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 way, 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 relative polar cordinates at which the agent is collocated at the timestep, and are disposed as a uniform 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 maximum 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 the time, so it made sense to let sensor 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.

The functions and operations that the sensors can do are:

- InRange() : function that verify the presence of the target in its working area

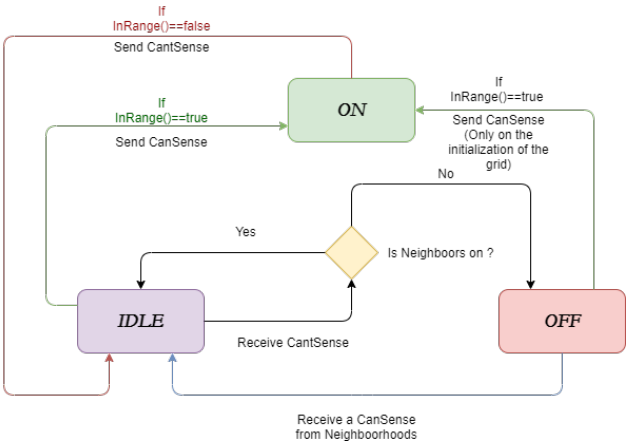

Figure 1: State machine sensor

- CanSense : messagge 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

Every sensor accordingly with its state can perform some action and send information following the rules listed below:

- On: participate in the measurement process while InRange() stay with positive result. It retrieve data on position and perform a IMM algorithm and with a selected rate perform a linear consensus calculation
- Idle : perform a InRange() verification each timestep to verify if it have to switch in On. If triggered to On initialization starts using data from the others sensors as a starting point. if it receive a CantSense from all the neighboors it turn itself off.
- Off : no measurment function or involvement. Triggered to Idle when receive one CanSense by the neighboors

So sensors can communicate with the devices adjacent to them (Neighboorhood), as represented in the figure 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 above.

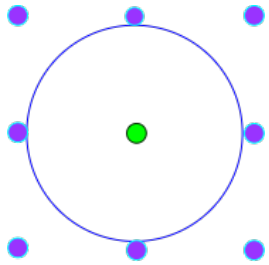

Figure 2: Neighboors for a sensors are the one immediately adiacent. The purples ones represent the neighboors of green sensor

Defing the range of the sensors equal to their spacing on the grid it follows, depending on agent's position, only a maximum of 4 sensors can be turned on. It follows a figure that show the idea with a 4-sensors grid

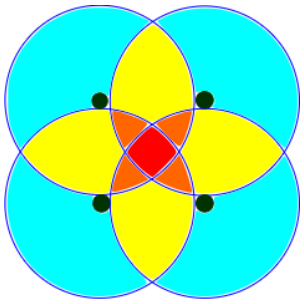

Figure 3

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

Whenever the target is in range of a sensor, the device takes a measurement at each timestep. The model we choose for our measurement device is a radar and as such the measurement function $z(k) = h(x(k)) + v(k)$ associated with it is the cartesian to polar transformation, as shown below

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

where $h(x)$ is the polar to cartesian cordinates transform and $v(k)$ is the noise associated to the measurement's operation.

$$\begin{bmatrix}\rho\\ \alpha\end{bmatrix} = \begin{bmatrix}x^2 + y^2\\ atan2(y/x)\end{bmatrix}$$

for sake of use in the extended Kalman filter, a linearization of $h(x)$ must be performed. The LS problem associate becomes

$$x^{SL}(k) = argmin(Z^k - h(x)W(Z^k - h(x))^T) = argminJ(x,k)$$

and we have to found

$$\frac{dJ(x,k)}{dx} = 0$$

After a bit of management we can compute

$$J(x,k) = Z^k$$

and than it follows that

$$\frac{dJ(x,k)}{dx} = -H(x)^T W(Z^K - h(x)) = 0$$

where $H(x) = \frac{dh(x)}{dx}$ So in this particular case:

$$H = \begin{bmatrix}\frac{x}{(x^2+y^2)^{1/2}} & \frac{y}{(x^2+y^2)^{1/2}} & 0 & 0\\ \frac{-y}{(x^2+y^2)} & \frac{x}{(x^2+y^2)} & 0 & 0\end{bmatrix} \quad (1)$$

Model used

To models the movement of the target has been employed two models: The discrete Wienerprocess acceleration model (DWPA) (reference) and the unicycle model Both are Markov process.

Discrete Wiener Process Acceleration Model

The Discrete Wienerprocess acceleration model choose are described by 5 different mode

DPWA		
Modes	Description	Input
mode 1	costant velocity	$u(k) = 0$
mode 2	north acceleration	$u(k) = a$
mode 3	south acceleration	$u(k) = -a$
mode 4	east acceleration	$u(k) = a$
mode 5	west acceleration	$u(k) = -a$

In the case of the DWPA we choosen to have 5 states: costant velocity, positive or negative acceleration on x direction and positive or negative deceleration on y direction. The linear system describing the evolution of the state is

$$x(k+1) = Ax(k) + B(s)u(k) + Gw(k) \quad (2)$$

Where $x(k)$ is the state at the current timestep, $u(t)$ is the input and $w(k)$ is the process noise. In this case the state is a vector of the cartesian cordinates and their relative speed, while the input is the acceleration which the noise will influence. State,state and noise matrix associate to the DWPA has shown below:

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

With δ time step of the simulation.
We hypothesize that the input is unknown in direction but known in magnitude. This set according to the current state at which the markov process is at the timestep. This is modeled though the switch between a set of matrices $B(s)$ that change according to the chain's state s.

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

It can be noticed that the matrix G is a merge between the second and the fourth matrices found in the set $B(s)$ as the noise will act randomly in direction and magnitude at every time-step.

Unicycle mode

The unycicle model has different input and state description. The input are represented by the angular acceleration of the wheel and the steering angle acceleration. possible modes are 5: constant angular velocity, acceleration and deceleration of the wheel and positice or negative acceleration of the steering angle.

Unicycle		
Modes		Input
mode 1	costant angular velocity	$u(k) = 0$
mode 2	angular acceleration	$u(k) = a$
mode 3	angular deceleration	$u(k) = -a$
mode 4	steer angle acceleration (CW)	$u(k) = a$
mode 5	steer angle acceleration (CCW)	$u(k) = -a$

Unycicle model instead is a Non-linear model. The

$$x(k+1) = Ax(k) + B(s)u(k) + Gw(k) \qquad (3)$$

The matrix that represent the state of system for this case are

$$X = \begin{bmatrix} x \\ y \\ \theta \\ \omega \end{bmatrix} A = \begin{bmatrix} 1 & 0 & \delta cos(\dot{\theta}) & 0 \\ 0 & 1 & 0 & \delta sin(\dot{\theta}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} G = \begin{bmatrix} \delta^2/2cos(\theta)r & 0 \\ \delta^2/2sin(\theta)r & 0 \\ \delta r & 0 \\ 0 & \delta \end{bmatrix}$$

We hypothesize that the input is unknown in direction but known in magnitude. This set according to the current state at which the markov process is at the timestep. This is modeled though the switch between a set of matrices $B(s)$ that change according to the chain's state s.

$$B(s) = \left\{ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta^2/2cos(\theta)r & 0 \\ \delta^2/2sin(\theta)r & 0 \\ \delta r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -\delta^2/2cos(\theta)r & 0 \\ -\delta^2/2sin(\theta)r & 0 \\ -\delta r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \delta \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -\delta \end{bmatrix} \right\}$$

Also in this case it's possible to notice that $G(s)$ is a merge of $B(s)$ Therefore in the EKF a linearised model is needed and som

$$A(k)$$

Data's simulation

Imm and Linear Consensus

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?)
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