Localization of a Cable Robot

Chintan Dalal
cadalal@seas.upenn.edu

Mubeen Bhatti
mubeen@seas.upenn.edu

Abstract

The problem of identifying the location of a robot in a global or relative coordinate system is very important. It is necessary for navigation and control of the robot in an environment. Simple techniques like integrating measurements using reading from encoders, gyros and accelerometers without considering the inherent noise associated with each measurement leads to a poor estimate of the robot which degrades with time and these estimates cannot be used for a practical application. In this project we plan to use an Unscented Kalman filter (UKF) approach to do a localization of the robot. UKF was preferred over Extended Kalman filer because of its higher accuracy and easier implementation.

Problem Definition

To localize the position of a cable robot using three noisy distance positions. As can be seen from the diagrams below the intersection of two spheres would form a ring and the intersection of third would narrow down the point to two points but since we know that the cable robot can only be below the plane of the ceiling that solution would be admissible for this problem.

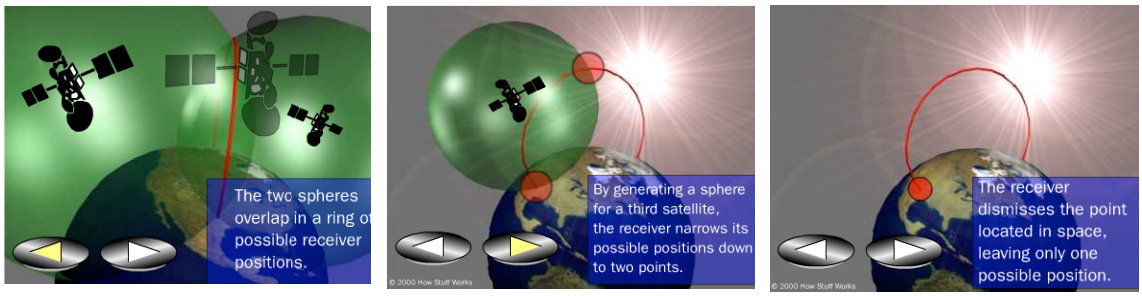


Figure: Explaining the Process of trilateration [5]

The method used to solve the location of a robot using three-distance measurement is done by method of trilateration. Two methods are investigated in [2] to solve the problem of finding the location of a device using three measurement readings. We used the technique of linearization of the minimization problem as presented in [2]. Details of the implementation have been provided in the algorithmic section of this report.

Report Framework:

This report is divided in four parts, the first part explains the mechanical design of the robot, it's CAD diagrams are presented and then the actual implementation that was made is shown. The second part describes the challenges that were faced for electrical design, like establishment of hardware in a loop for Matlab. The third part presents the algorithm for tracking the lengths of the robots from the different attachments (here attachment position is known) and then SLAM was implemented to localize the attachment position (here attachment position is unknown) and simultaneously track the robot with respect to localized attachment. The fourth and last part of the report presents the results from a simulation of the proposed algorithms and the work we plan to achieve in future.

Mechanical Design:

The basic Mechanical elements of the Cable Robot are:

Spool Assembly:

1) One end of the wire is wound around the Spool, its other end going through the Standoff assembly is connected to the hook on the wall.

2) One end of a spool is actuated by Maxon DC motors, while its other end is free to revolve in a housing with Bearings.

Standoff Assembly:

1) Pulley-rod assembly is made in a housing, for the wire to be directed towards the hook.

The major manufacturing problem was making entire Assembly as light and compact as possible. For which every part was made from thermoplastic, for its good strength, easy machining and low density.

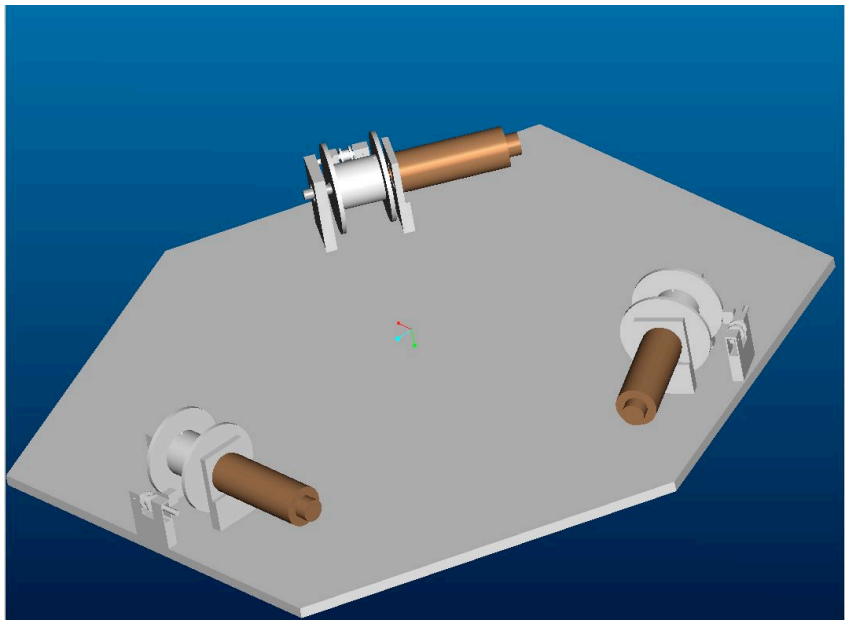


Figure: Assembly view:

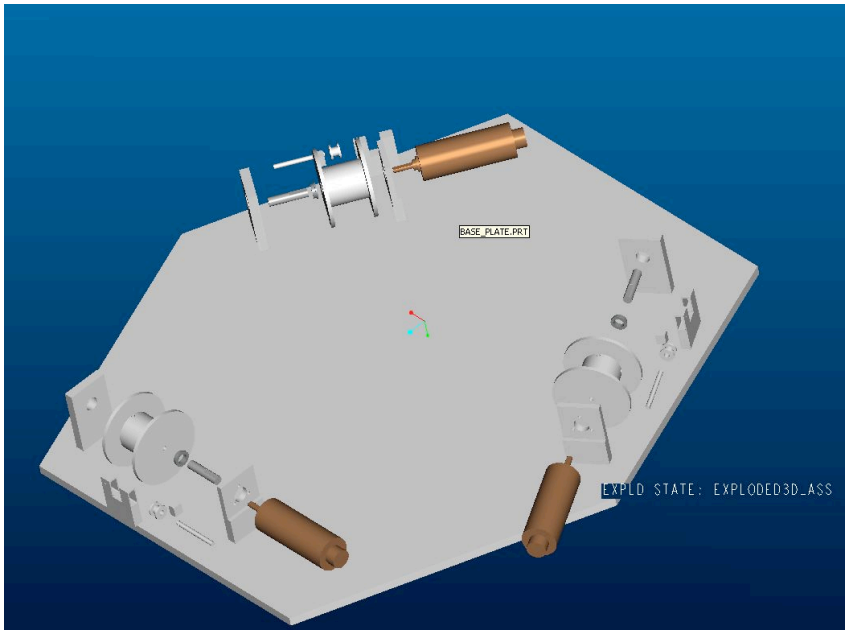
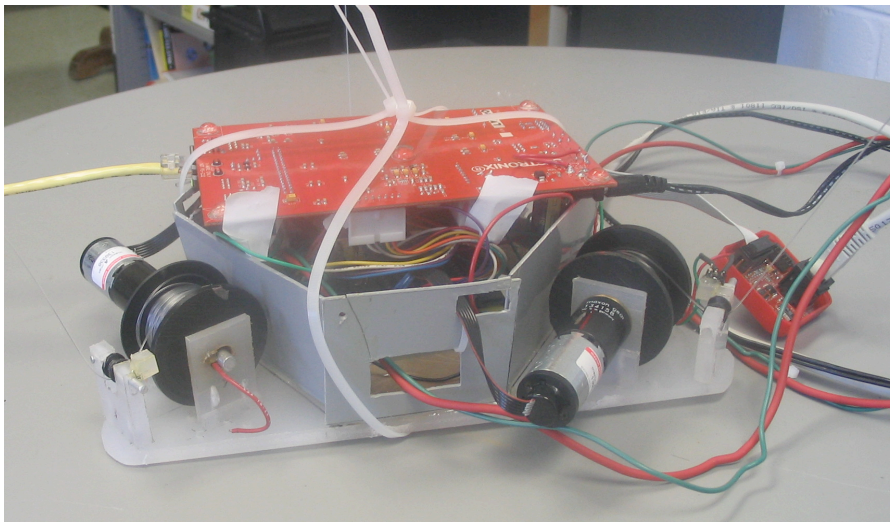
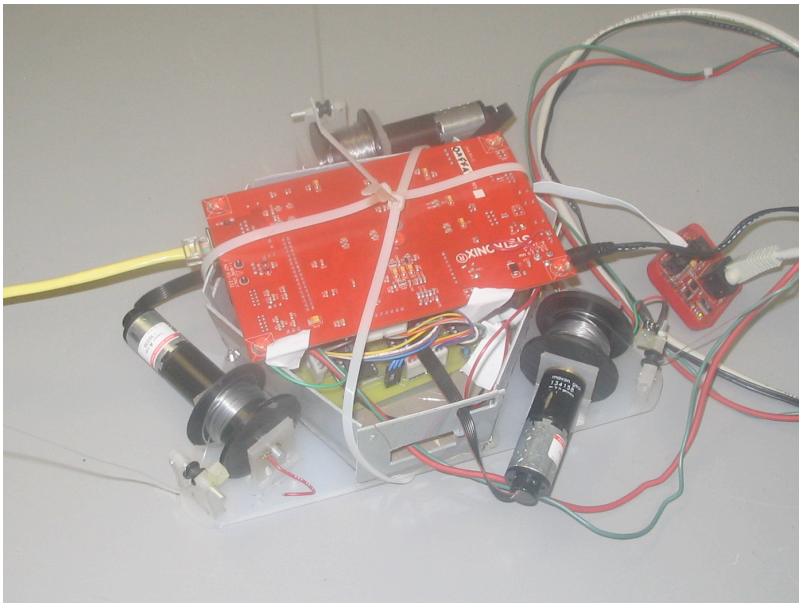


Figure: Disassembly View:





Electrical Design:

The electronic system for this cable robot consisted of an embedded server running on a aJ-100, Java core microcontroller. This allowed soft real time link up with a desktop PC running Matlab through its instrumentation toolbox. aJ-100 basically acted as a communication agent that parsed commands from Matlab and then passed on the required parameters to a Motor precision controllers, LM629 developed by National. This Motor controller along with a LMD 18200 allows very precise control of servo/dc motor allowing currents up to 3A.

The figure below shows the basic setup required for one motor control

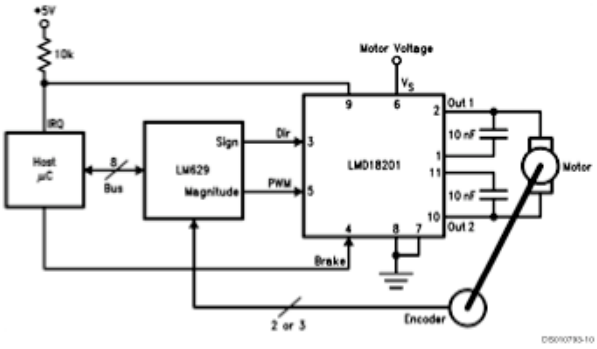
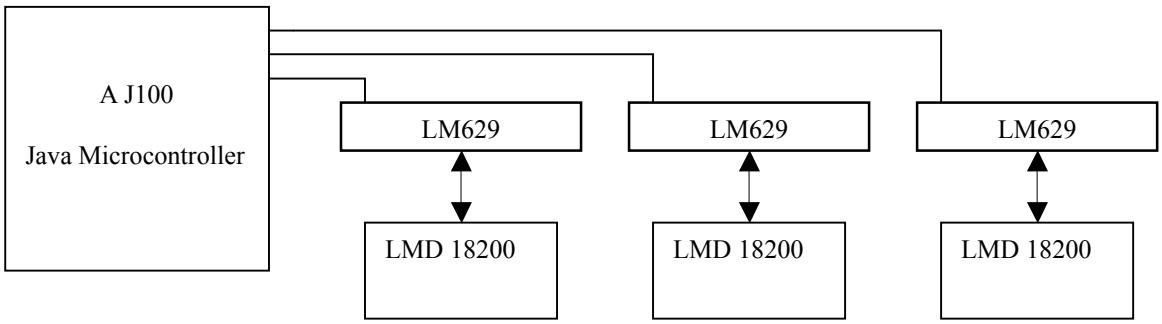


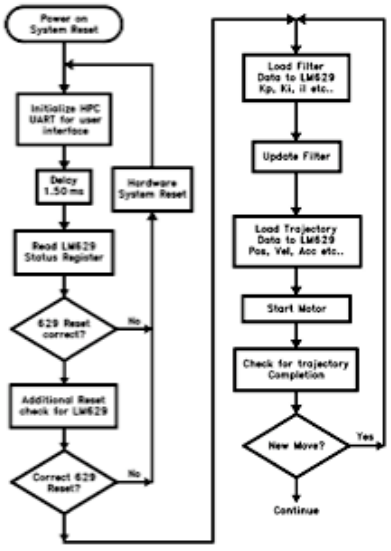
Figure: Basic Motor Control of LM629 and LMD18201

For the cable robot presented in this project we used a set of three of this kind of motor control circuits coupled together in parallel.

The schematic below shows the diagram for



The figure below shows is the basic instructions that cycle that have to be done in order to be able to communicate the filter parameters required for LM629 to operate properly.

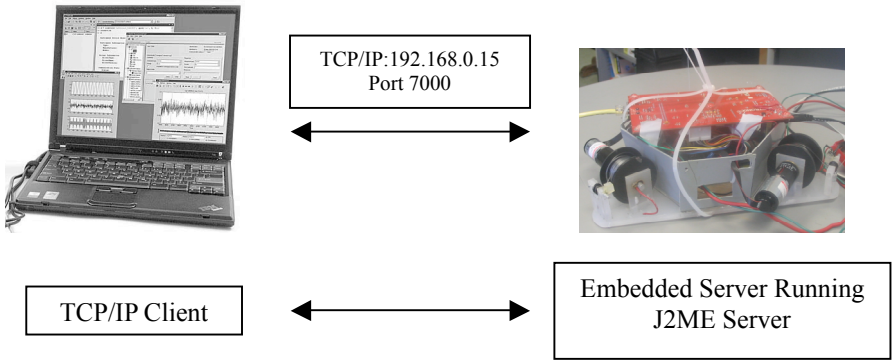


Flow Chart of Commands for LM629

The interface was implemented in such a way that the direction, distance and time were required and the filter parameters velocity, acceleration and position coefficients were calculated at runtime in order to be able to complete a standard 1:3:1 trapezoidal velocity profile. After the step is completed measurements

are taken from the two channel encoder to find out exactly how much distance was moved by the motors and this is reported to Matlab, along with the echo from of the command received.

Matlab Instrumentation ToolBox was used to establish a soft realtime connection between the Java embedded microcontroller and a desktop control station.



Algorithm:

For localizing the robot position in 3D space, Unscented Kalman Filter [3] was applied. Because of its ease in calculation and the precision in prediction upto fourth order Taylor series expansion.

The algorithm is as follows:
Initialization:

1) As all the information of the robot platform can be acquired from the three wires going from the motor to the hooks. The state is defined by the three lengths (l1, l2, l3). Precise upto 1mm (as all the measurement done in millimeter).
2) The initial covariance matrix (i.e. uncertainty in initial measured lengths l1, l2, l3) is assumed 10% of the lengths. The process noise (given by friction in mechanical parts and microcontroller error) and the measurement noise (given by encoder and the processor error) are assumed .03 for all the lengths.

$$\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$$
$$\mathbf{P}_0 = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]$$
$$\hat{\mathbf{x}}_0^a = E[\mathbf{x}^a] = [\hat{\mathbf{x}}_0^T \; \mathbf{0} \; \mathbf{0}]^T$$

$$\mathbf{P}_0^a = E[(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)^T] = \begin{bmatrix} \mathbf{P}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_\mathbf{v} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_\mathbf{n} \end{bmatrix}$$

Where, $\mathbf{x}^a = [\mathbf{x}^T \; \mathbf{v}^T \; \mathbf{n}^T]^T$ =state
$\mathbf{P}_\mathbf{v}$=process noise cov., $\mathbf{P}_\mathbf{v}$=measurement noise cov.

The state is then passed through discrete time steps. From the results it seems that after 100 steps the platform converges to the actual position

Sigma Points:

1) Cholesky decomposition is done on the covariance matrix and each column is then added to the state mean. Hence we get 19 points i.e. 2*L+1.where L is the dimension of the state. Note: transpose is done before adding, because of the function chol in matlab, to get lower triangular matrix.

$$\mathcal{X}_{k-1}^{a} = \left[\hat{\mathbf{x}}_{k-1}^{a} \;\; \hat{\mathbf{x}}_{k-1}^{a} \pm \sqrt{(L+\lambda)\mathbf{P}_{k-1}^{a}}\right]$$

Where $\mathcal{X}^{a} = [(\mathcal{X}^{x})^{T} \; (\mathcal{X}^{v})^{T} \; (\mathcal{X}^{n})^{T}]^{T}$, λ=scaling parameter=0,as was not needed to get convergence

Process Model:

1) Process noise is added to the sigma points to get the new sate. Here the rows of 4 to 6 from the sigma point's matrix are added to its 1 to 3 rows. Then the empirical mean and covariance of 19 points is taken, to get xkminus and Pkminus.

$$\mathcal{X}_{k|k-1}^{x} = \mathbf{F}[\mathcal{X}_{k-1}^{x}, \mathcal{X}_{k-1}^{v}]$$

$$\hat{\mathbf{x}}_{k}^{-} = \sum_{i=0}^{2L} W_{i}^{(m)} \mathcal{X}_{i,k|k-1}^{x}$$

$$\mathbf{P}_{k}^{-} = \sum_{i=0}^{2L} W_{i}^{(c)} [\mathcal{X}_{i,k|k-1}^{x} - \hat{\mathbf{x}}_{k}^{-}][\mathcal{X}_{i,k|k-1}^{x} - \hat{\mathbf{x}}_{k}^{-}]^{T}$$

Where Wi=1=assuming that all sigma points are equal weightage

Measurement Model:

1) Measurement noise is added to the sigma points(here rows 7 to 9 is added to 1 to 3) to get the measurement update. Mean ykminus and covariance Pyy is then found.

$$\mathcal{Y}_{k|k-1} = \mathbf{H}[\mathcal{X}_{k|k-1}^{x}, \mathcal{X}_{k-1}^{n}]$$

$$\hat{\mathbf{y}}_{k}^{-} = \sum_{i=0}^{2L} W_{i}^{(m)} \mathcal{Y}_{i,k|k-1}$$

$$\mathbf{P}_{\mathbf{y}_k \mathbf{y}_k} = \sum_{i=0}^{2L} W_{i}^{(c)} [\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_{k}^{-}][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_{k}^{-}]^{T}$$

2) After calculating covariance between measurement estimate and the state estimate (Pxy), Kalman gain is found. Applying the equation as shown in figure below we get the estimate of the new sate and its covariance, which convergence to the actual robots lengths after 100 iterations.

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sum_{i=0}^{2L} W_{i}^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_{k}^{-}][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_{k}^{-}]^{T}$$

$$\mathcal{K} = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\mathbf{y}_k \mathbf{y}_k}^{-1}$$

$$\hat{\mathbf{x}}_{k} = \hat{\mathbf{x}}_{k}^{-} + \mathcal{K}(\mathbf{y}_k - \hat{\mathbf{y}}_{k}^{-})$$

$$\mathbf{P}_{k} = \mathbf{P}_{k}^{-} - \mathcal{K} \mathbf{P}_{\mathbf{y}_k \mathbf{y}_k} \mathcal{K}^{T}$$

Where $\hat{\mathbf{x}}_k$, $\mathbf{P}_k$ =new mean and covariance estimate of the state.

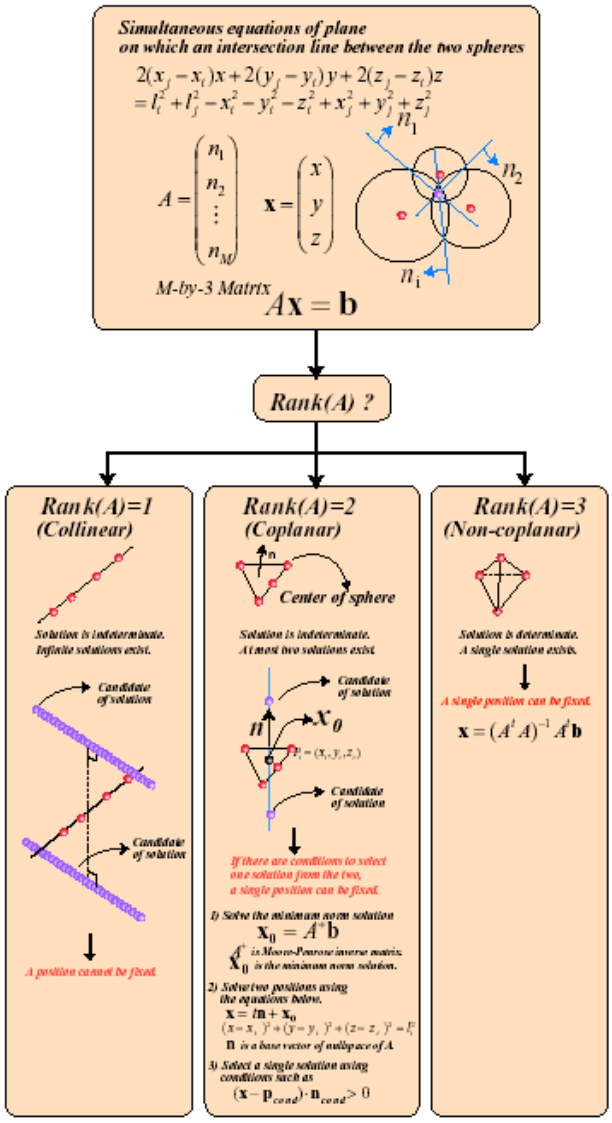
Multilateration:

Once the lengths are known, the platform is mapped to the Cartesian coordinates by using Multilateration. The multilatertion method used is called linearization of the minimization problem.
Its algorithm is follows:

1) Drawing three spheres around the hook points with radius equal to the length calculated will give intersection points. Where in, one is above the ceiling (which is eliminated) and one is below (which is selected).
2) Linear equations are: (X-Xi)^2-(Y-Yi)^2-(Z-Zi)^2=li^2
   where i=1,2,3, Xi,Yi,Zi are three hooks points and X,Y,Z is the platform coordinate (assuming point mass).
3) The simultaneous equation is: A*P=B

Where, A = [2*(x2 - x1) 2*(y2 - y1) 2*(z2 - z1);
2*(x3 - x1) 2*(y3 - y1) 2*(z3 - z1);
2*(x3 - x2) 2*(y3 - y2) 2*(z3 - z2)];
B= [l1*l1 - l2*l2 + x2*x2 - x1*x1 + y2*y2 - y1*y1 +z2*z2 -z1*z1;
l1*l1 - l3*l3 + x3*x3 - x1*x1 + y3*y3 - y1*y1 +z3*z3 -z1*z1;
l2*l2 - l3*l3 + x3*x3 - x2*x2 + y3*y3 - y2*y2 +z3*z3 -z2*z2];
P=(X,Y,Z) =platform position
4) If the rank of A is three then the position is calculated using
P=inverse (transpose (A)*A)*transpose (A)*B
5) If the rank of A is 2, then the inverse of A cannot be found so a pseudo inverse technique has to employed to solve this equation, this technique as suggested by [2] is 'Moore-Penrose' inverse. This inverse allows us to calculate proper values of X and Y however the value of Z is not correctly found. To calculate the value of Z the intersection of a line that is parameterized by the calculated values of X and Y with one of the spheres is calculated. The value of Z that is below the z values of the attachments is used.

The diagram below shows the process of calculating the values of x,y,z from the three distance measurments



Note: the results are obtained from simulated data. Hence once the platform is in complete working

condition we can obtain the results from the real data and then learn also the noise parameters.

SLAM:

The above tracking was done when the hook position was known and the platform was localized with respect to the hooks. But when the hook position is not known, then one has to localize both the platform and the hooks simultaneously. For this simultaneous localization and mapping we implement particle filter algorithm[4].

The algorithm is as follows:
Initialization:
1) The state contains the x,y,z position of the platform, the weights of each particle, the mean position and covariance of each hooks as observed by each particle.
2) Global position of three hooks is assumed (assigned initial maximum value of the room to incorporate all the particles in it).
3) Particles are initialized randomly in the convex hull formed by the hooks, all given equal weights.

The particles are then propagated by discrete time steps through process and observation model.
Process Model:
1) As the position of the platform and the Hooks are in Cartesian coordinates, the lengths between each hook and the platform is found by subtracting two vectors and finding norm to get l1,l2,l3 for each particle.
2) Each particle then takes the step, which is desired by the user, the process noise is included to incorporate the mechanical and electronics uncertainty in change in length.
3) Because the process step and the noise is in the l1,l2,l3 space, we add this change in l1,l2,l3 to particle's original l1,l2,l3 and then use trilateration algorithm to map it back into Cartesian coordinates.
4) We take three steps to get three state of each particle, reason is mentioned in below.
Measurement Model:
1) Observation is taken, which is the initial length measured from the encoder for three spools (before hooking the platform), plus the change recorded by encoders when each steps that was taken in process model, plus the noise in the encoder readings.
2) Now for each particle we know its three states, hence using that three points we draw three spheres, of the observed lengths as the radius, to intersect at a point which is our observed hook. Trilateration was used again to find the hook points from each particle.
Note: the problem arises when the points of particles are linear, which gives infinite intersection points. For that we take more number of steps to get some more points which gives unique answer.
3) Now the Global Hook points which was assumed in the starting is merged with the observed Hook for each particle to get updated local mean and covariance of the hooks for each particle.
4) The equation for merging is, from [1],
mergcov=observedHookCov-
observedHookCov*inv(observedHookCov+globalHookCov)*observedHookCov;
mergmean=observedHookMean+observedHookCov*inv(observedHookCov+globalHookCov)*(globalHo
okMean-observedHookMean); (mubeen write in equation form)

The heuristics for merging is al follows:

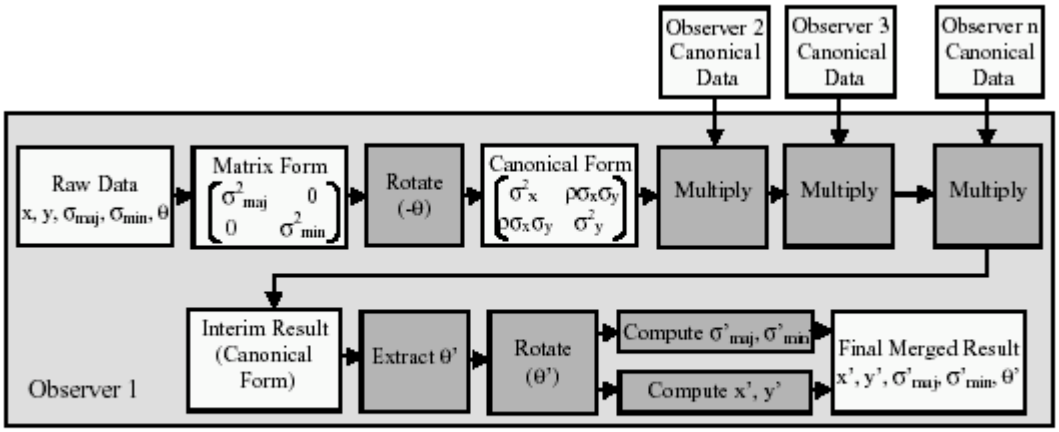


Figure 2. Block diagram of the multi-distribution merging process. Multiplication is conducted using the mathematical formulation described above. Each subsequent distribution is merged with the previous result, and the final parameters are extracted.

The canonical form is obtained as below, by rotating it with an angle theta it makes with the global reference frame.

$$C^{-1} = R(-\theta)^T C_L^{-1} R(-\theta) \quad \Rightarrow \quad C = R(-\theta)^T C_L R(-\theta)$$

where

$$C_L = \begin{bmatrix} \sigma^2_{maj} & 0 \\ 0 & \sigma^2_{min} \end{bmatrix}$$

Then the merged covariance and mean is given by,

$$C' = C_1 - C_1 \left[C_1 + C_2\right]^{-1} C_1$$

$$\hat{X}' = \hat{X}_1 + C_1 \left[C_1 + C_2\right]^{-1} \left(\hat{X}_2 - \hat{X}_1\right)$$

Here C' =merged canonical covariance, $\hat{X}'$ =Merged mean, C_1, C_2 = canonical covariance of two individual distribution. $\hat{X}_1$, $\hat{X}_2$= Mean for two individual distributions.

The principal axis angle is obtained from the merged covariance matrix entries:.

$$\theta' = \frac{1}{2} \tan^{-1}\left(\frac{2B}{A - D}\right)$$

A, B, and D are top left, top right/lower left, and lower right entries, respectively.

Lastly, the resulting major and minor axis standard deviations are extracted by rotating the covariance matrix to align with those axes and reversing Equation

$$C' = R(\theta')^T C' R(\theta')$$

5) weights are then assigned to each particle from the probability of the observed hook seen as global hooks. The equation is:

wdash=sqrt(det(observedHookCov)/(2*pi^3))*exp(-.5*(globalHookMean-observedHookMean)'*inv(covcon1)*(globalHookMean-observedHookMean));

Resampling:

1) After weights are assigned to each particle, resampling is done, by normalizing and taking cumulative sum of the weights, then random numbers are generated from 0 to 1. Number of random numbers is equal to no.of particles

2) Then each random number is compared with every particle's weights. All the particles which have cumulated summed weight greater then a particular random number is grouped. The minimum from that group is selected for the next state. Where in eventually, all the particle which shows maximum weight, ones whose belief in the observed and the global hook is as close as possible, is selected repeatedly for the next state.

.

Now belief of all the particles individual hook position is merged to get an updated Global Hook position (same method is used to merge[1] as shown above).
Hence finally the Hook converges to the actual hook position and the platform is also localized.

Results:

The graph below shows the results of trilateration using three measurement readings. As it can be seen there were multiple answers to the problem so steps had to be taken in order to select an admissible solution. One assumption that was made was that since the cable robot is hanging from the attachments it cannot be above the attachments so points were picked that were below the plane containing all the attachments. When this was done we got two sets of readings two select the proper cluster of readings from them the fact that was used was that the net unit vector from the cable robot to the landmarks should be in a pure positive z-plane, this was a requirement since the robot can be considered as a point mass hanging from the attachments so for stable equilibrium to exist the net force upwards should be purely z so that it would support the weight of the robot.

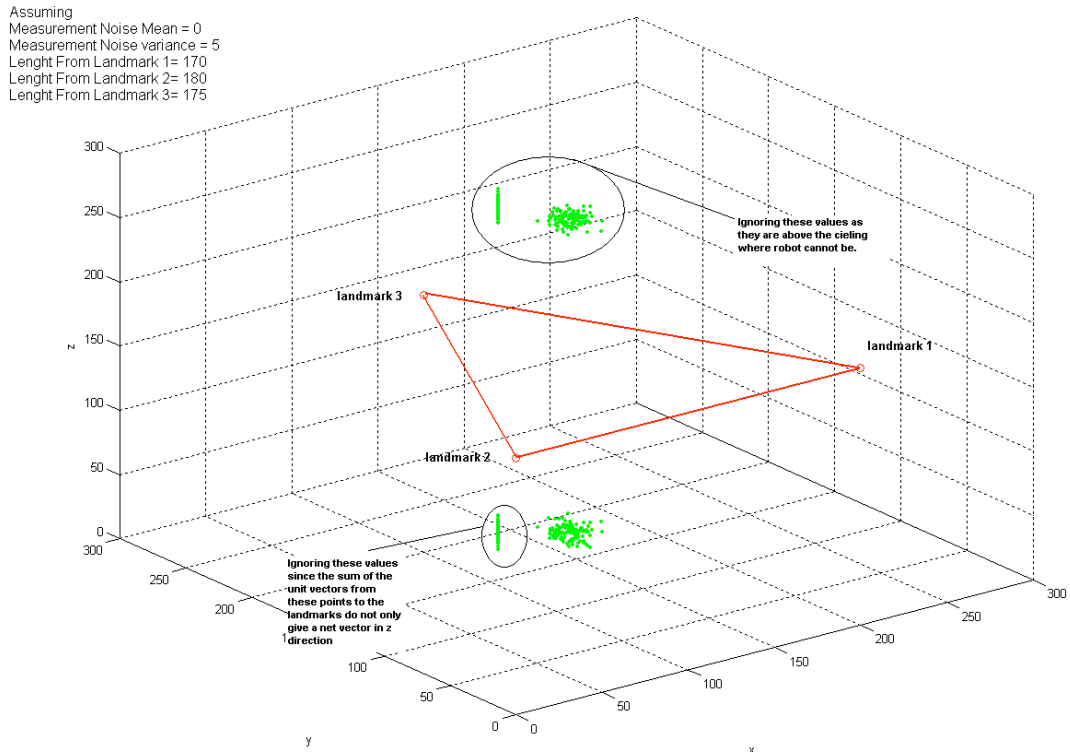


Figure: Estimation in 3D Cartesian coordinate

The diagram below shows the tracking of the value of the distance from the three attachment points. The path selected for this simulation was a random normally distributed of mean zero and variance 10.

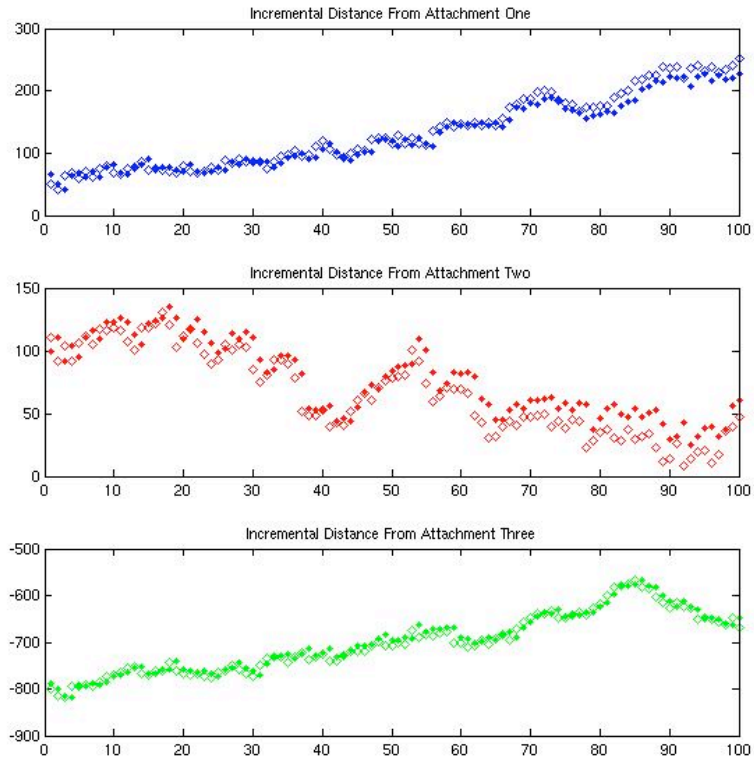
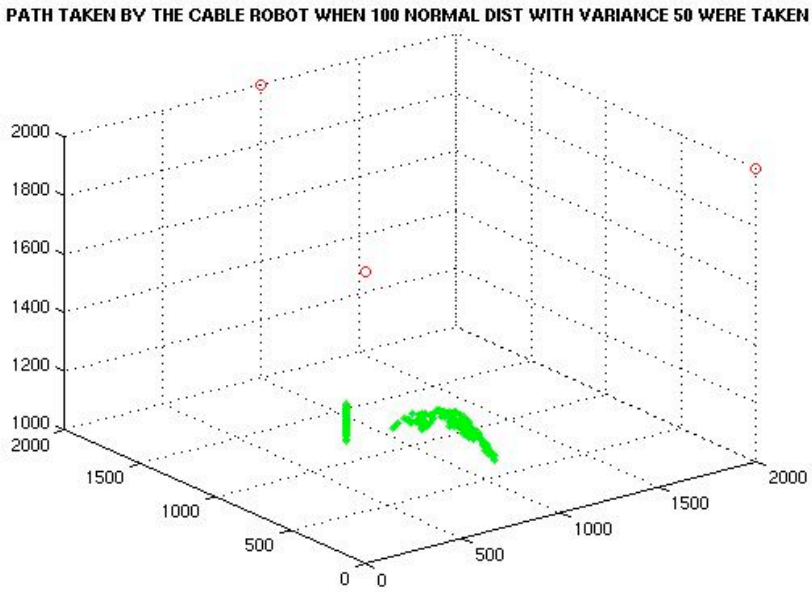


Figure: Result for Tracking Noisy Measurements Of Distance

The graph below shows the result of trilateration of the three length measurements using the mean value of the sigmapoints obtained by using the tracking above.



Results for SLAM:
There were number of problems while implementing SLAM which we are still working on:
1) There are some particles for which the spheres do not intersect at all. If that happens in process noise we can discard that particle, as for true particle state all the wires should converge to get a point mass. If it happens when doing trilateration in measurement model, we take more no of particles, and still if doesn't intersect we discard that particle too.
2) It was just an intuition that if the points in measurement model are linear then by taking more points we can get true observed hook mean, for that particle. But could not mathematically prove, if by going that way we get asymptotically true mean and covariance of the hook points.

Future Work:

1) Once the SLAM problem is completely solved, we would like to implement motion planning. This can be considered in couple of parts:
a) Planning for obstacle avoidance, in a room which present at different location in 3D space
b) Planning for avoiding its own singularities in space .i.e. the location it can't be reached, due to space constraints.
2) We also plan to implement Multi Robot coordination. Where in number of cable robots will be hanging in a room which will coordinate with each others space in such a way, that desired task is achieved.

References:

1) Ashley W.Stroupe, Martin C.Martin, and Tucker Balch 'Merging Gaussian Distribution for Object Localization in Multi-Robot System'.
http://www.ri.cmu.edu/events/iser00/papers/stroupe.pdf
2) Yoshifumi Nishida,Hiroshi Aizawa,Toshio Hori '3D Ultrasonic Tagging System For Observing Human Activity'
http://www.dh.aist.go.jp/~ynishida/papers/IROS2003.pdf
3)Eric A.Wan and Rudolph van der Merwe 'The Unscented Kalman Filter For Nonlinear Estimation'.http://graphics.snu.ac.kr/statistical2004/KalmanFilter/wan_AS-SPCC_2000_Unscented.pdf
4) Michael Montemerlo and Sebastian Thrun 'FastSLAM: A Factored Solution to the simultaneous localization and mapping problem'.
http://robots.stanford.edu/papers/montemerlo.fastslam-tr.pdf
5) www.howstuffworks.com

Codes:

1) Electronics:
   - Embedded Java Server: dataSocketHandler.java multithreaded handler for server handling
   - Embedded Java Server: CableRobot.java Motor.java Execute.java Clock.java are used for system level programming with LM629
   - MATLAB-client: commandTerminal.m used to send commands
   - MATLAB-client: takestep.m used to take one step at a time.
   - MATLAB-client: char2int.m int2char.m used to parse integer to bytes.
2) Tracking:
   - Matlab-localization: Simplelocalization contains locaization algorithm using trilateration and UKF
   - Matlab-localization: SLAM: the main file is slamcable.m, while triliterate.m is used to call for in process model and invTriliterate.m is used to call for in measurement model and findlength.m is used to find the length between the particle and the hook.