ME699 Final Project Proposal

Benton Clark, Ethan Howell, Brian Moberly

02-04-2020

1 Project Outline

For our project, we are proposing three primary phases:

- Modeling
- Perception and Planning
- Control

Each phase of the project will feed into the next. For instance, the model designed in the robotic modeling section will then be used in the control section of the project. By the end of the project, our goal is to have a fully modeled, controlled robotic manipulator capable of following a simple trajectory and moving to desired target locations in task space to accomplish simple goals.

2 Demonstration

The end goal of our project is to apply the modeling, perception and adaptive controls to the Panda Robot arm. The task will consist of the arm beginning in a configuration, moving towards an intermediate state where an object is located, picking up that object, and successfully moving that object to a desired goal state. The object will be of unknown and variable mass, meaning the arm must be able to effectively adapt to objects of differing weight online. The exact location of the object must also be identified by the "camera" attached to the Panda, meaning a valid path must be found to both reach the object and take it to the goal state. The goal state will be fixed.

For the modeling portion, both static free and dry friction models will be used. The perception and planning portion will be tasked with filtering out small amounts of Gaussian white noise from both the joint configurations and the camera readings in task space. It must also find a valid trajectory of configurations in both task and joint space to complete the objective. The adaptive controller will use both the modeling and perception values in determining the joint torques to command to follow the given trajectory. It will also compensate for the additional weight on the last link of the arm when the object of unknown mass is picked up.

3 Robotic Modeling

3.1 Approach

The general equation of motion for a robotic manipulator can be expressed as:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(g) = \tau \tag{1}$$

where $M(q) \in \mathbb{R}^{n\times n}$ is the mass matrix of the joints, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is the Coriolis matrix, $G(q) \in \mathbb{R}^n$ is the conservative force vector acted on the arm by gravity, and $\tau \in \mathbb{R}^n$ are the torques commanded to each joint. Each of the terms on the left hand side of the equation can be directly modeled when the manipulator joint information is known. However, when this information is not available, a neural network can be used to estimate the value of these matrices.

To model the conservative force vector, a robotic manipulator can be controlled with a simple PD controller and brought to a stop at a given configuration value q. When the manipulator is at rest, Eq. (1) simplifies to:

$$M(q) * 0 + C(q,0) * 0 + G(q) = \tau \tag{2}$$

$$G(q) = \tau \tag{3}$$

which allows the loss function for the function approximator $\hat{G}(q)$ to be expressed as:

$$\hat{G}(q) - \tau = \delta_{Loss} \tag{4}$$

To model the mass matrix with a function approximator $\hat{M}(q)$, we must again model a loss function to train the network. This proves more difficult, as the mass matrix must be isolated on the left hand side of the equation. However, if the number of acceleration samples taken at a given configuration q satisfy $rk(M(q)) = n_{samples}$, we could then construct an acceleration matrix $\ddot{Q} = [\ddot{q_1}, \ddot{q_2} \dots \ddot{q_n}]$ which satisfies $rk(\ddot{Q}) = rk(M(q)) = n$. Properties of linear algebra now guarantee that a unique inverse of this matrix must exist, and the equation can then be expressed as:

$$M(q)\ddot{Q} + C(q,\dot{Q})\dot{Q} + G(Q) = \mathrm{T} \tag{5}$$

$$M(q) = \ddot{Q}^{-1}(\mathrm{T} - G(Q) - C(q,\dot{Q})\dot{Q}) \tag{6}$$

where $G(Q) \in \mathbb{R}^{n\times n}$ is a one dimensional matrix where each column satisfies $G_i(Q) = G(q)$.

The problem with this method is that the Coriolis term is still in place, and we have no estimate for this value. However, if we take our $\ddot{q}$ samples at the initial point when we first begin accelerating, we should see that $\dot{q} \approx 0$, and thus Eq. (6) can be expressed as:

$$M(q) = \ddot{Q}^{-1}(\mathrm{T} - G(Q)) \tag{7}$$

which now allows for a loss function to be formulated.

For the Coriolis matrix, it directly depends on the mass matrix $M(q)$, and thus once an estimate for $M(q)$ is formulated, the Coriolis matrix can be derived from the given values of the $\hat{M}(q)$ matrix.

3.2 Experiments

The experiments for this section will include learning both $\hat{G}(q)$ and $\hat{M}(q)$ using the proposed method above in two settings: One in a friction free environment, and another when static friction is present. The friction free case will be used primarily as a base case, and the accuracy of both the $\hat{G}(q)$ and $\hat{M}(q)$ will be compared to their true values. This will serve to see how effective this method is in general, as friction begins complicating the process.

The frictional case will be further broken down into two sections. The first will consist of learning the two maps without static friction compensation, and the second will include static friction compensation. Static friction will be modeled as a constant μ_s, so this can easily be added into the above formulations (primarily for the mass matrix case).

4 Perception and Planning

4.1 Uncertainty in Configurations

The robot configuration q_1, q_2, ..., q_n determines how the robot will interact with the task space and the end-effector. Assuming that the end-effector has a camera on its end and can understand its surroundings, the perception through said camera will be used to achieve motion and interaction with an object in the task space. A combination of the RRT path planning algorithm, Kalman filtering, and Forward/Inverse kinematics will lead to the configuration that the robot is in to have the end-effector reach the object. An object with some mass M_o will be placed in the task space. The robot's purpose is to move to the object, pick it up, and move it to a desired position.

4.2 Approach

The end-effector perception problem requires many disciplines of robotics. First, a landmark based Kalman filter will be applied to the robot. This will test our ability to move to the object. Then, applying kinematic theory, the configuration of the robot to achieve the end-effector location and orientation corresponding to the object will be achieved.

5 Control

5.1 Global Stability

Using the dynamics of the n-link manipulator described in Eq. (1), we will design an adaptive PD controller with the goal of reaching the desired task space. In order to ensure that we can do this in a stable manner, we derive our control law using Lyapunov stability analysis. To do this, consider the Lyapunov candidate

$$V(t) = \frac{1}{2}\left(\dot{\tilde{q}}^T M(q)\dot{\tilde{q}} + \tilde{a}^T \Gamma \tilde{a} + \tilde{q}^T K_p \tilde{q}\right) \tag{8}$$

where $a \in \mathbb{R}^{m\times 1}$ are the unknown manipulator parameters, $K_p, \Gamma \in \mathbb{R}^{m\times m}$ are positive definite and symmetric, $\tilde{q}$ is the error between the current space and desired task space, and $\tilde{a}$ is the error in our estimation of unknown parameters. To understand how candidate 8 changes over time, we differentiate to obtain

$$\dot{V}(t) = \dot{\tilde{q}}^T \left[\tau - M(q)\ddot{q}_d - C(q,\dot{q})\dot{q}_d - G(q) + K_p\tilde{q}\right] + \tilde{a}^T\Gamma\dot{\tilde{a}} \tag{9}$$

where $q_d \in \mathbb{R}^{n\times 1}$ is the desired trajectory of joint properties. We then select the control law to be

$$\tau = \hat{M}\ddot{q}_d + \hat{C}(q,\dot{q})\dot{q}_d + \hat{G}(q) - K_d\dot{\tilde{q}} \tag{10}$$

where K_d is positive definite and symmetric. Choosing the control law in this manner allows our estimation vector, a, to be dependent only upon unknown parameters of the manipulator. We can then re-write Eq. (9) in terms of our state error as

$$\dot{V}(t) = \dot{\tilde{q}}^T \left[\tilde{M}(q)\ddot{q}_d + \tilde{C}(q,\dot{q})\dot{q}_d + \tilde{G}(q) - K_d\dot{\tilde{q}}\right] + \tilde{a}^T\Gamma\dot{\tilde{a}} \tag{11}$$

Note that since our state matrices are linear we can further simplify to obtain

$$\dot{V}(t) = -\dot{\tilde{q}}^T K_d\dot{\tilde{q}} + \tilde{a}^T\left[\Gamma\dot{\tilde{a}} + Y^T\dot{\tilde{q}}\right] \tag{12}$$

Thus, for a reasonably close approximation of our state matrices, we can select our gains, K_d such that we obtain a globally stable controller.

5.2 Implementation

In order to implement this control law, we will first need a desired task space consisting of joint displacements, velocities and accelerations. Once we have these parameters, we will simulate sensors to measure these values as well as a random number generator to replicate sensor noise. These sensor values will then be passed into the Kalman filter to eliminate as much noise as possible. Once this process is complete, the state will be updated and passed into the controller for the appropriate action to be taken.

6 Team Contributions

For the individual sections of the project, Benton will focus on the robotic modeling, Ethan on the robotic controls, and Brian on the perception and tracking. However, since each part of the project will slowly feed into one another, each team member will help contribute to all three parts of the project equally.