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 traking. 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, we have limited the range of measurement to the distance between one sensor and the following one in any direction in the grid, as soon as the agent exceed the range 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 that we 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. The sensor is modeled as a state machine as shown.
AGGIUSTA FIGURA CHE E' SBAGLIATA E NON CI SONO GLI INRANGE, I CANSENSE/CANNOT SONO I SEGNALI MANDATI, NON LE ROBE CHE FANNO CAMBIARE STATO
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 before.
QUESTO METTILO NEL CONSENSUS, NON HA SENSO PARLARNE QUI
The set of active sensors 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.
The nodes composing the graph changes dynamically with iterations according with the movements of target. The numbers of sensors turned on depends on working range choosed and position of target. The mechanism that regulate the transition from sensors state relay on the information exchanged with nearby sensor. 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 for the initialization step. When a sensor have the target in range it does a measure operation each timestep. the sensor works a radar in this model and remembering

$$z(k) = H(k)x(k) + \mathbf{R} \tag{1}$$

where the **H** is

$$H = \begin{bmatrix} dopo & la \\ scr & vo \end{bmatrix} \quad (2)$$

Imm and Linear Consensu

The IMM algorithm is based on the

Model used

Introduction

Random walk and Unycicle model

The target in this simulation can move accordingly to two families of models, both are Markov processes.
TABELLA CON DA UNA PARTE LE MODES DI UNO E DELL'ALTRO AL POSTO DI SPIEGARLO A PAROLE
Each of this model has a Markov chain that regulate how the given input goes in the system. For the random walk we have 5 state: costant velocity, positive or negative acceleration on x direction and positive or negative deceleration on y direction. So the input are $\ddot{x}$ and $\ddot{y}$. The Transition matrix associated with the Markov's chain is shown below

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

The linear system describing the evolution of the state is

$$X^+ = \mathbf{A}X(k) + \mathbf{B}u(k) + \mathbf{G}w(k) \quad (3)$$

State,state and noise matrix associate to the random walk 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. The **B** input matrix change accordingly with the acceleration input (of what we suppose we know only the magnitude)

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

The **G** matrix is a merge between the "positive"matrices .

The unycicle model has different input and state description. The inputare represente 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

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

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