Report : Adaptive Beam Scheduling Algorithm for an Agile Beam Radar in Multi-Target Tracking

Ram Shirazi
id : 20427205

***Abstract*—An adaptive beam scheduling algorithm based on covariance control strategy is proposed for an agile beam radar in multi-target tracking applications. The algorithm uses IMMKFs (Interacting Multiple Model Kalman Filters) to track maneuvering targets. The optimal beam scheduling model is built which can schedule radar beam based on the difference between the desired covariance matrix and that of the actual covariance of each target for maintaining the target's state estimate covariance near a desired level. Two kinds of beam scheduling algorithms, the minimum mean bias criterion and the minimax bias criterion, are proposed, respectively. Simulation results show that the algorithms based on covariance control can quickly achieve the desired tracking state and allocate sensor resources effectively**

Keywords—component, formatting, style, styling, insert *(key words)*

I. Introduction

Future radar multiple-target tracking systems will utilize the agile beam (electronically scanned array or phased array)radar. The agile beam radar has the capacity to switch beam pattern, beam direction and waveform and can perform adaptive sampling by directing the radar beam without inertia in any direction. Thus the radar can switch between the tasks of tracking existing targets and acquiring new targets essentially instantaneously. When this kind of sensor is used to track multi-target, a strategy to schedule radar beam among those targets is needed. This problem is a beam scheduling problem, and is therefore also denoted measurement scheduling. Generally, beam scheduling of agile beam radars is studied in the field of sensor management. the paper author in [1] (based on previous works) develop an adaptive beam scheduling algorithm based on covariance control for agile beam radars in multi-target tracking. The proposed algorithm can schedule radar beam based on the difference between the desired covariance matrix and that of the actual covariance of each target to make the target maintain the desired tracking state. The validity of the proposed algorithm is proved by the simulation results.

II. Kalman Filter and IMMKFs

A. *Kalman filter matimatical model*

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

The Kalman filter addresses the general problem of trying to estimate the state $x \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{1}$$

with a measurement $z \in \mathbb{R}^m$ that is

$$z_k = Hx_k + v_k \tag{2}$$

- A, a $n \times n$ matrix , is the state transition matrix relating the previous time $k-1$ step to the current state k.
- B, a matrix $n \times l$ is the control input matrix to the optional control input u_{k-1}.
- H, a matrix $m \times n$ is a transformation matrix that transform the state into the measurement domain.
- The random variables w_k and v_k represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distribution:
  $$p(v) \sim N(0, R)\ ;\ p(w) \sim N(0, Q)$$

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

B. *Prediction*

In time update equation we're going to calculate the predicted state estimate (*a priori* state estimates) $\hat{x}_k^-$ and predicted error covariance(*a priori* error covariance estimates) P_k^-. First, the *a priori* state estimate $\hat{x}_k^-$ is predicted by using the state dynamic equation model that projects forward in time as follows:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{3}$$

where $\hat{x}_{k-1}$is the previous estimated state (*a posteriori* state estimate). Next, the error covariance matrix P_k^- is predicted by:

$$P_k^- = AP_{k-1}A^T + Q \tag{4}$$

where P_{k-1} is the previous estimated error covariance matrix and Q is the process noise covariance.

C. *Update*

During the update stage, we compute the Kalman gain K_k as follows:

$$S_k = HP_k^- H^T + R \tag{5}$$

$$K_k = P_k^- H^T S_k^{-1} \tag{6}$$

where R is the measurement noise covariance. After that we preform the actual measurement z_k, and S_k is the uncertainty (covariance) of the residual. In order to update the predicted state estimate x_k^-, we need to measure the measurement residual. It is the difference between the true measurement z_k and the previous estimated measurement $H\hat{x}_k^-$, so the measurement residual is $z_k - H\hat{x}_k^-$. To update the x_k is proceed by performing the summation of the previous updated state estimate x_k^- to the product of the Kalman gain and the measurement residual. It can be written as follows:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{7}$$

After obtaining the updated state estimate, the filter calculates the updated error covariance P_k which will be used in the next time step.

$$P_k = (I - K_k H)P_k^- \tag{8}$$

where I is the identity matrix.

D. *The IMMKFs algorithem*

The basic principle of an IMMKFs estimator is to run a bank of Kalman filters corresponding to some unknown model parameters and subsequently, the estimates from each filter are fused based on the probability of each filter, which is typically calculated using their error residual and covariance. The algorithm mainly consists of four steps: filtering, mode probability updating, state combination and filter interaction. Notice the filtering step has been described in the previous section.

n IMMKFs, a bank of KFs is run in parallel to provide an estimate $\hat{x}_k^j$ and each filter has its own mode probability μ_k^j, where $j \in \{1,2 \ldots, r\}$ is the filter number and $r \in \mathbb{N}$ is the number of total filters. These mode probabilities are updated from the filter likelihood, which is interpreted as how likely the filter provides a good state estimate from the measurement. Assuming the error residual is in Gaussian distribution, the likelihood is defined as follows:

$$\Lambda_k^j = \frac{1}{\sqrt{2\pi \det\left(S_k^j\right)}} \exp\left(-\frac{1}{2}\left(\hat{z}_k^j\right)^T \left(S_k^j\right)^{-1} \hat{z}_k^j\right) \tag{9}$$

where $\hat{z}_k^j = z_k^j - H\hat{x}_k^-$ is the residual between the noisy measurement and the priori estimate.

Based on the Bayes' theorem, the mode probabilities are updated as follows:

$$\mu_k^{j-} = \frac{\mu_k^j \Lambda_k^j}{\Sigma_{j=1}^r \mu_k^j \Lambda_k^j} \tag{10}$$

The output (state $\hat{x}_k^-$ and error covariance P_k^-) of the IMMKFs estimator are calculated by combining the weighted estimate $\hat{x}_k^{j-}$ and state error covariance P_k^{j-} of each filter, defined as follows:

$$\hat{x}_k^- = \Sigma_{j=1}^r \mu_k^j \hat{x}_k^{j-} \tag{11}$$

$$P_k^- = \Sigma_{j=1}^r \mu_k^j \left[P_k^{(j)+} + \left(\hat{x}_k^- - \hat{x}_k^{j-}\right)\left(\hat{x}_k^- - \hat{x}_k^{k-}\right)^T\right] \tag{12}$$

The filter interaction is that the filters with higher probabilities modify the estimates of the ones with lower probabilities. This step is particularly important for systems with time-varying uncertain parameters. The less likely filters are updated with better state and error covariance, thus, yielding a quicker response to changes in the model parameters. In IMMKFs, each filter is considered as a mode and the switching process of the modes is modelled by a time-invariant Markov chain. The mode transition probability π_{ij} describes how likely the mode i is changed to mode j and notice that it is a design parameter. The priori mode probability for mode j at next cycle $k + 1$ is defined as follows:

$$\mu_{k+1}^{j-} = \Sigma_{i=1}^r \pi_{ij} \mu_k^i \tag{13}$$

Subsequently, based on the Bayes' theorem, the mixing mode weight from mode i to mode j is calculated as follows:

$$\mu_k^{(i|j)-} = \frac{\pi_{ij}\mu_k^i}{\mu_{k+1}^{j-}} = \frac{\pi_{ij}\mu_k^i}{\Sigma_i \pi_{ij}\mu_k^i} \tag{14}$$

Finally, the mixed state $\hat{x}_k^{j-}$ and mixed error covariance P_k^{j-} are computed as follows:

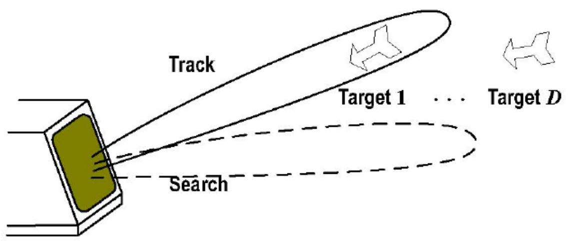
$$\hat{x}_k^{j-} = \Sigma_{i=1}^r \mu_k^{(i|j)-} \hat{x}_k^{i-} \tag{15}$$

$$P_k^{j-} = \Sigma_{i=1}^r \mu_k^{(i|j)-} \left[P_k^{i-} + \left(\hat{x}_k^j - \hat{x}_k^{i-}\right)\right)\left(\hat{x}_k^j - \hat{x}_k^{i-}\right)^T\right] \tag{16}$$

III. Beam Scheduling Model

A. *Intreduction*

In general the agile beam radar can track multi-target and search the undetected target simultaneously. We assume that total D targets are being tracked currently by the radar with only one steerable beam and the target coordinates evolve independently of each other. Since we have assumed that there is only one steerable beam, we can obtain noisy measurement of at the most one target at any given time instant. Our aim is to answer the follow question: Which single target track task or search task should the radar system choose to perform at each time instant to optimize some specified tracking performance? Fig. 1 shows a schematic representation of the problem.



1 Fig1 Multi-target tracking with an agile beam radar

For the beam control problem in multi-target tracking, the covariance control strategy can be used to allocate the sensor time resources effectively. Its basic ideal is as follows: first set a desired covariance for each target, and control the sensor beam to obtain the measurement data so as to minimize the difference between the desired covariance and the actual covariance. The residual time can be used to perform the new target searching task.

B. Mathematical modling

If the sensor tracks D tagets with the desired covariance matrix P_{des}^{i} , $(i = 1,2, \dots D)$, the control vector is expressed as :

$$U(t_k) = \{u(t) \mid u(t) = 0,1,2, \dots, D; t = t_1, t_2, \dots, t_k\} \quad (17)$$

Where $u(t)$ is the sensor work mode at time t, and $u(t) = 0$ corresponds to the search mode while $u(t) = j, (j \neq 0)$ to the jth target tracking. Thus, beam searching model is:

$$u(t_{k+1}) = j_0(t_{k+1}) = \arg \min_{j \in \{0,1,\dots,D\}} F\big[P_0^i, P^i(t_{k+1} \mid U(t_k), u(t_{k+1}) = j), i = 1,2, \dots, D\big] \quad (18)$$

Where $P^i(t_{k+1} \mid U(t_k), u(t_{k+1}) = j)$ denotes the tracking error covariance of the ith target on condition that the sensor tracks the jth target at time t_{k+1} and $F[\cdot]$ is the measurement function of two group of covariance matrices. Furthermore, two criterions, the minimum mean bias criterion, and the minimax bias criterion, are proposed for selecting the expression $F[\cdot]$. Those two criterions are marked as F-1 criterion and F-2 criterion. For the F-1 criterion, the beam scheduling model can be written as:

$$u(t_{k+1}) = j_0(t_{k+1}) = \arg \min_{j \in \{0,1,.,D\}} \frac{1}{D} \sum_{i=1}^{D} f\left(P_0^i, P^i(t_{k+1} \mid U(t_k), u(t_{k+1}) = j)\right) \quad (19)$$

For the F-2 criterion, the beam scheduling model can be written as:

$$u(t_{k+1}) = j_0(t_{k+1}) = \arg \min_{j \in \{0,1,\dots,D\}} \left[\max_{i \in \{1,2,\dots,D\}} f\left(P_0^i, P^i(t_{k+1} \mid U(t_k), u(t_{k+1}) = j)\right)\right. \quad (20)$$

Where the function $f(A, B)$ denotes the difference between the matrix A and the matrix B, it can choose many forms but the author picked the $g_1(\cdot,\cdot)$ metric as follows:

$$f(P_1, P_2) \triangleq g_1(P_1, P_2) = trace\ [abs(\varDelta P)] = \textstyle\sum_{i=1}^{n} |\varDelta p_{ii}| (20)$$

Where $\Delta P = P_1 - P_2$.

I also added another metric $g_2(\cdot,\cdot)$:

$$f(P_1, P_2) = g_2(P_1, P_2) = \det(abs(\varDelta P))$$

Important note: there is "no target" for $j = 0$ meaning that the minimum, in equation (18), is given for no update(no measurement is given) for any target, meaning that all targets are sufficiently close (in the sense of $f(\cdot,\cdot)$) to their desired covariance matrices, thus the optimal control is given by no update step for any target, and the radar will search for the next time-step.

C. IMMKFs for multi target tracking

The IMMKF blends motion models matched to different flight regimes to achieve improved performance against maneuvering target. For the ith target, let $\{t_{k(i)}, \hat{x}^i(t_{k(i)}), P^i(t_{k(i)})\}, (t_{k(i)} < t_k)$ denotes its state at current time t_k. $t_{k(i)}$ is the latest update time and $\hat{x}^i(t_{k(i)}), P^i(t_{k(i)})$ are the state and covariance estimate for all the models combined using the updated model probability weights μ_j^i as in equations (11) and (12).

D. Realization of Beam Scheduling Algorithem

1. *Start of prediction* – predict all targets states at time t_k for all the filters (without updating their covraince matrices) using eqution (3).
2. *Aggregation* - aggregate the state model and the error covariance metrices for each target using equation (11) and (12). and save the error covariance metrices.
3. *Finish prediction* – finish the prediction step by updating all the covariance matrices for all filters for all the targets.
4. *Optimal control* – using the saved covariance matrices, calculate for each target :

$$P_{pre}^i(t_{k+1} \mid U(t_{k+1})) = \begin{cases} P_{pre}^i(t_{k+1}^-) & , u(t_{k+1}) = i \\ P_{pre}^i(t_{k+1}) & , u(t_{k+1}) \neq i \end{cases} \quad (21)$$

Where $P_{pre}^i(t_{k+1}^-)$ and $P_{pre}^i(t_{k+1})$ denote the predicted covariance and the updated covariance resulting from the jth model, respectively.
$P_{pre}^i(t_{k+1}^-)$ is calculated using the saved covariances filters for each target , as follows:

$$P^i_{pre}(t^-_{k+1}) = \sum_{j=1}^{r} \mu^i_j(t^-_{k+1})\{P^i_j(t^-_{k+1}) + \left[\hat{x}^i_j(t^-_{k+1}) - \hat{x}^i(t^-_{k+1})\right] \cdot \left[\hat{x}^i_j(t^-_{k+1}) - \hat{x}^i(t^-_{k+1})\right]^T\} \quad (22)$$

and $P^i_{pre}(t_{k+1})$ is calculated using the latest predicted covariances for each target filters (determined in step 3) , as follows:

$$P^i_{pre}(t_{k+1}) = \sum_{j=1}^{r} \mu^i_j(t^-_{k+1})\{P^i_j(t^-_{k+1}) + \left[\hat{x}^i_j(t^-_{k+1}) - \hat{x}^i(t^-_{k+1})\right] \cdot \left[\hat{x}^i_j(t^-_{k+1}) - \hat{x}^i(t^-_{k+1})\right]^T\} \quad (22)$$

lastly, we can determine via equation (18) the optimal control $u(t_{k+1}) = j_0$ using $\{P^i_{pre}(t^-_{k+1})\ , P^i_{pre}(t_{k+1})\}, i \in \{0,1 \ldots D\}$.

5. *Target filters update* – update all the filters state and covariances according to (7) and (8) for the selected target (j_0).

6. *Update selected target filters weights, and aggregate the model state and covariance* – In this step we can update the model probability weights in two different methods, fallows by different state and covariance aggregation:

    a. Do not update the target model probability weights, meaning that the weights will stay constant throughout the whole simulation. aggregate the target state and covariance using equations (11) and (12).

    b. update the model probability weights using equations (9) and (10) and proceed to aggregate the selected target state and covariance using equations (14), (15) and (16).

Let $k = k + 1$ and return to step 1 until the end.

IV. Important notes

The author of the article had some ambiguities in his simulation guidelines and parameters which made the task of repeating his simulation difficult. Thus, I had to change the simulation

$$\mu^i_j(t^-_{k+1}) = \sum_{k=1}^{r} p^i_{jk}\mu^i_k(t_k)$$

fig2 – the update rule according to [1]

guidelines and even the algorithm to see sufficient results, I have listed the problems and their proposed solution :

- this is the first problem appearing after trying to simulate the algorithm as the author of [1] intended – if we look at the model probability weights update rule according to the author (Figure 2), we can see that the weights for each time t_k can be computed via the target switching matrix, regardless to the target state at time t_k (closed form solution for an initial μ^j_0), and the simulation diverges (the prediction for one of the target state diverges) if one uses this update rule. After looking at different implementation for the IMMFKs algorithm in the literature, I found an article about wind speed estimation for wind turbines[3], which implemented the IMMFKs algorithm. All of section II-D is inspired by the found article [3] and specifically option *b* in step 6 in the beam scheduling algorithm realization, as one can see, the update rule of the model weights in option *b* is not a closed form solution, and it depends on the current state and covariance of the target.

- The criteria F1 given in equation (19) is problematic. If we select the target with minimum metric $f(P_k, P_{des})$, at time t_k with covariance P_k , we are updating the target with the minimum uncertainty (relative to P_{des}). If we choose to update the target according to F1 as the author in [1] intended, instead of updating the target with more uncertainty, which will bring his covariance matrix closer to P_{des} and reduce the cumulative uncertainty of all the target , we will keep updating (selecting) the target that have less uncertainty which will result in constant selection of one target and neglecting the other targets. More reasonable criteria is to update the targe with maximum uncertainty, which will reduce the cumulative uncertainty of all the target, so the F1 criteria in my simulation is :

$$u(t_{k+1}) = j_0(t_{k+1}) = \arg \max_{j\in\{0,1,..,D\}} \frac{1}{D}\sum_{i=1}^{D} f\left(P^i_0, P^i(t_{k+1} \mid U(t_k), u(t_{k+1}) = j)\right) \quad (23)$$

- The beam scheduling realization presented in [1] is not the same as my suggested realization. The realization as the author in [1] wrote is more a "clairvoyant" version of the algorithm, because in each time t_k ,he updates all the targets with a measurement and then he selects the target which caused the cumulative uncertainty of all the targets to be a minimum, resulting in keeping the current measurement only for the selected target. In practice, one cannot foresee in such a deterministic way which target will benefits the most from a measurement, and in my simulation, I choose to update the target that the next prediction, will cause the overall cumulative uncertainty to be maximum, that is way I have more realization steps then in [1].

- The simulation results in [1] is very hard to reconstruct due to ambiguity in the simulation guidelines and parameters thus, I implemented my own simulation that shows the algorithm performance.

V. Simulation Guidlines.

A. Evaluation parameters

Covariance-misadjustment-ratio (CMR) quantifies the covariance control capability of the beam scheduling algorithm, and define as follows:

$$\varepsilon_i(t) = f\left(P^i(t) - P^i_0\right)/f\left(P^i_0\right), (i = 1,2, \ldots, D) \quad (24)$$

Where $f(\cdot)$ denotes the selected matrix metrices, $P^i(t), P_0^i$ are the actual covariance matrix of the ith target and its desired covariance, respectively. Thus, CMR provides a measure of how close the actual covariance is to the desired one.

B. *Trejectory generator:*

To evaluate the algorithm performance, I implemented a 2d trajectory generator which for each target, generate a trajectory with number of segments , each segment represents a target maneuver and is defined as follows :

$$x(t) = a_x \cdot t^2 + v_x \cdot t + p_x$$
$$y(t) = a_y \cdot t^2 + v_y \cdot t + p_y$$

where p_x, p_y are pre-determined only for the first segment, for the next segment these values are determined via the last value of the previous segment to keep a continues trajectory between segments, and a_x, a_y , v_x, v_y are independent random variables with a pdf:

$$a_x, a_y \sim U[-5,5] \ , \ \sigma_a = 2.82 \left[\frac{m}{s^2}\right]$$
$$v_y, v_x \sim U[-25,25] \ , \ \sigma_v = 14.42 \left[\frac{m}{s}\right]$$

Where σ_a and σ_v are the standard deviation of the acceleration and velocity, respectively. After constructing all the time segment, we smooth them via using cubic spline interpolation. If the random trajectory is too much random for proper interpolation, meaning that the number of knots points for the spline is over 45 due to a difficult maneuver, the generator will generate a new trajectory until the cubic interpolation can be done properly. Another important notice is that the random trajectory transition to the next time segment (simulating the target maneuver) is done rapidly, even after the interpolation smoothing, while the author in [1], evaluated the tracking on a much simpler trajectory with a slow targets maneuver, thus, we can expect different results.

C. *Simulation parameters.*

The state vector x_k for a constant acceleration (CA) model for motion in 2d plane is:

$$x_k \triangleq [x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}]_k^T$$

our dynamic equation for CA is :

$$x_{k+1} = x_k + \dot{x}_k \cdot \Delta t + \ddot{x}_k \cdot \frac{1}{2}\Delta t^2$$
$$y_{k+1} = y_k + \dot{y}_k \cdot \Delta t + \ddot{y}_k \cdot \frac{1}{2}\Delta t^2$$
$$\dot{x}_{k+1} = \dot{x}_k + \ddot{x} \cdot \Delta t$$
$$\dot{y}_{k+1} = \dot{y}_k + \ddot{y} \cdot \Delta t$$
$$\ddot{x}_{k+1} = \ddot{x}_k$$
$$\ddot{y}_{k+1} = \ddot{y}_k$$

considering we have no input to our dynamic system ($u(t) \equiv 0$) we can write our transition matrix from equation (1) as follows:

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

sensor for the radar is a multisensor data fusion with acceleration ($\ddot{x}$ *and* $\ddot{y}$) and position (x *and* y) thus our transformation matrix can be written as :

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

we defined the measurement noise matrix R as :

$$R = \begin{bmatrix} 30 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 \\ 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0.3 \end{bmatrix}$$

and I defined 3 Kalman filters for each target with process error covariance parameters q_1, q_2, q_3 . Where each filter is associated with process error covariance :

$$Q_j = \hat{Q} \cdot q_j$$

Where the process noise covariance matrix $\hat{Q}$ or error in the state process is basically can be written as follows (assuming the process noise axis independent):

$$\hat{Q} = \begin{bmatrix} \sigma_x^2 & 0 & \sigma_{x\dot{x}} & 0 & \sigma_{x\ddot{x}} & 0 \\ 0 & \sigma_y^2 & 0 & \sigma_{y\dot{y}} & 0 & \sigma_{y\ddot{y}} \\ \sigma_{\dot{x}x} & 0 & \sigma_{\dot{x}}^2 & 0 & \sigma_{\dot{x}\ddot{x}} & 0 \\ 0 & \sigma_{\dot{y}y} & 0 & \sigma_{\dot{y}}^2 & 0 & \sigma_{\dot{y}\ddot{y}} \\ \sigma_{\ddot{x}x} & 0 & \sigma_{\ddot{x}\dot{x}} & 0 & \sigma_{\ddot{x}}^2 & 0 \\ 0 & \sigma_{\ddot{y}y} & 0 & \sigma_{\ddot{y}\dot{y}} & 0 & \sigma_{\ddot{y}}^2 \end{bmatrix}$$

where $\sigma_x, \sigma_{\dot{x}}, \sigma_{\ddot{x}}$ are the standard deviations of the position, velocity, and acceleration, respectively (the explanations for the y axis are the same). We can define the standard deviation of position

$$\sigma_x = \sigma_{\dot{x}} \Delta t \cdot \sigma_{\ddot{x}} \frac{\Delta t^2}{2}$$

The position is not random generated in all the trajectory, in fact, given the initial position (which is given to each filter) the position is a deterministic function of the acceleration and velocity. because the way I constructed the time segments, the acceleration and velocity can be considered as constant throughout each segment, thus we can evaluate the standard deviation as:

$$\sigma_{\ddot{x}} = \frac{length\ of\ transition}{length\ of\ time\ segment} \cdot \sigma_a \leq \frac{1}{2}\sigma_a = 1.41 \left[\frac{m}{s^2}\right]$$

$$\sigma_{\dot{x}} = \frac{length\ of\ transition}{length\ of\ time\ segment} \cdot \sigma_v \leq \frac{1}{2}\sigma_v = 7.21 \left[\frac{m}{s}\right]$$

where $length\ of\ time\ segment = 100$ and $\max(length\ of\ transition) = 45$. we can also concur from the independence between the velocity and the acceleration that:

$$\sigma_{x\dot{x}} = \sigma_x\sigma_{\dot{x}}\ , \qquad \sigma_{x\ddot{x}} = \sigma_x\sigma_{\ddot{x}}\ , \qquad \sigma_{\dot{x}\ddot{x}} = \sigma_{\dot{x}}\sigma_{\ddot{x}}$$

Thus we can easily compute $\hat{Q}$.

The initial model probabilities are : $\mu_0 = [0.6, 0.3, 0.1]$. With model switching probabilities:

$$\pi = \begin{bmatrix} 0.95 & 0.025 & 0.025 \\ 0.025 & 0.95 & 0.025 \\ 0.05 & 0.05 & 0.9 \end{bmatrix}$$

the radar dwell time of each beam is $0.2sec$. I also choose different desired covariances for each simulation type to associate the results with the desired covariance.

VI. Simulations

All my simulations performed in Python, git link: https://github.com/ramshi236/Beam-Scheduling-Algorithm-/tree/main

A. *tracking targets uner F1 criteria with constant filters probability weight(option a).*

The desired covariance for 2 targets was equal, and determined as :

$$P_0^1 = P_0^2 = diag(250, 250, 25\ ,25,1.5,1.5)$$

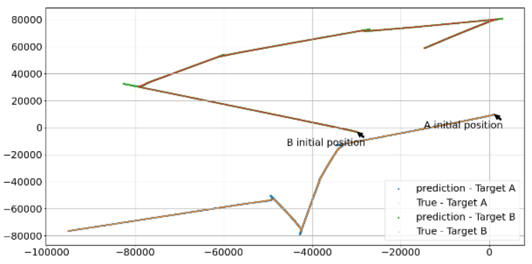


Figure3 – two targets simulation with equal desired covariance with F1 criteria and $g_1(\cdot,\cdot)$, with constant model probability weights preforming 4 maneuvers

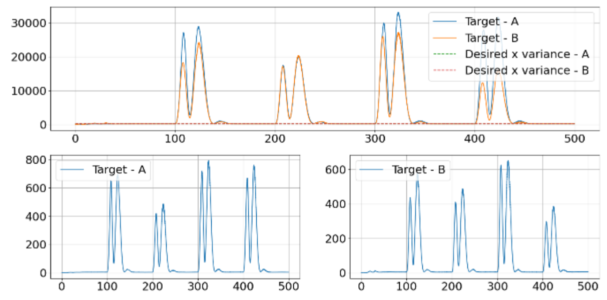


Figure4 – upper plot is the averaged error variance in the x axis for 2 targets with the same desired covariance with constant model probability weights using F1 criteria and g_1. Lower plots are the averaged CMR of each target

And the criteria is F1 with $g_1(\cdot,\cdot)$ function, the process error covariance parameters was defined as :

$$q_1 = 0.01, q_2 = 0.1, q_3 = 1$$

As we can see from Fig3, the tracking algorithm preforms well, and have a slight drawback when close to a target maneuver, the reason for that is that Kalman filter will not perform well in nonlinear environment of the target maneuver. I also preformed 100 Monta Carlo simulation to approximate the variance in the x-axis (the same results are for the y-axis) and the CMR for each target as we can see in Fig4. The 4 lobes in Fig4 are the error covariance caused by the 4 rapid maneuvers of each target, the reason for that phenomenon is that the Kalman filer for each model, at the middle of a time segment gains confidence in his prediction ,because he receives a fair number of updates that almost equal (almost in the sense of the additive measurement noise) to his prediction, thus he needs to receive enough measurement updates to reduce his prediction confidence to readjust himself to the rapidly target maneuver

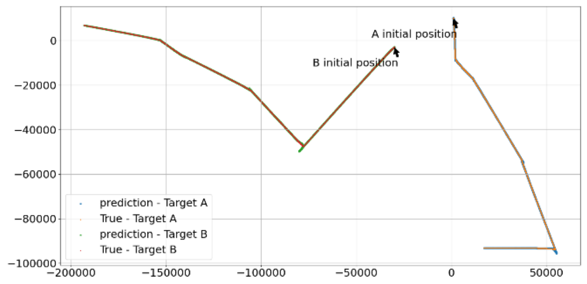


Figure5 – two targets simulation with equal desired covariance with F1 criteria and $g_2(\cdot,\cdot)$, with constant model probability weights preforming 4 maneuvers

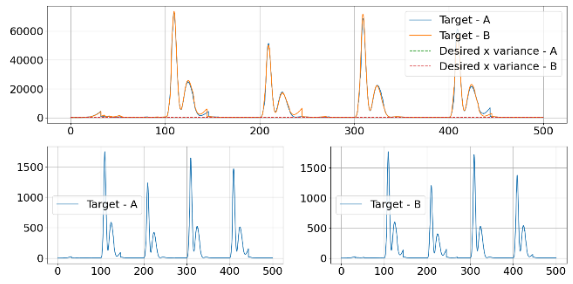


Figure6- upper plot is the averaged error variance in the x axis for 2 targets with the same desired covariance with constant model probability weights using F1 criteria and g_2. Lower plots are the averaged CMR of each target.

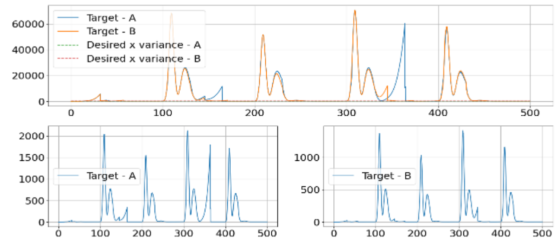


Figure7- upper plot is the averaged error variance in the x axis for 2 targets with different desired covariance with constant model probability weights using F1 criteria and g_2. Lower plots are the averaged CMR of each target

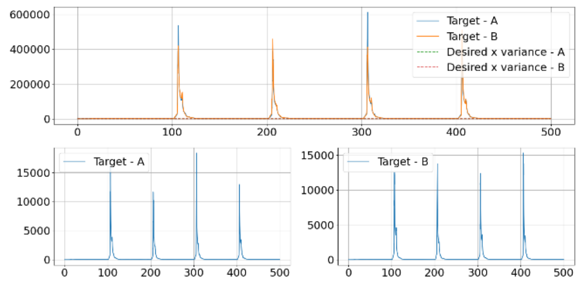


Fig8- The averaged error variance in the x axis for 2 targets with the same desired covariance in the x -axis using F1 criteria and $f(\cdot,\cdot) = g_2(\cdot,\cdot)$. with process error covariance noise q_1 =0.1 ,q_2=10 ,q_3 =100. Lower plots are the averaged CMR of each target

In Fig5 and Fig6 the same simulations preformed as Fig3 and Fig4 (equal covarince and constant model probabilty weights), but $f(\cdot,\cdot) = g_2(\cdot,\cdot)$. We can see the tradeoff between using g_1 or g_2 in their lobes caused by the target manuver. by using g_2 we gain higher peak in he left lobe but lower peak in the right one.

In Fig7, the same Monta Carlo simulation preformed as in Fig4 evuaulted ,but the deseird covarince matrix of the targets was alterd, more explictly:

$$P_0^1 = diag(200, 300, 25 ,25,1.5,1.5).$$
$$P_0^2 = diag(300, 200, 25 ,25,1.5,1.5).$$

and i also choose $f(\cdot,\cdot) = g_2(\cdot,\cdot)$. We can see that changing the desired covariance in specifc axis changes the error, and gives more attention (in the sennse of more updateds) to a specfic target, causing the apprence of a third lobe. In Fig8 the simulation was the same as the first simulation, but i used g_2 and different kind of process noise for each filters :

$$q_1 = 0.1 , q_2 = 10 , q_3 = 100$$

We can see that the lobes became much thinner but higher. We can cuncur that incresing the process varince, which reduce the prediction confidence, also reduces the recovery time after the target manuvers. In Fig9 I preformed the same simulation as in Fig8 but with F2 critiria.

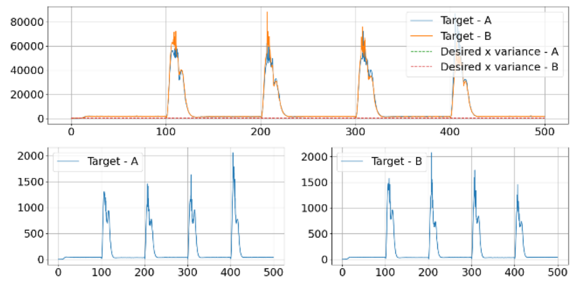


Figure9- The averaged error variance in the x axis for 2 targets with the same desired covariance in the x -axis using F2 criteria and $f(\cdot,\cdot) = g_2(\cdot,\cdot)$. with process error covariance noise q_1 =0.1 ,q_2=10 ,q_3 =100. Lower plots are the averaged CMR of each target

B. *tracking targets with updated filters probability weight(option b)*

In this simulation I set the same parameters as the previous section but updating the model probability weights using option b as shown in the realization. after carefully reading through [2], I have implemented the IMMKFs accordantly but achieved poor results as can see in Fig9. This might happen because I have implemented the code wrong, or because my trajectory generator is too difficult to predict due to rapid maneuvers. After the first maneuver the error covariance matrix for all the targets diverges right after receiving an update for a maneuvering target.

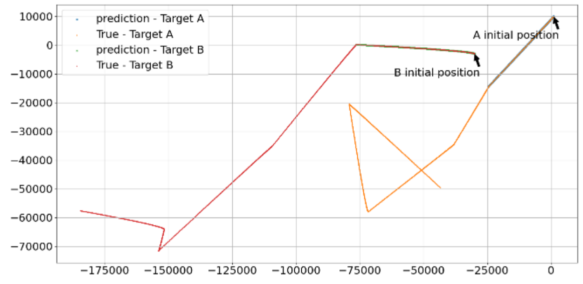


Figure 10- two targets simulation with equal desired covariance with F1 criteria and $g_2(\cdot,\cdot)$, with updated model probability weights according to option b preforming 4 maneuvers.

REFERENCES

[1] L. Jianbin, H. Weidong and Y. Wenxian, "Adaptive Beam Scheduling Algorithm for an Agile Beam Radar in Multi-Target Tracking," 2006 CIE International Conference on Radar, 2006, pp. 1-4, doi: 10.1109/ICR.2006.343564.

[2] arXiv:2104. Kalman-based interacting multiple-model wind speed estimator for wind turbines. https://arxiv.org/abs/2104.10063

[3] scipy- splprep () https://docs.scipy.org/doc/scipy/reference/reference/generated/scipy.interpolate.splprep.html#scipy.interpolate.splprep.

[4] An Introduction to the Kalman Filter Greg Welch and Gary Bishop https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf