Estimation and Control Library

Pan

***Abstract*— A library for optimal estimation and control, as well as classical/modern control theories. Algorithms are applied on simulated robots using Ignition Gazebo.**

I. INTRODUCTION

A library for optimal estimation and control, as well as classical/modern control theories. Algorithms are applied on simulated robots using ROS [1] and Ignition Gazebo [2].

II. CARTPOLE DYNAMICS

This section derives the equation of motion of the cartpole system in the following two ways: (1) standard vector analysis; (2) using Lagrangian.

A. Standard vector analysis

The position of the cart is given as

$$\mathbf{r}_1 = x\mathbf{E}_x \tag{1}$$

Computing the 1st and 2nd order rate of change of $\mathbf{r}_1$ in reference frame $\mathcal{F}_i$, we obtain the velocity and acceleration as

$$\mathbf{v}_1 =^{\mathcal{F}_i} \frac{d}{dt}(\mathbf{r}_1) = \dot{x}\mathbf{E}_x \tag{2}$$

$$\mathbf{a}_1 =^{\mathcal{F}_i} \frac{d}{dt}(\mathbf{v}_1) = \ddot{x}\mathbf{E}_x \tag{3}$$

The position of the pole is given as

$$\mathbf{r}_2 = x\mathbf{E}_x + l\mathbf{e}_r \tag{4}$$

Similarly, the velocity and acceleration of the pole can be computed as

$$\mathbf{v}_2 =^{\mathcal{F}_i} \frac{d}{dt}(\mathbf{r}_2) = \dot{x}\mathbf{E}_x + l\dot{\theta}\mathbf{e}_\theta \times \mathbf{e}_r = \dot{x}\mathbf{E}_x - l\dot{\theta}\mathbf{e}_z \tag{5}$$

$$\mathbf{a}_2 =^{\mathcal{F}_i} \frac{d}{dt}(\mathbf{v}_2) = \ddot{x}\mathbf{E}_x - l\ddot{\theta}\mathbf{e}_z - l\dot{\theta}^2\mathbf{e}_r \tag{6}$$

Rewriten $\mathbf{a}_2$ in $\mathbf{E}_x$ and $\mathbf{E}_z$ as

$$\mathbf{a}_2 = (\ddot{x} + l\ddot{\theta}\cos\theta - l\dot{\theta}^2\sin\theta)\mathbf{E}_x - (l\ddot{\theta}\sin\theta + l\dot{\theta}^2\cos\theta)\mathbf{E}_z \tag{7}$$

Applying Netwon 2nd law on the $\mathbf{E}_x$-direciton of the cart and pole, we have

$$F - T_x = m_1\ddot{x} \tag{8}$$

$$T_x = m_2(\ddot{x} + l\ddot{\theta}\cos\theta - l\dot{\theta}^2\sin\theta) \tag{9}$$

Adding the above two euqations we get the first equation,

$$F = (m_1 + m_2)\ddot{x} + m_2(l\ddot{\theta}\cos\theta - l\dot{\theta}^2\sin\theta) \tag{10}$$

[1] The Pennsylvania State University, University Park, PA 16802, USA. pan.liu@psu.edu

The cartpole system is consisted of two systems, i.e. the cart and the pole. The internal force can be canceled out by analyzing the two systems as a whole. A simplier way is to apply the balance of angular momemtun relative to the cart.

$$\frac{d}{dt}\mathbf{H}_c = \mathbf{M}_c - (\mathbf{r}_2 - \mathbf{r}_1) \times m_2\mathbf{a}_1 \tag{11}$$

where $\mathbf{H}_c$ is the angular momemtun of the cartpole system relative to the cart.

$$\mathbf{H}_c = (\mathbf{r}_2 - \mathbf{r}_1) \times m_2(\mathbf{v}_2 - \mathbf{v}_1) = m_2 l^2\dot{\theta}\mathbf{E}_y \tag{12}$$

And $\mathbf{M}_c$ is the moment relative to the cart.

$$\mathbf{M}_c = (\mathbf{r}_2 - \mathbf{r}_1) \times (-m_2 g)\mathbf{E}_z = m_2 g l\sin\theta\mathbf{E}_y \tag{13}$$

$$(\mathbf{r}_2 - \mathbf{r}_1) \times m_2\mathbf{a}_1 = l\mathbf{e}_r \times m_2\ddot{x}\mathbf{E}_x = m_2 l\ddot{x}\cos\theta\mathbf{E}_y \tag{14}$$

Thus, we get the second equation

$$\cos\theta\ddot{x} + l\ddot{\theta} = g\sin\theta \tag{15}$$

In summary, the equation of motion is

$$(m_1 + m_2)\ddot{x} + m_2(l\ddot{\theta}\cos\theta - l\dot{\theta}^2\sin\theta) = F \tag{16}$$

$$\cos\theta\ddot{x} + l\ddot{\theta} = g\sin\theta \tag{17}$$

B. Linearization

Assuming $\mathbf{q} = [x, \theta]^T$, the equations of motion of the cartpole system can be written in the standard form as

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{B}\mathbf{u} \tag{18}$$

where $\mathbf{q} = [x, \theta]^T$ is a n-vector called the *generalized coordinates vector*, $\mathbf{H}(\mathbf{q})$ is a $n \times n$ nonsingular symmetric positive-definite matrix called the *mass matrix*, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is a $n \times n$ matrix called the *centrifugal/Coriolis/friction matrix*, $\mathbf{G}(\mathbf{q})$ is a n-vector sometimes called the *conservative forces vector*. **This equation of motion is valid for systems that follow classical Newton-Euler mechanics or Lagrangian mechanics** with a kinetic energy that is quadratic in the derivative of the generalized coordinates and a potential energy that may depend on the generalized coordinates, but not on its derivative. Such systems include robot arms, mobile robots, airplanes, helicopters,underwatervehicles,hovercraft,etc.

$$\mathbf{H}(\mathbf{q}) = \begin{bmatrix} m_1 + m_2 & m_2 l\cos\theta \\ \cos\theta & l \end{bmatrix}, \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} 0 & -m_2 l\dot{\theta}\sin\theta \\ 0 & 0 \end{bmatrix},$$
$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} 0 \\ -g\sin\theta \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{19}$$

Let $x_1 = \mathbf{q}$, $x_2 = \dot{\mathbf{q}}$, and $x = [x_1, x_2]^T$. Then we have the **standard form for linearization**, i.e. $\dot{x} = f(x, u)$

$$\dot{x} = \begin{bmatrix} x_2 \\ \mathbf{H}(\mathbf{q})^{-1}(-\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q}) + \mathbf{B}\mathbf{u}) \end{bmatrix} \tag{20}$$

Linearize around the equlibrium point $x^* = 0$,

$$\dot{x} = Ax + Bu \tag{21}$$

where

$$A = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{H}(\mathbf{q})^{-1}\frac{\partial}{\partial \mathbf{q}}\mathbf{G}(\mathbf{q}) & -\mathbf{H}(\mathbf{q})^{-1}\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix}, \tag{22}$$

$$B = \begin{bmatrix} 0 \\ \mathbf{H}^{-1}\mathbf{B} \end{bmatrix} \tag{23}$$

Note that the term involving $\frac{\partial}{\partial \mathbf{q}}\mathbf{H}(\mathbf{q})^{-1}$ disappears because $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q}) + \mathbf{B}\mathbf{u}$ must be zero at the fixed point. Many of the $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ derivatives drop out, too, because $\dot{\mathbf{q}}^* = 0$. The $-\mathbf{H}(\mathbf{q})^{-1}$ can be calculated by Matlab symbolic inversion,

$$\mathbf{H}(\mathbf{q})^{-1} = \begin{bmatrix} \frac{1}{m_1 + m_2 - m_2\cos^2\theta} & \frac{-m_2\cos\theta}{m_1 + m_2 - m_2\cos^2\theta} \\ \frac{-\cos\theta}{m_1 l + m_2 l - m_2 l\cos^2\theta} & \frac{m_1 + m_2}{m_1 + m_2 - m_2\cos^2\theta} \end{bmatrix} \tag{24}$$

and

$$\frac{\partial}{\partial \mathbf{q}}\mathbf{G}(\mathbf{q}) = \begin{bmatrix} 0 & 0 \\ 0 & -g\cos\theta \end{bmatrix} \tag{25}$$

Substitute $x^* = 0$ and rearrange x to be $x = [x, \dot{x}, \theta, \dot{\theta}]$, we have

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{m_2}{m_1}g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{m_1 + m_2}{m_1 l}g & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1/m_1 \\ 0 \\ -1/(m_1 l) \end{bmatrix} \tag{26}$$

III. Sensors

This section describes comonly used sensors in robotics area, including LiDAR and IMU.

A. LiDAR

LiDAR is used as an acronym of "light detection and ranging" or "laser imaging, detection, and ranging".

LiDAR operating principles: Measuring distance with time-of-flight using three components: a laser, a photodetector, and a very precise stopwatch.

$$d = \frac{1}{2}ct \tag{27}$$

where c is speed of light. Since light travels much faster than cars, it's a good approximation to think of the LIDAR and the target as being effectively stationary during the few nanoseconds that it takes for all of this to happen. It is worth noting that the intensity information of the return pulse is can also be useful. It provides some extra information about the geometry of the environment and the material the beam is reflecting off of. it turns out that it's possible to create 2D images from LIDAR intensity data that you can then use the same computer vision algorithms you'll learn about in the next course. 2D and 3D build a rotating mirror into the LIDAR that directs the emitted pulses along different directions. add an up and down nodding motion to the mirror along with the rotation, you can use the same principle to create a scan in 3D.

LiDAR measuring model in 2D and 3D LiDAR sensor gives $[r, \alpha, \epsilon]^T$, which are distance, azimuth (heading) and elevation angle. They can be convert to the Euler coordinate as (inverse sensor model)

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = h^{-1}(r, \alpha, \epsilon) = \begin{bmatrix} r\cos\alpha\cos\epsilon \\ r\sin\alpha\cos\epsilon \\ r\sin\epsilon \end{bmatrix} \tag{28}$$

And the forward sensor model is

$$\begin{bmatrix} r \\ \alpha \\ \epsilon \end{bmatrix} = h(x, y, z) = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arctan\frac{y}{x} \\ \arcsin\frac{z}{r} \end{bmatrix} \tag{29}$$

For 2D model, $z = 0$ and $\epsilon = 0$.

Sources of measurement noise

- uncertainty in the exact time of arrival of the reflected signal
- uncertainty in the exact orientation of the mirror in 2D and 3D LIDARs
- interaction with the target surface which can degrade the return signal

These factors are commonly accounted for by assuming additive zero-mean Gaussian noise $v \sim \mathcal{N}(0, \Sigma)$.

LiDAR point clouds All points of LiDAR are stacked horizontally as $p = [p_1, p_2, ..., p_n]$, where $p_* = [x, y, z]^T$. translation, rotation and scaling in SE(3)

$$g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{30}$$

Find road with 3D plane fitting the plane model is $z = ax + by + c$, least square fitting is

$$\begin{bmatrix} z_1 \\ z_2 \\ \dots \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \dots & .. & .. \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{31}$$

point cloud library for c++

State estimation via point set registration Problem description The point set registration problems says, given 2 point clouds in two different coordinate frames, and with the knowledge that they correspond to or contain the same object in the world, how shall we align them to determine how the sensor must have moved between the two scans? More specifically, we want to figure out the optimal translation and the optimal rotation between the two sensor reference frames that minimizes the distance between the 2 point clouds.

The problem is that, in general we don't know which points correspond to each other. The most popular algorithm for solving this kind of problem is called the Iterative Closest Point algorithm or ICP for short.

References:

- State Estimation and Localization for Self-Driving Cars, Module 4 LiDAR Sensing.

(https://www.coursera.org/learn/state-estimation-localization-self-driving-cars)

REFERENCES

[1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[2] I. Gazebo, "https://ignitionrobotics.org."