The Robotic Jenga Player

6.4210: Robotic Manipulation
Final Project Proposal

David Bombara and Christy Jestin

November 3, 2023

Abstract

In our project, we propose a comprehensive framework for developing an end-to-end robotic controller designed to autonomously play the game of Jenga using a Franka Panda arm. Our initiative aims to address a wide array of interconnected challenges in robotics, encompassing geometric perception, inverse kinematics, and adaptive reasoning essential for motion planning and gameplay. We have organized the project into distinct but interrelated tasks, including environment setup, perception and state estimation, control, sensing, and decision-making. These tasks will enable our robot to perform complex manipulative actions, from spatially recognizing block configurations via cameras to executing sophisticated block extractions based on optimized algorithms. Ultimately, we seek to apply the concepts of this course by developing a full-stack manipulation system that addresses a specific sub-problem in the field of robotic manipulation.

1 Introduction

We want to build a full-stack robotic manipulation system that plays the game Jenga in simulation with a Franka Panda arm. Jenga is a complex manipulation task that requires not only careful and controlled motion but also adaptive reasoning and planning to test blocks and figure out which moves are most viable. It is already challenging for humans, and we don't foresee any clear advantages for a robot. While many robotics tasks may force the robot to think about multiple candidate actions, Jenga is unique because the robot must update its beliefs and replan as it tests these candidate actions.

This project will require topics like geometric perception, inverse kinematics, contact and motion planning, and grasping. Some tasks the robot will need to perform include:

1. Use cameras to understand where the blocks are.
2. Consider candidate moves based on its past observations and the static equilibrium of the tower.
3. Test candidate blocks and update its belief about loose and tight blocks.
4. Plan and execute block extractions (which will almost always require multiple modes of contact).
5. Raise blocks and place them on top of the stack.

1.1 Related Work

Previous studies have considered the manipulation problem. Full-stack systems have been developed in [1], [2], and [3]. The instance segmentation and visual servoing aspects were explored by Marchionna et al [4]. The force-based manipulation aspects were explored by Kimura et al [5].

1.2 Gameplay Overview

Firstly, one robot or two must assemble the Jenga tower. After the tower is assembled, the robots begin to play Jenga. At each turn, a robot decides the optimal move to play. If the Jenga tower falls during a robot's turn, that robot is the loser of the game. After the tower falls, the robots reassemble the tower (as in the first step.

2 Project Tasks

This project contains many sub-problems/sub-problems that many individual tasks. Some may be solved independently whereas others are interdependent. It will be too challenging to solve the entire Jenga-playing problem (with its entire constraints and rules) without first relaxing some of these rules and solving the resulting sub-problems. In this project, the following constraints/scenarios will be enforced and relaxed.

1. *Block friction modeling:* with and without.
2. *Block state estimation*: (1) Given through an oracle versus (2) estimated from RGB-D cameras.
3. *Block friction estimation:* (1) Given through an oracle versus (2) estimated from fingertip force sensors.
4. *Gameplay decision making:* (1) Human player tells the robot which block to move, versus (2) the robot determining that for itself via an optimization problem.

5. *Block modeling fidelity:* (1) All blocks have exactly equivalent geometry versus (2) giving them slight deviations from their nominal dimensions to make some blocks “tighter” and other blocks ”loose”.

6. *Strategy of players:* (1) Adversarial versus (2) cooperative.

7. *Tower setup:* (1) Robots assembling the tower versus (2) starting with an assembled tower.

We organize this project’s tasks into the following categories: (1) Environment Setup, (2) Perception and State Estimation, (3) Modeling Tower Physics, (4) Sensing, (5) Control, and (6) Gameplay Decision Making.

2.1 Environment and simulation setup

2.1.1 Creating and sourcing of `URDF` files

This tasks includes the meshes (`.stl` files), collision geometry, and `.urdf` files for each object. We plan to use the Franka panda robotic arm, which is the same arm purchased in our research group (the Computational Robotics Lab at Harvard).

2.1.2 Initialize component placement during simulation

This task may be trivial, but nonetheless necessary for the execution of the project’s other tasks. If the Jenga tower is initially unassembled, this task can also include the assembly process.

2.2 Perception

2.2.1 Segmentation and Pose Estimation of individual blocks from RGB-D cameras

Object segmentation is a fundamental component of practically every end-to-end manipulation system. Several cameras will be used to prevent occlusion and partial views. We will likely use a neural network for point cloud segmentation.

Once the individual blocks are segmented, their unique position, orientation, and velocity of each block must be estimated. Geometric pose estimation will be used to obtain the positions and orientations. Velocity estimation is not crucial because the blocks are nearly always static unless the tower is toppling.

2.2.2 Tower State Estimation

The state of the overall tower can be discretized along the continuum from unassembled, partially-assembled, and full-assembled. This knowledge will be useful for the robots to detect when the tower falls, thus ending the game.

2.3 Sensing

The problem here is to correlate block friction coefficients and their nominal dimension deviations with the fingertip force feedback. This sensing component will help the robot ”feel” which blocks are tight versus loose, thus helping it play a move that will not topple the tower.

2.4 Control

Firstly, the manipulator(s) should be able to push and grip *frictionless* blocks, without requiring fingertip force feedback. After this control problem is solve, the control should should be modified to use fingertip force feedback for gentle pushing and gripping. The basic static equilibrium can be computed analytically.

2.5 Modeling the Tower Physics

2.5.1 Rigid-body *statics without* block friction

Static equilibrium (absence external disturbances) is the most fundamental property that the tower must have in order for it to not topple. To start, masses and inertias of the blocks will be used to compute the net wrench (force and moment) due to gravity, *without considering* frictional effects or minor deviations in the blocks’ nominal dimensions.

2.5.2 Rigid-body *dynamics and statics* of the tower block friction and deviations from nominal dimensions

Because this modeling framework is more complicated, analytical solutions may either be intractable or suboptimal to numerical techniques. The Drake physics engine will be leveraged for this task and to simulate disturbances to the tower.

2.6 Gameplay and Decision Making

The computation of the optimal move will be the result of an optimization problem. There are different levels of modeling fidelity that can be incorporated into this optimization problem. The optimization problem is conceptually the following:

$$\begin{aligned} \min_{i} \quad & -\|\texttt{Structural stability of the tower}\| \\ & \text{s.t.} \\ & \texttt{The system remains in static equilibrium.} \\ & \texttt{Block } i \texttt{ has not been picked yet.} \end{aligned}$$

where i is the index of the block. “Structural stability” is a concept that can be quantified in different ways. For example, it could be measured in terms of:

- The maximum vibration frequency and amplitude that the tower can endure without toppling.
- The maximum vertical force that the can be exerted onto any point from above the tower without it toppling.

When friction and block dimension deviations are incorporated into the modeling framework, the optimization landscape may change accordingly. The cost function will be modified to account for the more realistic physics. These physical effects may also be incorporated into the constraints of the optimization problem. The bottom line is that the optimal move may likely change due to the completeness of the physics modeling framework.

2.7 Task-Level Architecture and System Integration

Many of the tasks above may be completed independently. Thus, the integration of the above tasks into one simulation will a crucial but non-trivial exercise. We will organize the high-level states of the Jenga game using finite state machine(s) and/or behavior trees.

3 Deliverables and Approximate Timeline

The final deliverable for this project will be a simulation (including code, video, and written report) that shows the one robotic Jenga game from initialization (fully-assembled tower) to completion (toppled tower). During the first project check-in, the simulation environment will be complete, including the hardware stations, block physics, and perception systems. During the second project check-in, some preliminary success be shown where a robotic manipulator can successfully execute some form of pick-and-place without the tower toppling.

3.1 Distribution of Work

Below is a summary of the work distribution between Christy and David. Each item in the list represents one Jupyter notebook that implements the task.

1. Setup: Creating and sourcing URDF files: **David**
2. Setup: Initializing component placement during simulation: **Christy**
3. Perception: Segmentation and Pose Estimation of individual blocks from RGB-D cameras: **Christy**
4. Perception: Tower State Estimation: **David**
5. Sensing: **Christy**
6. Control: **David**
7. Modeling Tower Physics: **David**
8. Gameplay and Decision Making: **David**
9. Task-Level Architecture and System Integration: **Christy**

References

[1] T. Kroger, B. Finkemeyer, S. Winkelbach, L.-O. Eble, S. Molkenstruck, and F. M. Wahl, "A manipulator plays jenga," *IEEE robotics & automation magazine*, vol. 15, no. 3, pp. 79–84, 2008.

[2] M. CHIABERGE and G. PUGLIESE, "Visual and tactile perception to play jenga with a robotic arm," 2021.

[3] J. Wang, P. Rogers, L. Parker, D. Brooks, and M. Stilman, "Robot jenga: Autonomous and strategic block extraction," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 5248–5253.

[4] L. Marchionna, G. Pugliese, M. Martini, S. Angarano, F. Salvetti, and M. Chiaberge, "Deep instance segmentation and visual servoing to play jenga with a cost-effective robotic system," *Sensors*, vol. 23, no. 2, p. 752, 2023.

[5] S. Kimura, T. Watanabe, and Y. Aiyama, "Force based manipulation of jenga blocks," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 4287–4292.