small asymptotic tracking error at time of contact  $t_c$ , which would violate Condition 3 (4). It turns out that there exist many functions that would result in a velocity and acceleration matching contact, in a similar fashion to (5), and a few can even satisfy the additional conditions (4) and (3). However, the construction of polynomials is the simplest solution and allows a procedure equivalent to spline generation. In the following sections we will derive a spatial geometric controller based on polynomials of the object position for the vertical (Section 5.1) and the horizontal (Section 5.2) components.

### 5.1. Vertical Component

A quintic function of the vertical object position satisfies all the grasping conditions:

$$r_{d,z}(b_z) = \sum_{n=0}^5 \kappa_{z,n} b_z^n = q_z(b_z). \quad (7)$$

The three gains  $\kappa_{z,0}, \kappa_{z,1}, \kappa_{z,2}$  can be obtained from Condition 1 (2). For the vertical robot position at contact we obtain from (7)

$$r_{d,z}(t_c) = q_z(b_z(t_c)) = q_z(0) = \kappa_{z,0}$$

and to satisfy the grasping condition (2)

$$r_{d,z}(t_c) = b_z(t_c) = 0$$

requires  $\kappa_{z,0} = 0$ . Similarly, the velocity and acceleration conditions result in  $\kappa_{z,1} = 1$  and  $\kappa_{z,2} = 0$ . The remaining three gains  $\kappa_{z,3}, \kappa_{z,4}, \kappa_{z,5}$  are derived from the three initial conditions (4):

$$r_{d,z}(t_0) = q_z(b_{z,0}) = r_{z,0}$$

with

$$r_{d,z,0} = r_{d,z}(t_0), \quad r_{z,0} = r_z(t_0).$$

The velocity and acceleration initial conditions result in

$$\dot{q}_z(b_{z,0}) = \dot{r}_{z,0}, \quad \ddot{q}_z(b_{z,0}) = \ddot{r}_{z,0}.$$

The solution for the last three gains can be expressed as

$$\begin{bmatrix} \kappa_{z,3} \\ \kappa_{z,4} \\ \kappa_{z,5} \end{bmatrix} = \mathbf{B}_{z,0}^{-1} \cdot \begin{bmatrix} r_{z,0} - b_{z,0} \\ \dot{r}_{z,0} - \dot{b}_{z,0} \\ \ddot{r}_{z,0} - \ddot{b}_{z,0} \end{bmatrix}$$

where

$$\mathbf{B}_{z,0} = \begin{bmatrix} b_{z,0}^3 & b_{z,0}^4 \\ 3b_{z,0}^2\dot{b}_{z,0} & 4b_{z,0}^3\dot{b}_{z,0} \\ 6b_{z,0}\dot{b}_{z,0}^2 + 3b_{z,0}^2\ddot{b}_{z,0} & 12b_{z,0}^2\dot{b}_{z,0}^2 + 4b_{z,0}^3\ddot{b}_{z,0} \\ b_{z,0}^5 \\ 5b_{z,0}^4\dot{b}_{z,0} \\ 20b_{z,0}^3\dot{b}_{z,0}^2 + 5b_{z,0}^4\ddot{b}_{z,0} \end{bmatrix}. \quad (8)$$

Subscript 0 refers to the initial condition. We have now obtained the desired robot trajectory which satisfies the grasping conditions (2) and (4). Notice that this computation is performed only once at  $t = 0$ , and subsequently only the evaluation of the polynomial (7) is required (in addition to its derivative for tracking purposes).

If the initial object velocities and accelerations are not available, a third order polynomial is sufficient to match the initial position only. In this case,  $\kappa_{z,3} = (r_{z,0} - b_{z,0})/b_{z,0}^3$  and the geometric controller for the vertical component takes on the extremely simple form

$$r_{d,z} = b_z + \kappa_{z,3}b_z^3. \quad (9)$$

Note that Condition 2 (3) is satisfied by (9) as long as it is satisfied at the initial time,

$$\kappa_{z,3} = (r_{z,0} - b_{z,0})/b_{z,0}^3 < 0.$$

## 5.2. Horizontal Components

In this section, the algorithm for online generation of a robot reference trajectory is developed which achieves velocity and acceleration matching for the  $r_x$  and  $r_y$  components. The critical task here is that this matching must be accomplished at the object's spatial position and **at the same time**  $t_c$  when the vertical component is contacting the object. This is an illustrative example how purely geometric controllers are used to achieve time synchronization. Time synchronization between two processes has been achieved once before with purely geometric controllers, however in an entirely different fashion. In [4] it was critical for juggling two balls which required the control of a certain phase relationship between them. For brevity, only the  $r_x$  component is developed. The algorithm is identical for the  $r_y$  component.

In order to satisfy the grasping conditions at  $t_c$  and the initial condition at  $t_0$  in the x coordinate, a quintic polynomial of the **combined** ball positions in x and z coordinates must be used.

$$r_x = b_x + \sum_{n=3}^5 \kappa_{x,n} b_z^n \quad (10)$$

satisfies all the requirements, since on the contact surface  $b_{z,c} = 0$ , and thus

$$\dot{r}_{x,c} = \dot{b}_{x,c} + \dot{b}_{z,c} \sum_{n=3}^5 n\kappa_{x,n} b_{z,c}^{n-1} = \dot{b}_{x,c} \quad (11)$$

$$\begin{aligned} \ddot{r}_{x,c} &= \ddot{b}_{x,c} + \dot{b}_{z,c}^2 \sum_{n=3}^5 n(n-1)\kappa_{x,n} b_{z,c}^{n-2} + \ddot{b}_{z,c} \sum_{n=3}^5 n\kappa_{x,n} b_{z,c}^{n-1} \\ &= \ddot{b}_{x,c}. \end{aligned} \quad (12)$$

The solution for the last three gains can be expressed from (10), (11), (12) and Condition 3 (4) as

$$\begin{bmatrix} \kappa_{x,3} \\ \kappa_{x,4} \\ \kappa_{x,5} \end{bmatrix} = \mathbf{B}_{z,0}^{-1} \cdot \begin{bmatrix} r_{x,0} - b_{x,0} \\ \dot{r}_{x,0} - \dot{b}_{x,0} \\ \ddot{r}_{x,0} - \ddot{b}_{x,0} \end{bmatrix}$$

with  $\mathbf{B}_{z,0}$  as defined in (8).

Again, if only initial position matching is required, we can use the third order polynomial version of (10). The constant  $\kappa_{x,3}$  is determined as above,  $\kappa_{x,3} = (r_{x,0} - b_{x,0})/b_{z,0}^3$  and

$$r_{d,x} = b_x + \kappa_{x,3}b_z^3. \quad (13)$$

Notice that there is no interference problem since the vertical controller assures Condition 2, namely that the robot's end effector is always below the object. So there are no spatial restrictions on the horizontal components.

## 5.3. Spherical coordinates

For common non-Cartesian manipulators, it would be advantageous to express the desired robot end point and the object position in spherical coordinates  $\mathbf{r} = [r_r, r_\theta, r_\phi]$ ,  $\mathbf{b} = [b_r, b_\theta, b_\phi]$ . Now the simplest catch surface equivalent to (1) would be a sphere around the origin, within the robot's workspace,  $b_r(t_c) = r_0$ . The geometric controller for  $r_r(b_r)$  is now derived analogously to the vertical component above (7). The geometric controllers both  $r_\theta(b_r, b_\theta)$  and  $r_\phi(b_r, b_\phi)$  are analogous to the horizontal component  $r_x(b_z, b_x)$  (10) where the subscripts  $x, y, z$  are replaced by  $r, \theta, \phi$ , respectively.

## 5.4. Orientation

For any practical implementation, the end effector orientation, angular velocity and acceleration should be matched to the object's at time of contact as well. In addition, the initial desired end effector orientation should match the actual orientation. In the Cartesian case this is implemented in the geometric controller framework by specifying the Euler angles of gripper ( $r_\phi, r_\theta, r_\chi$ ) with respect to the base coordinates as a polynomial of the object's Euler angles w.r.t. the gripper frame (Figure 3) ( $b_\phi, b_\theta, b_\chi$ ) expressed in the base

coordinates and the vertical object component  $b_z$

$$r_i = b_i + \kappa_{i,3} b_z^3; \quad \kappa_{i,3} = \frac{r_{i0} - b_{i0}}{b_{z0}^3} \quad i = \phi, \theta, \chi. \quad (14)$$

For illustration Figure 4 shows the planar case in the vertical x-z-plane. Corresponding robot-object trajectory points are connected via a dashed line, and the robot's end effector orientation is indicated by a short solid line. Notice how the initially large orientation error is eliminated by (14) to match the object's angle, angular velocity and angular acceleration with respect to the robot's end effector.

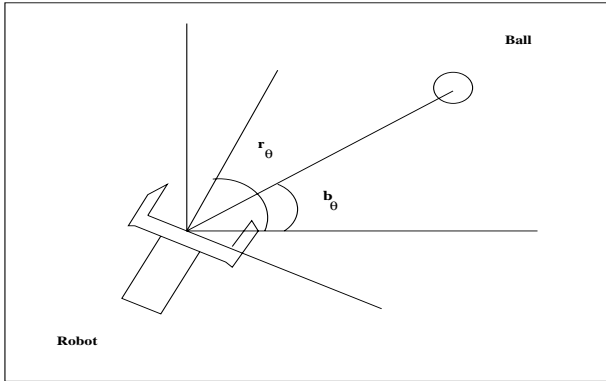


Figure 3. End Effector Orientation Control Variables

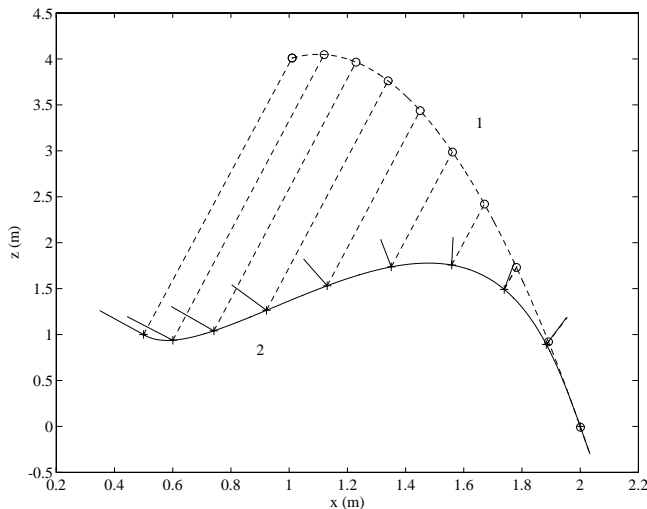


Figure 4. Orientation Control for Parabolic Object Trajectory, 1: object, 2: robot

## 6. Comparisons

In this section we compare the performance of the geometric control approach to smooth grasping as described above to the more conventional trajectory planning approach described in Section 3. Both versions, which assume either constant velocity or acceleration

for estimating the future object trajectory are evaluated. For ease of exposition we limit ourselves to the planar problem, and endpoint position only. The results for the full spatial case with orientation would be analogous. Therefore the geometric control is completely described by equations (9) and (13)

$$r_{d,z} = b_z + \kappa_{z,3} b_z^3, \quad r_{d,x} = b_x + \kappa_{x,3} b_z^3.$$

The results are shown in Figure 5 for a parabolic object trajectory, Figure 6 and 7 for two sinusoidal object trajectories of different frequencies. In all three plots the object trajectory is trajectory 1 (dashed line), the geometric controller is trajectory 2 (solid line), the replanning approach based on constant acceleration object prediction is trajectory 3 (dashed dot) and the one on constant velocity object prediction is trajectory 4 (dot).

Overall, the robot trajectories generated by all three approaches are quite similar, despite the drastic difference between the replanning and the geometric controller. However, the geometric controller performs more closely to the better one of the replanning approaches. For example, for the parabolic object trajectory (Figure 5), the replanning approach assuming constant object acceleration (3) is predicting correctly. In this case the geometric controller (2) results in almost the same trajectory. In the case of the sinusoidal object trajectories (Figures 6 and 7), the replanning approach assuming constant object acceleration (3) predicts incorrectly, and the constant velocity (4) replanning approach results in a shorter, smoother trajectory. In this case, the geometric approach (2) results in a robot reference trajectory similar or better than the constant velocity approach.

## 7. Conclusion

We have presented a sensor based approach to spatial online trajectory generation for smooth dynamic grasping, including orientation of the end effector. The new geometric controller is based only on current object state measurements and requires no information about the object dynamical model nor its future trajectory.

## References

- [1] R. L. Andersson. Real time intelligent visual control of a robot. In *IEEE Workshop on Intelligent Control*, Aug 1986.
- [2] M. Buehler, D. E. Koditschek, and P. J. Kindlmann. A Simple Juggling Robot: Theory and Experimentation. In V. Hayward and O. Khatib, editors, *Experimental Robotics I*. Springer-Verlag, 1989.
- [3] M. Buehler, D. E. Koditschek, and P. J. Kindlmann. A Family of Robot Control Strategies for Intermittent Dynamical Environments. *IEEE Control Systems Magazine*, 10(2), Feb 1990.
- [4] M. Buehler, D. E. Koditschek, and P. J. Kindlmann. Planning and Control of Robotic Juggling and Catch-

ing Tasks. *Int. J. Robotics Research*, 13(2):101–118, 1994.

- [5] G. Buttazzo, B. Allotta, and F. Fanizza. Mousebuster: A robot for real-time catching. *IEEE Control Systems Magazine*, 14(1), Feb 1994.
- [6] P. Allen et al. Trajectory filtering and prediction for automated tracking and grasping of a moving object. In *Proc. IEEE Int. Conf. Robotics and Automation*, May 1992.
- [7] C. Gosselin, J. Côté, and D. Laurendeau. Inverse kinematic functions for approach and catching operations. *IEEE Trans. Systems, Man, and Cybernetics*, 23(3), 1993.
- [8] N. Houshangi. Control of a robotic manipulator to grasp a moving target using vision. In *Proc. IEEE Int. Conf. Robotics and Automation*, 1990.
- [9] B. Hove and J.-J. E. Slotine. Experiments in robotic catching. In *American Control Conference*, 1991.
- [10] Z. Lin, V. Zeman, and R. V. Patel. On-line robot trajectory planning for catching a moving object. In *Proc. IEEE Int. Conf. Robotics and Automation*, 1989.
- [11] J. Lloyd and V. Hayward. Trajectory generation for dynamic, sensor-driven environments. *Int. J. Robotics Research*, 12(4), 1993.
- [12] A. A. Rizzi and D. E. Koditschek. Further progress in robot juggling: The spatial two-juggle. In *Proc. IEEE Int. Conf. Robotics and Automation*, Atlanta, Georgia, May 1993.

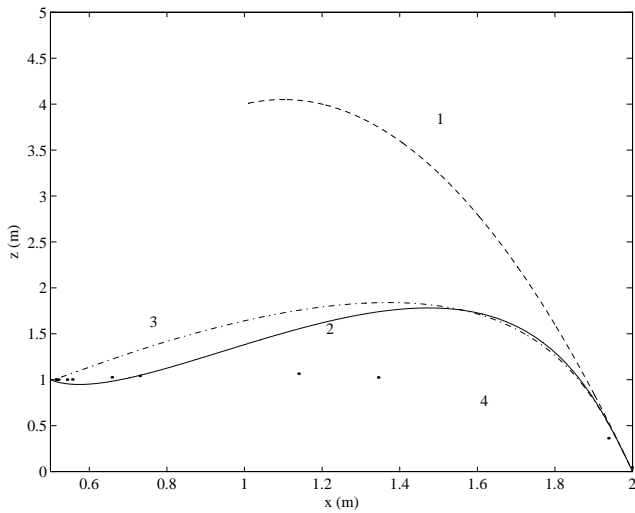


Figure 5. Parabolic Object Trajectory, (1: object, 2: geometric controller, 3: replanning, const. acceleration, 4: replanning, const. velocity)

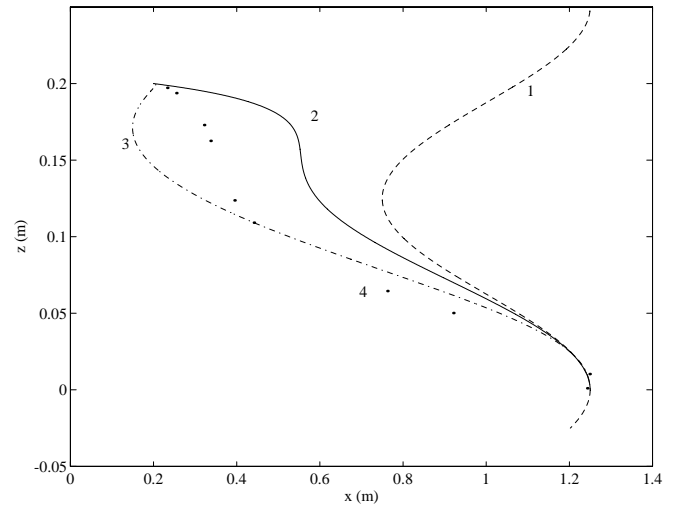


Figure 6. Sinusoidal Object Trajectory,

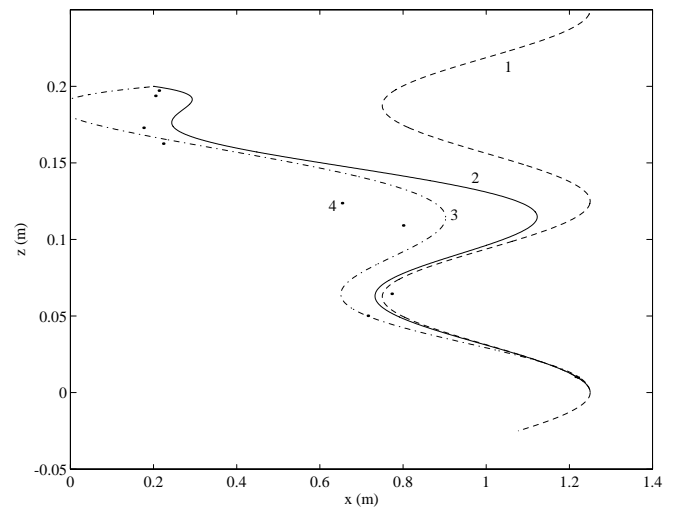


Figure 7. Sinusoidal Object Trajectory,