Report and analysis on implementation of IMM algorithm for multiple-model dynamics tracking

Paiola Lorenzo 198573 Zanolli Erik 198852

January 2020

Introduction

The purpouse of this project is to analyze and evaluate the performance of an IMM algorithm's implementation in a distributed environment for multiple-model dynamics tracking. The tracked agent switches between linked models of movement by the means of a Markov chain. The goal is to evaluate the best trade-off between error on estimated position and real position and number of messages regarding consensus involved in the tracking.

Setting

The general objective of this report is to provide a robust tracking that works in a distributed manner for some object that can move in a number of different ways (modelized as a set of dynamical systems $\mathbb{M}$). The environment in which this object moves is one of a large room that contains multiple sensors that can talk to each other.

Sensor's model

The sensors chosen are radars measuring the polar cordinates relative to themselves at which the agent is collocated at the timestep, and are disposed in a uniform square grid. In order to simulate the real workings of a sensor, range of measuremnt has been limited to the distance between one sensor and the following one in any direction of the grid, as soon as the agent exceed the imposed maxiumum distance from the sensor, the device will stop sensing. This property of the sensor grid, coupled by its geometry, ensures that no more than 4 sensors can measure the agent position at any time, so it made sense to let the sensors switch between 3 different states named ON, OFF and IDLE. This can be justified as a way to make the system more power efficient and to avoid useless data stream towards sensors that aren't currently in range and sensing. The sensor is modeled as a state machine as shown in figure 1

Sensors in different states differ between each other by the actions that are allowed in the state they are currently in.
Here we enumerate such allowed actions:

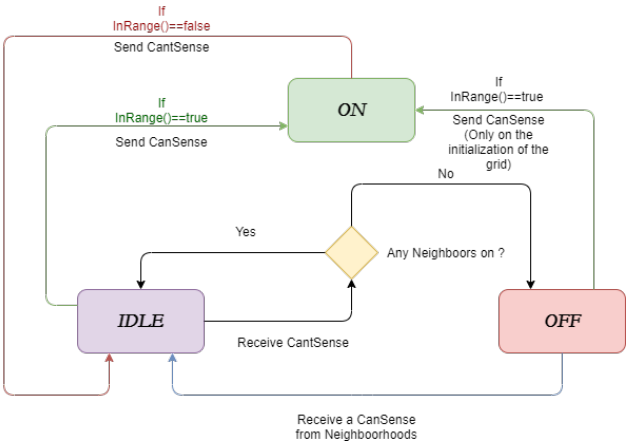

Figure 1: State machine sensor

- ON: at each time-step checks if the agents is still in range with the function inRange(), takes a measurement, computes the IMM algorithm and at a tunable rate computes the consensus with the other nearby ON sensors. If the agent is not in range anymore then turns IDLE.
- IDLE : checks if the agent is now in range with the function inRange(), if it is then initialize itself with the data from nearby sensors that are ON. If it recieves a CantSense signal, it checks if at least one of its neighboors is still ON, else it turns OFF itself.
- OFF : does nothing but waits for a CanSense signal sent by a neighboor and turns IDLE in the case it has received one (this means that a nearby sensor has turned ON).

So sensors can communicate with the devices adjacent to them (Neighboorhood), as represented in the figure 2 below, and exchange with them signals named CanSense and CantSense which state respectively whenever the agent gets in or goes out of the communicating sensor's range. This check is done through the function inRange() that also serves as a switch between the states of the sensor, as already shown by the figure 1 above. The messages that the sensors exchange with their Neighboorhood are

- CanSense : message sent by a sensor switching from IDLE to ON triggered by a positive result from InRange() function

- CantSense : message sent by a sensor switching from ON to IDLE triggered by a negative result from InRange() function

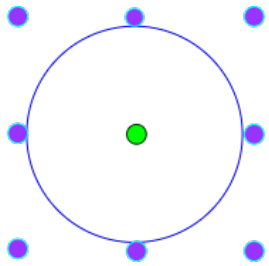

Figure 2: The neighboors of a sensors are the one immediately adiacent, represented as purple in this image

By defing the range of the sensors equal to their spacing on the grid it follows that only a maximum of 4 sensors can be turned on. This fact is shown in the following picture that illustrates the different cases

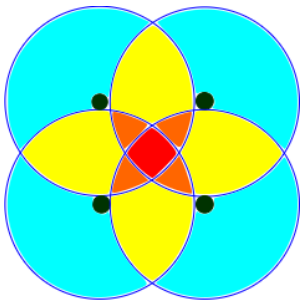

Figure 3

- blue : 1 sensors ON
- yellow : 2 sensors ON
- orange : 3 sensors ON
- red : 4 sensors ON

Note that the blue case happens only at the borders and corners of the grid.
Whenever the target is in range of a sensor at given timestep, the device takes a measurement. The model chosen for our sensor is that of a radar, and as such the non-linear measurement function associated to it is the cartesian to polar transformation at timestep k, as shown below

$$z(k) = h(x(k)) + v(k)$$
$$\begin{bmatrix} \rho(k) \\ \theta(k)(k) \end{bmatrix} = \begin{bmatrix} \sqrt{x_1(k)^2 + x_2(k)^2} \\ atan2(x_2(1)/x_1(k)) \end{bmatrix} + \begin{bmatrix} v_1(k) \\ v_2(k) \end{bmatrix}$$

where $h(x)$ is the polar to cartesian cordinates transform that takes in $x(k)$ that is the state (of which the first 2 elements are the cartesian cordinates relative to the sensor), $z(k)$ is the measure and $v(k)$ is the noise associated to the measurement's operation. The measurement noise $v(k)$ is associated to a diagonal power spectral density matrix R.

$$R = \begin{bmatrix} \sigma_\rho^2 & 0 \\ 0 & \sigma_{\theta(k)}^2 \end{bmatrix}$$

The matrix $H^k = (\nabla_x h(x))_{|x=x(k)}$ used in the linearized model with $v(k) = 0$, necessary in the IMM filter at each timestep k, computes to

$$H^k = \begin{bmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_2}{\sqrt{x_1^2+x_2^2}} & 0\dots \\ \frac{-x_2}{x_1^2+x_2^2} & \frac{x_1}{x_1^2+x_2^2} & 0\dots \end{bmatrix}_{|x=x(k)}$$

Where $0\dots$ is a vector of zeroes for all the other states that do not influence the measurement function.

Model used

As mentioned already, the agent we want to track has a variable-model dynamic that can switch between different elements in a set of models that we denote as $\mathbb{M}$. We will consider 2 families of such sets that we will call: Random Accelerated Walk $\mathbb{M}_1$ and Random Unicycle Turning $\mathbb{M}_2$. Every element of said families is a discrete Markovian process, as all are influenced by noise on their inputs.

Random Accelerated Walk

For the Random Accelerated Walk we consider the set $\mathbb{M}_1$ to be 5 elements large. Here the elements are described

Random Walk	
Mode	Behaviour
Mode 1	Constant Speed
Mode 2	Positive Acceleration in x
Mode 3	Negative Acceleration in x
Mode 4	Positive Acceleration in y
Mode 5	Negative Acceleration in y

Every member of the set has the same structure to describe their dynamics, in state space this their shared linear model

$$x(k+1) = Ax(k) + B(s_k)u + Gw(k)$$

where A is the state matrix, $B(s_k)$ the input matrix function of the mode/index s_k that indicates the element picked of $\mathbb{M}_1$ at timestep k, u is the input that stays constant at all timesteps k and in all modes s_k, G the noise matrix and $w(k)$ the process noise at timestep k with associated Q_1 diagonal power spectral density matrix.

$$Q_1 = \begin{bmatrix} \sigma_{a_x}^2 & 0 \\ 0 & \sigma_{a_y}^2 \end{bmatrix}$$

Below we show the constant matrices and how the state is structured

$$x = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} A = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} G = \begin{bmatrix} \frac{\delta^2}{2} & 0 \\ 0 & \frac{\delta^2}{2} \\ \delta & 0 \\ 0 & \delta \end{bmatrix}$$

Where x and y are the cartesian coordinates in the absolute reference frame and δ is the duration of each timestep k. To model different behaviours of the agent/elements in the set $\mathbb{M}_1$ we use switching matrices $B(s_k)$ while keeping the vector u constant. The input u here is a vector of acceleration in x and y that the noise $w(k)$ will influence. We hypothesize that the input is always known in magnitude and constant. Here is the set $B(s_k)$ of input matrices.

$$B(s_k) = \left\{ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\delta^2}{2} & 0 \\ 0 & 0 \\ \delta & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -\frac{\delta^2}{2} & 0 \\ 0 & 0 \\ -\delta & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & \frac{\delta^2}{2} \\ 0 & 0 \\ 0 & \delta \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & -\frac{\delta^2}{2} \\ 0 & 0 \\ 0 & -\delta \end{bmatrix} \right\}$$

It is to note that the matrix G is a merge between the second and the fourth matrices found in the set $B(s_k)$ as we hyphotesized that the noise will act randomly in direction and magnitude at every time-step.
The jump between behaviours is modeled by a Markov chain that has a constant stochastic transition matrix T. In the simulations the matrix is fixed at

$$T = \begin{bmatrix} p_0 & p & p & p & p \\ b & q_0 & 2q & q & 2q \\ b & 2q & q_0 & 2q & q \\ b & q & 2q & q_0 & 2q \\ b & 2q & q & 2q & q_0 \end{bmatrix}$$

Where p_0, q_0 and b are fixed while p and q are computed so that the matrix is stochastic.

Random Unicycle Turning

In the case of the Random Unicycle Turning we also consider the set $\mathbb{M}_2$ to be 5 elements large, but the states are changed.

Unycicle	
Mode	Behaviour
Mode 1	Constant Tangential Velocity and Angle
Mode 2	Positive Angular Wheel Acceleration
Mode 3	Negative Angular Wheel Acceleration
Mode 4	Positive Yaw Rate
Mode 5	Negative Yaw Rate

Here all the elements in the set $\mathbb{M}_2$ are non-linear models $x(k+1) = f(x(k), u(k), w(k), s_k)$, but if we can still organize them in such a way

$$x(k+1) = A(x(k))x(k) + B(s_k, x(k))u(k) + G(x(k))w(k)$$

By defining

$$x = \begin{bmatrix} x(k) \\ y(k) \\ v_t(k) \\ \alpha(k) \end{bmatrix} A(x(k)) = \begin{bmatrix} 1 & 0 & \delta cos(\alpha(k)) & 0 \\ 0 & 1 & 0 & \delta sin(\alpha(k)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} G(x(k)) = \begin{bmatrix} \frac{\delta^2}{2} cos(\alpha(k))r & 0 \\ \frac{\delta^2}{2} sin(\alpha(k))r & 0 \\ \delta r & 0 \\ 0 & \delta \end{bmatrix}$$

and

$$B(s_k, k) = \left\{ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\delta^2}{2} cos(\alpha(k))r & 0 \\ \frac{\delta^2}{2} sin(\alpha(k))r & 0 \\ \delta r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -\frac{\delta^2}{2} cos(\alpha(k))r & 0 \\ -\frac{\delta^2}{2} sin(\alpha(k))r & 0 \\ -\delta r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \delta \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -\delta \end{bmatrix} \right\}$$

where δ is the length of the timestep k, r the radius of the unicycle's wheel, $x(k)$ and $y(k)$ absolute cartesian cordinates of the unicycle, $v_t(k)$ tangential velocity and α yaw angle. The input u is a vector of the angular acceleration of the wheel ω and yaw rate Ω with

$$Q_2 = \begin{bmatrix} \sigma_\omega^2 & 0 \\ 0 & \sigma_\Omega^2 \end{bmatrix}$$

Just as in $\mathbb{M}_1$, G is build on the hypothesis that the process noise $w(k)$ influences both inputs at any timestep k.
The stochastic transition matrix in this case will be

$$T = \begin{bmatrix} p_0 & p & p & p & p \\ q & q_0 & q & 0 & 0 \\ q & q & q_0 & 0 & 0 \\ q & 0 & 0 & q_0 & q \\ q & 0 & 0 & q & q_0 \end{bmatrix}$$

where p_0 and q_0 are fixed while p and q are computed to make the matrix stochastic.

IMM

Being the agent's dynamic costantly switching between the models in the set chosen, a single Kalman filter won't give us acceptable results while tracking and we have to rely on the Interacting Multiple Models algorithm (IMM). In essence the algorithm let various estimation filters work in parallel, and outputs a mixture of the filters' estimate based on the probability of each of the models used in the filter to be correct.
The hyphotheses we have to make in order to use the IMM are that the sequence of the true mode at timestep k, s_k, is a Markovian process, i.e. the mode evolves to the memoryless random process $P\{s_{k+1}^{(j)} | s_k^{(j)}\} \pi_{ji} \ \forall \ j, i$ and that the model set $\mathbb{M}$ is the correct one and is time-invariant. In our case those hypotheses are known to be true.
The design parameters of the IMM are the model set $\mathbb{M}$, the process and measurement noise spectral density matrices Q and R for each model and sensor, the initial state and covariance $\hat{x}_{0|0}$ and $P_{0|0}$ and the transition matrix T containing all the π_{ji}.
Following is a short recap of the algorithm, to see it in much more detail refer to CITA QUA
The algorithm can be subdivided in 4 subsections, :

- Model Conditioned reinitialization $\forall m_i, m_j \in \mathbb{M}$

$$\mu_{k|k-1}^{(i)} = \sum_j \pi_{ji} \mu_{k-1}^{(j)}$$
$$\mu_{k-1}^{(j|i)} = \pi_{ji} \mu_{k-1}^{(j)} / \mu_{k|k-1}^{(i)}$$
$$\bar{x}_{k-1|k-1}^{(i)} = \sum_j \hat{x}_{k-1|k-1}^{(j)} \mu_{k-1}^{j|i}$$
$$\bar{P}_{k-1|k-1}^{(i)} = \sum_j [P_{k-1|k-1}^{(j)} + (\bar{x}_{k-1|k-1}^{(i)} - \hat{x}_{k-1|k-1}^{(j)})(\bar{x}_{k-1|k-1}^{(i)} - \hat{x}_{k-1|k-1}^{(j)})^T] \mu_{k-1}^{j|i}$$

Here we take all the output model probabilities computed at the previous iteration of the IMM and mix them together with the transition matrix to then use the result to combine both the previous step state and covariance estimation. It is to notice that the transpose of the matrix formed by the collection of $\mu_{k-1}^{(j|i)}$ is stochastic, this makes sense since it works as a normalized transition matrix while computing $\bar{P}_{k-1|k-1}^{(i)}$ and $\bar{x}_{k-1|k-1}^{(i)}$.

- Model Conditioned filtering $\forall m_i \in \mathbb{M}$

$$\hat{x}_{k|k-1}^{(i)} = f_k^{(i)}(\bar{x}_{k-1|k-1}^{(i)}, u)$$
$$P_{k|k-1}^{(i)} = F_{k-1}^{(i)} \bar{P}_{k-1|k-1}^{(i)} F_{k-1}^{(i)^T} + G_{k-1}^{(i)} Q_{k-1}^{(i)} G_{k-1}^{(i)^T}$$
$$\bar{z}_k^{(i)} = z_k - h(\hat{x}_{k|k-1}^{(i)})$$
$$S_k^{(i)} = H_k^{(i)} P_{k|k-1}^{(i)} H_k^{(i)^T} + R_k^{(i)}$$
$$W_k^{(i)} = P_{k|k-1}^{(i)} H_k^{(i)^T} S_k^{(i)^{-1}}$$
$$\hat{x}_{k|k}^{(i)} = \hat{x}_{k|k-1}^{(i)} + W_k^{(i)} \bar{z}_k^{(i)}$$
$$P_{k|k}^{(i)} = P_{k|k-1}^{(i)} - W_k^{(i)} S_k^{(i)} W_k^{(i)^T}$$

This part of the algorithm is the one that includes all the estimation filters running in parallel. In the case of $\mathbb{M}_1$ we use a Kalman filter with a non-linear $h(x)$, so $f_k^{(i)}(x,u)$ is just the linear model (i) inside $\mathbb{M}_1$ while $F_{k-1}^{(i)}$ is A and $G_{k-1}^{(i)}$ is G specified in the model. In $\mathbb{M}_2$ case an Extended Kalman Filter is used since $f_k^{(i)}(x,u)$ is non-linear for every model (i), so here $F_{k-1}^{(i)}$ is the Jacobian of the non-linear function without noise evaluated in $\bar{x}_{k-1|k-1}^{(i)},\ u$ while $G_{k-1}^{(i)}$ is $\nabla_w f_k^{(i)}(x,u)$ evaluated in the same point with noise= 0. Both power spectral density matrices Q and R are constant by assumption. The matrix $H_k^{(i)}$ will be model (i) agnostic, computed as already discussed in section and evaluated in $\hat{x}_{k|k-1}^{(i)}$ (each different sensor will calculate its relative position from that coordinate).

- Model Probability update $\forall m_i, m_j \in \mathbb{M}$

$$L_k^{(i)} = \mathcal{N}(\bar{z}_k^{(i)}, 0, S_k^{(i)})$$
$$\mu_k^{(i)} = \frac{\mu_{k|k-1}^{(i)} L_k^{(i)}}{\sum_j \mu_{k|k-1}^{(j)} L_k^{(j)}}$$

Here we compute what is the actual probability of the model $m_i \in \mathbb{M}$ to have guided the dynamic of our agent at timestep k. To do so we assume that the the probability distribution of the error $\bar{z}_k$ should be a Gaussian with 0 mean and as covariance the innovation covariance $S_k^{(i)}$ given by the filter (i) at timestep k, and we compute the $L_k^{(i)}$ probability density function in $\bar{z}_k^{(i)}$. Using this result we re-normalize the previously computed $\mu_{k|k-1}^{(i)}$ to give us the output vector of model probabilities μ_k.
In this step two adjustment are needed: the first one is the choice to use a logaritmic pdf instead of a normal one due to the fact that it gives better numerical results and lower bound on the likelihood is implemented since a likelihood vector of 0 breaks the algorithm.

- Estimate fusion

$$\hat{x}_{k|k} = \sum_i x_{k|k}^{(i)} \mu_k^{(i)}$$
$$P_{k|k} = \sum_i [P_{k|k}^{(i)} + (\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)})(\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)})^T] \mu_k^i$$

Using the vector μ_k computed at the previous step we mix all of the predictions computed by the filters to get the outputs. This last step is only for the sake of the output, the algorithm forms a loop without it as the starting step of the next iteration uses the outputs of the filters at the steps before this one.

Consensus

Right after having computed the IMM's estimate for every ON sensor, we use a consensus algorithm to try to lower the estimation error even more. The consensus will happen at a variable rate and only between the ON sensors. The set of active sensors inside the grid is dynamic, as vertices keep getting added and popped. However no more than 4 sensors will be ON at any point of time, see 3, and they will all be adjacent to each other, this let us model the graph of the active sensors as fully-connected, since the distance between them is short and we imagine them to be directly linked.
This modeling let us use a Weighted Least Squared estimation of the best measure between all the ON sensors, since this operation gives us a consensus in a single timestep, which is what we need with both a dynamic graph and a dynamic system. So for our system we get

$$P_{cons}(k) = (H^{k^T} P^{k^{-1}} H^k)^{-1}$$
$$x_{cons}(k) = P_{cons} H^{k^T} P^{k^{-1}} Z^k$$

where P^k is the block diagonal matrix built with all the covariance matrices from the IMM's output, Z^k is the measurement vector built by stacking all the state predictions from the IMM's output and H^k is the matrix build by stacking all the H_i for $z_i = H_i x + \epsilon_i$, luckily in our case all H_i are identity matrices.
It has to be noted that the WLS can be seen as 2 parallel linear consensus algorithms, running one on top of the other. The consensus is computed both on the output of the IMM filter and all the outputs of its internal Kalman filters. The same WLS algorithm is used to initialize IDLE sensors that turn ON using data from nearby ON sensors.
We also tentatively tried to use the same consensus algorithm on the prediction vector of model probabilities μ_k using the collection of all $\mu_{k-1}^{j|i}$ in matrix form

instead of the covariance matrix for the weighting matrix, but the resulting matrix was too close to singularity and the simulation was having numerical problems so this road was abandoned, instead we just computed the mean between the various μ_k.

Results

Here we present the results of our implementation of the algorithm and sensors' grid. The agent was left wandering inside the set virtual room and at each timestep the estimation error of the algorithm was measured. The room is set to be a square with a $100m$ side with a 10×10 grid of sensors with range$= 10m$.
The simulations were performed in a loop according to the Montecarlo's Method. We set a number of iterations for any specific case to $n_{monte} = 100$ and each iteration stop wheter the agent leaves the room limits or a number of timesteps $n_k = 1000$ is reached. The starting position and starting model are randomized at each iteration. The performance indexes that we evaluated are:

- The Root Mean Square of the means of the errors at each Montecarlo's iteration
- The Root Mean Square of the maximum error at every Montecarlo's iteration
- The Maximum error between all Montecarlo's iterations

We evaluated both model sets $\mathbb{M}_1$ and $\mathbb{M}_2$ for different consensus rate (how many timestep between each consensus), different R and Q and we considered both the errors given by the Consensus and by each individual sensor's IMM. We've also run simulations with noconsensus at all.
The timestep δ is set at $0.05s$, the radius of the wheel r at $0.5m$ while the inputs are

$$u_1 = \begin{bmatrix} 5\frac{m}{s^2} \\ 5\frac{m}{s^2} \end{bmatrix} \qquad u_2 = \begin{bmatrix} 3\frac{rad}{s^2} \\ 6\frac{rad}{s} \end{bmatrix}$$

The power spectral density matrices considered are $R1 = diag(0.1, (\frac{2\pi}{180})^2)$ and $R2 = diag(0.01, (\frac{2\pi}{360})^2)$ for all the sensors, and while $Q2 = diag(1, 1)$ is shared by $\mathbb{M}_1$ and $\mathbb{M}_2$, $Q1 = diag(0.2, 0.2)$ for $\mathbb{M}_2$ and $Q1 = diag(0.1, 0.1)$ for $\mathbb{M}_1$. The Transition matrices were initializated as follows

$$T_1 = \begin{bmatrix} 0.6 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.2 & 0.5 & 0.12 & 0.06 & 0.06 \\ 0.2 & 0.12 & 0.5 & 0.12 & 0.06 \\ 0.2 & 0.06 & 0.12 & 0.5 & 0.12 \\ 0.2 & 0.12 & 0.06 & 0.12 & 0.5 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} 0.8 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.25 & 0.5 & 0.25 & 0 & 0 \\ 0.25 & 0.25 & 0.5 & 0 & 0 \\ 0.25 & 0 & 0 & 0.5 & 0.25 \\ 0.25 & 0 & 0 & 0.25 & 0.5 \end{bmatrix}$$

The first case is $\mathbb{M}_1$, the second $\mathbb{M}_2$. Here are reported all the results of the simulations, all in meters.

Sensor			
Measure	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.3985	1.2407	1.7790
R2,Q1	0.1574	0.5454	0.7139

This table is needed to compare the simulations results, because the power spectral density matrices R regard the polar coordinates, while here we have the errors in cartesian. If the algorithm gives an error greater in all cases than what is given by the sensors, then it is useless.

case 1: consensus rate = 20			
Consensus	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1078	0.2578	0.3960
R1,Q2	0.1094	0.2694	0.4671
R2,Q1	0.0505	0.1210	0.2491
R2,Q2	0.0514	0.1313	0.2549
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1609	0.1762	2.0937
R1,Q2	0.1625	0.1907	1.0347
R2,Q1	0.0799	0.0877	0.4310
R2,Q2	0.0815	0.0926	0.4445

case 1: consensus rate = 10			
Consensus	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1110	0.2831	0.4460
R1,Q2	0.1129	0.2940	0.5227
R2,Q1	0.0516	0.1406	0.2226
R2,Q2	0.0519	0.1331	0.2429
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1455	0.1678	0.9
R1,Q2	0.1478	0.1926	1.5835
R2,Q1	0.0751	0.0853	0.4441
R2,Q2	0.0756	0.0824	0.4522

case 1: consensus rate = 5			
Consensus	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1158	0.3516	1.2838
R1,Q2	0.1192	0.3245	0.5018
R2,Q1	0.0553	0.1649	0.3532
R2,Q2	0.0542	0.1649	0.2246
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1302	0.2460	1.808
R1,Q2	0.1359	0.1700	1.7787
R2,Q1	0.0666	0.0762	0.762
R2,Q2	0.0661	0.0738	0.4450

case 1: consensus rate = 2			
Consensus	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1158	0.3516	1.2838
R2,Q1	0.0598	0.1925	0.2654
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1320	0.2460	1.808
R2,Q1	0.0617	0.0741	0.4368

case 1: consensus rate = 1			
Consensus	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.4265	1.6287	4.5672
R2,Q1	0.0757	0.204	0.2612
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.3755	0.8706	5.9070
R2,Q1	0.0766	0.1128	0.4872

case 1: consensus rate = 2			
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1790	0.2278	1.7442
R1,Q2	0.1800	0.2782	6.4459
R2,Q1	0.3961	0.0930	0.0850
R2,Q2	0.0864	0.0973	0.7242

case 2: consensus rate = 20			
Consensus	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1167	0.3027	0.7750
R1,Q2	0.1132	0.2847	0.6577
R2,Q1	0.0512	0.1325	0.2315
R2,Q2	0.0539	0.1454	0.2334
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.0771	0.0903	0.4918
R1,Q2	0.1664	0.2726	1.0990
R2,Q1	0.0771	0.0903	0.4918
R2,Q2	0.0804	0.0947	0.4190

case 2: consensus rate = 10			
Consensus	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1228	0.3438	0.5992
R1,Q2	0.1214	0.3291	0.7441
R2,Q1	0.0535	0.1536	0.2525
R2,Q2	0.0539	0.1439	0.2396
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1566	0.1938	0.8766
R1,Q2	0.1577	0.2278	1.4168
R2,Q1	0.0742	0.0906	0.3968
R2,Q2	0.0759	0.0935	0.4782

case 2: consensus rate = 5			
Consensus	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1984	0.7467	2.3522
R1,Q2	0.1925	0.5871	2.2983
R2,Q1	0.0604	0.1829	0.4389
R2,Q2	0.0628	0.1873	0.2912
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.2135	0.7295	2.4394
R1,Q2	0.2116	0.5412	2.7371
R2,Q1	0.0722	0.1012	0.4599
R2,Q2	0.0742	0.1077	0.5699

case 2: consensus rate = 2			
Consensus	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.6709	3.2770	8.4954
R2,Q1	0.1392	0.6907	2.0562
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.07629	2.5799	9.3296
R2,Q1	0.1585	0.5868	2.1211

case 2: consensus rate = 1			
Consensus	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	1.3654	5.5341	16.2066
R2,Q1	0.3883	2.3273	5.1490
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	1.4520	5.6335	16.4740
R2,Q1	0.3867	1.3242	5.3175

case 2: no consensus			
Individual IMM	RMS of Means	RMS of Maxes	Max of Maxes
R1,Q1	0.1784	0.2516	1.5541
R1,Q2	0.1795	0.2145	0.9872
R2,Q1	0.0845	0.0962	0.6888
R2,Q2	0.0843	0.1090	0.9289

What we gather from these tables is that, specifically in the case of the unicycle $\mathbb{M}_2$ (that is a more complex model set), a consensus computed at a high frequency (rate=1, rate=2) gives disastrous results with any magnitude of noise. This is probably caused by the nature of the IMM algorithm that gets impeded by the consensus which homogenizes the internal Kalman filters input to a point which the IMM can't compute their probabilities efficiently. In fact we see a much better results with lower frequency such as rate=10 and rate=20. We also tried the case in which we do not compute any consensus and we let individual sensors work alone and, while we see good results, it doesn't seem to be the best compromise.

In the case of the Random Walk with a rate=2 we have good results with low R, but the algorithm is really sensible to changes to this parameter and performs really bad with $R1$.

While we see that with both model sets the case with rate=20 gives very good results, but we also notice that the maximum of both the individual IMMs and of the consensus can get pretty high so if we have an application where the frequency of estimation must be high and we can't have a big overshoot, then this rate can fail. All things considered the best trade-off may be the rate=10, which always gives good results and has double the frequency of rate=20.

Following are some graphs of the best and worst cases computed by the simulations. More graphs and a video of the algorithm working can be found at ...