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 as 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:

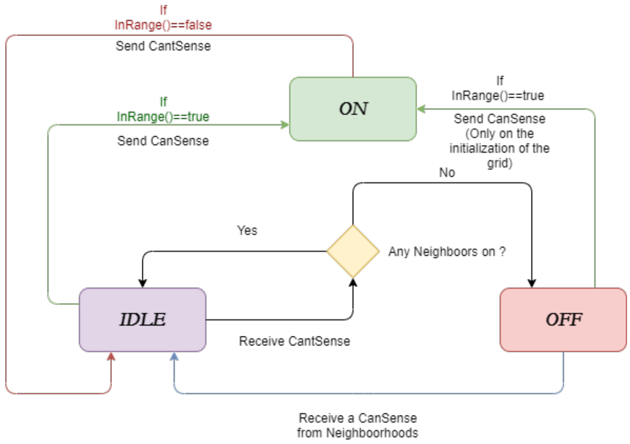

Figure 1: State machine sensor

- ON: at each time-step checks if it 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 it's not in range anymore then turns IDLE.
- IDLE : checks if it 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

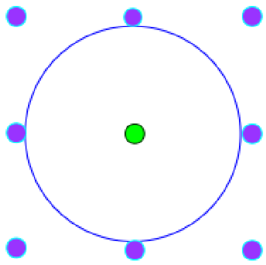

Figure 2: The neighboors of a sensors are the one immediately adiacent, represented as purple in this image

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

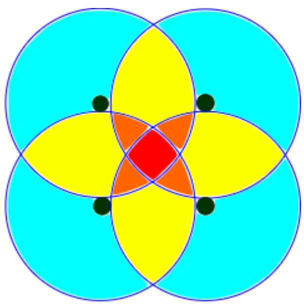

Figure 3

- blue : 1 sensors ON, happens only at the conrners of the grid
- yellow : 2 sensors ON
- orange : 3 sensors ON
- red : 4 sensors ON

Whenever the target is in range of a sensor at given timestep, the device takes a measurement. The model choosen for our sensor is 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 $w(k)$ 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

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

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 Accellerated 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 a white noise on their inputs.

Random Accellerated Walk

For the Random Accellerated 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

$$x(k+1) = Ax(k) + B(s)u + Gw(k) \tag{1}$$

where A is the state matrix, $B(s)$ the input matrix function of the mode/index s that indicates the element picked of $\mathbb{M}_1$, u is the input that stays constant at all timesteps k and in all modes s, 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 $B(s)$ 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 $B(s)$ of input matrices.

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

It is to note that the matrix G is a merge between the second and the fourth matrices found in the set $B(s)$ 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 $x(k+1) = f(x(k), u(k), w(k), s)$, but if we can still organize them in such a way

$$x(k+1) = A(x(k))x(k) + B(s, x(k))u(k) + G(x(k))w(k) \tag{2}$$

By defining

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

and

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

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

In order to reach a more accurate prediction of the trajectory a IMM algorithm is implemented in the sensors's grid. The working principles involves a general knowledge or hypothesis of the model of agent's movement. Every model is used in a Kalman filter stage that use the same measurement and make a prediction: the one with the smaller uncertainty is choose as the model for our agent at that timestep. A general scheme of the working principles is shown here (image?)

Consensus

The set of active sensors inside the grid is dynamic, as vertices keep getting added and popped. However it has to be noticed, as stated before, that no more than 4 sensors will be ON at any point of time 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 they are directly linked.

Result

The performance evaluation of the system are evaluted by computing the RMS of the predicted trajectory eith the respect to the actual one choosing different rate of performind the WSL. The final result are listed in the table below

1 step	2 step	3 step	4 step
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344