Chapter 4

Centroid Detection and Tracking Algorithm

In image-based air traffic control or air defense system, automatic target recognition (ATR) is extremely important for the safety and early warning of the perceived threat. ATR process involves automatic target acquisition, identification and tracking by processing a sequence of images. Thus, ATR requires algorithm for detection, segmentation, feature computation, selection, classification and tracking. An important application of ATR is in guiding pilots of high performance aircraft flying close to the ground during bad weather or at night and in air traffic management (ATM) [34].

4.1 Target Tracking using Imaging Sensor Data

Detection and tracking of moving target is a challenging problem in forward-looking infrared (FLIR) image sequences because of low signal-to-noise ratio, low contrast, background clutter, partial occlusion of target. Tracking moving targets using image data, involves processing images from a target of interest and producing at each time step, an estimate of the target's current position and velocity vectors. Uncertainties in the target motion and in the measured values, usually modeled as additive random noise, lead to corresponding uncertainties in the target state. Also, there is additional uncertainty regarding the origin of the received data, which may or may not include measurements from the targets and may be due to random clutter (false alarms). This leads to the problem of data association [34]. In this situation tracking algorithms have to include information on detection and false alarm probabilities [35].

The main focus is on the implementation and validation of the algorithm for precision tracking with segmentation from imaging sensors [36]. This "Centroid Detection and Tracking Algorithm" (CDTA) is independent of the size of targets and is less sensitive to the intensity of the targets. In general typical characteristics of the target obtained by motion recognition or by object (pattern) recognition methods are used in associating images to the target being tracked. Motion recognition characteristics of a target are its location, velocity and acceleration (i.e. state vector) which could be generated using data from successive frames (inter-scan level). Object (pattern) recognition characteristics are its geometric structure (shape, size), energy level

distribution (i.e. different gray level in the image) in one or more spectral bands which are obtained using image data at the intra scan level.

The CDTA combines both object and motion recognition methods for practical target tracking from imaging sensors. The characteristics of the image considered are the intensity and size of the cluster. The pixel intensity is discretized into several layers of gray level intensities and it is assumed that sufficient target pixel intensities are within the limits of certain target layers. The CDTA implementation involves the conversion of the data from the image into a binary image by applying upper and lower threshold limits for the "target layers". The binary target image is then converted to clusters using nearest neighbor criterion. If the target size is known, the information is used to set limits for removing those clusters that differ sufficiently from the size of the target cluster to reduce computational complexity. The centroid of the clusters is then calculated and this information is used for tracking the target. Thus, CDTA involves the following steps (see Fig-4.1):

Intra-scan (single image) level:

- Identifying potential targets by image segmentation methods
- Calculation of centroid of the identified targets

Inter-scan (between images) level:

- Tracking centroid using single or multiple target tracking techniques
- Separation of the true and false targets by association based on both motion and object characteristics

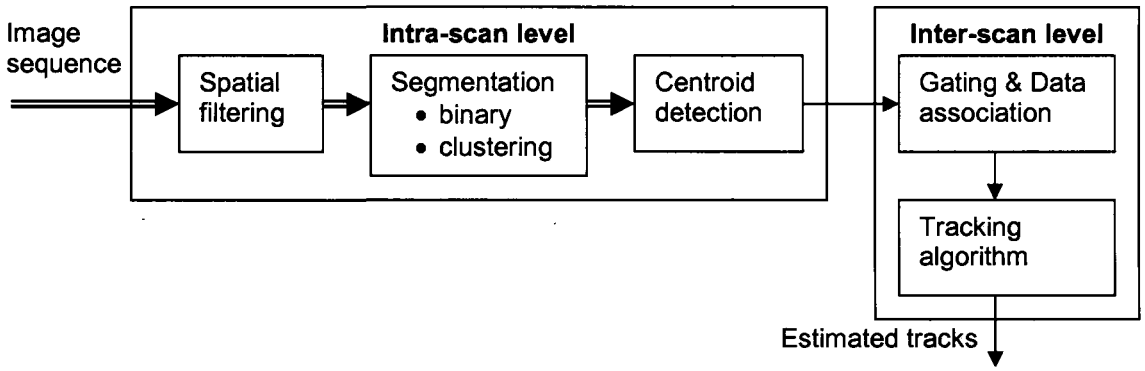


Fig-4.1 Information flow in CDTA

4.2 Spatial Filter

Images acquired through modern sensors may be contaminated by a variety of noise sources, such as sensor problems in a camera, variation in the detector sensitivity, environmental variations, discrete nature of radiation, dust covering the optics, quantization error or during a point to point transmission of an image. Noise reduction can be done in two ways: spatial domain filtering and frequency domain filtering. Spatial domain filtering refers to the image plan itself and it is based on direct manipulation of pixels in the image. Frequency domain filtering is based on modifying the Fourier transform of an image. In spatial filters, a mask (also called as template, kernel, window or filter) is a small 2D (say 3x3) array/image in which the co-efficients determine the nature of the process such as image smoothening, sharpening etc [37]. Image noise can be reduced by averaging many images of the same static scene. This approach is not feasible in many applications particularly image based target tracking for the reason that the sensor may be moving or the scene may be changing or it may be compulsory to use every scene in the processing. The two primary categories of spatial filters for noise removal from image are mean filters and order filters. The mean filters are linear and the order filters are non-linear operations.

4.2.1 Linear Spatial Filters

Linear filtering in the domain of image space is usually achieved by local convolution with size nxn kernel mask as follows. The convolution is created by a series of shift-multiply-sum operators with nxn kernel (convolution mask). When the image is convolved with a smoothing kernel mask, the brightness value of each pixel is replaced by a weighted average of the pixel brightness vales in a neighborhood surrounding the pixel. The weighting factor for various pixels in the neighborhood is determined by the corresponding value in the convolution mask. If the brightness value of any neighborhood pixel is unusually high or low (due to the influence of noise) then brightness averaging will tend to reduce the effect of the noise by distributing it among the neighboring pixels.

$$y(i,j) = \sum_{k=i-w}^{i+w} \sum_{l=j-w}^{j+w} x(k,l)h(i-k,j-l) \qquad (4.1)$$

where x : input image

h : filter function or impulse response or convolution mask

y : output image

(i, j) : index of the image pixel, $i, j = 1,2,..., N$

$w = \frac{n-1}{2}$

N : size of the input image

nxn : order of the filter (usually an odd number)

Mean filter is a simplest linear spatial filter, intuitive and easy to implement method of reducing noise in an image. The coefficients in the kernel (convolution mask) are non-negative and equal. Different size masks can be obtained as:

$$h_k = \frac{ones(k,k)}{k^2} \tag{4.2}$$

where $ones(k,k)$ is a kxk square matrix having all elements are unity and the subscript k indicates the mask size.

4.2.2 Non-Linear Spatial Filters

Non-linear filters do not have the same kind of output for each pixel. These filters are data dependent. Order filters are one kind of non-linear filters. Order filters are based on a specific type of image statistics called order statistics. These filters are implemented by arranging the neighborhood pixels in order from smallest to largest gray-level values and using this ordering to select the correct value. The most useful of the order filters is the median filter. The median filter selects the middle pixel value from the order set. The median filtering operation is performed on an image by applying the sliding window concept similar to what is done with convolution in linear spatial filter. The median is calculated by first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. Unrepresentative pixel in a neighborhood will not affect the median value significantly. This filter would not create any unrealistic pixel values when the filter straddles an edge because the median value exactly equal to one of the pixel value in the neighborhood. This could be the reason that the median filter is better at preserving sharp edges.

4.2.3 Image Noise Generation

There are two ways to corrupt an image with noise viz., image data independent noise and image data dependent noise. Image data independent noise can be expressed by an additive noise model. The measured image $x(i,j)$ is the sum of the true image $s(i,j)$ and the noise $v(i,j)$ i.e.

$$x(i,j) = s(i,j) + v(i,j) \tag{4.3}$$

It is assumed that the noise $v(i,j)$ is zero mean and evenly distributed over the frequency i.e. white noise described by its variance σ^2. Where as image data dependent noise, it could be possible to model the noise with a multiplicative or non-linear model. The models are mathematically more complex.

Salt and Pepper Noise

Salt and pepper noise is commonly referred as intensity spikes, speckle or data drop-out noise. This noise is caused by the errors in the image data transmission, by malfunctioning pixel elements in the camera sensors, faulty memory locations, or timing errors in the digitization process. The corrupted pixels are set alternatively to zero or maximum value that gives the image a salt and pepper like appearance. Uncorrupted pixels remain unchanged.

$$x(i,j) = \begin{cases} 0 & rand < 0.5d \\ 255 & 0.5d \le rand < d \\ s(i,j) & rand \ge d \end{cases} \tag{4.4}$$

where $rand$: uniformly distributed random numbers in the interval zero to one

d : noise density and it is real positive

Gaussian Noise

The noise is due to electronic noise in the image acquisition system. The noise can be generated with zero mean Gaussian distribution described by its standard deviation (σ).

$$v(i,j) = randn * \sigma \tag{4.5}$$

$$x(i,j) = s(i,j) + v(i,j) \tag{4.6}$$

where $randn$: normally distributed random numbers with mean and unit standard deviation and σ : standard deviation of the noise

4.2.4 Performance Evaluation Metrics

The performance of the above discussed algorithms is evaluated with the following metrics. The basic idea is to compute single number that reflects the quality of the smoothed/filtered image.

1) Mean Square Error

It is computed as the mean square error of true and filtered images. It produces a single number and this number will be zero when the corresponding true and filtered image pixels are truly alike. This value will increase when the corresponding filtered image pixels deviate from the true image pixels. The filtering algorithm that gives minimum value is preferable.

$$MSE = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} [s(i,j) - y(i,j)]^2 \tag{4.7}$$

where s : true image (ground truth)

y : smoothed/filtered image

N : size of the image

(i,j) : index of the pixel

2) Root Mean Square Error

It is computed as the root mean square error of true and filtered images. It produces a single number and this number will be zero when the corresponding true and filtered image pixels are truly alike. This value will increase when the corresponding filtered image pixels deviate from the true image pixels. The filtering algorithm that gives minimum value is highly preferable.

$$RMSE = \sqrt{\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}[s(i,j)-y(i,j)]^2} \tag{4.8}$$

3) Mean Absolute Error

It is computed as the mean absolute error of true and filtered images. It produces a single number and this number will be zero when the corresponding true and filtered image pixels are truly alike. This value will increase when the corresponding filtered image pixels deviate from the true image pixels. The filtering algorithm that gives minimum value is highly preferable.

$$MAE = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}|s(i,j)-y(i,j)| \tag{4.9}$$

4) Percentage Fit Error

It is computed as the norm of the true and filtered images to the norm of the true image. It produces a single number and this number will be zero when the corresponding true and filtered image pixels are truly alike. This value will increase when the corresponding filtered image pixels deviate from the true image pixels. In general, up to 5% will be acceptable. The filtering algorithm that gives least PFE is preferable.

$$PFE = \frac{norm(s-y)}{norm(s)} * 100 \tag{4.10}$$

5) Signal to Noise Ratio

The impact of the noise on the image is often described by SNR. Usually it is expressed in decibels (dB). It produces a single value and this value is not meaningful, but the comparison between two values for different filtered gives one comparison of quality. This value will be infinity when both true and filtered images are truly alike. The filtered algorithm that gives high SNR value is highly preferable.

$$SNR = 10\log_{10}\left[\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} s(i,j)^2}{\sum_{i=1}^{N}\sum_{j=1}^{N}[s(i,j)-y(i,j)]^2}\right] \qquad (4.11)$$

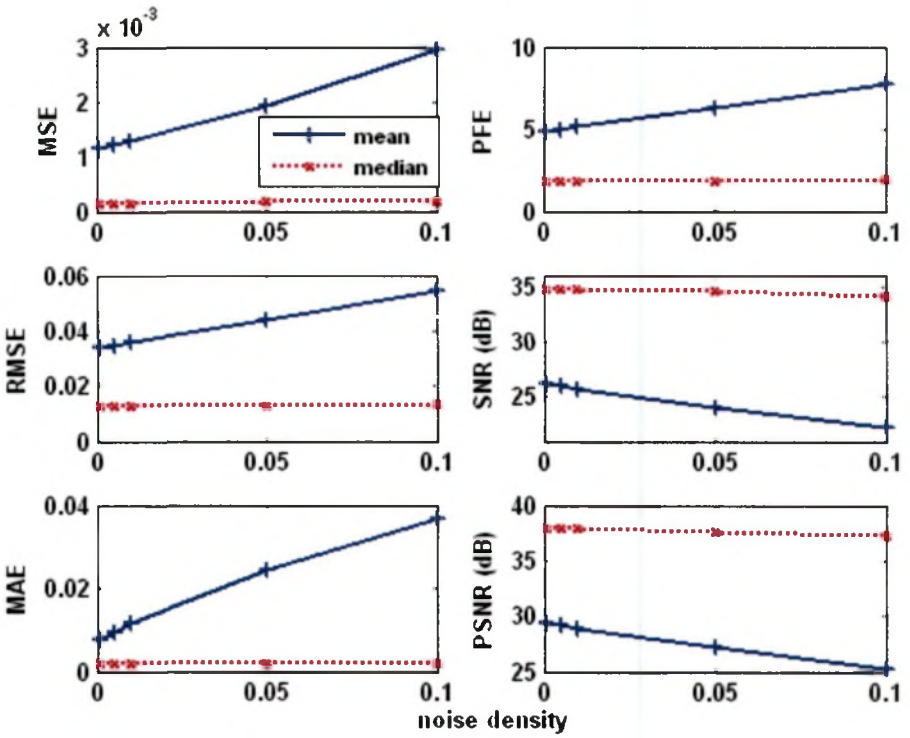
6) Peak Signal to Noise Ratio

It is also expressed in decibels (dB). It produces a single value and this value will be infinity when both true and filtered images are alike. This value will decrease when the filtered image is deviated from the true image. The filtered algorithm that gives high SNR value is highly preferable.

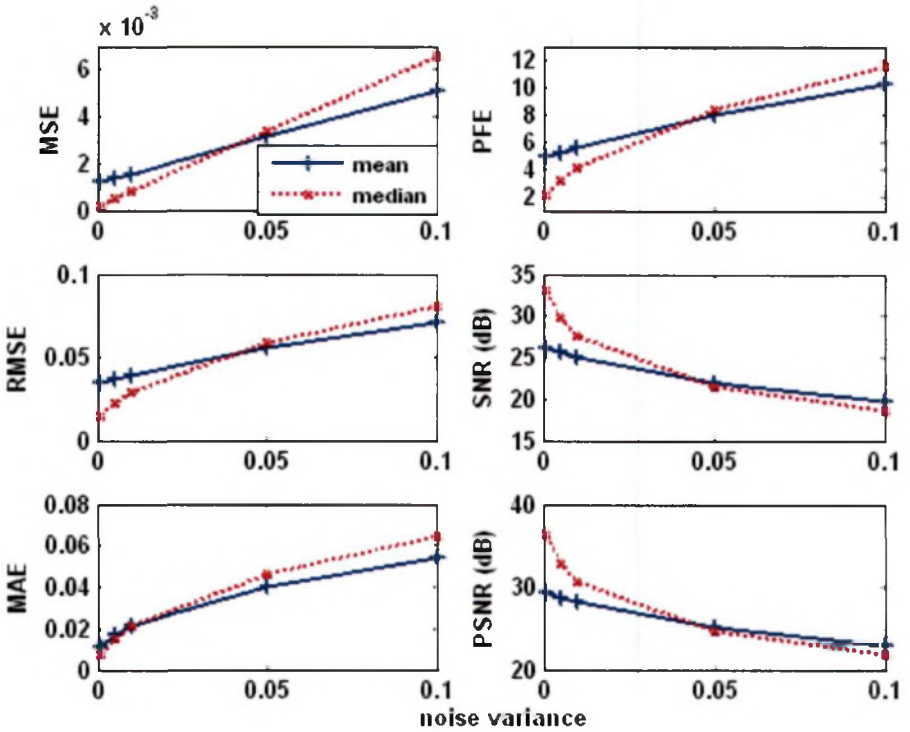
$$PSNR_{af} = 10\log_{10}\left[\frac{I_{\max}^2}{\sum_{i=1}^{N}\sum_{j=1}^{N}[s(i,j)-y(i,j)]^2}\right] \qquad (4.12)$$

where $I_{\max}$: maximum pixel intensity value

The performance of the mean and median filter to remove the salt & pepper noise with density 0.03 and Gaussian noise with mean zero & standard deviation $\sigma = 1.4$ are shown in Fig-4.2a and Fig-4.2b respectively. Mean filter performs well on reducing Gaussian noise or uniform noise and is poor for salt-and-pepper noise present in an image. The disadvantage of this filter is blurring the image edges or details because the coefficients are all positive. Median filter is very suitable to filter out the salt and pepper noise. And mean filter is suitable to remove Gaussian noise.



a) Salt & Pepper noise



b) Gaussian noise

Fig-4.2 Performance of mean and median filters in presence of salt & pepper noise and Gaussian noise

4.3 Segmentation and Centroid Detection Technique

4.3.1 Segmentation

Segmentation means decomposition of the image under into different regions. There are two kinds of segmentation: texture segmentation and particle segmentation. In texture segmentation, an image is partitioned into different regions (micro images) and each region is defined by a set of feature characteristics. This type of segmentation is used in medical imaging application, segmentation of text and halftones in document pages. Where as, in particle segmentation, an image is partitioned into object regions and background regions. Here segmentation refers to the task of extracting object or particles of interest as precisely as possible from the image. In CDTA algorithm, particle segmentation is used to separate the target (object of interest) from background, when target is not fully visible [36]. It is assumed that the pixel intensities are discretized into 256 gray levels. Particle segmentation is done in two steps: i). the gray level image is transformed into binary image using lower and upper threshold limits of the target. These thresholds of target are determined using the pixel intensity histograms from the target and its surroundings. ii) The detected pixels are grouped into clusters with nearest neighbor technique [36,38].

The gray image $\mathrm{Im}(i,j)$ is converted into binary image with intensity $\beta(i,j)$ by a hard limit on the intensity:

$$\beta(i,j)=\begin{cases}1 & I_L \le \mathrm{Im}(i,j) \le I_U \\ 0 & \text{otherwise}\end{cases} \qquad (4.13)$$

where I_L and I_U are the lower and upper threshold limits of the target intensity.

The detection probability of the pixel (i,j) can be defined as:

$$\begin{aligned} P\{\beta(i,j)=1\} &= p(i,j) \\ P\{\beta(i,j)=0\} &= 1-p(i,j) \end{aligned} \qquad (4.14)$$

where $p(i,j)=\dfrac{1}{\sigma\sqrt{2\pi}}\int_{I_L}^{I_U} e^{\frac{-(x-\mu)^2}{2\sigma^2}}\,dx$, considering the gray image $I(i,j)$ as having a Gaussian distribution with mean μ and variance σ^2

The binary image is then grouped into clusters using the nearest neighbor technique [36,38]. A pixel is considered as belonging to the cluster only if the distance between this pixel and at least one other pixel of the cluster is less than the proximity distance d_p. The d_p is chosen as:

$$\sqrt{\frac{1}{p_t}} < d_p < \sqrt{\frac{1}{p_v}} \tag{4.15}$$

where p_t and p_v are detection probabilities of target and noise pixels respectively.

By choosing the proximity distance as in Eq. (4.3), fewer noise clusters are obtained. The d_p affects the size, shape and number of clusters obtained by clustering. In practice, it is better to use d_p close to $\sqrt{\frac{1}{p_t}}$ to minimize the gaps in the target image [36,38].

4.3.2 Centroid Detection

The centroid of the cluster is determined using non-convolution method as:

$$(x_c, y_c) = \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}} \left(\sum_{i=1}^{n}\sum_{j=1}^{m} iI(i,j), \sum_{i=1}^{n}\sum_{j=1}^{m} jI(i,j) \right) \tag{4.16}$$

where $[x_c, y_c]$ is the centroid of the cluster, I_{ij} is the intensity of the $(i,j)^{th}$ pixel and n & m are the dimensions of the cluster.

4.4 Data Association and Tracking Algorithms

Tracking comprises of estimation of the current state of a target based on uncertain measurements selected according to a certain rule as sharing a common origin and calculation of the accuracy and credibility associated with the state estimate. The problem is complex even for single target tracking because of target model

uncertainties and measurement uncertainties. The complexity of the tracking problem increases further when multiple targets are to be tracked from measurements of multiple sensors.

Data association i.e. to determine from which target, if any, a particular measurement originated, is the central problem in multi sensor multi target tracking. The problem is complex due to uncertain data and disparate data sources. The identity of the targets responsible for each individual data set is unknown, so there is uncertainty as how to associate data from one sensor which are obtained at one time and location to those of another sensor at another point in time and location. Also, false alarms and the clutter detections may be present which are not easily distinguishable from the true target measurements.

Gating and data association enable tracking in multi sensor multi target (MSMT) scenario. Gating helps in deciding if an observation (which includes clutter, false alarms and electronic counter measures) is a probable candidate for track maintenance or track update. Data association is the step to associate the measurements to the targets with certainty when several targets are in the same neighborhood. Two approaches to data association are possible: i) using the nearest neighbor (NN) approach in which a unique pairing is determined so that at most one observation can be paired with a previously established track. The method is based upon likelihood theory and the goal is to minimize an overall distance function that considers all observation-to-track pairings that satisfy a preliminary gating test, ii) decision is achieved using probabilistic data association PDA algorithm in which a track is updated by a weighted sum of innovations from multiple validated measurements. The NNKF or PDAF is necessary for the centroid tracking application because in the neighborhood of the predicted location for the target centroid during tracking, several centroids could be found due to splitting of the target cluster or due to noise clusters. A comparative study of NNKF and PDAF was done in Ref.8 and it was concluded that PDAF shows better performance in presence of high clutter and measurement loss.

4.4.1 Gating Principle

Gating is a method, which is used to resolve the prediction of track positions problem in data association by eliminating unlikely observations to track pairings [39-42]. It is

used in early stages of data association process in order to reduce computational load. Here the basic logic is to differentiate target from clutter based on the "distance" from predicted target position. Calculating these distances and comparing them for all measurements is computationally intensive. Hence gating is used to reduce the number of candidate measurements to be considered for comparison.

Gate selection is also one of the problems during tracking in clutter environment. A gate is introduced around the predicted measurement (as origin) in which the true measurement of interest will possibly lie. The measurement falling within the gate is to be considered as valid measurement for data association whether it may or may not be related to the target. The gating principle is illustrated in Fig-4.3.

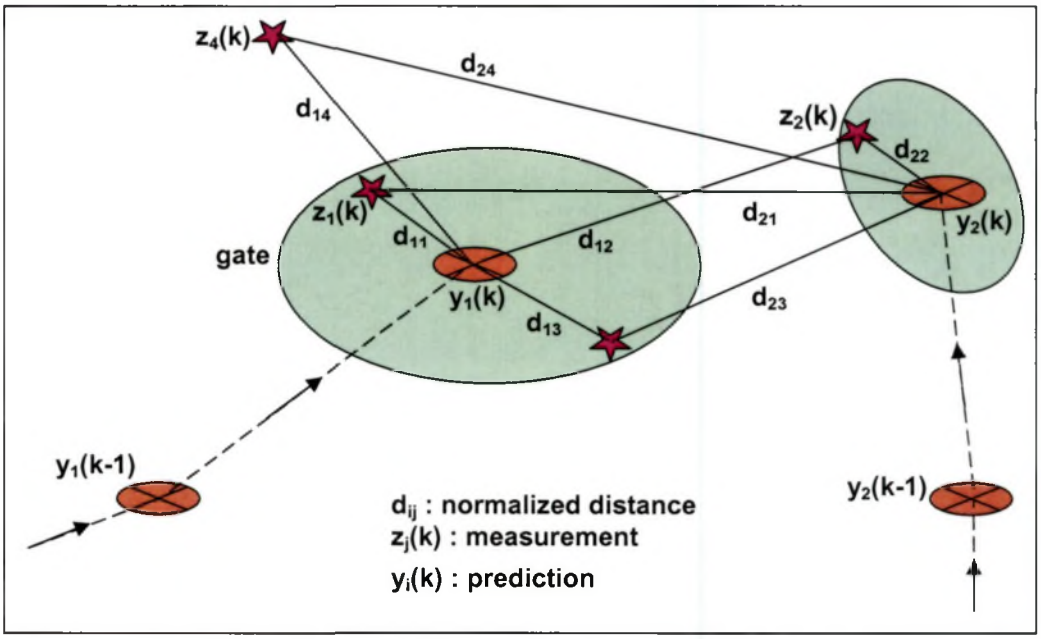


Fig- 4.3 Illustration of gating principle

From Fig-4.3, one can see that there is two tracks y_1 and y_2 at time $k-1$. These tracks are predicted to time k. At time k, there are four measurements z_1 to z_4. The quantity d_{ij} is the normalized distance between i^{th} track to j^{th} measurement. If d_{ij} is less than the gate threshold then the j^{th} target is within the i^{th} target's gate other wise out side the gate. Measurement z_4 is not fall any gate that means it is not associated to any existing targets so it will be used to initiate a new target. Measurement z_2 is within the target y_2 and it does not fall in any other gate so this measurement is taken for y_2 track updation. The measurements z_1 and z_3 are within

the y_1 track gate. Now the problem is which measurement is taken for updation. This is the situation where data association comes into picture.

Let us consider measurement received from the sensor at scan k as given by:

$$z(k+1) = HX(k+1) + v(k+1) \tag{4.17}$$

At scan k, the Kalman tracking filter forms the prediction $\tilde{X}(k+1|k)$ for the state vector $X(k)$ to be used at time $k+1$. The difference between measured and predicted quantities is given by

$$\vartheta = z(k+1) - \tilde{z}(k+1|k) \tag{4.18}$$

and associated residual covariance matrix S is given by

$$S(k+1) = H\tilde{P}(k+1|k)H^T + R \tag{4.19}$$

It is assumed that the measurement $z(k+1)$ at frame $k+1$ conditioned on Z^k is Gaussian distributed:

$$p[z(k+1)|Z^k] = N[z(k+1); \tilde{z}(k+1|k), S(k+1)] \tag{4.20}$$

where $N[.,.,.]$ denotes the Gaussian probability density function with argument the vector-valued random variable $z(k+1)$, mean $\tilde{z}(k+1|k)$ and innovation covariance matrix $S(k+1)$.

Then the true measurement will be in the region defined by the following equation

$$\begin{aligned} V(k+1,\gamma) \cong & \left\{ [z(k+1) - \tilde{z}(k+1|k)]^T S^{-1}(k+1)[z(k+1) - \tilde{z}(k+1|k)] \le \gamma \right\} \\ & = \vartheta^T(k+1) S^{-1}(k+1) \vartheta(k+1) \end{aligned} \tag{4.21}$$

where γ is the gate threshold that determines the probability that the measurement $z(k+1)$ is inside the gate.

It is noted that the quantity $\vartheta^T(k+1)S^{-1}(k+1)\vartheta(k+1)$ is the sum of the squares of n_z independent Gaussian variables with zero mean and unit standard deviation. Therefore the quantity is $\chi^2_{n_z}$ distribution. Hence the gate threshold γ is obtained from the tables of the χ^2 distribution [11].

4.4.2 Probabilistic Data Association Filter

The PDAF algorithm calculates the association probabilities for each valid measurement at the current time to the target of interest. The information flow in PDAF algorithm is shown in Fig-4.4. This probabilistic information is used in a tracking filter (PDAF) that accounts for the measurement origin uncertainty. If there are m measurements falling within the gate and it is assumed that there is only one target of interest and track has been initialized [9,11], the association events

z_i = { y_i is the target originated measurement}, i=1,2,...,m,

{none of the measurements is target originated}, i=0 (4.24)

are mutually exclusive and exhaustive for $m \geq 1$. The conditional mean of the state can be written as

$$\hat{X}(k/k) = \sum_{i=0}^{m} \hat{X}_i(k/k) p_i \tag{4.25}$$

where $\hat{X}_i(k/k)$ is the updated state conditioned on the event that the i^{th} validated measurement is correct and p_i is the conditional probability of this event. The estimate conditioned on measurement 'i' being correct is given by

$$\hat{X}_i(k/k) = \tilde{X}(k/k-1) + K\vartheta_i(k), \; i = 1,2,...,m \tag{4.26}$$

The conditional innovation is given by

$$\vartheta_i(k) = z_i(k) - \tilde{z}(k/k-1) \tag{4.27}$$

The gain K is the same as in Kalman filter eq. (4.5). For $i = 0$, i.e. if none of the measurements is valid ($m = 0$), then

$$\hat{X}_0(k/k) = \tilde{X}(k/k-1) \qquad (4.28)$$

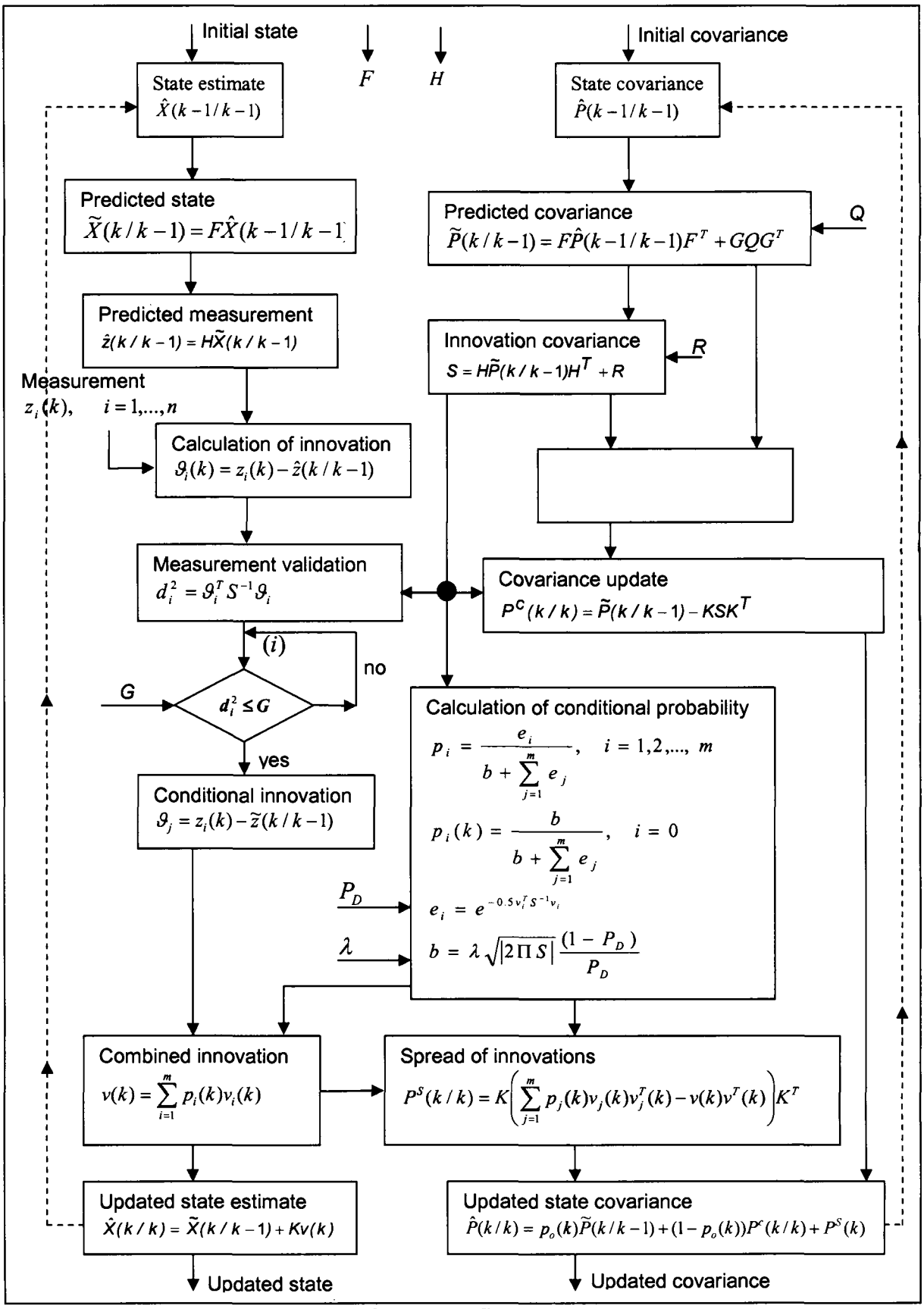


Fig-4.4 Information flow in PDAF

Combining the equations (4.21), (4.23) & (4.20) yield, the state update equation of the PDAF

$$\hat{X}(k/k) = \tilde{X}(k/k-1) + K\nu(k) \tag{4.29}$$

The combined innovation is given by:

$$\nu(k) = \sum_{i=1}^{m} p_i(k)\nu_i(k) \tag{4.30}$$

The covariance associated with the updated state is:

$$\hat{P}(k/k) = p_0(k)\tilde{P}(k/k-1) + (1-p_0(k))P^c(k/k) + P^s(k) \tag{4.31}$$

where the covariance of the state updated with correct measurement

$$P^c(k/k) = \tilde{P}(k/k-1) - KSK^T \tag{4.32}$$

and the spread of the innovations

$$P^s(k/k) = K(\sum_{i=1}^{m} p_i(k)\nu_i(k)\nu(k)_i^T - \nu(k)\nu(k)^T)K^T \tag{4.33}$$

The conditional probability is calculated using Poisson clutter model

$$p_i(k) = \frac{e^{-0.5\nu_i^T S^{-1}\nu_i}}{\lambda\sqrt{|2\Pi S|}\frac{(1-P_D)}{P_D} + \sum_{j=1}^{m} e^{-0.5\nu_j^T S^{-1}\nu_j}}, i = 1,2,...,m$$

$$= \frac{\lambda\sqrt{|2\Pi S|}\frac{(1-P_D)}{P_D}}{\lambda\sqrt{|2\Pi S|}\frac{(1-P_D)}{P_D} + \sum_{j=1}^{m} e^{-0.5\nu_j^T S^{-1}\nu_j}}, \text{ i=0} \tag{4.34}$$

where λ =false alarm probability

P_D = Detection probability

4.5 Models for Data Generation

4.5.1 Synthetic image data generation

The mathematical model of FLIR (forward looking infrared sensor) for generation of synthetic image [38, 43] is briefly described. Consider two-dimensional array of

$$m = m_{\xi} \times m_{\eta} \tag{4.35}$$

pixels where each pixel is represented by a single index $i = 1,\cdots,m.$ and the intensity I of pixel i is given by:

$$I_i = s_i + n_i \tag{4.36}$$

where, s_i is the target intensity and n_i is the noise intensity in pixel i, which is assumed to be Gaussian with zero mean and covariance σ^2.

The total target-related intensity is given by:

$$s = \sum_{i=1}^{m} s_i \tag{4.37}$$

If the number of pixels covered by the target is denoted by m_s, then the average target intensity over its extent is given by:

$$\mu_s = \frac{s}{m_s} \tag{4.38}$$

The average pixel SNR (over the extent of the target) is

$$r = \frac{\mu_s}{\sigma} \tag{4.39}$$

In order to simulate the motion of the target in the frame, kinematic models of target motion are used. Constant velocity kinematic model is used for generation of the data which determines the position of the target in each scan.

4.5.2 State Model

The following state and measurement models are used to describe the constant velocity target motion.

$$X(k+1)=\begin{bmatrix}1 & T & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & T\\ 0 & 0 & 0 & 1\end{bmatrix}X(k)+\begin{bmatrix}\frac{T^2}{2} & 0\\ T & 0\\ 0 & \frac{T^2}{2}\\ 0 & T\end{bmatrix}w(k) \qquad (4.40)$$

where $X(k)=[x \quad \dot{x} \quad y \quad \dot{y}]^T$ state vector

T : sampling period and

$w(k)$: zero mean Gaussian noise with variance $Q=\begin{bmatrix}\sigma_w^2 & 0\\ 0 & \sigma_w^2\end{bmatrix}$

4.5.3 Measurement Model

$$z(k+1)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}X(k+1)+v(k+1) \qquad (4.41)$$

where $v(k)$ is the centroid measurement noise that is zero mean Gaussian noise with covariance matrix:

$$R=\begin{bmatrix}\sigma_x^2 & 0\\ 0 & \sigma_y^2\end{bmatrix} \qquad (4.42)$$

Both process noise and centroid measurement noise are assumed to be uncorrelated.

4.6 Results and Discussions

A two-dimensional array of 64×64 pixels is considered for the background image, which is modeled as white Gaussian random field with mean μ_n and variance σ_n^2 i.e $N(\mu_n, \sigma_n^2)$. A two-dimensional array of pixels, which is modeled as white Gaussian random field with a mean μ_t and variance σ_t^2 (i.e. $N(\mu_t, \sigma_t^2)$) are used to generate a rectangular target of size (9x9). Three sets of data are used to test the performance of the ICTA algorithm. In all sets, the total number of scans is fifty and image frame rate (T) is 1 frame/sec.

Data set – I: The initial state vector of the target in the image frame is:

$$X = [x \quad \dot{x} \quad y \quad \dot{y}]^T = [10 \quad 1 \quad 10 \quad 1]^T$$

The target and noise parameters used in this simulation are: target pixel intensity is $N(100,0)$ and noise pixel intensity is $N(50,0)$. The synthetic image with these parameters is shown in Fig-4.5. The image is converted into binary image using the upper $(I_U = 110)$ and lower $(I_L = 90)$ limits of a target layer (eq.4.1) and then grouped into clusters by the nearest neighbor technique using the optimal proximity distance d_p (d_p=2).

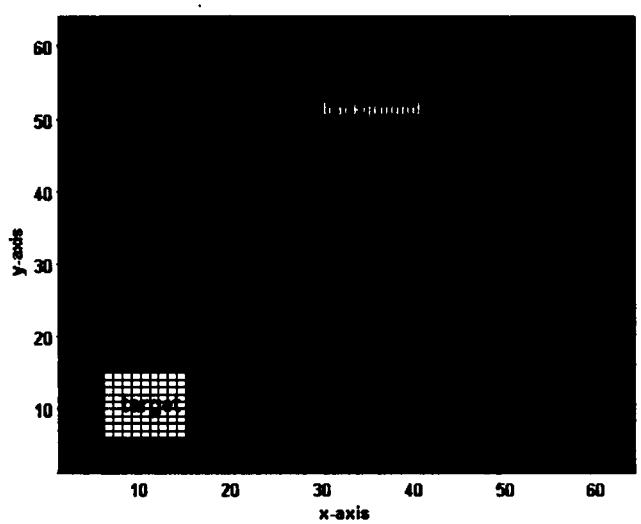


Fig-4.5 Synthetic image with target pixel intensity is $N(100,0)$ and background pixel intensity is $N(50,0)$ - Data set I

The centroid of each cluster is computed and used for state estimation in the measurement update part of the NNKF or PDAF filters to track the target. The true

and estimated target trajectory is shown in Fig-4.6. Since there is no back ground noises i.e. clear background, the estimated and true target positions are alike as shown in Fig-4.7 and the target position state errors are zero. The percentage fit error in both x- & y-position and the root mean square position error & root mean square velocity error are shown in Table-4.2. These parameters are with in the acceptable range that shows the tracker consistency.

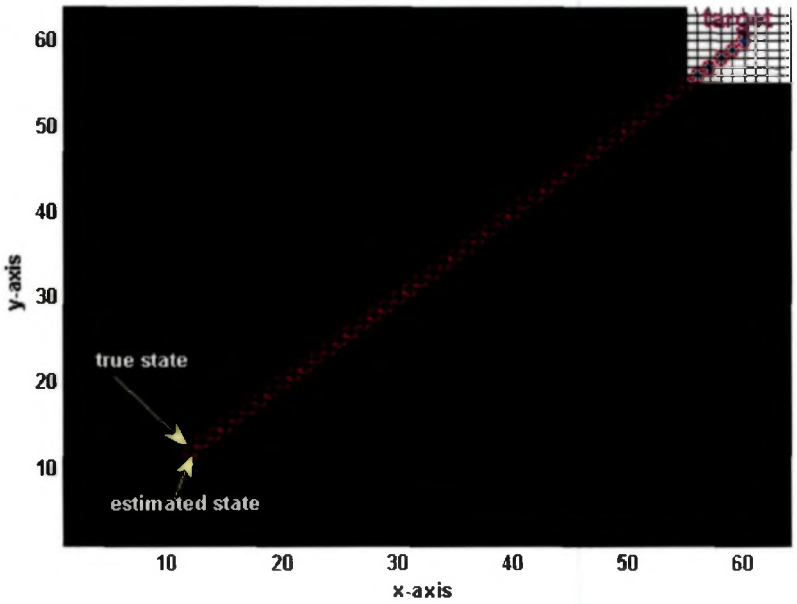


Fig-4.6 True and estimated trajectory (estimated - red circle, true – blue star) - Data set I

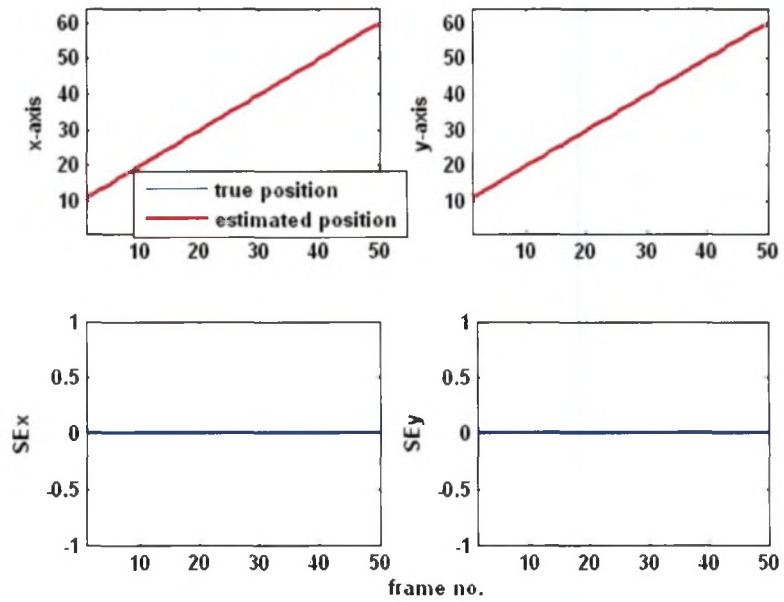


Fig-4.7 True & estimated x- and y-positions and state errors in x- and y-positions - Data set I

Tabel-4.1 PFE, RMSPE and RMSVE – Data set I

PFEx	*PFEy*	*RMSPE*	*RMSVE*
0.0	0.0	0.0	0.0

Data set – II: The initial state vector of the target in the image frame is:

$$X = [x \quad \dot{x} \quad y \quad \dot{y}]^T = [10 \quad 1 \quad 10 \quad 1]^T$$

The target and noise parameters used in this simulation are: target pixel intensity is $N(100,10)$ and noise pixel intensity is $N(50,50)$. The synthetic image with these parameters is shown in Fig-4.8. The image is converted into binary image using the upper ($I_U = 110$) and lower ($I_L = 90$) limits of a target layer and then grouped into clusters by the nearest neighbor technique using the optimal proximity distance d_p (d_p=2). The centroids of these clusters are computed. Since the background is very noise, the cluster algorithm produces more clusters hence more centroids. This necessitates the NNKF or PDAF to associate the true measurement to the target. The estimated and true target trajectory is shown in Fig-4.9.

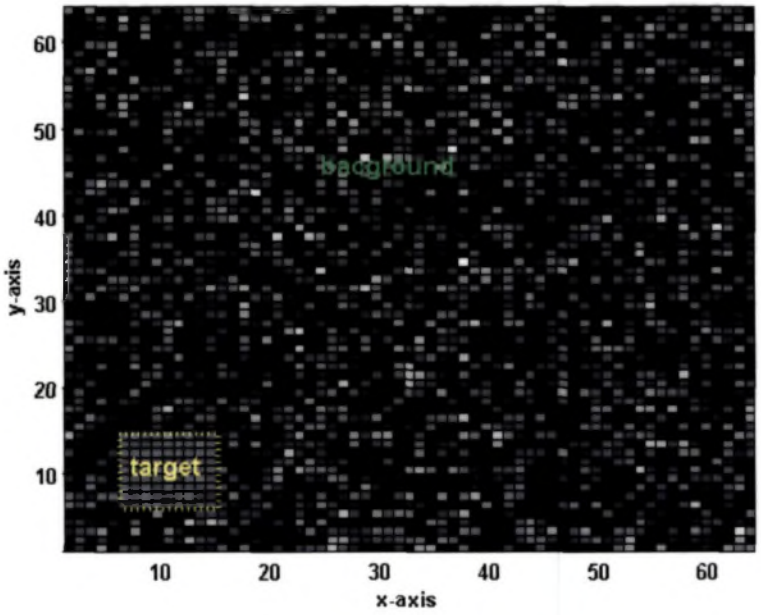


Fig-4.8 Synthetic image with target pixel intensity is $N(100,10)$ and background pixel intensity is $N(50,50)$ - Data set II

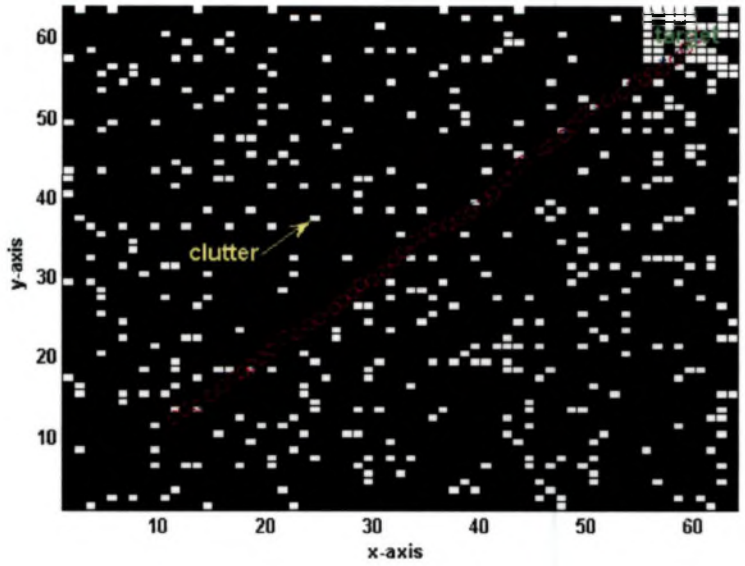


Fig-4.9 True and estimated trajectory (estimated: red circle, true: blue star) - Data set II

The true and estimated target position states and state errors are shown in Fig-4.10. Similarly, the true and estimated target velocity states and state errors are shown in Fig-4.11. Since the images are very noise, the estimates are little bit noisy but still the tracker tracks the target.

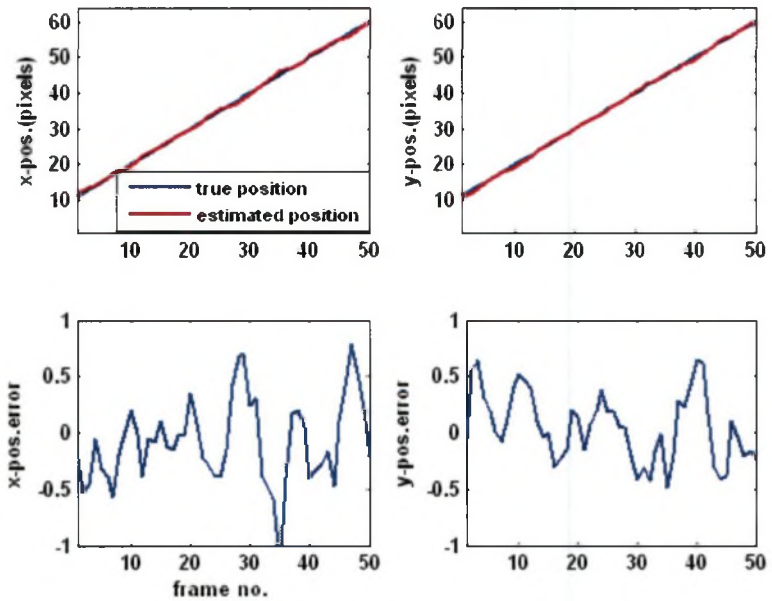


Fig-4.10 True & estimated x- and y-positions and state errors in x- and y-positions- Data set II

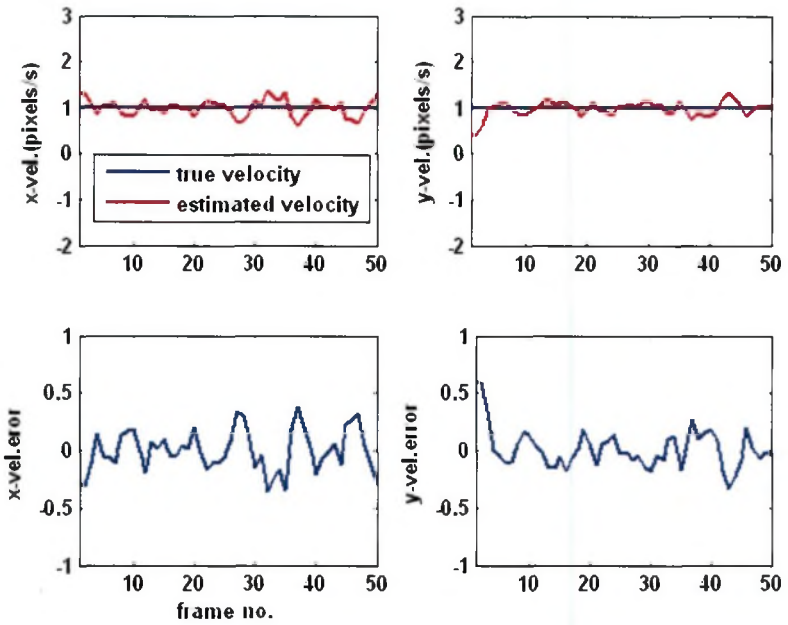


Fig-4.11 True & estimated x- and y-velocities and state errors in x- and y-velocities - Data set II

The root sum square error in position and velocity are shown in Fig-4.12. The RSSPE is less than 1pixel and RSSVE is less than 0.4 pixels/sec. State errors with theoretical bounds and innovation sequence with theoretical bounds are shown in Fig-4.13 and Fig-4.14 respectively. The percentage fit error and root mean square position error & root mean square velocity error are shown in Table-4.3. From the Figs-4.12 to 4.14 and Table-4.3, it observed that the parameters are within the acceptable limits that show the tracker consistency.

Tabel-4.2 PFE, RMSPE and RMSVE – Data set II

PFEx	*PFEy*	*RMSPE*	*RMSVE*
0.99	0.79	0.49	0.26

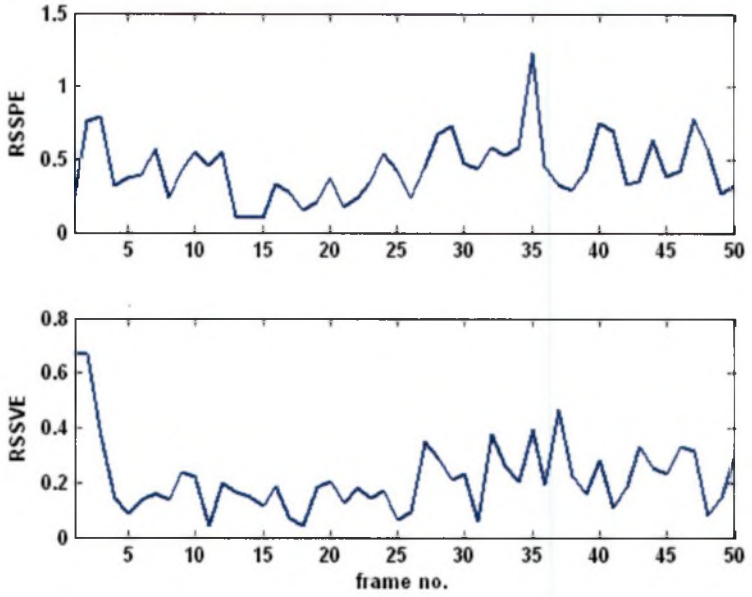


Fig-4.12 Root sum square error in position and velocity-Data set II

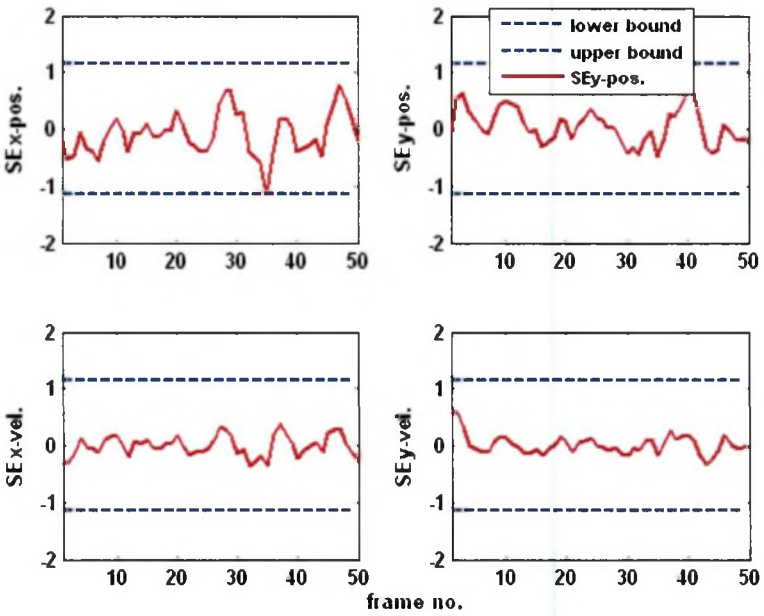


Fig-4.13 State errors with theoretical bounds- Data set II

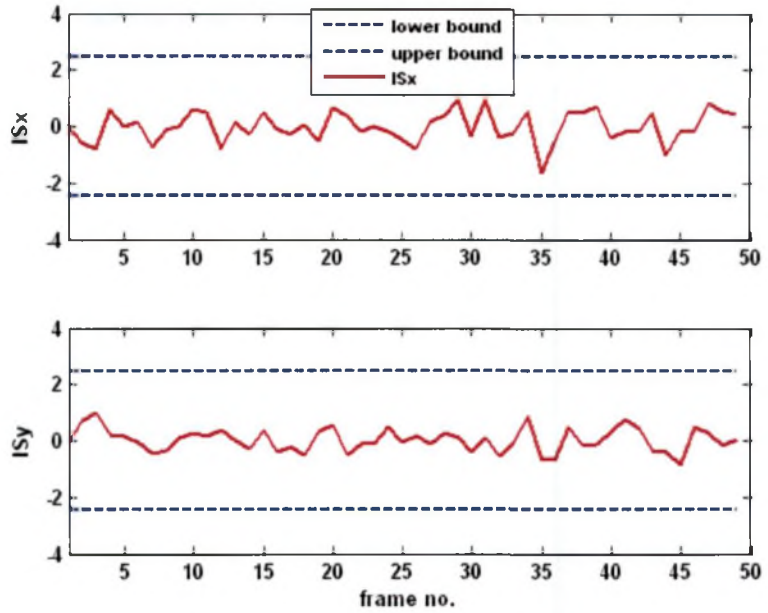


Fig-4.14 Innovation sequence with theoretical bounds- Data set II

4.7 Radar and Imaging Sensor Track Fusion

Having validated the performance of the ICTA for multiple-target tracking, the algorithm is used for providing input to state vector fusion when the data of position from ground based radars is available in Cartesian coordinate frame. Flow diagram for fusion of data from imaging sensor and ground based radar is shown in Fig-4.15.

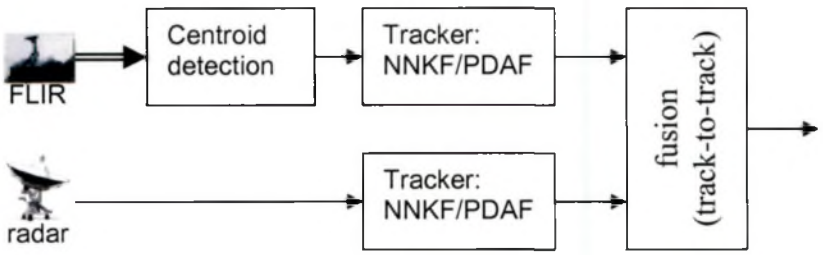


Fig-4.15 Fusion of data from imaging sensor and ground based radar

The two state vectors are fused using the following relations [11,38,44]. The tracks which are state vector estimates from the imaging sensor (track i) and ground based radar (track j) and their covariance matrices at scan k are shown below:

$$\text{Track } i\text{: } \hat{X}_i(k),\ \hat{P}_i(k) \quad \text{and} \quad \text{Track } j\text{: } \hat{X}_j(k),\ \hat{P}_j(k) \tag{4.43}$$

The fused state estimate is given by:

$$\hat{X}_c(k) = \hat{X}_i(k \mid k) + \hat{P}_i(k \mid k)\hat{P}_{ij}(k)^{-1}\left[\hat{X}_j(k \mid k) - \hat{X}_i(k \mid k)\right] \quad (4.44)$$

The combined covariance matrix associated with the estimate of Eq. (4.44) are given by:

$$\hat{P}_c(k) = \hat{P}_i(k \setminus k) - \hat{P}_i(k \mid k)\hat{P}_{ij}(k)^{-1}\,\hat{P}_i(k \mid k) \quad (4.45)$$

where $\hat{P}_{ij}$ is cross covariance between $\hat{X}_i(k \mid k)$ and $\hat{X}_j(k \mid k)$, and is given by

$$\hat{P}_{ij}(k) = \hat{P}_i(k \mid k) + \hat{P}_j(k \mid k) \quad (4.46)$$

The true and estimated x-pos, y-pos, x-vel, and y-vel from imaging sensor ground based radar are shown in Fig-4.16and Fig-4.17 respectively.

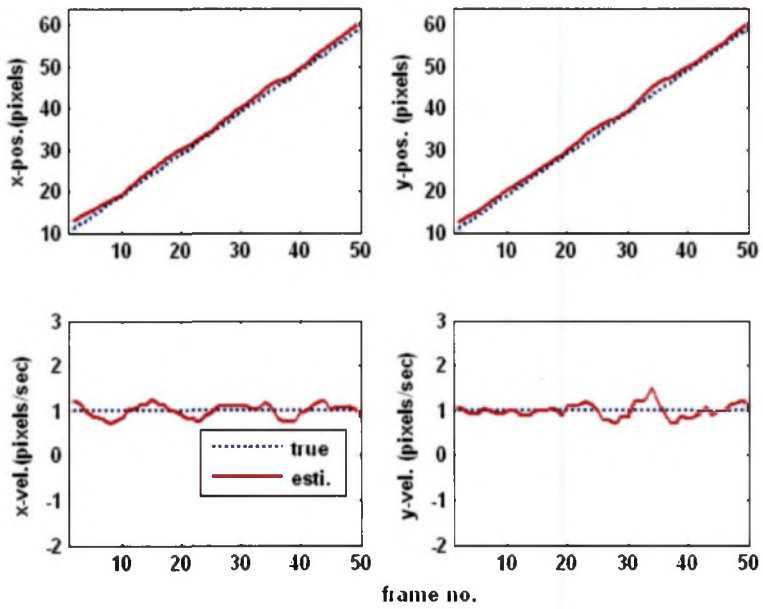


Fig-4.16 True and estimated x-pos., y-pos., x-vel. and y-vel. from imaging sensor

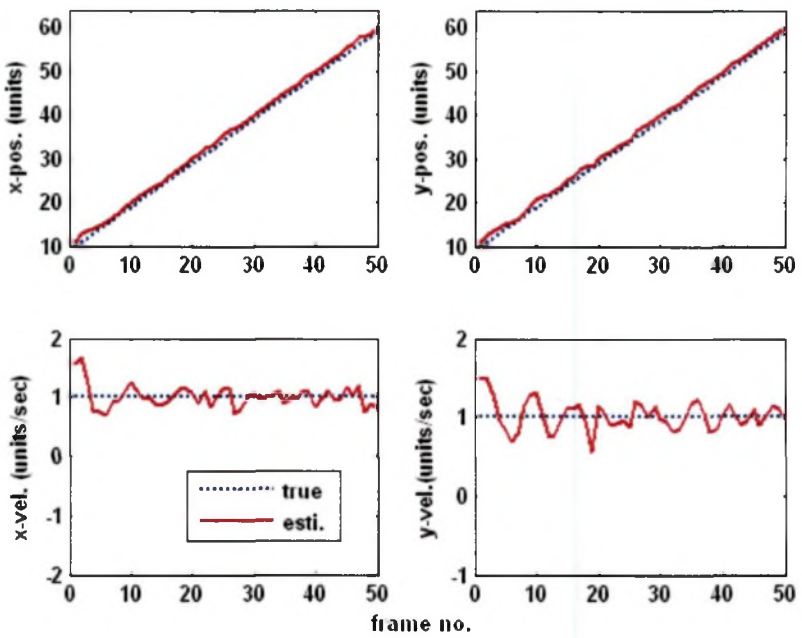


Fig-4.17 True and estimated x-pos., y-pos., x-vel. and y-vel. from ground based radar

The true, estimated and fused trajectories are shown in Fig-4.18 from which it is clear that the fused trajectory matches the true trajectory. The percentage fit error in x-pos & y-pos root mean square error in position and velocity is shown in Table-4.4. From Fig-4.18 and Table-4.4, it is observed that fusion of radar and imaging sensor data would provide better target state estimation.

Table-4.3 PFE, RMSPE and RMSVE – without measurement loss

	PFEx	*PFEy*	*RMSPE*	*RMSVE*
Imaging sensor	3.01	2.95	1.618	0.213
radar	2.78	2.74	1.497	0.233
fused	0.78	0.69	0.39	0.355

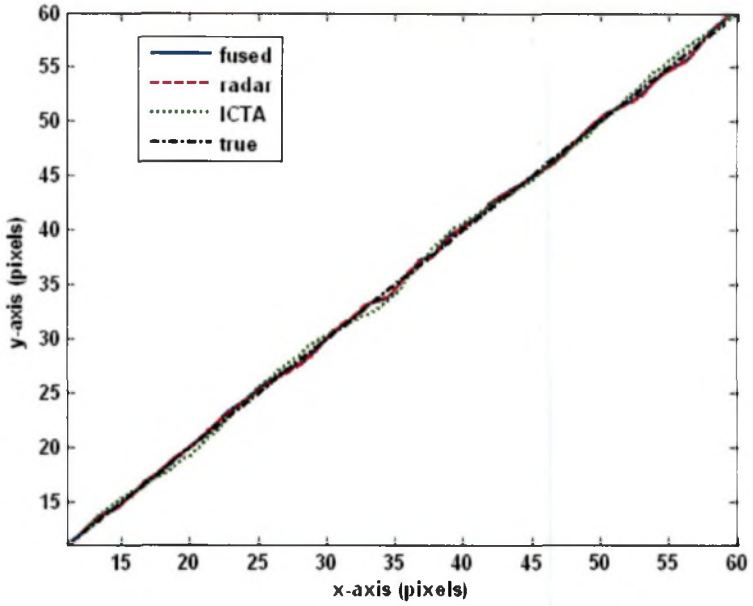


Fig-4.18 Fused trajectory – without measurement loss

A measurement loss in imaging sensor is simulated from 15sec to 25sec and in the ground based radar from 30sec to 45sec. Track extrapolation has been done during these periods. Track deviation can be observed in Fig. 4.19 and Fig-4.20 during these durations.

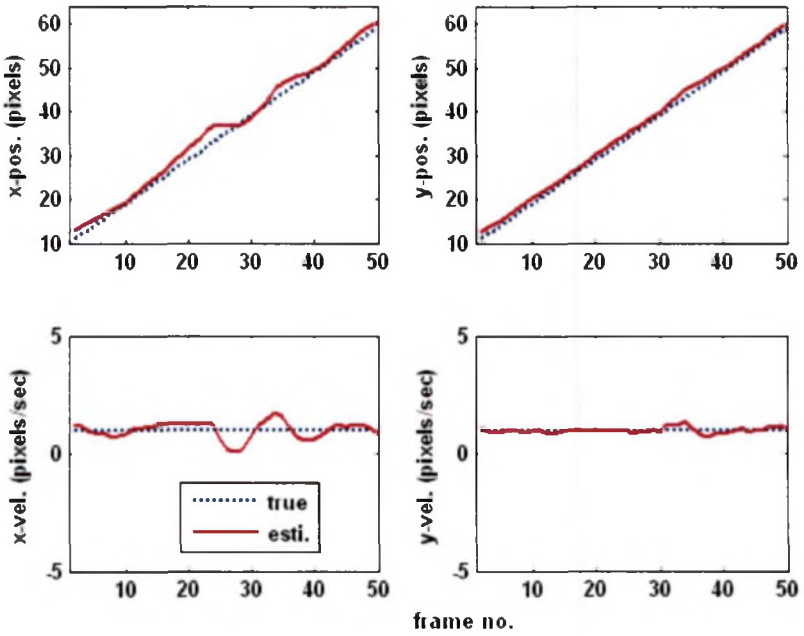


Fig-4.19 True and estimated x-pos., y-pos., x-vel. and y-vel. from imaging sensor with data loss from 15 to 25 frames

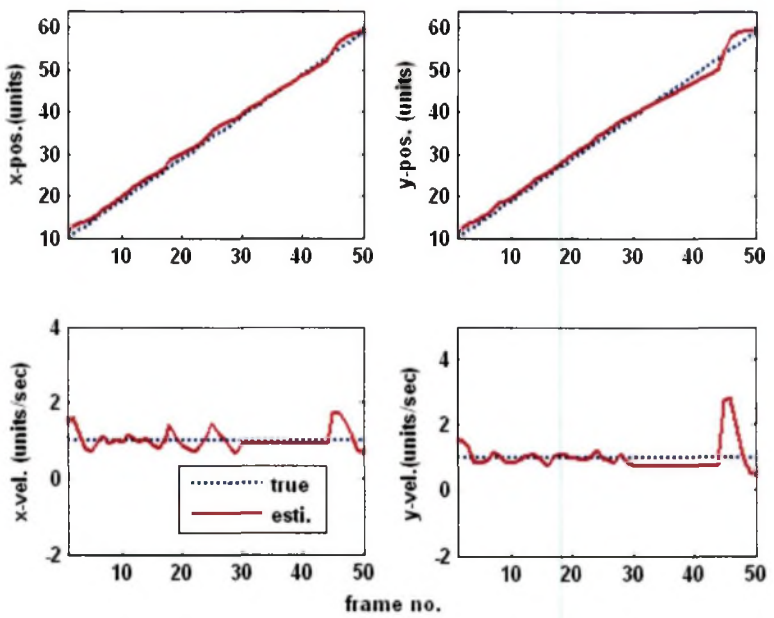


Fig-4.20 True and estimated x-pos., y-pos., x-vel. and y-vel. from ground based radar with data loss from 30 to 45 frames

Percentage fit error in x & y directions and root mean square error in position & velocity for before and after fusion are shown in Table-4.5. From Fig. 4.21 and Table-4.5, it is observed that fusion of data gives better results when there is a measurement loss in either of the sensors thereby demonstrating the robustness and better accuracy achieved because of fusion.

Table-4.4 PFE, RMSPE and RMSVE – with measurement loss

	PFEx	*PFEy*	*RMSPE*	*RMSVE*
Imaging sensor	4.29	2.86	1.984	0.374
radar	2.66	3.59	1.688	0.51
fused	1.31	1.46	0.736	0.478

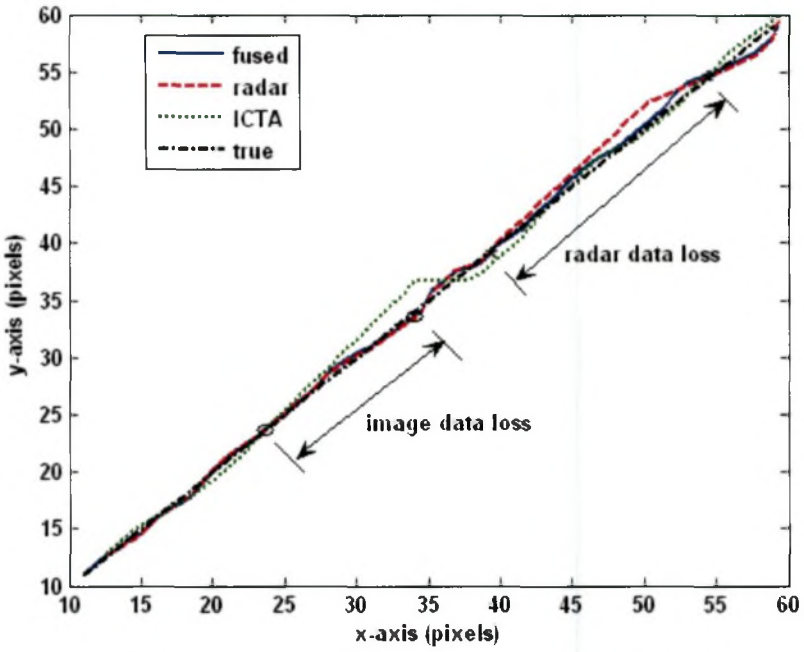


Fig-4.21 Fused trajectory – with measurement loss

4.8 Conclusion

An algorithm called ICTA (Image centroid tracking Algorithm) for tracking a target in Cartesian coordinate system using imaging sensor data is presented. Segmentation & centroid detection algorithm is used to find the target location in the image and gating & data association algorithms viz., NNKF &PDAF are used to track the target. It was shown that PDAF shows better performance even in the presence of clutter and measurement loss. The performance of the proposed algorithm is evaluated using three different sets of simulated data. A procedure for fusing the tracks obtained from imaging (ICTA) and radar sensors is presented. It was concluded that the fusion of tracks obtained from these sensors would give better results.