Report on the recovery of motion of 2d-linked robot

Initial data

2 samples of data obtained from acceleration sensors from each link (is attached in Application 1)

Initial position velocity is zero for both links

Initial position of link1 (body A) pose expressed in euler sequential angles rotations is X = 178 deg,Y = -38.96 deg, Z = -144.15 deg

Initial position of link2 (body B) pose expressed in euler sequential angles rotations is X = -90.50

deg, Y = 63.73 deg, Z = -99.97 deg

Coordinates of link 1 sensor point in link1 coordinate frame is

Coordinates of link 1 end point in link1 coordinate frame is

Coordinates of link 2 sensor point in link2 coordinate frame is

Coordinates of link 2 end point in link2 coordinate frame is

Timestep dt = 0.01 seconds

# The task

Recover motion of every link, including:

Link1 position angles, angular velocities, angular accelerations in reference to base

Link2 position angles, angular velocities, angular accelerations in reference to link1 frame

Link2 position angles and rotation matrix in reference to base frame

Numerically motion recovery is to be done with Matlab.

# Solution

## Coordinate systems

In frame of this report we are using three different coordinate systems, connected by pure rigid rotation one to each other. They are:

* Base coordinate system (denoted with lower index B, for instance PB)
* Coordinate system, referenced with Link 1 (Body A) and denoted as L1, for instancePL1
* Coordinate system, referenced with Link 2 (Body B) and denoted as L2, for instancePL2

### Mathematical model

Ground point of mathematical model is describing of angles rotations of links to each other and base. Form the one hand the general case of this description is using of rotation matrix. But from the other hand numerical integration with rotation matrix is not trivial task and could require such mathematical apparatus like Jacobians to integrate all three angle at once. Also we are unable to use Euclidian angles to describe increment of angles because Euclidian angles are applied in sequence and we have to do it in once. Besides operation with Euclidian angles can cause well-known problems as gimbal lock. Also we have manually take care of keeping in frame of 0-360 degrees. Quite elegant way to describe rotation avoiding this problems is using a quaternions. Also quaternions are supported in Matlab with a number of functions for its quite complicated mathematics. So we here denote total amount of rotation with quaternion Q. So here in our report quaternion QL1 describes total amount of link1 from zero position to currently described position in base coordinate frame, and QL2 contains total amount of rotation of link 2 from its zero position to current position in reference to Link 1. For description of position of link 2 point in base coordinate frame we use quaternions QL1L2.

So we denote here that having a point position in local frame PL1 and know rotation from base zero position to current position QL1 we imply rotation of at quaternioin to obtain point actual position in base frame [1]:

(1)

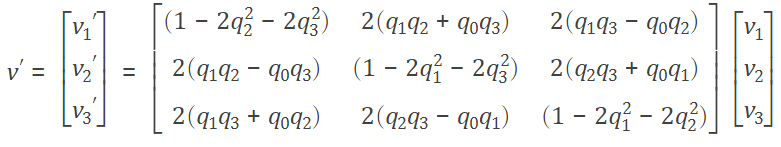
If we want to know coordinates of link 2 point PL2 in link 1 reference frame we do the same:

(2)

We note that operation \* here is not dot multiplication but special operation in hyperspace and introduced for simplicity, but in full case for quaternions **q** and original point **v** it is:







We will not refer to this full explanation more and for point and quaternions we will use sign \* for the operation in this report.

Having known coordinate of link 1 in base coordinate frame we can define its linear velocity through vector multiplication of angular velocity of link 1 in base frame [1] , formula 5.10:

(3)

The same for link1 and link 2 respectively:

(4)

Quaternion increment can be obtained from angular velocity numerical integration, we show it below. In the general case angular velocity is derivative on angle vector change:

(5)

Linear acceleration of these point (PB and PL1) will be [1](formula 6.12):

(6)

(7)

Where is angular acceleration of link 1 frame in base frame. Substituting (3) in (5) and (4) in (6) we obtain:

(8)

(9)

And linear acceleration is the derivative of the linear velocity on time:

(10)

(11)

So we now have basic mathematical appararus for solving the task of recovery of robot motion. But all we know about motion is acceleration of link middlepoint. So we start solving robot motion starting from this point. Obtained acceleration from link1 sensors is the sum of own motion of link1 and gravity acceleration

So our link1 motion in base coordinate frame is obtain sensors acceleration minus gravity acceleration, which is supposed to be constant (lines 114-117 of code)

(12)

From (10) we obtain increment of velocity (line 125):

(10)

This increment on k-th step we have to add to the previous step velocity to obtain actual velocity value

So now we have actual velocity of sensor point of link 1 and we have to numerically integrate it to obtain shift of physical angle. For this purpose we use quaternions properties. As far as link 1 is connected to the base allowing only link1 rotation, all of our *Vs* can be expressed as angular rotation, and this is will be angular rotation for the whole link in global/base frame. From (3) we obtain (line 148):

(13)

To apply quaternion paradigm of numerical integration we are going to use our actual (on the current step *k*) angular velocity in local link1 frame*,* sensor point position and sensor point velocity . Mathematical operation is identical to 11 (line 130). And local velocity we obtain by inverse rotation on our quaternion (line 129):

On the next step we are integrating this local to obtain addition to rotation (angles shift). We express this addition in quaternion in the following way. From (5) we obtain(line 133-139):

(14)

And having this addition in rotation all we have to do is implement addition of angle shift expressed in to total accumulated amount of rotation expressed by (line 145):

Initial sensor data, obtained from 4 accelerometers have to be put together and aligned with link direction. Sensors are located in circles like it is shown in Figure 1.

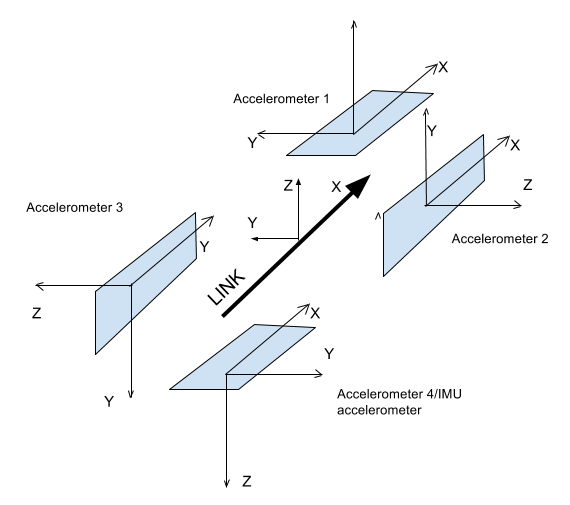


Figure 1 – Sensor location sheme on the link1

Accelerometer 1 is located in the top and it is already aligned to link 1 coordinate frame.Acelerometer 2 is on the right side and its measurement vector is to be rotated at -90 degrees. Accelerometer 3 is located at theleft side and is to be rotated at -270degrees (or 90 degrees).Bottom Accelerometer 4 on IMU is to be rotated at 180 degrees. All this operations are implemented for link 1(line 108-111) and link 2 (line 174-177).

The similar operation on motion recovery we are doing for the link2 considering that link 1 is stationary. Acceleration of link2 is the sum of link1 induced motion and gravity. So link 2 motion is(line 184-187) in link 1 frame is (lines 184-187,217):

can be found from (9) (line 209-211). Rotation of required point is obtained with (2) (line 201-205). Integration of acceleration to linear velocity (lines 222-224), linear velocity to angular(line 229) velocity, and angular velocity integration to quaternion (lines 233-241) are performed on the formulae written above.

# Literature

1. John J. Craig Introduction to Robotics: Mechanics and Control; 408 pages; Pearson; 3 edition (August 6, 2004) ISBN-10: 0201543613 ISBN-13: 978-0201543612 http://www.mech.sharif.ir/c/document\_library/get\_file?uuid=5a4bb247-1430-4e46-942c-d692dead831f&groupId=14040
2. http://stackoverflow.com/questions/12053895/converting-angular-velocity-to-quaternion-in-opencv