Term Paper

Cooperative Localization Using Posterior Linearization Belief Propagation

Group-9

Author:	*Student Number:*
SHIVRAM MEENA	150686
SHASHI KANT GUPTA	160645
MAMILLA SIVASANKAR	17104091
PRADEEP KUMAR	18104074
ALLAPARTHI VENKATA SATYA VITHIN	18104265

Statistical Signal Processing EE602A

November 15, 2018

Contents

1 AIM/OBJECTIVE

To infer the positions of the sensor nodes in cooperative fashion using the posterior linearization belief propagation (PLBP) algorithm with nonlinear measurements.

2 PROBLEM DEFINITION

In cooperative localization, there are some anchor nodes whose positions are known accurately. The remaining nodes infer their positions based on intercommunication between themselves, each node is connected to some other nodes which pass some messages to it for example distance. The measurements are generally non linear. Hence, we linearize the model by using statistical linearization using Sigma Points. Belief Propagation algorithm will be applied on the linearized posteriors to update the node position using Kalman filter updates. Strategies which increases the computational efficiency and noise robust are required.

3 SYSTEM MODEL/METHODOLOGY

A graph $G = (V, E)$ is formed by a collection of vertices/nodes $V = (1, ..., m)$, where m is the number of nodes, and a collection of edges $E \subset V \times V$. Each edge consists of a pair of nodes $(i, j) \in E$. The state of node i is represented by $x_i \in R^{n_x}$

We assume x_i has Gaussian PDF

$$p_i(x_i) = \mathcal{N}(x_i; \overline{x}_i, P_i) \tag{1}$$

With x_i representing the state of the node and $\overline{x}_i, P_i$ are mean and co-variance respectively

1. Defining the system model.

$$z_{i,j} = h_{i,j}(x_i, x_j) + n_{i,j} \tag{2}$$

   $h_{i,j}$ represent the measurement fuction between two nodes i and j. $n_{i,j}$ is the noise measurement.

2. Nonlinear measurement function.

$$h_{i,j}(x_i, x_j) = \sqrt{(p_{x,i} - p_{x,j})^2 + (p_{y,i} - p_{y,j})^2} \tag{3}$$

3. Linearization model of non linear measurements

$$h_{i,j} \approx A^1_{i,j} x_i + A^2_{i,j} x_j + b_{i,j} + e_{i,j} \tag{4}$$

   (a) Select m sigma points $\chi_0, \chi_1, \cdots, \chi_m$.

   (b) Propagate sigma points $Z_j = h(\chi_j)$

   (c) Compute mean and variance.

$\overline{z} = \sum_{j=1}^{m} \omega_j \mathcal{Z}_j (5)$

$$\Psi = \sum_{j=1}^{m} \omega_j \left(\mathcal{X}_j - \overline{x}\right) \left(\mathcal{Z}_j - \overline{z}\right)^T \tag{6}$$

$$\Phi = \sum_{j=1}^{m} \omega_j \left(\mathcal{Z}_j - \overline{z}\right) \left(\mathcal{Z}_j - \overline{z}\right)^T \tag{7}$$

$$\begin{aligned} A^+ &= \Psi^T P^{-1} \\ b^+ &= \overline{z} - A^+ \overline{x} \\ \Omega^+ &= \Phi - A^+ P \left(A^+\right)^T \end{aligned} \tag{8}$$

4. Belief propagation on linearized model.

   (a) The message $\mu_{i \to j}$ from node i to j is given as

$$\mu_{i \to j} \propto \int l_{i,j}(z_{i,j} | x_i, x_j) \mathcal{N}(x_i; \bar{x}_i, P_i) \prod_{p \in n(i) \setminus \{j\}} \mu_{p \to i}(x_i) dx_i \tag{9}$$

   (b) under approximation

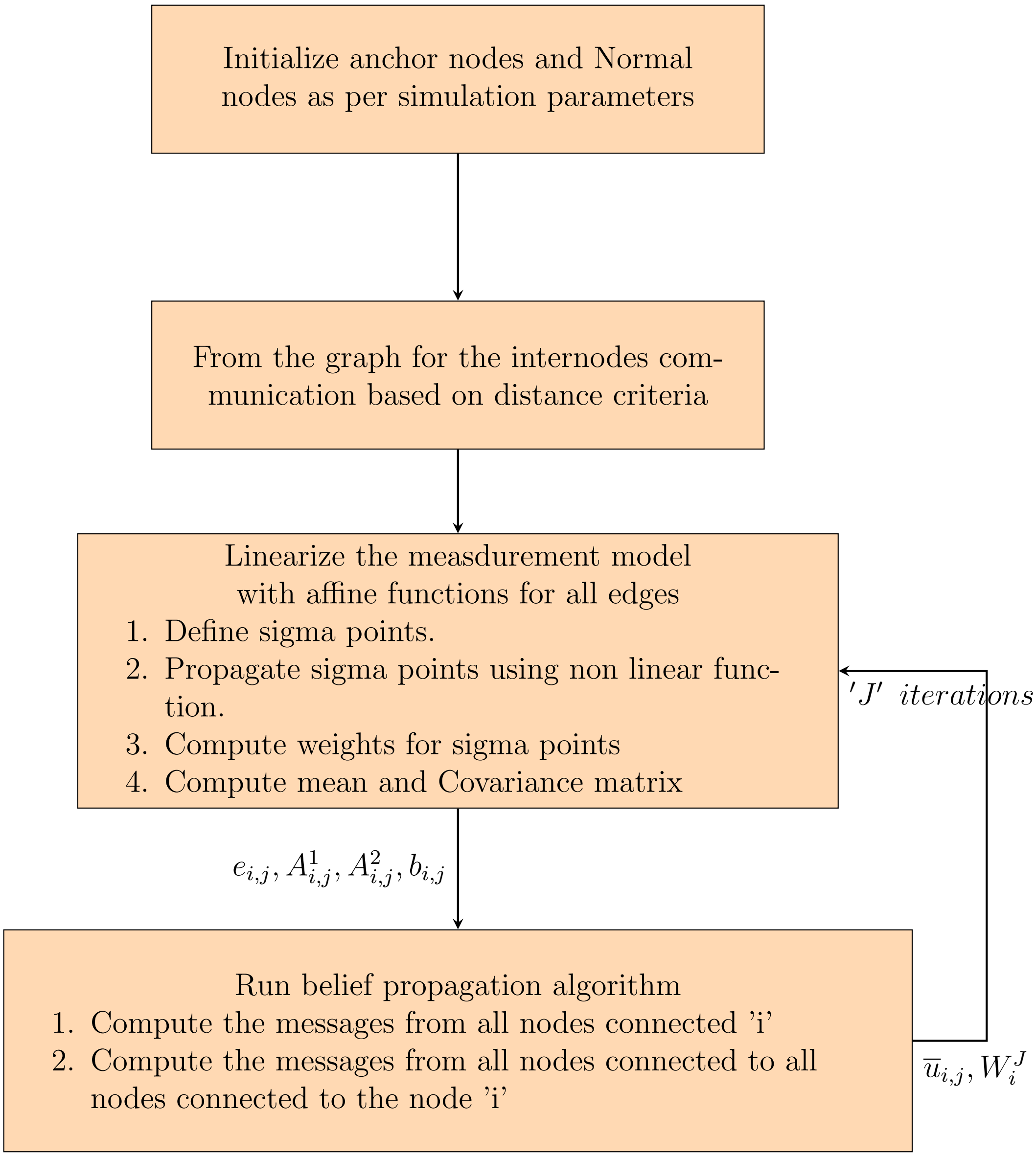
$$\mu_{i \to j}(x_j) \propto \mathcal{N}(\alpha_{i \to j}; H_{i \to j} x_j, \tau_{i \to j}) \tag{10}$$

$$\alpha_{i \to j} = z_{i,j} - A^1_{i,j} \bar{x}_{i \to j} - b_{i,j} \tag{11}$$

$$H_{i \to j} = A^2_{i,j} \tag{12}$$

$$\tau_{i \to j} = R_{i,j} + \Omega_{i,j} + A^1_{i,j} P_{i \to j} (A^1_{i,j})^T \tag{13}$$

3.1 Algorithm flow



4 MATLAB SIMULATIONS AND RESULTS

A MATLAB simulation is performed according to the system model described below.

$$\begin{aligned}
Area : &\quad 100mX100m \\
Number\ of\ Anchor\ nodes : &\quad 13 \\
Number\ of\ Normal\ nodes : &\quad 100 \\
Variance\ of\ Normal\ Nodes : &\quad 100m \\
Variance\ of\ Anchor\ Nodes : &\quad 0.01m \\
Number\ of\ Iterations : &\quad 20 \\
Range\ Measurement\ error : &\quad 1m
\end{aligned}$$

Results for the simulation is shown below. Please find the attached Matlab Code in Appendix at the end of this document.

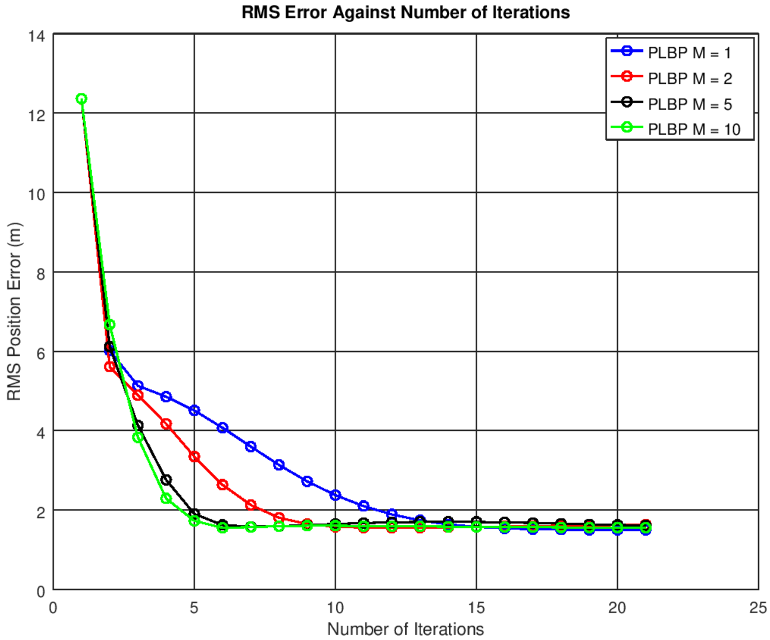


Figure 1: RMS error against number of iterations. Performance improves with M , number of BP iterations per linearisation.

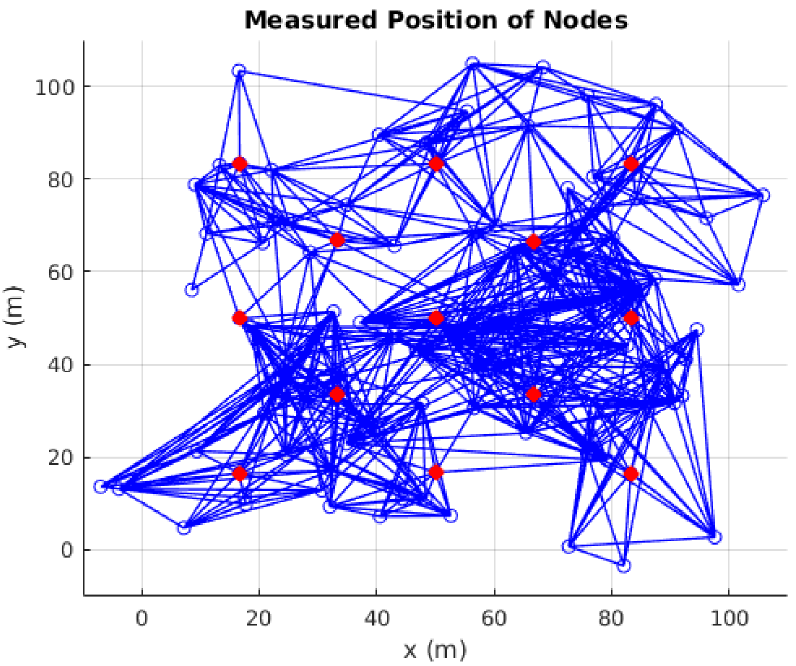
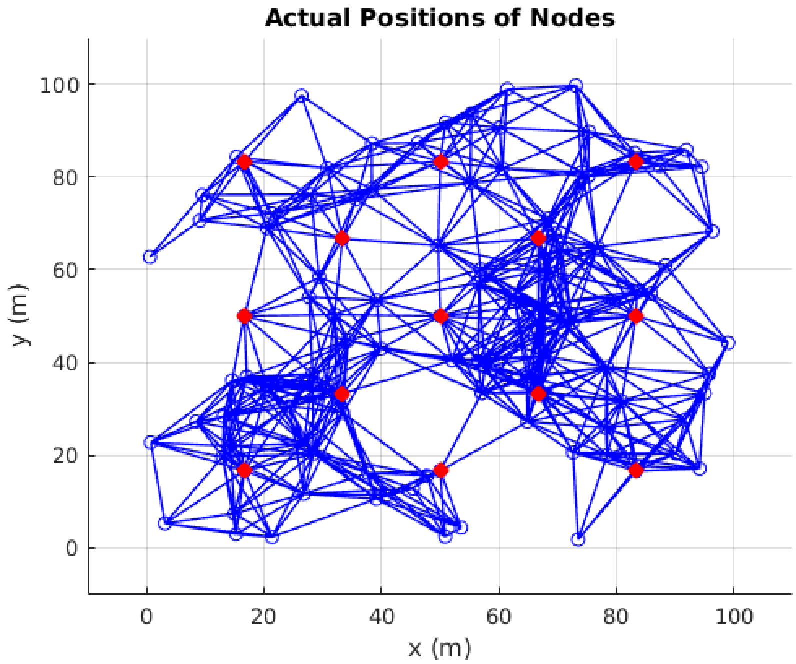


Figure 2: Position of measured nodes i.e. with noise. Red circles indicate the positions of 13 anchor nodes, blue circles the positions of the other 100 nodes and blue lines the edges of the graph. Communication radius is 20 m.



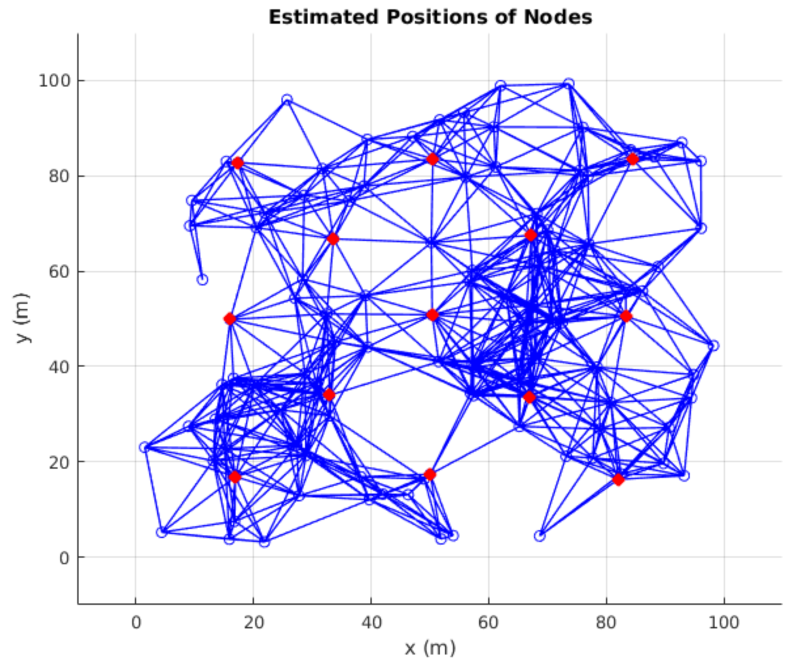


Figure 3: Comparison of actual node position and estimated node position using PLBP algorithm.

5 CONCLUSION

Posterior Linearizaation Belief propagation algorithm is used to infer the positions of the unknown nodes in a cooperative manner in a wireless sensor network with less computational complexity.

6 References

1 H. Wymeersch, J. Lien, and M. Win, “Cooperative localization in wireless networks,” Proc. IEEE, vol. 97, no. 2, pp. 427–450, Feb. 2009.

2 S. Kianoush, A. Vizziello, and P. Gamba, “Energy-efficient and mobile- aided cooperative localization in cognitive radio networks,” IEEE Trans. Veh. Technol., vol. 65, no. 5, pp. 3450–3461, May 2016.

3 T. V. Nguyen, Y. Jeong, H. Shin, and M. Z. Win, “Least square cooperative localization,” IEEE Trans. Veh. Technol., vol. 64, no. 4, pp. 1318–1330, Apr. 2015.

4 W. Yuan, N. Wu, B. Etzlinger, H. Wang, and J. Kuang, “Cooperative joint localization and clock synchronization based on Gaussian message passing in asynchronous wireless networks,” IEEE Trans. Veh. Technol., vol. 65, no. 9, pp. 7258–7273, Sep. 2016.

5 S. J. Julier and J. K. Uhlmann, “Unscented filtering and non- linear estimation,” in Proc. IEEE, vol. 92, no. 3, pp. 401–422, Mar. 2004.

6 F. Meyer, O. Hlinka, and F. Hlawatsch, “Sigma point belief prop- agation,” IEEE Signal Process. Lett., vol. 21, no. 2, pp. 145–149, Feb. 2014.

7 W. Sun and K.-C. Chang, “Unscented message passing for arbitrary con- tinuous variables in Bayesian networks,” in Proc. 22nd Nat. Conf. Artif. Intell., 2007, pp. 1902–1903.

8 S. Särkkä, Bayesian Filtering and Smoothing. Cambridge, MA, USA: Cambridge Univ. Press, 2013.

9 D. Bickson, “Gaussian belief propagation: Theory and application,” Ph.D. dissertation, The Hebrew Univ. Jerusalem, Jerusalem, Israel, 2008.

10 Q. Su and Y.-C. Wu, “On convergence conditions of Gaussian belief propagation,” IEEE Trans. Signal Process., vol. 63, no. 5, pp. 1144–1155, Mar. 2015.

11 P. Tichavsky, C. H. Muravchik, and A. Nehorai, “Posterior Cramér–Rao bounds for discrete-time nonlinear filtering,” IEEE Trans. Signal Process., vol. 46, no. 5, pp. 1386–1396, May 1998.

12 S. Li, M. Hedley, and I. B. Collings, “New efficient indoor cooperative localization algorithm with empirical ranging error model,” IEEE J. Sel. Areas Commun., vol. 33, no. 7, pp. 1407–1417, Jul. 2015.

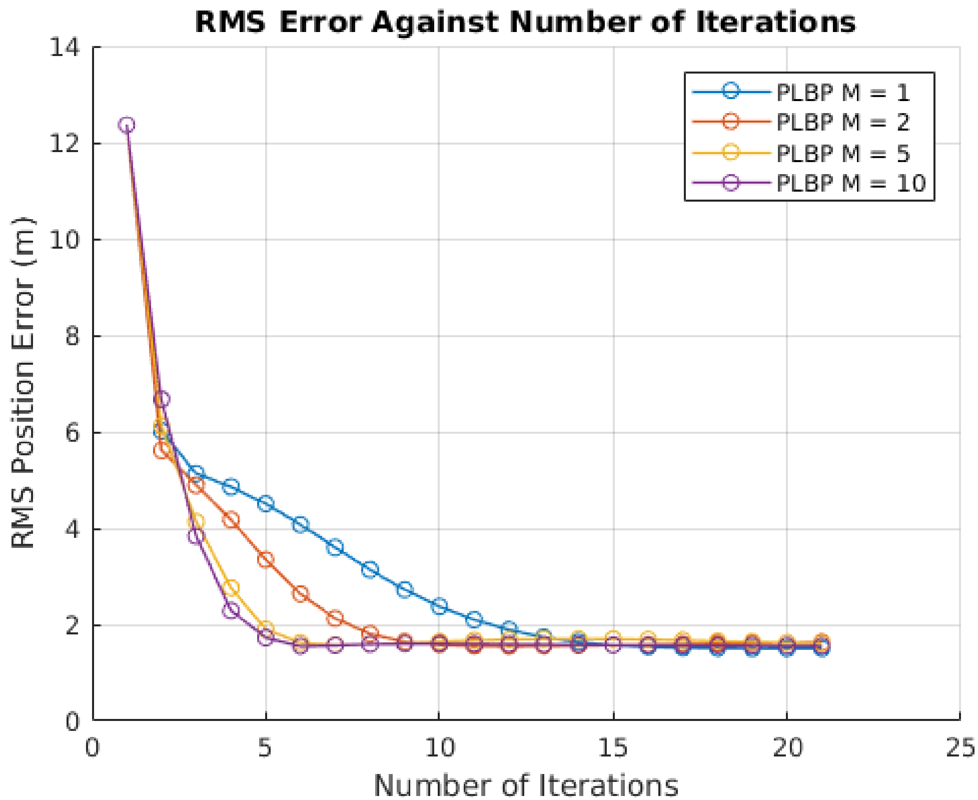
Appendix

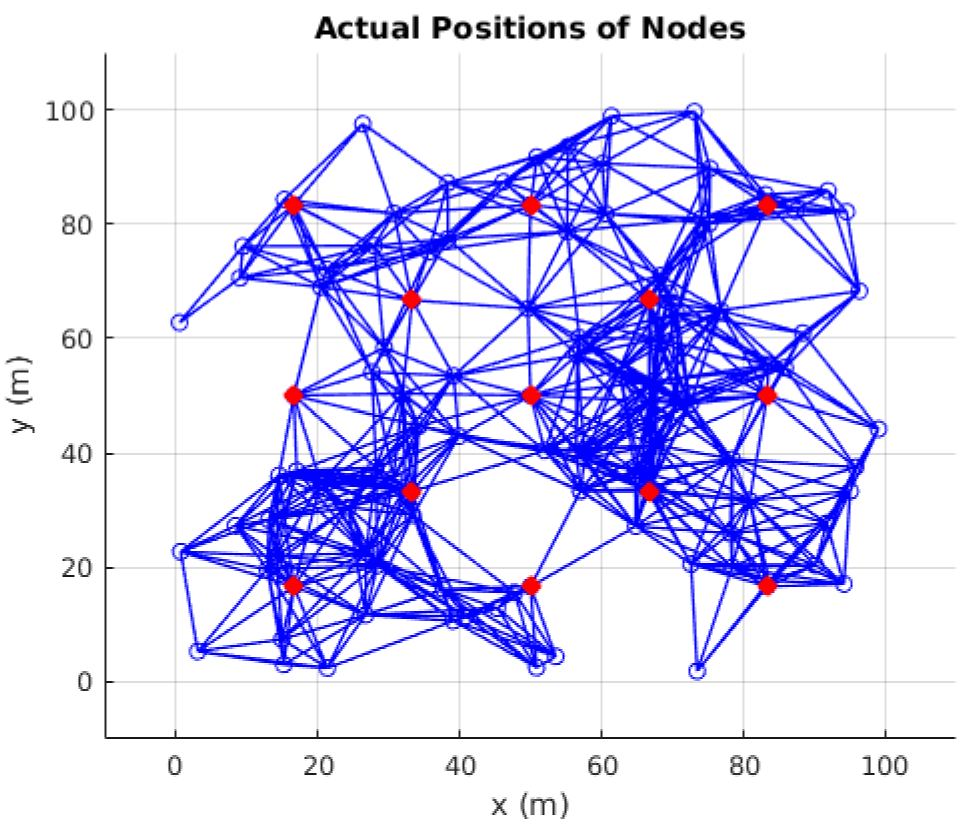
Main PLBP Algorithm

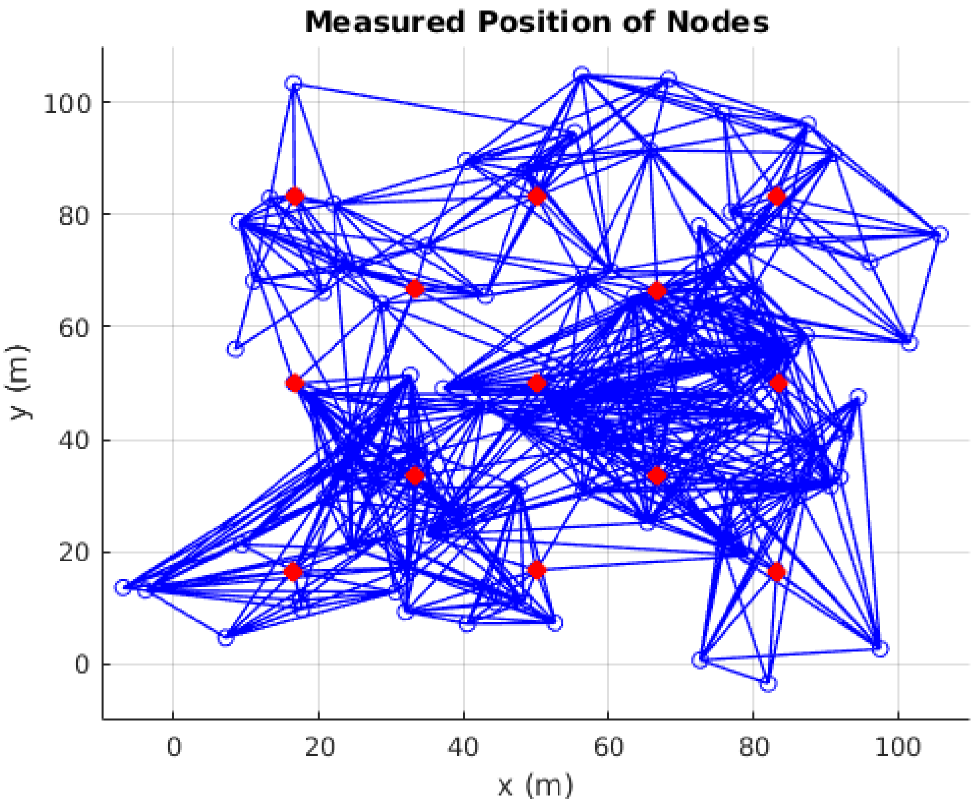
Node (1 to 100 - Normal Nodes) and (101 to 113 - Anchor Nodes) ------

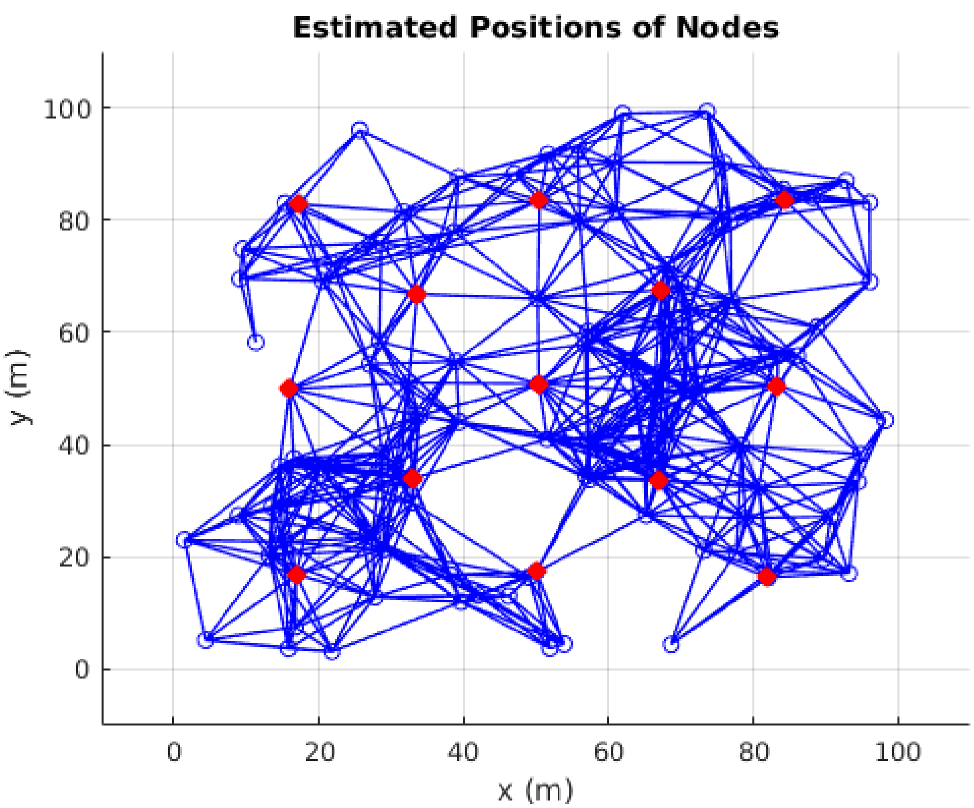
```
clear;

% Load the generated data --------
load data.mat

% Setting up variance for different measured data
 ---------------------
for i=1:100
  P(:,:,i) = 100.*eye(2);
end

for i=101:113
  P(:,:,i) = 0.01.*eye(2);
end

R = 1;
J = 20;
%----------------------------------------------------------------------

% Run four iterations for 1 PLBP, 2 PLBP, 5 PLBP and 10 PLBP, where M
 PLBP means M BP iterations.
for M=[1 2 5 10]
  u = x_observed;
  W = P;

  A(:,:,113,113) = zeros(1,4);
  b = zeros(113,113);
  sigma = zeros(113,113);
  Error = x_actual - u;
  RMSE = sqrt(sum(sum(Error.*Error))/113);

  % No of iteration for PLBP i.e. J times
 -------------------------------
  for k=1:J
    waitbar(k/20)

    % Run SLR Algorithm for i,j edges
 --------------------------------------
    for i=1:113
      for j=1:113
        if E(i,j)&&(i~=j)
          ul = transpose([u(i,:), u(j,:)]);
          Wl = [W(:,:,i),zeros(2,2);zeros(2,2),W(:,:,j)];
          [A(:,:,i,j), b(i,j), sigma(i,j)] = doSLR(ul, Wl);
        end
      end
```

```
    end
    %
 ----------------------------------------------------------------------

    % Run BP for M times for every nodes from 1 to 113
 ----------------------
    for m=1:M
      for r=1:113
        [u(r,:), W(:,:,r)] = doBP(A, b, sigma, u, W, r, E, h_observed,
 R);
      end
    end
    %
 ---------------------------------------------------------------------

    Error = x_actual - u;
    RMSE(:,k+1) = sqrt(sum(sum(Error.*Error))/113);
  end

  hold on;
  plot(1:21,RMSE(:,1:21),'o-', 'LineWidth', 1);
end


legend('PLBP M = 1', 'PLBP M = 2', 'PLBP M = 5', 'PLBP M = 10')

title('RMS Error Against Number of Iterations');
xlabel('Number of Iterations')
ylabel('RMS Position Error (m)')
grid on;

figure(2)
plotGraph(x_actual,E)
title('Actual Positions of Nodes');
xlim([-10 110])
ylim([-10 110])
xlabel('x (m)')
ylabel('y (m)')

figure(3)
plotGraph(x_observed,E)
title('Measured Position of Nodes');
xlim([-10 110])
ylim([-10 110])
xlabel('x (m)')
ylabel('y (m)')

figure(4)
plotGraph(u,E)
title('Estimated Positions of Nodes');
xlim([-10 110])
ylim([-10 110])
xlabel('x (m)')
ylabel('y (m)')
```


RMS Error Against Number of Iterations
PLBP M = 1
PLBP M = 2
PLBP M = 5
PLBP M = 10
RMS Position Error (m)
Number of Iterations


Actual Positions of Nodes
y (m)
x (m)


Measured Position of Nodes
y (m)
x (m)
0
20
40
60
80
100


Estimated Positions of Nodes
y (m)
x (m)
0
20
40
60
80
100

Published with MATLAB® R2017a

This function is to generate the required positions of nodes to setup the model.

Output: data.mat = matlab workspace file which will contain the node positions x_actual - Actual position of nodes x_observed - Measured position of nodes i.e. a gaussian noise is added to actual positions h_actual, h_observed - Message (Distance) between two nodes, actual and with noise(measure) respectively E - Matrix giving information about which node can communicate with each other ie. distance between them is within 20m E(i,j) = 0 for no communication between i and j nodes, E(i,j) = 1 for communication between i and j nodes.

```
%-------------------------------------------------------------------------------

clear;

Xa =
 [16.666,50,83.333,33.333,66.666,16.666,50,83.333,33.33,66.66,16.666,50,83.33];
Ya =
 [16.666,16.666,16.666,33.333,33.333,50,50,50,66.66,66.66,83.333,83.33,83.33];

x_actual = zeros(113, 2);
x_actual(1:100,:) = 100.*rand(100,2);
x_actual(101:113,:) = [transpose(Xa), transpose(Ya)];

x_observed = zeros(113, 2);
x_observed(1:100,:) = x_actual(1:100,:) + 10.0.*randn(100, 2);
x_observed(101:113,:) = x_actual(101:113,:) + 0.1.*randn(13, 2);

h_actual = zeros(113,113);
E = zeros(113,113);
for i = 1:113
  for j = 1:113
    h_actual(i,j) = norm(x_actual(i,:) - x_actual(j,:));
    if h_actual(i,j) <= 20.0
      E(i,j) = 1;
    end
  end
end

h_observed = h_actual + 1.*randn(113, 113);

save data.mat
```

Published with MATLAB® R2017a

SLR based on Sigma Points

Inputs: ul, Wl - mean and Covariance matrix for two nodes which is to be linearised Outputs: A, b, sigma - Linearisation Parameters. A = [A1 A2]

```
function [A, b, sigma] = doSLR(ul, Wl)
  N = 4;
  X = zeros(4,9);

  % Sigma Points and Corresponding Weight Generation
 --------------------------
  X(:,1) = ul;
  w1 = 1/3;
  wo = (1-w1)./(2.*N);
  T = chol(Wl);
  f = (N/(1-w1))^(1/2);

  % Approximating Linearisation based on the sigma points selected
 above -------
  for i=2:5
    X(:,i) = ul + f.*(T(i-1,:)');
    X(:,i+N) = ul - f.*(T(i-1,:)');
  end


  Z = sqrt((X(1,:) - X(3,:)).^2 + (X(2,:) - X(4,:)).^2);
  z = w1.*Z(:,1) + wo.*sum(Z(:,2:9));

  shi = w1.*(X(:,1) - ul).*(Z(:,1) - z);
  for j=2:9
    shi = shi + wo.*(X(:,j) - ul).*(Z(:,j) - z);
  end

  phi = w1.*(Z(:,1) - z).*(Z(:,1) - z);
  for j=2:9
    phi = phi + w1.*(Z(:,j) - z).*(Z(:,j) - z);
  end

  A = (shi')*(Wl^(-1));
  b = z - A*ul;
  sigma = phi - A*Wl*(A');

end
```

```
Not enough input arguments.

Error in doSLR (line 10)
  X(:,1) = ul;
```

Published with MATLAB® R2017a

Function to run belief propagation on a node 'k'

Inputs: A, b, sigma - Linearisation parameters obtained from SLR u, W - Old mean and variance of node 'k' E - matrix containing info about the edges which can communicate z - message(distance) matrix between two nodes R - Variation of measured message 'z' Outputs: ui, Wi - updated mean and variance for node 'k'

```
%---------------------------------------------------------------------------

function [ui, Wi] = doBP(A, b, sigma, u, W, k, E, z, R)

  % Kalman update for all neighbouring nodes.
  for p=1:113
    if (E(p,k)&&(p~=k))

      alpha = z(p,k) - A(:,1:2,p,k)*(transpose(u(p,:))) - b(p,k);
      H = A(:,3:4,p,k);
      T = R + sigma(p,k) +
 A(:,1:2,p,k)*W(:,:,k)*transpose(A(:,1:2,p,k));

      ze = H*(u(k,:)');
      S = H*W(:,:,k)*(H') + T;
      shi = W(:,:,k)*(H');
      a = u(k,:)' + shi*(S^(-1))*(alpha - ze);
      Ae = W(:,:,k) - shi*(S^(-1))*(shi');

      u(k,:) = a';
      W(:,:,k) = Ae;
    end
  end
  ui = u(k,:);
  Wi = W(:,:,k);
end
```

```
Not enough input arguments.

Error in doBP (line 14)
    if (E(p,k)&&(p~=k))
```

Published with MATLAB® R2017a

To plot the graphs with all nodes and vertices --------

Inputs: V - Nodes/ Vertex E - edges

```
function plotGraph(V, E)
  for i = 1:113
    for j = 1:113
      if E(i,j)
        hold on;
        plot([V(i,1) V(j,1)], [V(i,2) V(j,2)], 'o-b', 'LineWidth', 1);
      end
    end
  end
  plot(V(101:113,1), V(101:113,2), 'xr', 'LineWidth', 5);
  grid on
end
```

```
Not enough input arguments.

Error in plotGraph (line 8)
      if E(i,j)
```

Published with MATLAB® R2017a