Individual-Gating-by-Sorting in MHT

Viet Duc Nguyen and Tim Claussen
Digital Signal Processing and System Theory
Christian-Albrechts-Universität zu Kiel
Kiel, Germany
Email: vng@tf.uni-kiel.de, tic@tf.uni-kiel.de

***Abstract*—An efficient individual gating method is described and tested. Compared to other approaches it does not solely calculate distances between the predicted and the actual measurement but checks for overlapping frames centered around them. In contrast to covariance ellipses, these frames are created only by using the main diagonal entries of covariance matrices. The overlap-check is done by sorting algorithms and binary search. Only measurements which pass the overlap-check are further processed by ellipsoid gating. In doing so, not only the computation time itself but also the complexity can be reduced compared to classical ellipsoid or rectangular gating. This Individual-Gating-by-Sorting is integrated into a Multi-Hypothesis tracker and applied to a multistatic sonar dataset. Results show outstanding performances. The gain compared to classical ellipsoid gating in terms of computation even further increases as the amount of data increases.**

I. Introduction

Gating is an essential step within many tracking algorithms. This becomes more and more important as nowadays tracking systems often have to deal with large sensor networks rather than with one or few sensors and hence they have to process more information, contacts/measurements, in particular. Gating prevents unlikely measurement-track associations from being processed by the tracking system. This saves computational resources. Global Nearest Neighbor (GNN), Joint Integrated Probabilistic Density Association (JIPDA), or Multi-Hypothesis Tracking (MHT) are examples of tracking algorithms which have been implemented with gating [1]–[3].

A commonly used gating method is ellipsoid gating: A measurement $\boldsymbol{y}$ is considered to fall within an ellipsoidal gate centered around the predicted measurement $\hat{\boldsymbol{y}}$ if the Mahalanobis distance (see (3)) is below a certain threshold. Ellipsoid gating is considered to be precise [4]. On the other hand, the Mahalanobis distance has to be calculated for each possible pairing of predicted and actual measurement. As such it suffers from a $\mathcal{O}(m \cdot n)$ complexity, with n being the number of measurements and m being the number of predictions. Rectangular gating requires less computational load but it is also less precise [4]. In addition, its time complexity is also $\mathcal{O}(m{\cdot}n)$. In [5], a method called *Gating-by-Sorting* is presented. It is a two-step approach and uses sorting algorithms and binary search to conduct a coarse gating before using ellipsoid gating for fine gating. In doing so, a loglinear complexity $\mathcal{O}((n{+}m)\log(n))$ can be achieved. As a drawback it is *not* an individual gating anymore since it uses fixed gate sizes to achieve the lower computational load. This results in various problems which are explained in Section III-C as well as in [5].

In this paper, *Individual-Gating-by-Sorting* (IGS) is presented. The work here can be seen as an extension of *Gating-by-Sorting* given in [5] as well as of [6]. In contrast to [6], not only gating itself but its interaction within a complete tracking system is investigated. Hence the results can be better utilized for further applications. IGS is implemented within an MHT algorithm and applied on simulated multistatic sonar datasets. These datasets are generated by a multistatic sonar simulator provided by the University of Connecticut [7]. Results are obtained by Monte Carlo simulations and assessed regarding tracking performance metrics and computation time. The performance metrics are given by [8] and widely used in the literature, e.g., in [9]–[12].

The outline is as follows: Section II explains the MHT algorithm. The subsequent section depicts known gating methods as well as their advantages and disadvantages in detail. In Section IV the idea of IGS as well as its implementation are described. Simulation results are given in Section V. The last section provides a conclusion and an outlook.

II. Multi-Hypothesis Tracking

A. Concept

MHT is an approach to track targets in cluttered environments. The main idea of MHT is to create a hypothesis tree of target states. Each hypothesis is weighted according to its probability of representing the actual target state and new hypotheses are spawned by associating existing hypotheses with new contacts. The decision whether a contact used for hypotheses-spawning is target- or clutter-originated can be postponed until more information (e.g., subsequent contacts) is processed. Despite missing contacts or cluttered environments, MHT can calculate target-originated tracks.

Amongst different ways of MHT realization [13] the one implemented in this paper is based on [14]. Hence the following description of MHT refers to the implementation of this work and it is only one of many ways to implement MHT. However, the proposed gating method given in Section IV is not limited to this implementation.

B. Modeling and Filtering

In this work tracking is done in the 2-dimensional Cartesian plane and the *Nearly Constant Velocity* (NCV) model is used [15]. In order to take measurement noise and the kinematic model into account, the Kalman filter [16] is used in the MHT-implementation. If only the target position in Cartesian coordinates is used as measurement data, a linear Kalman filter (LKF) can be used.

In general, MHT can be realized with many other filter methods and kinematic models. E.g., in case of a nonlinear system, one may consider using the extended [16] or the unscented Kalman filter [17], or the particle filter [18].

C. Creating and Managing Hypotheses

Since it is not known *a priori* whether a contact is target-originated or not, all contacts are considered for association with existing hypotheses. So (in the simplest case) each hypothesis state $\hat{\boldsymbol{x}}^i_{k|k-1}$ is associated with each contact $\boldsymbol{y}^j_k$ with $i \in \{1, \ldots, m_{k-1}\}$ and $j \in \{0, \ldots, n_k\}$; n_k is number of contacts and m_{k-1} is the number of hypotheses at time k and time $k-1$, respectively. The preliminary weight of the resulting hypothesis is

$$\hat{\omega}^{ij}_k = \begin{cases} \omega^i_{k-1} \frac{P_D}{f_c} \mathcal{N}(\boldsymbol{y}^j_k; \hat{\boldsymbol{y}}^i_k, \boldsymbol{S}^{ij}_k), & j \neq 0\,, \\ \omega^i_{k-1}(1 - P_D), & j = 0\,. \end{cases} \tag{1}$$

Note that $\hat{\boldsymbol{x}}^i_{k|k-1}$ is the result of the Kalman prediction step, also known as time update step [16], and $\hat{\boldsymbol{y}}^i_k$ denotes the corresponding predicted measurement. $\boldsymbol{S}^{ij}_k$ is the innovation covariance, P_D the probability of detection, and f_c the clutter density. $\mathcal{N}$ denotes the normal distribution. There is a chance that no contact actually belongs to the targets. This case is covered by $j = 0$. In case of multi-sensor measurements, the measurements are processed simultaneously without prior fusion.

Final weights are calculated by normalizing preliminary weights such that

$$\omega^{ij}_k = \frac{\hat{\omega}^{ij}_k}{\sum_{i=1}^{m_{k-1}} \sum_{j=0}^{n_k} \hat{\omega}^{ij}_k}\,. \tag{2}$$

As it can be seen in (1) and (2), the number of hypotheses increases by a factor of $n_k + 1$ at each time step. In order to ensure computational feasibility, the number of hypotheses is reduced using different techniques such as gating, pruning, and merging [19]. Furthermore, track confirmation and deletion is covered by Sequential Track Extraction [20].

III. Known Gating Methods

A. Ellipsoid Gating

A measurement $\boldsymbol{y}$ lies within an ellipsoidal gate if it fulfills the condition

$$[\boldsymbol{y} - \hat{\boldsymbol{y}}]^\top \boldsymbol{S}^{-1} [\boldsymbol{y} - \hat{\boldsymbol{y}}] \leq G \tag{3}$$

with $\hat{\boldsymbol{y}}$ being the predicted measurement and $\boldsymbol{S}$ being the innovation covariance matrix.

The result of the left-hand side of (3) is called Mahalanobis distance. It can be seen as the normalized Euclidean distance between the predicted and the actual measurement, and the normalization is carried out by the innovation covariance $\boldsymbol{S}$. As such, the resulting gate shape and spatial extension is determined by $\boldsymbol{S}$ and G and, in general, it is an ellipse[1]. In this paper the gate size G corresponds to an association gate probability of 99%. This value is commonly found in the literature, e.g., [2], [11].

If a measurement does not fall within the gate, the corresponding measurement-hypothesis association is considered to be too unlikely. Hence further calculations such as measurement update, weight calculation as well as hypothesis merging and pruning do not need to be carried out. This reduces the computational load.

The gating procedure comprises different operations: matrix inversion ($\boldsymbol{S} \rightarrow \boldsymbol{S}^{-1}$), calculating the difference between the predicted and the actual measurement and comparing the distance with G. All these operations need to be carried out for every possible measurement-hypothesis pairing individually. Using the so called *big O notation* $\mathcal{O}(.)$ the complexity of the computation time T_{PEG} for all gating procedures is characterized by

$$T_{\text{PEG}} \in \mathcal{O}(m \cdot n)\,. \tag{4}$$

Ellipsoid gating is a precise gating method [4] but, as can be seen in (4) and considering the necessary matrix inversion, it is also very time consuming. In the following this classical gating method is denoted as "pure" ellipsoid gating (PEG) since this method is solely used. In order to reduce the time consumption, one could use known methods such as rectangular gating or Gating-by-Sorting. These two approaches are explained along with their advantages and disadvantages in the next subsection.

B. Rectangular Gating

In rectangular gating, the gating condition is

$$|y_d - \hat{y}_d| < G_d, \quad d \in \{1, \ldots, n_{\text{dim}}\}\,. \tag{5}$$

y_d and $\hat{y}_d$ are elements of the measurement $\boldsymbol{y}$ and the predicted measurement $\hat{\boldsymbol{y}}$, respectively. d denotes the dimension and n_{dim} is the number of dimensions. G_d is the gate in dimension d. In [4] it is suggested to use the gate

$$G_d = g\sigma_{\boldsymbol{S},d} \tag{6}$$

with $\sigma_{\boldsymbol{S},d}$ being the standard deviation of $\boldsymbol{S}$ of dimension d. g is suggested to be ≥ 3 [4]. As a faster alternative one can set G_d to be fixed.

As no matrix inversion is required and gating can be done for each dimension independently, each rectangular gating procedure can be computed faster and is easier to implement. As a drawback, it is less precise than PEG. Furthermore, it has to be mentioned that, though each rectangular gating procedure takes less time than using PEG, the complexity still remains $\mathcal{O}(m \cdot n)$.

C. Gating-by-Sorting

Gating-by-Sorting (GbS) is a two-step gating method which conducts a coarse but fast gating step before proceeding with precise ellipsoid gating. It has been used in [21] and is further explained in [5].

In principle, the first coarse gating step is similar to rectangular gating, but instead of checking the gating condition given in (5) for each prediction-measurement pairing individually, GbS uses efficient sorting algorithms and binary search to check the condition. This is possible since the coarse gating

[1]In case the innovation covariance matrix is an identity matrix the resulting shape is a circle and the Mahalanobis distance equals the Euclidean distance.

step uses fixed gate sizes. Only those measurements which pass the coarse gate are further gated by ellipsoid gating. In doing so GbS can achieve low computation times. Moreover, the complexity can be reduced to $\mathcal{O}((n+m)\log(n))$. However, the usage of fixed-size gates makes GbS a non-individual gating method since the individual innovation covariance matrix $\boldsymbol{S}$ of each prediction-measurement pairing is not taken into account during the first gating step. Thereby, fixed gate sizes may cause some falsification if the gates are too small and cut the ellipses, and, if they are too big, the efficiency of this method is reduced [5].

In the following, a method for individual Gating-by-Sorting is presented. It resolves the fixed-gate-size issue, thus achieving the precision of PEG, while keeping the complexity at $\mathcal{O}((n+m)\log(n))$.

IV. Individual-Gating-by-Sorting (IGS)

A. Basic Idea

Individual ellipsoid gating is based on the Mahalanobis distance. PEG checks if the Mahalanobis distance between the predicted and the actual measurement is less than a certain bound. This is equivalent to checking if the measurement falls within the ellipse[2] which is centered around the predicted measurement. Considering a frame whose sides[3] are parallel to the coordinates and tangential to the ellipse, it is clear that being inside the frame is a necessary condition for being inside the ellipse. This is shown in Fig. 1.

The frame size or the lengths of the sides are determined by the variances $\{\sigma^2_{\boldsymbol{S},1}, \sigma^2_{\boldsymbol{S},2}, ..., \sigma^2_{\boldsymbol{S},n_{\mathrm{dim}}-1}, \sigma^2_{\boldsymbol{S},n_{\mathrm{dim}}}\}$ of $\boldsymbol{S}$, i.e., its main diagonal entries:

$$\boldsymbol{S} = \begin{bmatrix} \sigma^2_{\boldsymbol{S},1} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \sigma^2_{\boldsymbol{S},n_{\mathrm{dim}}} \end{bmatrix} \tag{7}$$

and by the Gate G or its square root $\sqrt{G}$, respectively [6] (see Fig. 1). In order for a measurement to be within the frame, the requirement

$$\Delta_d < \sqrt{G} \cdot \sigma_{\boldsymbol{S},d}\,, \quad d \in \{1, \dots, n_{\mathrm{dim}}\} \tag{8}$$

with

$$\Delta_d := |y_d - \hat{y}_d| \tag{9}$$

must be met. Note that (8) is similar to (5) but in case of IGS the value G is already determined by the association probability which also defines the size of the ellipse. Furthermore, assuming additive noise, the innovation covariance $\boldsymbol{S}$ is

$$\boldsymbol{S} = \hat{\boldsymbol{C}} + \boldsymbol{C} \tag{10}$$

with $\hat{\boldsymbol{C}}$ being the covariance matrix of the prediction and $\boldsymbol{C}$ the covariance matrix of the measurement. For example, in case of the LKF the innovation covariance would be

$$\boldsymbol{S} = \underbrace{\boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^\top}_{\hat{\boldsymbol{C}}} + \underbrace{\boldsymbol{R}}_{\boldsymbol{C}} \tag{11}$$

with $\boldsymbol{H}$ being the measurement matrix, $\boldsymbol{R}$ being the measurement covariance, and $\boldsymbol{P}$ being the *a priori* state covariance. $\hat{\boldsymbol{C}}$ and $\boldsymbol{C}$ in combination with $\sqrt{G}$ also form ellipses centered around the prediction and the measurement, respectively. In addition one can also form circumscribing frames around these ellipses as it is done in case of $\boldsymbol{S}$. If these two "smaller" frames do not overlap in a certain dimension d, the following inequation applies:

$$\Delta_d > \hat{\sigma}_d\sqrt{G} + \sigma_d\sqrt{G}\,. \tag{12}$$

$\hat{\sigma}_d$ and σ_d denote the standard deviations given by the main diagonal entries of $\hat{\boldsymbol{C}}$ and $\boldsymbol{C}$, respectively. Furthermore, not overlapping in one dimension automatically results in not overlapping at all. Considering (7) and (10) we get

$$\sigma^2_{\boldsymbol{S},d} = \hat{\sigma}^2_d + \sigma^2_d \tag{13}$$

$$\Longrightarrow \sigma^2_{\boldsymbol{S},d} < \hat{\sigma}^2_d + 2\hat{\sigma}_d\sigma_d + \sigma^2_d \tag{14}$$

$$< (\hat{\sigma}_d + \sigma_d)^2 \tag{15}$$

and so

$$\sigma_{\boldsymbol{S},d} < \hat{\sigma}_d + \sigma_d\,. \tag{16}$$

These inequations and (12) result in

$$\Delta_d > \sigma_{\boldsymbol{S},d}\sqrt{G}\,. \tag{17}$$

Hence one can make the following implications:

The "smaller" frames do not overlap in all dimensions.
⇒ These frames do not overlap at all.
⇒ Measurement is not within the $\boldsymbol{S}$-frame.
⇒ Measurement does not fall within the $\boldsymbol{S}$-ellipse.

It needs to be kept in mind that these implications hold only one way. Overlapping frames is *not a sufficient* condition. But it is a *necessary* condition for the measurements being in the $\boldsymbol{S}$-gate.

Considering that, in many tracking scenarios, the variances and hence also the frames are small compared to the complete tracking area [5], [21], the number of contacts whose frames overlap with the frames of a predicted measurement will be much smaller than the total number of contacts. So, applying PEG only on these contacts will require much less time than applying PEG on all contacts. In addition, as the frame-overlap condition is only a necessary but not sufficient condition it will never filter out contacts which would not be filtered out by PEG anyway. So it does not suffer from the fixed-gate-size problems. Summarized, Individual-Gating-by-Sorting conducts an individual and coarse gating step by checking for overlapping frames followed by classical ellipsoid gating. In the following subsection the implementation of IGS is presented, including a quick way to check whether the two small frames overlap.

B. Implementation of IGS

As pointed out in Section IV-A, checking for frame-overlap can be done dimensionwise. So, the following steps are to be conducted for each dimension: Firstly, one needs to sort the set of measurements along each dimension. After that the relative position of a predicted measurement within the sorted list of measurements needs to be found as it is illustrated

[2]In multi-dimensional cases it is actually an ellipsoid. For better understanding the term ellipse is used here since visualizations are 2-dimensional.

[3]In multi-dimensional cases these are hyperplanes.

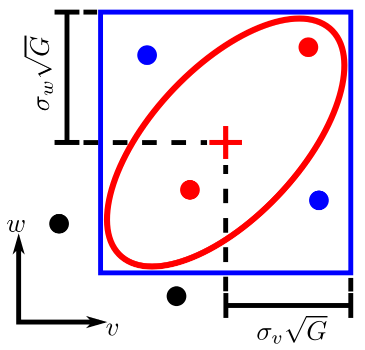

Figure 1: Visualization of an ellipsoidal gate and its circumscribing frame. It completely covers the ellipse. Hence no measurements (dots) which are outside the frame can ever be inside the ellipse.

in Fig. 2. This is carried out by means of binary search [22]. For all measurements whose positions are below the predicted measurement position, one needs to check if the upper side of the measurement-frame is above the lower side of the prediction-frame, i.e.,

$$y_d + \sigma_d\sqrt{G} > \hat{y}_d - \hat{\sigma}_d\sqrt{G}\,, \tag{18}$$

and for the measurements whose positions are above the predicted measurement position (see Fig. 2):

$$y_d - \sigma_d\sqrt{G} < \hat{y}_d + \hat{\sigma}_d\sqrt{G}\,. \tag{19}$$

Beginning at the position of the predicted measurement, this checking procedure is done sequentially downwards until the measurements do *not* meet the requirement

$$y_d + \max_i\left(\sigma_{i,d}\right)\sqrt{G} + \hat{\sigma}_d\sqrt{G} > \hat{y}_d \tag{20}$$

and upwards until the measurements do *not* meet the requirement

$$y_d - \max_i\left(\sigma_{i,d}\right)\sqrt{G} - \hat{\sigma}_d\sqrt{G} < \hat{y}_d \tag{21}$$

anymore; $\max_i\left(\sigma_{i,d}\right)$ denotes the largest variance of all measurements in dimension d.

After the frame-overlap checking has been conducted in all dimensions, results are lists of measurements who fulfill the conditions (18) or (19) for the corresponding dimensions. Hence the intersection set of these lists contains only measurements whose frames overlap in all dimensions with the frame of the predicted measurement, i.e., they actually overlap. Only these few remaining measurements are finally gated using PEG.

As an alternative one could check the frame-overlap stepwise. I.e., frame-overlap is checked for one dimension and only the remaining contacts are then checked regarding the next dimension. Although this seems to be a very fast method the implementation is more complex and the additional program-operations may need extra computation time. In [6] not sorting and binary search but multidimensional search trees are used. This method requires a lot more effort in programming since search trees are not part of the standard library of the programming language C++ used in this work.

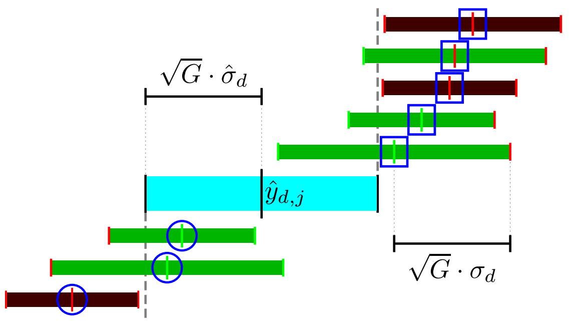

Figure 2: Illustration of frame overlapping. The broad blue bar visualizes the $\pm\sqrt{G}\hat{\sigma}_d$-range of the predicted measurement j around its position $\hat{y}_{d,j}$ in dimension d. Circles mark the positions of measurements which are below the position of the predicted measurement and rectangles mark the position of the measurements which are above. Dark red thin bars represent the $\pm\sqrt{G}\sigma$-range of measurements whose frames do not overlap with the frame of the predicted measurement in dimension d. Bright green bars represent measurements which overlap in this dimension.

C. Complexity Analysis

As mentioned in [5], a precise prediction of the computation time is hardly possible since it depends on the performance of the computer, the exact way of implementation, the optimization capability of the compiler, and many more things. Though, an approximate analysis of the computation time and the complexity is feasible.

The sorting algorithm used in this work is given by the C++ Standard-Template-Library and its average computation time $T_{\text{sort}}(n)$ is

$$T_{\text{sort}}(n) \in \mathcal{O}(n\ln(n)) \tag{22}$$

with n being the number of elements to be sorted [23], i.e., in this work the number of measurements. The computation time $T_{\text{search}}(n)$ of binary search is

$$T_{\text{search}}(n) \in \mathcal{O}(\ln(n)) \tag{23}$$

with n being the number of elements of the sorted list through which is searched. Considering that searching has to be conducted for each of the m predicted measurements the total computation time T_{tot} is

$$T_{\text{tot}}(n) \in \mathcal{O}((m+n)\ln(n))\,. \tag{24}$$

So the IGS computation time can be assumed to have a loglinear complexity which is an "improvement" to PEG's complexity of $\mathcal{O}(m\cdot n)$. It has the same complexity as GbS but is does not suffer from the fixed-gate-size problems (see Section III-C). Furthermore, the complexity analysis indicates that the benefit of IGS increases as the number of contacts increases. This behavior is investigated in Section V-B.

Calculating the intersection set as mentioned in Section IV-B as well as checking for the exit conditions (20) and (21) require time as well. But taking into account that the variances are small compared to the tracking area (see Section IV-A), these operations will not gain too much weight.

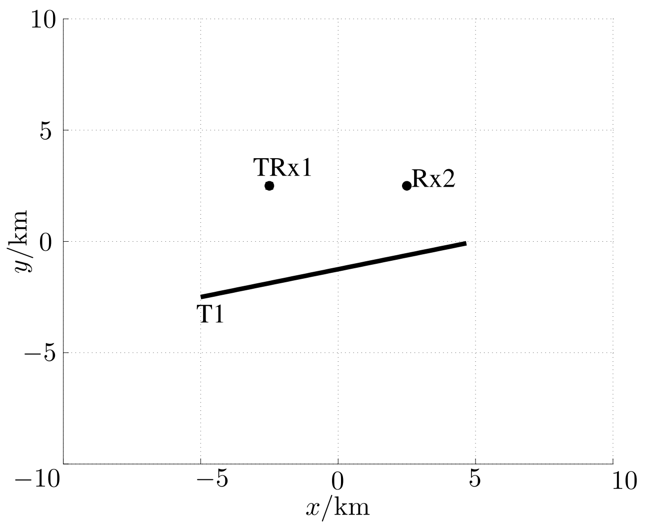

Figure 3: Setup of scenario 1. There is one transceiver (TRx1) and one additional receiver (Rx2). The target is denoted by T1.

V. Simulation and Results

GbS has been tested in [21] on a real radar dataset recorded at a sea trial in 2011 [24]. One disadvantage of real datasets is that some performance metrics given in [8] cannot be obtained. E.g., since the sea area was not controlled, the number of false tracks can hardly be quantified. In this work, the simulation data is generated by the multistatic sonar simulator of the University of Connecticut [7]. This enables the possibility to precisely obtain tracking performance metrics as well as running Monte Carlo simulations. Furthermore, compared to [5] not MATLAB-script language but the compiled programming language C++ is used to implement the algorithms. Hence the computation-time-results are more useful for further considerations in real-time applications or fast simulations in the presence of high amount of data.

The simulations are divided into two parts. The first part serves as a verification for the reliability of the presented tracking algorithm and especially of IGS. The second part investigates only the computation time of PEG compared to IGS with respect to the number of contacts.

The hardware used for the simulations is a standard personal computer. Its specifications and other development information are:

- Intel XEON CPU, 2.8 GHz (4 cores, no multicore usage programmed),
- Memory: 24 GB, 1333 MHz,
- Windows 7 Professional Edition,
- Microsoft Visual Studio 2012,
- C++ Standard Template Library, Boost (uBlas).

A. *Reliability of IGS*

In this part of the simulation, MHT is applied on a multistatic sonar dataset, once using IGS and once using PEG.

Table I: Simulation results in terms of tracking performance metrics.

	$\mathrm{TP_D}$	RMSE/m	$\mathrm{TFAR/s^{-1}} \times 1000$
Scenario 1, PEG	0.82	475	1.072
Scenario 1, IGS	0.82	475	1.072

Table II: Statistics of the computation time of 100 Monte Carlo simulations. The values refer either to the gating procedure only or to the complete tracking procedure. The upper and lower adjacent delimit the 99.3% coverage. All units are milliseconds (except for the number of outliers).

	Gating		**Tracking**	
	IGS	PEG	IGS	PEG
Mean	11.2	41.2	53.4	86.3
Standard Deviation	1.2	5.2	4.6	9.2
Min	9.4	32.6	45.0	70.0
Max	17.0	57.9	66.0	115.1
25th Percentile	10.8	38.1	49.7	80.1
Median	11.7	40.9	53.2	85.0
75th Percentile	12.5	44.8	56.0	91.3
Lower Adjacent	9.4	32.6	45.0	70.0
Upper Adjacent	14.8	54.7	65.2	106.8
#Outliers	1	3	1	3

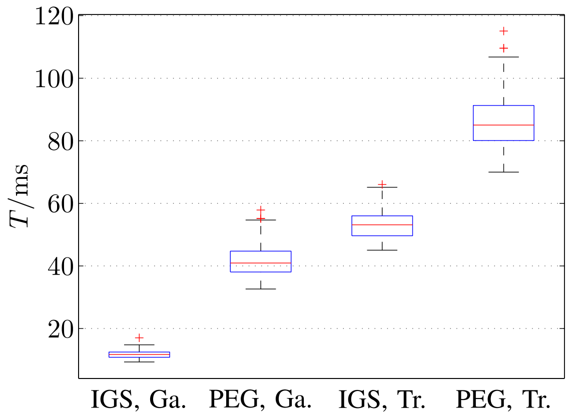

Figure 4: Boxplots for the computation time. Ga. denotes the computation time for the gating procedures only and Tr. stands for the complete tracking simulation (one run).

The main objective is to show that IGS and PEG actually create the same results. The simulation scenario is the predefined scenario 1 (K-distributed clutter) given in [25]. Scenario 1 features a single target moving in a straight line. It lasts for 3480 s (59 pings) and its configuration is shown in Fig. 3. For the Monte Carlo simulations, 100 datasets of this scenario are created using the multistatic sonar simulator.

1) Numerical Results: Three performance metrics were chosen: First, the track probability of detection $\mathrm{TP_D}$ which is the ratio of the total duration of all true tracks and the total scenario duration; second, the track false-alarm rate TFAR which is the number of false tracks normalized by the duration of the scenario; third, the aberration between the calculated true tracks and the ground truth given by the root-mean-square error (RMSE).

As expected and shown in Tab. I, the tracking results are the same. In fact, the complete tracking result (each track, each Monte Carlo run) is identical for both gating methods. The $\mathrm{TP_D}$ is 0.82. The RMSE is $475\,\mathrm{m}$ and the TFAR is $1.072 \cdot 10^{-3}\,\mathrm{s}^{-1}$. Although the actual performance is not as relevant as the validation that the performances of both gating methods are identical, one has the opportunity to compare the results of this scenario with those given in the literature, e.g., [12].

The results of the computation time are shown in Tab. II. The table shows the average computation time for a complete scenario run. When using PEG, the gating procedure takes in average $41.2\,\mathrm{ms}$ (milliseconds) whereas when using IGS, gating requires only $11.2\,\mathrm{ms}$. So, in average, IGS is 3.7 times faster than PEG. The complete tracking procedure is about 1.6 times faster. The main reason is that ellipsoid gating steps are conducted 46398 times when using PEG but only 2706 times when using IGS. Fig. 4 visualizes the results of Tab. II by using boxplots. It clearly shows the huge difference in computation time between the two gating methods. One can also see that tracking using IGS needs hardly more time than just gating using PEG. Furthermore, the boxplots visualize the relative low statistical spread of the results.

2) Summary: In this first simulation IGS has been proven to run many times faster than PEG while having no negative influence on the tracking result. In fact they are identical. The main reason for the acceleration is the frame-overlap check using sorting and binary search. This pre-gating step filters out quickly all contacts which do not satisfy the necessary condition of overlapping-frames. After that, the few remaining contacts can quickly be gated using ellipsoid gating. Since it is only a necessary condition, it is clear that IGS cannot alter the tracking result compared to PEG.

B. Computation Time of IGS

In the previous simulation the tracking environment has only low clutter and the number of contacts per ping is approximately only between 10 and 20. In order to further investigate the advantage of IGS regarding the computation time, IGS is now tested with respect to different numbers of contacts. The scenario remains identical, but it runs with Rayleigh-distributed clutter in a highly-cluttered environment [12]. The numbers of contacts per ping are set to $\{10, 20, \ldots, 90, 100\}$ in each simulation series. Again, each series comprises 100 Monte Carlo simulation runs. The number of contacts refers to those with the strongest SNR. The computer hardware remains the same.

1) Numerical Results: The results in computation time are shown in Tab. III, Tab. V, and Tab. IV, and they are visualized in Fig. 5, Fig. 6, and Fig. 8. In Tab. III, the mean computation times for the sum of all gating procedures within one simulation run are given and in Tab. IV the mean computation times for one complete tracking simulation are shown. Since gating and tracking using IGS is always faster than using PEG, the difference Δ_T is defined as

$$\Delta_T := T_{\mathrm{PEG}} - T_{\mathrm{IGS}} \tag{25}$$

Table III: Computation time statistics for the sum of all gating procedures within one run. $T_{\mathrm{IGS,Ga}}$ and $T_{\mathrm{PEG,Ga}}$ are mean values (one simulation run) and given in milliseconds. Δ_T denotes the difference and a is the acceleration.

#Contacts	$T_{\mathrm{IGS,Ga}}$	$T_{\mathrm{PEG,Ga}}$	Δ_T	a
10	3.5	14.8	11.3	4.2
20	11.6	69.3	57.7	6.0
30	25.0	194.0	169.0	7.8
40	46.6	422.4	375.9	9.1
50	80.4	787.0	706.6	9.8
60	126.0	1318.0	1192.1	10.5
70	185.8	2028.1	1842.3	10.9
80	255.6	2937.0	2681.4	11.5
90	331.2	3990.7	3659.5	12.0
100	417.9	5194.6	4776.6	12.4

Table IV: Computation time statistics for one complete tracking simulation. $T_{\mathrm{IGS,Tr}}$ and $T_{\mathrm{PEG,Tr}}$ are mean values (one simulation run) and given in milliseconds. Δ_T denotes the difference and a is the acceleration.

#Contacts	$T_{\mathrm{IGS,Tr}}$	$T_{\mathrm{PEG,Tr}}$	Δ_T	a
10	29.3	41.2	12.0	1.4
20	79.0	140.7	61.6	1.8
30	149.6	322.6	173.0	2.2
40	246.5	634.7	388.2	2.6
50	380.9	1112.2	731.2	2.9
60	563.7	1786.1	1222.4	3.2
70	780.9	2675.6	1894.7	3.4
80	1051.4	3808.2	2756.9	3.6
90	1338.4	5100.5	3762.1	3.8
100	1655.0	6582.0	4927.0	4.0

Table V: Mean number of ellipsoid gating procedures per simulation run. Δ_N is the absolute and a_N is the relative difference.

#Contacts	N_{IGS}	N_{PEG}	Δ_N	a_N
10	1060.7	18050.4	16989.7	17.0
20	3896.6	84236.4	80339.8	21.6
30	9240.3	235130.1	225889.8	25.4
40	17590.9	515273.2	497682.3	29.3
50	29188.2	959319.5	930131.3	32.9
60	44240.1	1601840.4	1557600.3	36.2
70	62724.1	2456672.4	2393948.4	39.2
80	84856.5	3546844.0	3461987.5	41.8
90	109455.1	4812201.0	4702745.9	44.0
100	136908.0	6262032.0	6125124.0	45.7

and the acceleration a is defined as

$$a := \frac{T_{\mathrm{PEG}}}{T_{\mathrm{IGS}}} \,. \tag{26}$$

The computation times for tracking as well as for gating are illustrated in Fig. 5. It is clearly visible that gating using IGS is much faster than using PEG. In fact, using PEG with more than 20 contacts per ping, the gating procedures themselves already need more time than the complete tracking procedure using IGS. Only for the lowest number of contacts (i.e., 10),

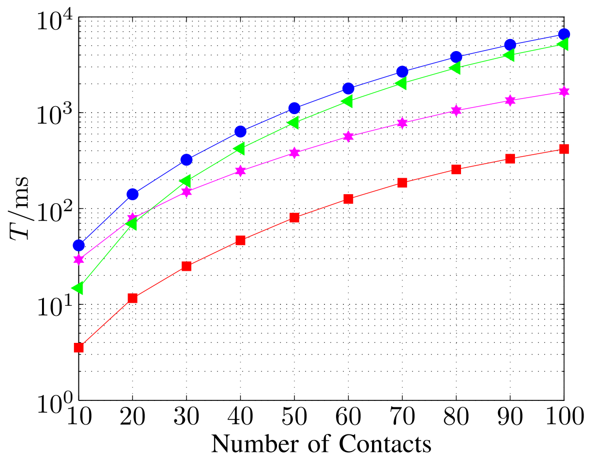

Figure 5: Computation time for one scenario run with respect to the number of contacts. The figure shows the mean computation time for all gating procedures using PEG (◄) and IGS (■) as well as for a complete tracking procedure using PEG (●) and IGS (★). Note that the time axis is logarithmic.

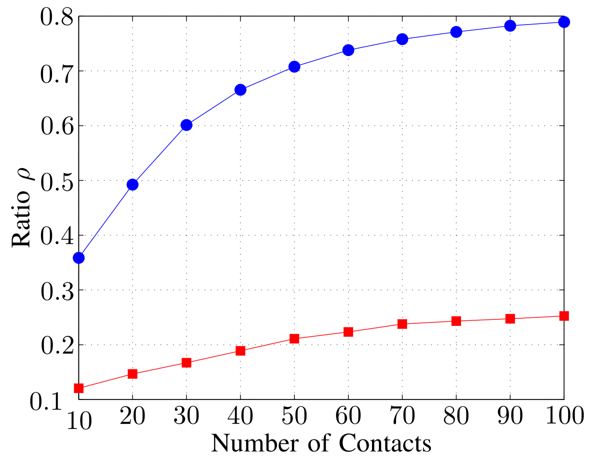

Figure 6: Ratio of the computation time of gating procedures to the complete tracking procedure. The blue curve (●) shows the PEG-related ratio and the red curve (■) the IGS-ratio.

PEG-gating is faster than IGS-tracking. Furthermore, the acceleration increases when the number of contacts increases. When 10 contacts are processed per ping, the acceleration of the gating procedures is 4.2 and that of the complete tracking procedure 1.4. But at 100 contacts the acceleration increases up to 12.4 and 4.0, respectively. In Tab. V the mean numbers of ellipsoid gating procedures of one simulation run are shown and they are visualized in Fig. 7. It clearly shows that the first gating step using frame overlap already sorts out most of the unlikely associations. Hence in IGS ellipsoid gating is carried out fewer times than in PEG. When simulating with 10 contacts per ping, IGS reduces the number by a factor of 17. Using 100 contacts it is even over 45 times fewer. Fig. 6 shows the ratio ρ of the computation time of gating procedures to that of the

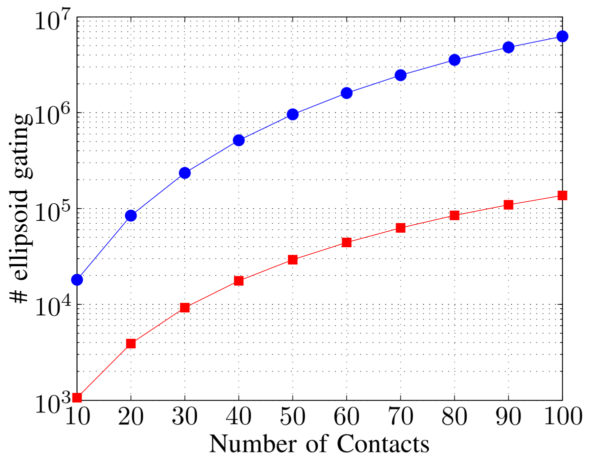

Figure 7: Mean number of ellipsoid gating procedures of one simulation run. The blue curve (●) represents the result using PEG and red curve (■) shows the result for IGS. Note that the #-axis is logarithmic.

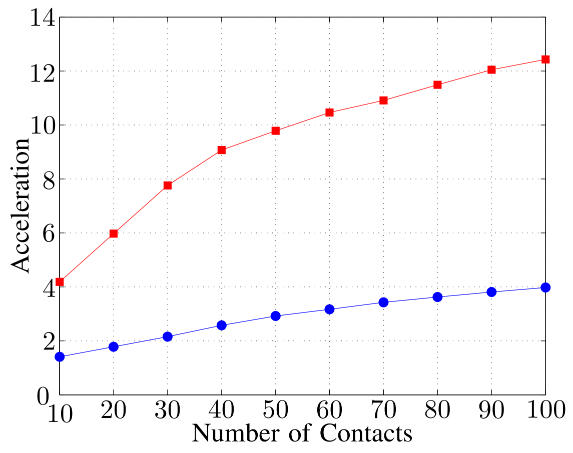

Figure 8: Mean acceleration of IGS compared to PEG for one simulation run. The blue curve (●) shows the acceleration for a complete tracking procedure and the red curve (■) for the gating procedures.

complete tracking procedure:

$$\rho := \frac{T_{\text{Gating}}}{T_{\text{Tracking}}} \left(:= \frac{T_{\text{Ga}}}{T_{\text{Tr}}} \right). \tag{27}$$

Using PEG the ratio starts at about 36 % and increases to almost 80 %. When using IGS the ratio ranges only from 12 % to 25 %.

2) Summary: In this second simulation, IGS has been investigated with respect to the number of contacts per ping. The results comply with those of the first part of the simulations (see Section V-A) in which IGS was faster than PEG. In this part it could additionally be shown that IGS becomes the more efficient compared to PEG the more contacts are involved. Not only the absolute difference between the two gating methods increases but also the relative difference, i.e., the acceleration, increases as the number of contacts increases. This makes

IGS suitable for scenarios with high numbers of contacts, in particular. In addition, the analysis of the ratio ρ indicates that gating becomes a bottleneck in efficiency in the tracking algorithm as the number of contacts increases. This bottleneck is overcome by using IGS.

VI. Conclusion and Outlook

In this paper, an approach for a low-complexity individual gating method, namely *Individual-Gating-by-Sorting* is presented. The main idea of IGS is to transform a distance-measurement problem into a frame-overlap problem. This can be solved efficiently using sorting algorithms and binary search. After this first step, only very few contacts remain, and they can be quickly gated using ellipsoid gating. Since the frames completely surround the ellipses, the frame-overlap condition actually cannot alter the result. IGS is integrated into an MHT-algorithm and applied to a multistatic sonar dataset of the University of Connecticut and analyzed in terms of reliability and computation. It has been shown that IGS accelerates the gating process significantly without altering the actual tracking result compared to classical ellipsoid gating. Furthermore, the benefit of IGS increases as the number of contacts increases.

These first results show that IGS is a promising approach for individual gating in the presence of a high amount of data. In order to further investigate its suitability, IGS should be applied to more scenarios. In these scenarios, one should further increase the number of contacts. However, not only the number of contacts, but also the contact density should be taken into account. I.e., one needs to know what happens when the number of contacts increases and the size of the tracking area increases proportionally (contact density remains equal) as well as what happens when the number of contacts increases and the size of the tracking area remains equal (contact density increases). Additionally, the benefit of IGS in combination with other tracking algorithms such as PHD or JIPDA could be investigated. Finally, IGS should be applied not only to simulated but also to real data sets to prove its suitability for real-life applications.

Acknowledgment

The authors are thankful to Prof. Peter Willett for providing his multistatic sonar simulator. Furthermore, the authors would like to thank Christoph Norrenbrock for helpful discussions and support. This work was supported by the Federal Ministry of Economics and Technology (Germany).

References

[1] P. Konstantinova, A. Udvarev, and T. Semerdjiev, "A Study of a Target Tracking Algorithm Using Global Nearest Neighbor Approach," in *Proceedings of the 4th International Conference on Computer Systems and Technologies: e-Learning*, ser. CompSysTech '03. New York, NY, USA: ACM, 2003, pp. 290–295.

[2] M. Munz, K. Dietmayer, and M. Mahlisch, "A Sensor Independent Probabilistic Fusion System for Driver Assistance Systems," in *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, Oct. 2009, pp. 438–443.

[3] K. Seget, A. Schulz, and U. Heute, "Multi-Hypothesis Tracking and Fusion Techniques for Multistatic Active Sonar Systems," in *Proceedings of the 13th International Conference on Information Fusion*, Edinburgh, Scotland, Jul. 2010, pp. 1–8.

[4] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, ser. Artech House radar library. Artech House, 1999.

[5] V. D. Nguyen and T. Claussen, "Reducing Computational Complexity of Gating Procedures Using Sorting Algorithms," in *Information Fusion (FUSION), 2013 16th International Conference on*, July 2013, pp. 1707–1713.

[6] J. Collins and J. Uhlmann, "Efficient Gating in Data Association with Multivariate Gaussian Distributed States," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 28, no. 3, pp. 909–916, Jul 1992.

[7] P. Willett. (2014, March) Department of Electrical & Computer Engineering, University of Connecticut Website. Storrs, CT, USA. [Online]. Available: http://www.engr.uconn.edu/~willett/current_papers/

[8] S. Coraluppi, D. Grimmett, and P. de Theije, "Benchmark Evaluation of Multistatic Trackers," in *Proceedings of the 9th International Conference on Information Fusion*, Florence, Italy, Jul. 2006, pp. 1 –7.

[9] K. Pikora and F. Ehlers, "Analysis of the FKIE Passive Radar Data Set with GMPHD and GMCPHD," in *Information Fusion (FUSION), 2013 16th International Conference on*, July 2013, pp. 272–279.

[10] K. Wilkens, V. D. Nguyen, and U. Heute, "Adaptive Clutter Density in Multi-Hypothesis Tracking," in *Proc. IEEE ISIF GI Workshop Sensor Data Fusion 2011, Berlin, Germany*, 2011.

[11] D. Grimmett, "SPECSweb Multistatic Tracking on a Truth-Blind Simulated Scenario of the MSTWG," in *Information Fusion, 2009. FUSION '09. 12th International Conference on*, Jul. 2009, pp. 1568 –1575.

[12] S. Schoenecker, P. Willett, and Y. Bar-Shalom, "Comparing Multitarget Multisensor ML-PMHT with ML-PDA for VLO Targets," in *Information Fusion (FUSION), 2013 16th International Conference on*, July 2013, pp. 250–257.

[13] Y. Bar-Shalom, X. Tian, and P. K. Willet, *Tracking and Data Fusion: A Handbook of Algorithms*. YBS Publishing, 2011.

[14] W. Koch, J. Koller, and M. Ulmke, "Ground Target Tracking and Road Map Extraction," *ISPRS Journal of Photogrammetry & Remote Sensing*, vol. 61, no. 3–4, pp. 197–208, 2006.

[15] Y. Bar-Shalom and X.-R. Li, *Estimation and Tracking: Principles, Techniques, and Software*. Norwood: ARTECH HOUSE, INC., 1993.

[16] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," UNC Chapel Hill, Tech. Rep., 2006.

[17] S. Julier and J. K. Uhlmann, "Unscented Filtering and Nonlinear Estimation," in *Proceedings of the IEEE*, vol. 92, 2004, pp. 401–422.

[18] A. Doucet, N. De Freitas, and N. Gordon, Eds., *Sequential Monte Carlo methods in practice*, 2001. [Online]. Available: http://www.worldcatlibraries.org/wcpa/top3mset/839aaf32b6957a10a19afeb4da09e526.html

[19] M. Daun and F. Ehlers, "Tracking Algorithms for Multistatic Sonar Systems," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, pp. 35:1–35:28, Feb. 2010.

[20] G. van Keuk, "Sequential Track Extraction," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 34, no. 4, pp. 1135–1148, Oct. 1998.

[21] V. D. Nguyen, "Small-Target Radar Detection and Tracking within the Pitas Hard-and Software Environment," in *Future Security*, ser. Communications in Computer and Information Science, N. Aschenbruck, P. Martini, M. Meier, and J. Tölle, Eds., vol. 318. Springer, 2012, pp. 347–358.

[22] R. Sedgewick, *Algorithms in C - parts 1-4: fundamentals, data structures, sorting, searching (3. ed.)*. Addison-Wesley-Longman, 1998.

[23] ——, *Algorithms in C*. Addison-Wesley, 1990.

[24] B. Culik, T. Lehmann, and C. Zebermann, "PITAS: Pirate and Terrorist Aversion System," in *Future Security*, ser. Communications in Computer and Information Science, N. Aschenbruck, P. Martini, M. Meier, and J. Tölle, Eds., vol. 318. Springer, 2012, pp. 337–346.

[25] S. Schoenecker, P. Willett, and Y. Bar-Shalom, "A Comparison of the ML-PDA and the ML-PMHT Algorithms," in *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, July 2011, pp. 1–8.

[26] N. Aschenbruck, P. Martini, M. Meier, and J. Tölle, Eds., *Future Security - 7th Security Research Conference, Future Security 2012, Bonn, Germany, September 4-6, 2012. Proceedings*, ser. Communications in Computer and Information Science, vol. 318. Springer, 2012.