Imm algorithm implementation for 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 agent's tracking. The agent switches between linked models of movement by the means of a Markov chain. The goal is evaluate the best trade-off between error on estimated position and real position and number of messagesses involved in the tracking.

Setting

NON CAPISCO COSA VUOI DIRE, IN QUALSIASI CASO QUA AGGIUNGI CHE PARLIAMO DI FILTRI DA DISCRETE-DISCRETE
Tracking an object with a switching dynamic require proper algorithms and filters.Retriving physical data with observations it's not enough in the case of Markov processes that switch between modes and we need to , models achiving the best-fit to our data is needed in order to make a prediction and estimate with reduced uncertainty. Here comes to play a major role the IMM algorithm (Interacting Multiple Models): The data from sensor are combined with different dynamical models through the algorithm...

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 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 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.

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

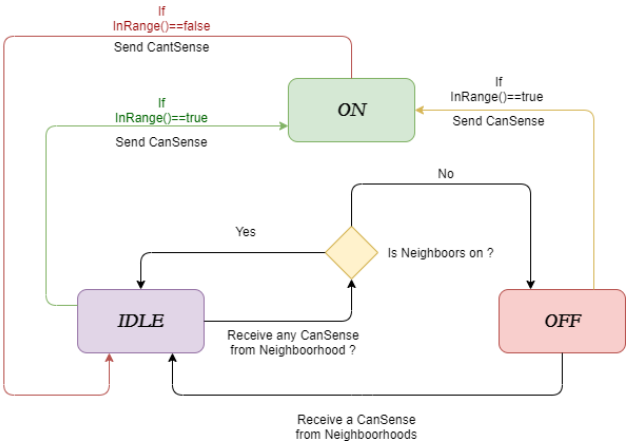

Figure 1: State machine sensor

	On	3 Idle	Off
InRange()==	True	False	False
Message sent	CanSense	CantSense	X
Functions	measurement	ready to switch to On	X

TABELLA CON LE FUNZIONI DEGLI STATI (mettere di fianco al paragrafo sotto una immagine che fa vederecosa intendiamo con neighboors (gli 8 sensori circostanti))
Sensors can communicate with the devices adjacent to them (Neighboorhood) 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.

Different state can receive and send different messagesses. A sensor in On state has the moving object in his range and send a CanSense to the nearby and when trackin moving away from the operating range send a CantSense turning itself to Idle. Sensors that receive a CanSense and are not already On turn in Idle state; during the Idle state sensors check if object appears in range sensors turn to On and send a CanSense, if nothing trigger the range it send to the nearby a CantSense signal. A sensor that is in Idle and during the information exchange receive only CantSense from the nearby turn itself Off and stop to check if somenthing is in range and can be turned again to Idle via a CanSense. This scheme implies that a sensors can never switch to On from Off except

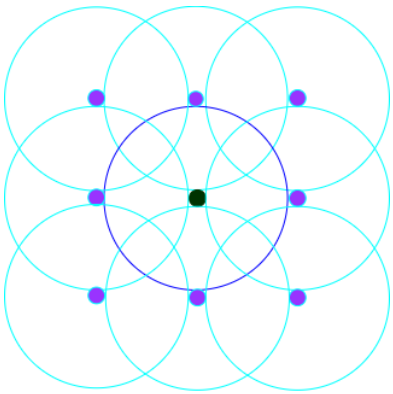

Figure 2: Neighboors for a sensors are the one immediately adiacent. The purples ones represent the neighboors of green sensor

for the initialization step.
When a sensor have the target in range it does a measure operation each timestep. the sensor works as a radar in this model and remembering

$$z(k) = h(x(k)) + v(k) \tag{1}$$

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 WLS problem associate becomes

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

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:

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

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?)

Model used

Introduction Discrete Wiener Process Acceleration Model

Discrete Wiener Process Acceleration Model and Unycicle model

The target in this simulation can move accordingly to two families of models, both are Markov processes: The Discrete Wienerprocess acceleration model and The Unycle model. The main caratheristic of both are represented in the table below

	DPWA		Unycicle
	Modes	Input	Modes
state 1	costant velocity	$\ddot{x}, \ddot{y} = 0$	constant angular veloci
state 2	north acceleration	$\ddot{x} = \delta$	angular acceleration
state 3	south acceleration	$\ddot{x} = -\delta$	angular deceleration
state 4	east acceleration	$\ddot{y} = \delta$	acceleration steering (C
state 5	west acceleration	$\ddot{y} = -\delta$	acceleration steering (CC

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.

state 1	costant velocity	$\ddot{x}, \ddot{y} = 0$
state 2	north acceleration	$\ddot{x} = \delta$
state 3	south acceleration	$\ddot{x} = -\delta$
state 4	east acceleration	$\ddot{y} = \delta$
state 5	west acceleration	$\ddot{y} = -\delta$

The Transition matrix associated with the Markov's chain is shown below

$$\begin{bmatrix}0.8 & 0.025 & 0.025 & 0.025 & 0.025//\end{bmatrix}$$

The linear system describing the evolution of the state is

$$x(k+1) = Ax(k) + B(s)u(k) + Gw(k) \tag{3}$$

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:

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

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.

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 state are 5: constant angular velocity, acceleration and deceleration of the wheel and positice or negative acceleration of the steering angle. The associate Markov chain for this case is

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

$$\begin{vmatrix} 0.8 & 0.025 & 0.025 & 0.025 & 0.025// \end{vmatrix}$$

Unycicle model instead is a Non-linear model. Thus is necessary to use a linearization in the Kalman filter implementation The matrix that represent the state of system for this case are

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

Data's simulation

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