Robotic Juggling

MIT 6.881 - Intelligent Robot Manipulation Final Project

Matt Beveridge
Department of EECS
Massachusetts Institute of Technology
Cambridge, MA, United States
mattbev@mit.edu

Ryan Shubert
Department of EECS
Massachusetts Institute of Technology
Cambrdige, MA, United States
rshubert@mit.edu

***Abstract*—In this paper we implement stable juggling of a ball in closed-loop by a single robotic arm and with a paddle end effector. We control the system using mirrored kinematics between the ball and end effector, motion planning to guide the juggled ball to a stable position, and first-order control to command joint velocities for orienting the paddle in relation to the ball and the space around it. We additionally examine adversarial scenarios unique to this problem and implement subsystems to resolve them. The proposed juggling arm is capable of maintaining a stable juggle about a central axis, resolving (reasonable) perturbations in the starting position of the ball.**

***Index Terms*—Robotic Manipulation, Motion Planning, Controls**

I. Introduction

There has been widespread interest in robotic manipulation problems in the past several decades. Many of these focus on systems that are not chaotic and are rather predictable. While these problems are challenging in their own way, manipulation in chaotic environments poses a new set of difficulties that did not previously have to be considered. In this paper, we explore one such chaotic environment: juggling of a ball. Besides the theoretical interest of robotic juggling, the results are applicable in a multitude of other areas concerning prehensile manipulation in a dynamic environment.

We implement a closed-loop system in simulation for robotic juggling of a ball. We use a KUKA IIWA as our robotic arm and an attached rigid paddle to make contact with the ball. We use differential pseudo-inverse kinematics to command velocities within the IIWA arm, and use mirrored kinematics relative to the ball to orient the paddle within the world. We handle xy linear and angular velocity of the end effector with first-order and proportional control respectively, mirror linear velocity in the z direction with logarithmic scaling, and omit control of angular velocity in the z direction. In doing so, we are able to create a system that stably juggles the ball about a central axis regardless of the starting position of the ball.

Code for this project is openly available on Github[1] and a video of the juggler is available on YouTube[2].

Special thanks to Prof. Russ Tedrake, Hyung Ju Suh, and Meng Feng.

[1]Available code: https://github.com/mattbev/robot-juggler

[2]Example video: https://youtu.be/cmH_37cLE1o

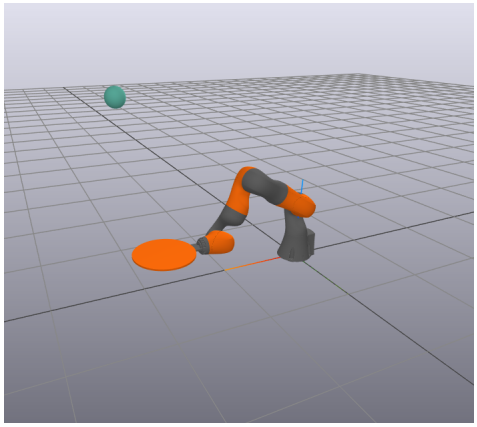

Fig. 1. The full manipulation station setup, including KUKA IIWA arm, paddle end effector, and ball. Note: the ball begins offset from the paddle, and the system accounts for such perturbations in starting pose.

II. Related Work

The state-of-the-art in robotic juggling involves the use of humanoid hands as end effectors. For example, in [1], we find that the catching of an object using a robotic arm can be reduced to a nonlinear optimization problem. The authors use a 7 degree of freedom (DOF) arm with a 12 DOF hand are used in tandem (19 total) and achieve a catch rate greater than 80%. This system is more complex than ours, and uses a cluster of 32 CPU cores to perform the necessary calculations to manage the system. However, one does not need such complexity to examine an interesting problem. It has been shown that it is possible to juggle with as little as a single degree of freedom [2], and in our case we extend this to 7 DOF. In such an environment we are able to explore a dynamically changing environment while keeping computation cheap.

III. Methodology

A. Experimental Setup

Our system consists of three parts: (1) the KUKA IIWA robotic arm, (2) the rigid paddle end effector, and (3) the ball. The end effector and ball should have high restitution coefficients as to simulate elastic collisions, however this is not a factor in Drake — the framework in which our system

is implemented. Further, we assume that both the paddle and ball have low friction (e.g. friction coefficient of zero) in our tests. See Figure 1 for a visualization of the setup.

In practice, we drop a ball from height and the control system will manipulate the arm to catch and juggle the ball above the paddle around the desired central axis. The central axis is in the z direction, meaning we constrain the x and y position of where the ball is juggled but not the height to which the ball is hit, the z position. This is akin to juggling a ping pong ball with a ping pong paddle. To test robustness of the system, we vary the starting position along the x and y axes to create different dynamic environments in simulation. A robust system converges to juggling the ball around a center point in the xy plane regardless of the starting position of the ball, given it is within reach of the arm. We omit variation in the z axis as it is less informative about the stability of the system as a whole and has negligible impact on performance of the final system. This also lends to the assumption that the robot is on a pedestal or some sort of raised platform, i.e. the arm and ball can go lower than $z = 0$.

B. Assumptions

In the setup of our manipulation station, we make the assumption that we have access to the known ball position p^B and velocity v^B at a given time step. In doing this we eliminate the need for a perception system that would further complicate the system.

C. Control

In order to juggle the ball, we implement a differential psuedo-inverse kinematic controller. In this we use the Jacobian with respect to the general velocity to command angular and linear velocity to the KUKA IIWA. However, in order to do so, we implemented two lower-level controllers to handle these tasks separately.

1) Velocity Mirroring: To follow the trajectory of the ball in space, we implement velocity mirroring. In essence this means matching the velocity of the ball along one axis, in our case the xy axis, and flipping the sign along the third axis, in our case the z axis. Purely, this means:

$$\begin{aligned} v_x^P &= v_x^B \\ v_y^P &= v_y^B \\ v_z^P &= -v_z^B \end{aligned}$$

Where v_x^P is the velocity of the paddle in the x axis, v_x^B is the velocity of the ball in the x axis, and so on.

However, in practice, exactly commanding velocities that mirror the ball's velocity in this way accumulates error in space. To fix this, we expand this to a first-order dynamical system that takes into account the positions of the ball and paddle. The first order system operates only on the x and y velocities as a different approach is taken for the z velocity. We don't use the first order system for the z velocity as the z position of the ball doesn't inform how fast we should move in that direction. For the desired z velocity, we instead compute

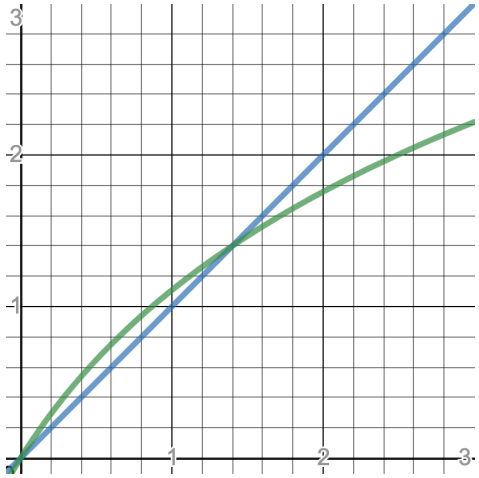

Fig. 2. Graph of $|v^P_{z,des}|$ vs. $|v^B_z|$. The blue curve is linear scaling while the green curve shows the logarithmic scaling described in Equation 1

a logarithmically-scaled mirror of the ball's z velocity. We do this for two reasons. One is that we can tune this scale so that at low v_z^B values (hinting that the ball has lost kinetic energy), the desired paddle z velocity $v^P_{z,des}$ will be greater than v_z^B and should help restore the ball's kinetic energy. The other reason is that as v_z^B gets larger, the logarithm tapers off so $v^P_{z,des}$ doesn't diverge. This prevents the system from experiencing a diverging feedback loop when the ball has high velocity, as the the paddle then hits ball at higher speed making the ball accelerate, and so on. See Figure 2 for a comparison of the logarithmic scaling versus linear scaling.

Our combined first-order system and logarithmic mirror system is as follows:

$$v^P_{xy,des} = \mathbf{K_p}(p^B_{xy} - p^P_{xy}) + \mathbf{K_v}(v^B_{xy} - v^P_{xy})$$

$$v^P_{z,des} = -\ell \cdot \text{sign}(v_z^B)\ \ln(|\, v_z^B \,| + 1) \tag{1}$$

Where $\mathbf{K_p}$ and $\mathbf{K_v}$ are gains that can be tuned to counteract errors in position and velocity, respectively, and ℓ is just a constant to scale the logarithmic response to z velocity.

2) Angular Velocity Tilt: While juggling the ball, we don't want it to go out of reach of the arm. Thus, our system aims to constrain the ball position relative to a central axis. In practice this takes the form of a tilt in the paddle where the angle of tilt can be thought of as a bowl in three dimensional around the central axis, opening in the $+z$ direction. We define this bowl region as:

$$\text{bowl} = c[1 - \cos(p^B_{x,y} - \text{center})] \tag{2}$$

Where c is a constant, $p^B_{x,y}$ is the x and y position of the ball in $\mathbb{R}^2$, and `center` is the vertical axis that we stabilize around in $\mathbb{R}^2$. We choose a sinusoid for `bowl` over a quadratic for it's milder behavior at extrema (assuming at most one period of the sinusoid is contained within the viable ranges of ball position), which is reflected in the gradient. In practice

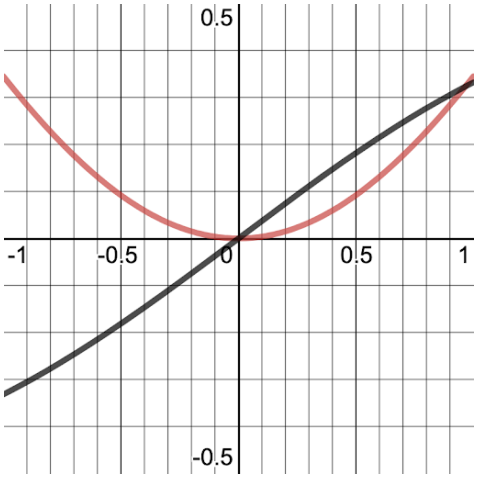

Fig. 3. Graph of angular velocity tilt as a function of ball offset. The red parabolic line represents `bowl` in Equation 2, and the black curve represents the commanded θ_{des} in Equation 3

we need to command an angular velocity, thus we use the following to determine the desired angle of the paddle tilt:

$$\begin{bmatrix}\theta_{y,des}\\ \theta_{x,des}\end{bmatrix} = \arctan\left(\frac{\text{bowl}}{p^B_{x,y} - \text{center}}\right) * \begin{bmatrix}-1\\ 1\end{bmatrix} \tag{3}$$

Where $*$, cos, and arctan are element-wise operations. Both the bowl function and the desired angle are graphed in Figure 3. An important note here is that these (and the following) equations work because the nominal orientation frame of our paddle is the same as the world frame, i.e. the x, y, z-axes of the paddle are in the same directions as the x, y, z-axes of the world. Also note the order change of x and y in the LHS — this is because when there is an offset in the x direction, we want to command the rotation in the y direction θ_y (pitch). Furthermore, we negate θ_y because a positive x offset results in the arctan of Equation 3 giving a positive θ_y, however we want it negative to bring the ball towards the central axis. Lastly, the desired rotation in the z axis $\theta_{z,des}$ is set to 0 as it is redundant due to the circular shape of the paddle.

Using the components of θ_{des} we implement a proportional controller to command an angular velocity on the paddle ω^P_{des} that drives the current θ to θ_{des}. The system is as follows:

$$\omega^P_{des} = \mathbf{K}_\theta(\theta^P_{des} - \theta^B)$$

Where $\mathbf{K}_\theta$ is a gain to tune how extreme the response is to angle error.

D. Scaling Velocities Under Special Conditions

As one can imagine, it is easy to hit the ball and cause the ball's lateral velocity to be greater than what the KUKA IIWA is capable of matching. In such cases when the ball is far from the central axis, the paddle has a high degree of tilt and we hit the ball with z velocity equal to that of the ball. This creates a rapid lateral velocity in the xy plane that we cannot match with the paddle in a controlled manner. Hence, it would be advantageous to lower the linear z velocity when the tilt of the paddle is high. From Equation 3 we know this only happens when the ball is far from the central axis, so it follows we then lower velocity $v^P_{z,des}$ when the ball is far from the central axis. Representing this above mathematically, we scale $v^P_{z,des}$ proportional to:

$$v^P_{z,des,scaled} = v^P_{z,des} \max[1 - (p^B_x - \text{center}_x)^2 - (p^B_y - \text{center}_y)^2, 0]$$

This equation represents the aforementioned intuition by penalizing $v^P_{z,des}$ with respect to the radial distance of the ball from the central axis.

Further, to make the system robust against situations where the ball has high enough velocity to make the KUKA IIWA behave unfavorable, we cap $v^P_{z,des,scaled}$. Stated explicitly,

$$v^P_{z,des,final} = \text{sign}(v^P_{z,des,scaled}) \min\left(\left|v^P_{z,des,scaled}\right|, \text{s}\right)$$

Where s is the velocity to cap the magnitude of $v^P_{z,des,scaled}$.

Additionally, there is scaling that helps the angular velocity tilt controller. Begin by imagining the case where the ball is far from `center` and the paddle has just changed the direction of the ball, i.e. the sign of the ball velocity in the x or y axis has flipped. In the configuration laid out in Section III-C2, we continue to command a paddle tilt that accelerates ball back towards the central axis until the ball is near said axis. In actuality this leads to rapid lateral velocities that the KUKA IIWA cannot match; the same reason we scaled and limited $v^P_{z,des,final}$. Intuitively, we want to minimize lateral accelerations and instead slowly bring the ball back to the central axis. To rectify this in terms of tilt, we implement a multiplier to limit acceleration:

$$\alpha_{xy} = \frac{r}{m|v^B_{xy}| + 1}$$

Where r and m are constants and the quantity is computed element-wise. In application we perform a check to apply α defined in Algorithm 1.

Algorithm 1 Acceleration-limiting of p^B_{xy}

function LIMITACCEL($\theta_{xy,des}, p^B_{xy}, v^B_{xy}$)
  if sign(v^B_x) $\neq$ sign($p^B_x - \text{center}_x$) **then**
    $\theta_{y,des,final} = \alpha \cdot \theta_{y,des}$
  else
    $\theta_{y,des,final} = \theta_{y,des}$
  end if
  if sign(v^B_y) $\neq$ sign($p^B_y - \text{center}_y$) **then**
    $\theta_{x,des,final} = \alpha \cdot \theta_{x,des}$
  else
    $\theta_{x,des,final} = \theta_{x,des}$
  end if
  return $\theta_{x,des,final}$, $\theta_{y,des,final}$
end function

IV. Results & Discussion

As a whole, the system performs well. Given reasonable perturbations in the starting position of the ball we are able to recover to a relatively stable juggle.

A. Stability

With a starting ball position directly on the center point, our system juggles the ball at a consistent height while keeping the ball almost perfectly center[2]. The simulation can do this for a minute to a minute-and-a-half with the ball perfectly center before it becomes unstable. An odd behavior we noticed is that around this minute to minute-and-a-half mark, the robot's movement becomes erratic, even if the ball is perfectly center, and it ends up hitting the ball way out of it's reach. We still do not know the source of this issue.

When the ball starts slightly off of the center point, our system is able to juggle it towards the central axis and keep it relatively near[3]. In most cases, the system isn't able to get the ball perfectly aligned with the center point, but rather it slowly oscillates around it. There are some cases, though, where the system is able to bring the ball to the central axis and stably juggle it there (until the movement becomes erratic as mentioned in the previous paragraph). Another end state the system will sometimes reach is stably bouncing the ball just slightly off center ($\sim$ 20 centimeters). This is because of how we are commanding the paddle orientation, the ball is close enough to the central axis that the commanded angle is not enough to push it closer. If we wanted to fix this, we could make our bowl function tighter, though this was a compromise we made for increased overall stability of the system.

B. Robustness

Given the ball starts within the radius of the KUKA IIWA's reach and is a reasonable distance in the xy plane from the starting pose of the paddle ($\leq .5m$), the system is able to recover to a stable juggle about the central axis. This holds true for perturbations in the x axis, the y axis, and the z axis — given z is high enough for the paddle to reach the ball when dropped. It is also robust to perturbations in a combination of these axes.

C. Next Steps

In the construction of our system we introduce numerous gains and coefficients. As such, it could make for an interesting problem to perform optimization over these parameters instead of hand-tuning them. This could be done several ways, namely stochastic gradient descent or bayesian optimization. In such a case we would define the loss as a function of the duration that the system can maintain a juggle and the proximity to the central axis. This way, we would more finely tune the parameters of our system. Further, this could be rephrased as a reinforcement learning problem with a similar loss as previously mentioned. These algorithms could be generalized across possible ball starting poses.

Additionally there are several modifications that would make interesting problems in their own right. One such would be switching the end effector to a humanoid hand to throw and catch instead of bounce, and another would be to flip our setup and dribble the ball against the ground instead of juggling it in the air.

[3]Stability video: https://youtu.be/MCTis1PE7GY

References

[1] B. Bäuml, T. Wimböck, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2592–2599.

[2] B. B. Zavala-Rio, A., "On the control of a one degree-of-freedom juggling robot," *Dynamics and Control*, vol. 9, pp. 67–90, 1999.

[3] R. Tedrake, "Robot manipulation: Perception, planning, and control (course notes for mit 6.881)," downloaded on Nov 22, 2020.