Rock'Em Sock'Em Bot

Sean McGrath, Megan Ochalek, and Valerie Pietrasz

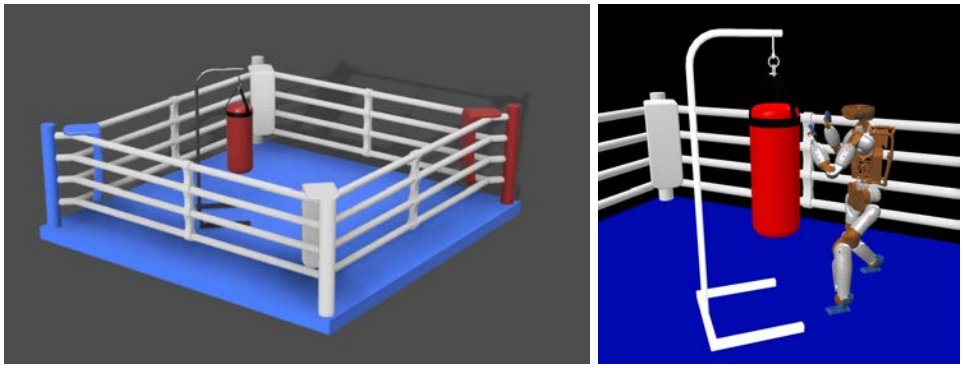
CS225, Spring 2021

Abstract

For our project, we designed a boxing robot that can punch (both jab and cross) and dodge a punching bag. For our boxing robot, we used the Toro bot, which has 32 degrees of freedom; we then modeled the punching bag as a three degree of freedom robot consisting of three overlapping revolute joints. Collision detection was enabled between the robot and the punching bag, and the bag hinged on a pivot point as it would in the real world in response to punches. The controller sensed the location of the swinging punching bag with noisy measurements, and Toro would respond by dodging away when the bag passed a threshold distance. Punches would alternate between jab (leading arm) and cross (rear arm) randomly. A combination of position, orientation, and joint tasks were utilized to achieve the desired motion control.

Introduction

Our simulation world consisted of two robots: a boxer Toro and a punching bag (Figure 1). Both were collision enabled, and the latter was modeled after a physical punching bag such that it would swing in response to landed punches. A static boxing ring and stand were added for aesthetic purposes and to aid in realism; all non-Toro items were modeled by hand in Blender. For world sensing, Gaussian white noise was implemented as an addition to the bag's true location to simulate real-life sensors.



(a) (b)

Figure 1: (a) world environment, consisting of a robot-modeled punching bag and static bag stand and ring; (b) SAI realization of Toro within the world environment.

Final Implementation

The punching bag robot was implemented as a simple 3DOF RRR robot, allowing it to swing dynamically beneath the hook. This robot used gravitational torques and joint velocity damping to simulate the real, damped pendulum motion of a punching bag.

The boxing robot was ultimately loaded as a 30DOF robot with five subsystems; five independent manipulators branching off from the hip joint, based on the provided 32DOF Toro robot model. The 2DOF discrepancy was a result of fixing two wrist joints in the robot's hands that were otherwise able to swing around in a motion not realistic in boxing due to the boxers' clenched fists. The systems were controlled largely independently, although obviously linked by the torso. Each limb had a dedicated controller, with an underlying overall posture controller, and they were managed by carefully selecting a control hierarchy for each high-level state.

The robot was controlled first through a series of operational space position and position-orientation controllers with dynamic decoupling for each end effector, with an additional joint space controller to provide a default posture. In all cases, the left leg position and orientation task in operational space was calculated first, then the null space was used for a right leg position and orientation task in order to fix the feet to the ground. In addition to fixing the feet, gravity torques were turned off for the robot to remove the challenging task of robot balance from the project.

The subsequent null space was used for an additional position task in operational space for the "punching" arm, when applicable. However, before doing so, this null space had to be modified slightly: in order to encourage the robot to punch with its arm rather that move its entire upper body forward, a task selection matrix that allowed the arm, and in the case of the cross-punch, the torso, to move more freely was introduced.

In addition, a position task, as opposed to position-orientation, was chosen to simplify the task -- when using position-orientation tasks for punches, the response was slow and unsatisfying as control effort was divided between actuating both the position and orientation. When the orientation task was removed, the robot arm was able to quickly move to the desired punching position.

The final null space was then fed into the joint space controller, which had a predetermined posture; four postures were developed using manipulation of joint positions in the absence of operational space controllers: "dodge", "orthodox" (a neutral stance), "jab", or "cross". Although command torques were written to the robot on every controller loop, the position/orientation tasks in operational space were programmed to update at a reduced frequency to save processing power and improve performance.

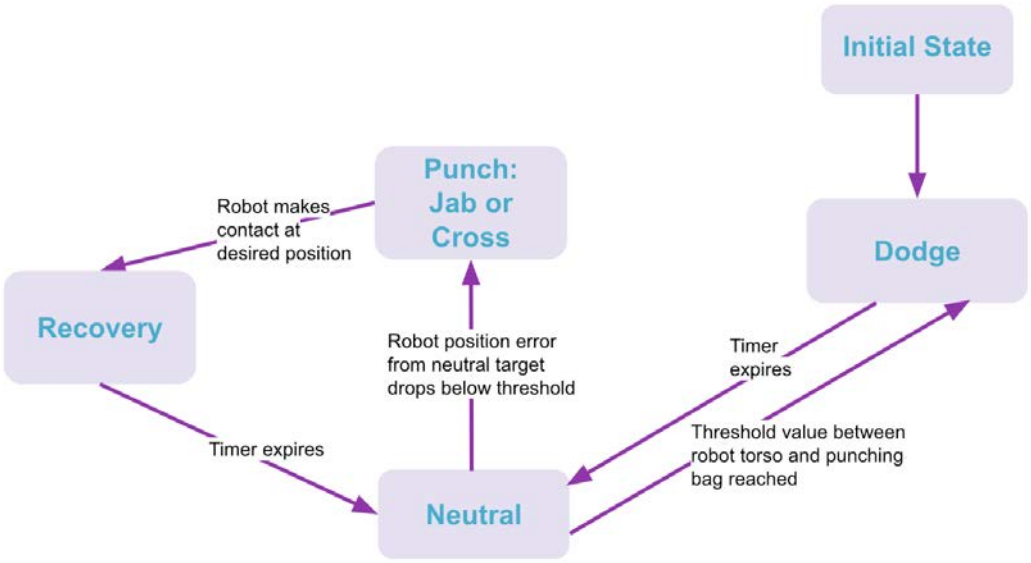


Figure 2: State diagram of robotic state machine for boxing Toro.

The state machine was implemented as shown in Figure 2, using a switch statement. At the start of the simulation, the robot was initialized in a completely relaxed position with all joint angles set to 0. The dodge posture (shown in Figure 3b) was selected as its first state -- as it entered this state, the feet moved to their desired fixed locations, and the dodge enabled the robot to move its arms up into their desired neutral position without clipping into the bag. With some tuning of the initial conditions, this could have been avoided, but was left as a fun beginning to the simulation.

To transition out of the dodge posture, a timer was implemented, which when expired allowed the robot to move to the neutral stance (shown in Figure 3a). The option of sensing when it was safe to stop dodging based on the bag's position and velocity was considered, but ultimately descoped in favor of achieving other goals. The neutral stance did not not use a position task for the arms, but simply enforced the orthodox joint posture. When the robots joints were within a small margin (ε) of the neutral stance, determined by the formula:

$$||q\_robot - q\_desired|| < \varepsilon\_neutral$$

the robot was considered to have reached the stance, and could proceed to perform one of three actions, depending on the state of the world. If the robot sensed that the bag was approaching, by detecting whether of the tip of the bag was too close to where it wanted its torso to be:

$$||x\_pos\_bag\_end - x\_pos\_hip\_desired|| < d\_dodge$$

the robot entered the dodge state. Initially, we had used the actual position of the hips to determine whether a dodge should be initiated, but ran into problems when recovering from a punch -- because the punching motion brought the robot hips forward, using the hip position meant that the robot immediately dodged after every punch, an undesired behavior. Further,

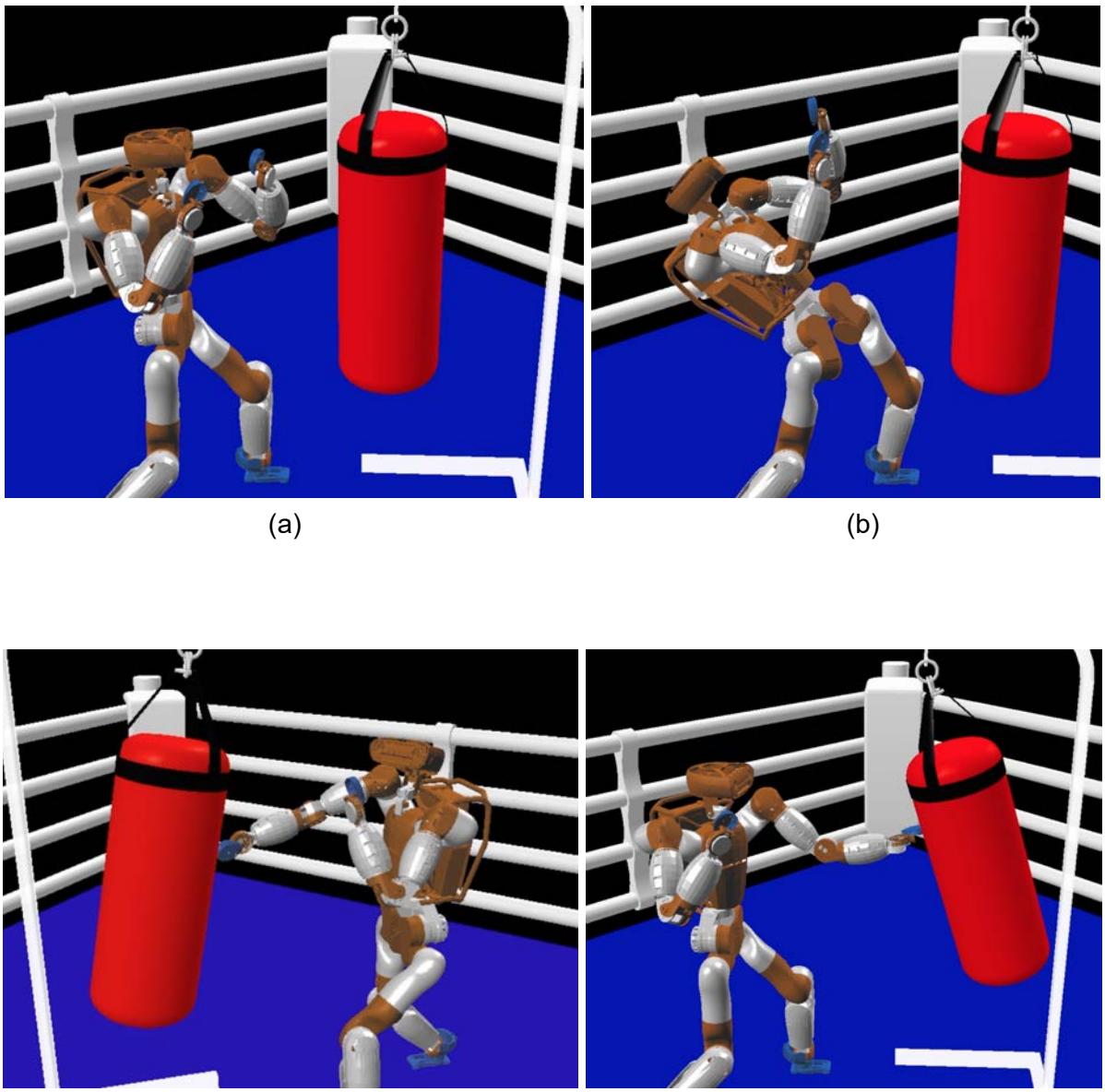
sensing noise was introduced to the measurement of x_pos_bag_end such that the robot might detect the bag as closer or farther than it actually was, leading the robot to occasionally dodge the bag when it was not actually moving toward it. We intentionally added this realistic behavior -- when humans box, they occasionally flinch when an opponent is not actually punching.

In the dodge stance, while the hips rotated backward, overactuation in the hips rotated the entire body of the robot forward, so that the feet stayed in place while the robot torso moved out of the bag's way.

If the bag was not sensed as approaching, the robot randomly chose one of two punches shown in Figures 3c and 3d, either a cross or a jab (with the right and left arms respectively). The end effector of the hand was programmed to target the center of mass of the bag -- again, with sensor noise, sometimes the robot targeted off-center. In either of the punch states, if the distance between the position of the bag at the time of punch initiation and the position of the hand dropped between a predetermined threshold, based on the formula

$$||x\_pos\_bag\_punch\_init - x\_pos\_rh|| < \varepsilon\_punch$$

the robot was considered to have succeeded in punching, and the robot entered a "recovery" state. Similar to the dodge case, when the robot targeted the actual position of the bag center of mass, the robot could never achieve its goal, as collisions with the bag meant the hand could never reach the bag center of mass. In the recovery state, the robot targeted the neutral position with temporarily high gains, so that it could reach the state more quickly. These gains were returned to their standard neutral gains once the robot appendages were sufficiently recovered, based on a timer, and the robot continued to return to neutral until one of the dodge or punch criteria were met. To make the punches and posture look realistic, the gains of each task were tuned extensively.



(a) (b)

(c) (d)

Figure 3: (a) Orthodox position (b) Dodge position (c) Cross position (d) Jab position

Challenges

Challenges were plentiful in our execution of this project. While many were listed in the final implementation, we highlight some of the bigger ones here. Our vision was not a simple one: Toro has 32 degrees of freedom, and controlling all his limbs separately was not a simple task. Mimicking realistic human motion (and corresponding joint positions) while using primitives to control hand positions posed a challenge in itself, and required lengthy tuning of joint positions, controller gains, and task timings by hand through extensive trial-and-error. While we

were able to achieve much of what we wanted through brute force, there are many goals we had to cut as they would have been too time-consuming.

One of the particularly challenging aspects of all the degrees of freedom was keeping the robot in place. In particular, we wanted the feet to be fixed on the ground while the robot boxed, and the hips to remain roughly in one place. This was easy enough to do in the beginning when we were first defining the robot's positions, but as the desired motions became more complex, so did maintaining the fixed feet. This was solved by both using a combination of brute force joint tuning and overactuation to keep the robot feet positions relatively close to the ground, and the additional operational space tasks locking the feet at the correct height.

Simulating this was also CPU intensive, resulting in the simulation and controller integration steps based on a timer being too slow, subsequently causing instabilities. A temporary fix was to forcibly slow down the simulation integration steps, but the problem was not truly solved until we were able to synchronize the simulation and controller loops using the teaching assistant's improved code.

Relatedly, due to the high cost of calculating collisions, most of the robot control development was performed with collisions disabled. Thus, we did not test our collision objects, which we initially developed as meshes in blender, as early as we should have. When it came time to turn collisions on, two of the team members were unable to run the simulation due to "bus errors" in the simulation. Therefore, we had to reduce collision complexity by replacing the collision meshes we developed (along with those provided with the toro robot) with simpler SAI primitive objects. For the punching bag, we used a cylinder that matched the shape of the bag fairly closely, and for the boxing robot, we turned off all collisions except for the hands, which were modelled as SAI spheres.

If we were to approach the project again from scratch, we would likely have attempted to further refactor the code and modularize it to more easily add and edit new stances and poses. We may also have introduced an alternative velocity based control for the punches, while targeting a surface position on the bag. We may also have spent less time on the appearance of the models in blender, as their rendering was much more limited in SAI2.

Conclusion

Ultimately, we were proud of the Rock'Em Sock'Em bot's performance. We were able to control the Toro to successfully target the punching bag repeatedly, with a convincing humanoid motion. This project gave us the opportunity to expand on the control principles and theory that we learned in the context of single arm manipulators, and apply them to a much more complex system, the 32DOF Toro robot. This forced us to look at robot control on several levels, from the high-level state machine architecture, to the tuning of specific joint parameters. We saw the value that robot simulation in SAI2 provided when planning complex motions, which would have been much more difficult to test on a physical robot. We also had the chance to improve our blender modeling skills, and our understanding of ROS URDF files when we designed our punching bag robot from scratch for the TORO to interact with. These are all skills that will serve us well in our future endeavors in the fields of robotic manipulation and controls.

Future Extensions

In our initial project concept, we intended to have two Toros boxing each other. The robots would each sense the location of the opposing Toro and decide the proper response: punch if their opponent is open, or dodge if they are punching. This required a lot of advanced work in advanced targeting (the trajectory of incoming punches would need to be interpolated) and better dodging. Due to time constraints and the plethora of challenges discussed above, we were unable to realize this design; the extension of our project from one Toro opposing a punching bag to two Toros facing off is a clear next step.

Additionally, our robot design could be extended to include more variation of punches, such as a hook or body shot, or include a variation of kicks, such as a snap kick or roundhouse kick. The realism of the punches could also be increased in comparison to a human executing them, which would include dynamic posturing of the legs (i.e. pivoting the rear foot when completing a cross punch) and possibly more joint tasks. Furthermore, more sophisticated dodging could be included, ideally using something such as an artificial potential field to force the robot away from the bag.

A key simplification in our design was the exclusion of balancing; introducing the action of balancing on the ground surface instead of utilizing “PosOri” tasks and overactuation would make our simulation much more real-world.

Appendix

- The project code can be found at: https://github.com/valerie-pietrasz/rockem_sockem_project
- Short videos of the robot in action can be found at:
    - https://drive.google.com/file/d/1noqz5S4akWAEyojT0UcBfWBTOTudfxOo/view
    - https://drive.google.com/file/d/1RL0_rZQLgZSaQT0Cwi3So1AwGfqJkA5k/view
    - https://drive.google.com/file/d/1wYtYYb7UYEYYFL0vF7Su-JAdisTEsKWj/view