Case Western Reserve University
Electrical Engineering and Computer Science Department
Algorithmic Robotics EECS-499
Project-1
April 4, 2017

Torque State Estimation on Baxter

Kevin Bradner, graduate student in EECS department
Ammar Nahari, graduate student in EECS department
Hieu (Tony) Tran, graduate student in EMBE department

Abstract: This project aims to detect collision that occurs on Rethink Baxter left arm by estimating the torque states. Given the joint angles commanded, the expected torque states are estimated using a particle filter. By getting the real torque values from the sensors, we can know if a collision has occurred or not, by looking at the average likelihoods of resampled particles. If the average of the weights of resampled particles is under a set threshold, then the algorithm will determine that a collision has happened, and the movement commands are stopped.
To implement and test the algorithm, Baxter was commanded to move randomly at every time step. The sensor data and commanded joint angles were passed through the particle filter algorithm. The algorithm was tested first with no collisions to determine if the algorithm have false positives. Then the algorithm was tested with collision to determine accuracy. Both tests showed that the algorithm can be used to detect collision with minimal possibility of false positives.

Introduction and Approach:

There are different approaches of collision detection that can be applied to Baxter. One way is to estimate the position state given torque inputs in the dynamic model. By using the data from the joint encoders, we can construct the sensing model easily. However, this approach requires complex calculation and consume large processing time.

Therefore, we decided to calculate the torques state estimation instead by commanding the joint angles, calculating expected torque, and use the integrated effort sensors to get the real torque data. To avoid complex inertial terms caused by computing velocity and acceleration, we calculated the torque estimation only during stationary states, after the motion is completed. A smooth motion can still be achieved using this algorithm by using small joint angle movements and small time rest in between each movement. The algorithm would also work under low maximum joint acceleration and velocity control modes.

This report covers the technical properties of Baxter's left arm, the dynamic model generation, error sampling procedure, code overview, and conclude with some experimental results of the project.

Properties of Baxter's Left Arm: [2]

DH Parameters:

Link	θ_i	d(m)	a(m)	α(rad)
1	θ_1	0.27	0.069	-pi/2
2	θ_2	0	0	pi/2
3	θ_3	0.362	0.069	-pi/2
4	θ_4	0	0	pi/2
5	θ_5	0.371	0.01	-pi/2
6	θ_6	0	0	pi/2
7	θ_7	0.28	0	0

Center of mass (in DH Parameter):

x (m)	y (m)	z (m)
-0.5177	0.07908	0.00086
0.00269	-0.00529	0.06845
-0.07176	0.08149	0.00132
0.00159	-0.01117	0.02618
-0.01168	0.13111	0.0046
0.00697	0.006	0.06048
0.005137	0.0009572	-0.06682

Link tensors:

$Link_i$	I_{xx}	I_{yy}	I_{zz}
1	0.0470910226	0.035959884	0.0376697645
2	0.027885975	0.020787492	0.0117520941
3	0.0266173355	0.012480083	0.0284435520
4	0.0131822787	0.009268520	0.0071158268
5	0.0166774282	0.003746311	0.0167545726
6	0.0070053791	0.005527552	0.0038760715
7	0.0008162135	0.0008735012	0.0005494148

Assumptions:

- I_{xy},I_{yz},I_{zx} can be ignored
- Coriolis can be ignored
- Robot joint angles are within +-0.00872664626 rad (default controller value limit)
- Joint angle errors are normally distributed, centered at 0, with variance equals controller limit
- Joint torque measurements are independent
- Joint torque sensor errors are normally distributed around 0, standard deviation of 0.5

“Bump detection” using particle filter:

- Goal: determine whether or not Baxter’s left arm has collided with an object, using particle filter in joint torque space
- Command input: joint angles
- “Movement” model: dynamics model
- Verify torque with torque sensors
- Re-sample the particles
- If resampled particles have average weights below set threshold, a collision has been detected. Halt movement.

Generating dynamics model:

Can get the following twists (ξ) from DH parameters:

Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7
0	0	-0.27	0	-0.201	0	-0.191
0	-0.27	0	-0.201	0	-0.191	0
0	0.069	0	0.433	0	0.808	0
0	-1	0	-1	0	-1	0
0	0	1	0	1	0	1
1	0	0	0	0	0	0

COM in global coordinates:

	x	y	z	Mass
Link 1	-0.00098	0.0178	0.19096	5.70044
Link 2	0.0053	0.13759	0.2673	3.22698
Link 3	-0.0013	0.3612	0.1195	4.31272
Link 4	0.0112	0.4592	0.1994	2.07206
Link 5	-0.0046	0.6769	0.2027	2.24665
Link 6	-0.0060	0.8685	0.1840	1.60979
Link 7	-0.0010	1.02128	0.1859	0.54218

We can place 7 frames at the center of mass, assuming they are aligned with principal inertia axis of each link, in the form:

$$g_i(0) = \begin{bmatrix} 1 & 0 & 0 & COM_i(x) \\ 0 & 1 & 0 & COM_i(y) \\ 0 & 0 & 1 & COM_i(z) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Link inertia matrix form:

$$\mathrm{M}_i = \begin{bmatrix} m_i & 0 & 0 & 0 & 0 & 0 \\ 0 & m_i & 0 & 0 & 0 & 0 \\ 0 & 0 & m_i & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \end{bmatrix}$$

$$[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad I_{zz}]$$

Body Jacobian

We then can calculate the body Jacobian corresponding to each g_i COM link frames using i twists that affect that frame:

$$Define:$$

$$Ad_g^{-1} = \begin{bmatrix} R^T & -R^T(\hat{p}) \\ 0 & R^T \end{bmatrix}$$

$$Body\ Jacobian:$$

$$J^b(\theta) = [\xi_1^+ \ \xi_2^+ \ \xi_3^+ \ \dots \ \xi_7^+]$$

$$With: \xi_i^+ = Ad^{-1}_{(exp(\hat{\xi}_i * \theta_i) * \dots * exp(\hat{\xi}_7 * \theta_7) * g_i(0))} * \xi_1$$

$$Where: \ exp(\hat{\xi}\theta) = \begin{bmatrix} exp(\hat{w}\theta) & (I - exp(\hat{w}\theta))(w \times v) + ww^T v\theta \\ 0 & 1 \end{bmatrix}$$

$$And: exp(\hat{w}\theta) = I + \hat{w}\,sin(\theta) + \hat{w}^2(1 - cos(\theta))$$

Some Jacobian results:

Joint 1 link body Jacobian:

```
[  0.0512, 0, 0, 0, 0, 0, 0]
[ -0.0791, 0, 0, 0, 0, 0, 0]
[       0, 0, 0, 0, 0, 0, 0]
[       0, 0, 0, 0, 0, 0, 0]
[       0, 0, 0, 0, 0, 0, 0]
[     1.0, 0, 0, 0, 0, 0, 0]
```

Joint 2 link body Jacobian:

```
[ - 0.00593*cos(T2 - 1.1) - 0.069,        0, 0, 0, 0, 0, 0]
[                -0.0685*cos(T2),  0.00529, 0, 0, 0, 0, 0]
[                -0.0685*sin(T2), -0.00269, 0, 0, 0, 0, 0]
[                              0,     -1.0, 0, 0, 0, 0, 0]
[                   -1.0*sin(T2),        0, 0, 0, 0, 0, 0]
[                        cos(T2),        0, 0, 0, 0, 0, 0]
```

Joint 3 link body Jacobian:

```
[       - 0.069*cos(T3) - 0.0718*sin(T2) - 5.46e-12*cos(T3)^2*sin(T2) - 0.00132*cos(T2)*cos(T3),       0.00132*sin(T3), 0.0718, 0, 0, 0, 0]
[                                                                 -0.109*cos(T3 + 0.722)*cos(T2), 0.109*cos(T3 - 0.849),      0, 0, 0, 0, 0]
[ - 0.0815*sin(T2) - 0.069*sin(T3) - 0.00132*cos(T2)*sin(T3) - 5.46e-12*cos(T3)*sin(T2)*sin(T3),      -0.00132*cos(T3), 0.0815, 0, 0, 0, 0]
[                                                                       -1.0*cos(T2)*sin(T3),          -1.0*cos(T3),      0, 0, 0, 0, 0]
[                                                                             -1.0*sin(T2),                     0,    1.0, 0, 0, 0, 0]
[                                                                          cos(T2)*cos(T3),          -1.0*sin(T3),      0, 0, 0, 0, 0]
```

Since there are too many terms for the remaining body Jacobians, results will not be shown here.

Inertia of whole system can be calculated:

$$M_{total}(\theta) = {}^{b^T}\sum_{i=1}^{7} J_i {}^{b^T} M_i J_i {}^{b}$$

We now calculate the gravity term of the system:
Have:

$$V(\theta) = g * \sum_{i=1}^{7} m_i * h_i(\theta)$$

Where $h_i(\theta)$ is the height of the COM of each link. These can be calculated by rotating the link COM frames:

$$g_i' = g_i(0) * \prod_{j=1}^{i} exp(\hat{\xi}\theta)$$

And pick out the (3,4) term of the matrix.
To calculate the gravitational component, we need:

$$N(\theta_i) = \delta(V)/\delta\theta_i$$

We can combine the results to get the following dynamics model:

$$M_{total}(\theta) * d(\theta)^2/dt^2 + N(\theta) = \tau$$

With τ being the 7x1 vector showing the joint torque values.
Due to the complicated nature of both $M_{total}(\theta)$ and $N(\theta)$ terms, also we expect that $d(\theta)^2/dt^2$ will remain small, we will ignore the $M_{total}(\theta)$ term of the model.

Limitations of the dynamics model:

Unfortunately, the inertia-mass matrix term of the dynamics equation was too complicated to calculate effectively in real time; hence only the gravitational term was used. The gravitational model derived here is then compared with the gravitational model built into Baxter to determine a collision event.

Error Sampling:

Joint Angle Error:

Assuming the joint position errors are normally distributed, we can generate samples from the distribution using rejection sampling:

$$\varepsilon_{\sigma^2}(x) = 1/(\sqrt{2 * pi * \sigma^2}) * exp(x^2/(2 * \sigma^2))$$

With x being the positional error
Define:

$$a = -3 * \sigma^2,\ b = 3 * \sigma^2,\ c = max(\varepsilon_{\sigma^2}(x))$$

We generate 2 random variables: r1 and r2 from uniform distribution, from 0 to 1
Have:

$r1' = a + r1 * (b - a)$

$r2' = r2 * c$

Evaluate:

$If\ r2' > \varepsilon_{\sigma^2}(r1')$

Then the sample is rejected. Resample r1 and r2 until:

$r2' <= \varepsilon_{\sigma^2}(r1')$

Then we accept the sample.

Our error then is $r2'$

And hence, joint position error is: $\theta_{input} + r2'$

Measurement Error:

Since torque measurement errors are assumed to be independent and normally distributed around zero, we have the following error distribution PDF:

$\varepsilon_{\sigma^2}(x) = 1/(\sqrt{2 * pi * \sigma^2}) * exp(x^2/(2 * \sigma^2))$

With $x = abs(T_{i,predicted} - T_{i,measured})$

And $T_{i,predicted}$ and $T_{i,measured}$ being the torque of joint i predicted from dynamic equations (with joint error included) and the torque measured from joint i.

Hence we can find the weight of particle n by multiplying the probability of the error of all 7 joints (since they are independent).

$p(z_t \mid x_t^n) = \prod_{i=1}^{7} \varepsilon_{\sigma^2}(x_i)$

Weights can then be normalized with:

$\eta = 1/N * (\sum_{n=1}^{N} p(z_t \mid x_t^n))$

With N being the total number of particles.

Code Overview

The code will be available to view in the github directory which contains this pdf. Here is an overview of what is contained in each directory:

Baxter_joint_position_commander: code to manually command baxter motions
Data_display: code to plot the torques experienced by baxter's limbs
Torque_tracker: code related to the dynamics of Baxter
Particle_filter: the actual particle filter code

Note that the first three directories listed were simply used during the development process. Particle filter is the only directory that is used in the final working version of our code, and so only that directory's contents will be detailed.

Inside particle_filter you will find a README, CMakeLists.txt with compilation instructions, and a src directory that contains the main code. There, torque_filter.cpp is where you'll want to look.

Overview of torque_filter.cpp:

Please note that the code is well commented, and should be fairly straightforward to understand. If you want to read the code yourself, I recommend starting on line 139 (particle_filter).

This code runs on ROS, and uses the ROS functionality to interface with Baxter. Lines 1-35 primarily set up this functionality, with the exception of two global variables which represent the number of particles (1000) and a tunable parameter to determine how sensitive baxter is to collisions.

Normal_pdf is a helper function which returns the density function of a gaussian (with provided mean and standard deviation) at a particular location. This is used in the rejection sampler functionality.

The rejection sampler acts as a generative gaussian model, allowing for samples to be taken from a gaussian with a specified mean and standard deviation. This functionality is in turn used by state_transition_sampler.

State_transition_sampler is a key piece of particle filter functionality. It takes a previous particle and an action (commanded position), and it samples a new particle from the distribution of particles conditioned on that choice of state and action.

The next function, importance_factor, evaluates the likelihood of a particular measurement of torques occurring at a particular robot state (in which the robot is motionless at specified joint angles). The particle filter uses a model of the robot's dynamics to supply the torques expected at the supplied state when this function is called.

Particle filter makes use of all of the above functionality. It closely follows the pseudocode provided in class, with one addition. The additional functionality evaluates overall consistency between observations and predictions, and uses this evaluation to determine whether a collision has been experienced or not.

We find that the robot can reliably detect collisions with almost no false positives under the conditions we specify. These conditions include a particular threshold sensitivity to discrepancies between prediction and observation, as well as commands that keep the robot in a specified region of joint space that agrees well with our dynamics model.

Finally, the main function commands baxter to make random arm motions, stopping only when the particle filter infers a collision or when baxter has completed ten motions in a row.

More details for the nuts and bolts of all of this functionality can be found in the code itself.

Conclusion and Experimental Results:

To test our approach, we ran particle_filter.cpp and the arm completed the movement successfully according to the commanded joint position ten times in a sequence using randomly generated joint angles with no collision messages. We ran it a second time, and after the third movement step, we blocked the movement by holding the arm in place. As a result, a warning message appeared saying that a collision is detected. We tried increasing and decreasing the threshold of the particle filter and it results in increasing and decreasing in the sensitivity of the collision detector respectively.

[WARN] [1491262192.761887619]: collision detected. breaking out of loop
[WARN] [1491262192.760291624]: average importance factor when collision triggered: 0.000010

In conclusion, we were able to detect collisions occurring at the left arm of Baxter using torque state estimation and particle filtering. We faced some challenges in modeling the arm due to the offset of the joints and the difficulty to properly detect the center of mass. Furthermore, because of the internal feedback system inside baxter, what supposed to be open loop commands are basically closed loop controlled internally. In future, we recommend using a self built and modeled robot, or a well technically detailed robot, to have more control over the computing parameters. In addition, including velocity in torque state calculation will result in a faster and more applicable response to collision. Also, Estimating the position instead of torque is a more direct approach, but it will require more time and higher processing power.

Links:

False Positive Test video: https://youtu.be/KIIvS2wJIs0
Collision Test video: https://youtu.be/_IrL2UueqPc
Github repository: https://github.com/McCheesecake/EECS-489-Baxter

References:

[1] Lijun Zhao, Xiaoyu Li, Peidong Liang, Chenguang Yang, Ruifeng Li, "Intuitive robot teaching by hand guided demonstration", *Mechatronics and Automation (ICMA) 2016 IEEE International Conference*, pp. 1578-1583, 2016, ISSN 2152-744X.

[2] C.Yang et al., “Robot Kinematics and Dynamics Modeling,” in *Advanced Technologies in Modern Robotic Applications*, Springer Singapore, 2016, pp. 27-48