Enhancing Predicted Distributions for Constant Acceleration and Turn Rate Motion Models: A Deep Learning Approach

Ole Halvard Sætran, Sigmund Rolfsjord
Norwegian Defence Research Establishment
Kjeller, Norway
Ole-Halvard.Satran@ffi.no, Sigmund-Johannes-Ljosvoll.Rolfsjord@ffi.no

***Abstract*—Gating and association are key components of target tracking, most commonly using the predicted distribution of the track to evaluate new measurements. While a Gaussian predicted distribution is widely used, it is not optimal for some groups of targets, such as fixed-wing aircraft and surface vessels. This article introduces the Constant Acceleration and Turn Rate - Neural Network (CAT-NN) method, which uses a neural network to calculate the predicted distribution for such targets. Our work can be seen as an expansion of the CAT distribution by correcting for track uncertainty, thereby unlocking CAT distributions for a much wider range of tracking applications.**

Simulations show that the CAT-NN predicted distribution is a better match for the true predicted distribution than both the CAT and Gaussian distributions for a fixed-wing aircraft. It also outperforms the Gaussian distribution in a simulation tracking scenario with a single fixed-wing aircraft in heavy clutter.

The CAT-NN model is runtime efficient and implemented as a custom hypothesiser component for the Stone Soup tracking framework.

***Index Terms*—association, deep learning, gating, probability distribution, stone soup, target tracking**

I. Introduction

Multitarget tracking can be broken down into three components: Detection, association and estimation. Detection is the task of obtaining a set of measurements, using one or more sensors. Association is the task of assigning measurements to targets or concluding that the measurement is a false alarm or clutter. Estimation is the task of estimating the state of a target based on the measurements assigned to it and the predicted state. This article focuses on association and the related task of gating.

Association can be done in many different ways, from the simple SNN (Suboptimal Nearest Neighbor) and GNN (Global Nearest Neighbor) [1] methods, to more complicated methods such as JIPDA [2] and MHT [3], to the most recent developments in tracking using Random Finite Sets [4]. All these association methods need to rank the potential track-to-measurement associations with a numeric value for the association algorithm to work. This is usually done with a distance metric that measures the distance between the predicted distribution of the track and the measurement.

One such distance metric commonly used in gating and association is the Mahalanobis distance [5]. This is the distance from one point to a distribution. To calculate the Mahalanobis distance between the measurement, which is itself a distribution, and the predicted distribution, we treat the measurement as a point and replace the covariance of the predicted distribution with the innovation covariance. While the form of the gate is elliptical, the size needs to be decided. Wang et al. [6] use the performance of the filter in calculating the gate size, while Duan et al. [7] use a forecast of the residual error. Gates consisting of overlapping Gaussians, for instance in the case of IMM filters, have been studied among others by Wang et al. [8].

While Gaussian predicted distributions are easy to use and robust, they are not always optimal. This might be due to the non-linearity of the measurement or the target motion model. Bailey et al. [9] use Monte Carlo simulations to find the gate for a non-gaussian PDF using a bearing-only measurement model. Singh et al. [10] propose a set of gates, amongst which are ”wild maneuver gates” that are clearly non-Gaussian, meant for highly maneuverable airborne targets. And Gade et al. [11] introduce a non-ellipsoidal gate for ship-to-measurement association.

The Constant Acceleration and Turn rate (CAT) distribution was introduced by Gade et al. in [12], inspired by their previous work [11]. It is based on the assumption that the target has a constant along-track acceleration and a constant turn rate in each scan interval. It also assumes that the along-track acceleration and the turn rate are Gaussian distributions. This is a reasonable assumption for certain categories of targets, such as fixed-wing aircraft or ships, if the time between measurements is not too long. The CAT distribution is banana-shaped, with the width and curvature of the banana being decided by the potential acceleration and turn rate of the target. Gade et al. showed in [12] that in simulations using a zig-zag vehicle trajectory, the CAT distribution outperforms Gaussian distributions.

The CAT distribution is calculated using the initial velocity of the tracked target, as well as standard deviations for the acceleration and turn rate. These standard deviations are given as parameter inputs to the tracker. It does not consider the uncertainty of the tracked target state, i.e. the covariance matrix of the track. In this paper the CAT-NN distribution is presented, which includes this uncertainty.

The CAT distribution is calculated analytically. This calcu-

lation is not easily expandable to include track uncertainty. We propose to use neural networks to estimate the target probability density functions used for association and gating. Artificial neural networks (NN) have successfully been used to estimate different probability distributions [13], [14], and have demonstrated the ability to learn complex shapes, e.g. in [15]. Learned density functions can both give faster inference and a more flexible framework to incorporate different motion models.

There are several papers that use NN for association [16]–[19], but these approaches solve association directly with neural networks instead of learning spatial density functions. While these methods can learn advanced association functions, they require large labelled datasets for each tracking scenario. NN are also used for motion prediction [20]–[23], e.g. in the Waymo Motion Forecasting Challenge [24], but the prediction is only used for target estimation and not association.

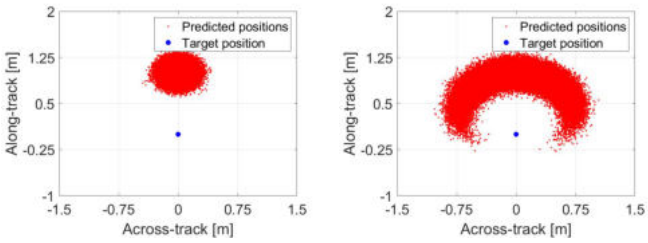
The rest of the article is as follows. Section II describes the motivation for the CAT and the CAT-NN. Section III describes the CAT distribution. The CAT-NN distribution is introduced in section IV. Section V contains comparisons of Gaussian distribution, the CAT distribution and the CAT-NN distribution. Conclusions are found in section VI.

II. Motivation and scenario

A. Motivation for the CAT distribution

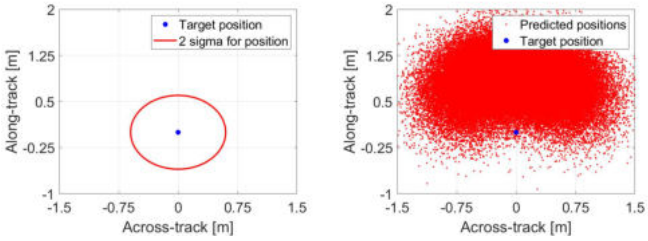
In target tracking, it is normal to assume that the target moves with a constant velocity between measurements. Acceleration is included as process noise in the form of a zero mean Gaussian distribution. As we see it, there are two main reasons for this. The first is that using using Gaussian zero mean process noise allows for the use of the linear Kalman filter when predicting the next position, which is easy to implement and quite robust. The second reason is that in many scenarios the actual target dynamics might be unknown. If acceleration is equally likely in all directions it makes sense to model it as a Gaussian distribution. And for some targets, such as quadcopter drones, the acceleration might indeed be modelled quite well by a Gaussian distribution.

However, there are targets for which modelling the acceleration as a Gaussian seem suboptimal. One such category of targets consists of fixed-wing aircraft (if we track in the 2D plane) and surface vessels. Somewhat simplified, we can say that they are limited to two types of acceleration: Performing a coordinated turn, and increasing/decreasing speed. This difference means that the predicted distribution can be far from Gaussian. An example is shown i Fig. 1. In both (a) and (b) we have predicted ahead 10^5 state vectors with position $[0, 0]$, velocity $[0, 1]$. In (a), the acceleration for each state is drawn from a multivariate Gaussian with $\sigma_{a_x} = \sigma_{a_y} = 0.2\frac{m}{s^2}$. In (b), each state vector draws an along-track acceleration from $\mathcal{N}(0, \sigma_a = 0.2\frac{m}{s^2})$, and a turn rate ω from $\mathcal{N}(0, \sigma_\omega = 1\frac{rad}{s})$. The difference between (a) and (b) motivated Gade et al. to create the CAT distribution and to use it for gating and association. For any given scenario σ_a and σ_ω are selected to match the expected motion of the target, and are called *scenario parameters* by Gade et al.



(a) (b)

Fig. 1: Predicted positions for a target moving with a) constant velocity, b) with constant acceleration and turn rate. The clearly non-Gaussian distribution in b) motivated the development of the CAT distribution.



(a) (b)

Fig. 2: Fig. (a) shows the mean of a track and its 2 sigma positional uncertainty. Fig. (b) shows the predicted positions of that track, when it has the same motion model and initial velocity as the track in Fig. 1b.

B. Motivation for improving the CAT distribution

The CAT distribution predicts where the target will be after a small time interval, given the position of the target and the *scenario paramters* σ_a and σ_ω. However, a track is usually given as a state vector and an uncertainty covariance matrix. As such, it is itself a distribution, and while it has a mean, one cannot be certain of the target's actual position.

Fig. 2 demonstrates the effect this has on the predicted distribution. Fig. 2a shows the position and 2 sigma uncertainty of the track position. Fig. 2b is analogous to Fig. 1b, i.e. 10^5 sampled state vectors propagated ahead one timestep. The difference is that in addition to sampling the acceleration and turn rate, each state vector also samples its initial position from the track distribution. The predicted positions in Fig. 2b form a wider distribution, one that can be interpreted as a combination of a CAT distribution and a Gaussian distribution. The motivation for the CAT-NN is that neither the CAT nor the Gaussian distributions are good matches for when a target has a CAT motion model and the track has non-negligible uncertainty.

C. Scenario and rescaling

In order for us to design and train a neural network to generate the CAT-NN distribution we need to decide on the input parameters. We wish to generate a predicted distribution based on the motion of the target and the state of the track. For simplicity we have decided to look at a 2D scenario, with a state vector containing position and velocity. With neither acceleration nor turn rate being tracked, these are assumed to be zero-mean, in addition to being Gaussian. The input to the neural net generating the predicted distribution is therefore σ_a, σ_w, and the covariance of the track.

The predicted distribution shown in Fig. 1 and Fig. 2 has the target at the origin at $t = t_0$ with a speed of 1 $\frac{m}{s}$ heading straight north. Fig. 3 shows a measurement being made at $t = t_1$. This is a very particular case, and many relevant targets, such as fighter aircraft, will have velocities greater than this by orders of magnitude. However, any scenario can be rescaled to be equal to the one in Fig. 3 by introducing the length unit $l^* = v_0 t$ and the time unit $t* = \Delta t$. By doing so the rescaled velocity will be $v_0^* = \begin{bmatrix} 0 \\ 1 \end{bmatrix} l^*/t*$ and the update time will be $t_1 = 1t^*$. This allows us to train the neural network for the CAT-NN distribution on this specific scenario while still being able to use the resulting trained network on any given scenario.

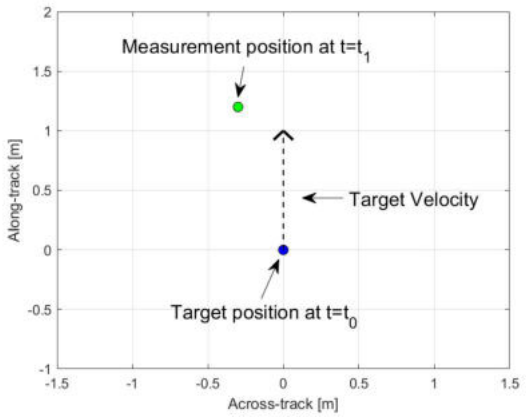


Fig. 3: The scenario used in this article. There is a target at the scene, and the origin is put at the target position, with the velocity of the target pointing *along-track*. The velocity vector is to scale, so the predicted position of the target is at [0,1]. A measurement is made at the next timestep.

III. CAT

This section describes the CAT distribution [12]. When an association between a track and a measurement is made it is usual to calculate the probability that the track and the measurement should be associated, and compare this to other possible associations, or the possibility of a missed detection. A common assumption is that the target moves with a constant velocity, which yields a Gaussian probability distribution for the predicted position. If we assume a constant along-track acceleration and turn rate instead we obtain the CAT distribution. The procedure to calculate the CAT probability for a given target and measurement is shown in Fig. 4, and described in greater detail below.

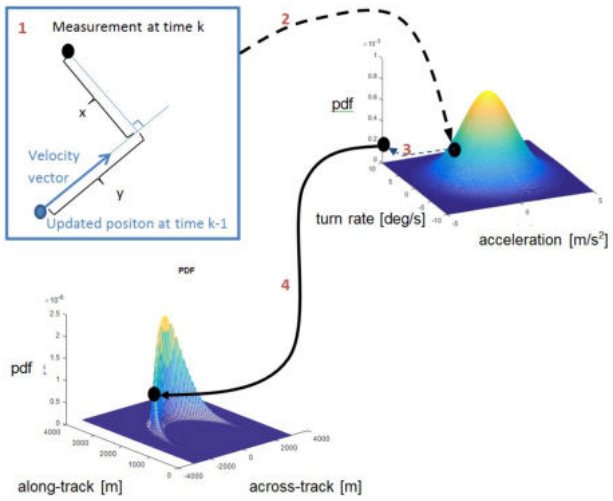


Fig. 4: The procedure to calculate the CAT probability for a given track and measurement. The figure is created by Gade et al. and was first presented in [12].

1) Find the position of the measurement relative to the target. This is the Cartesian position, with the target in the origin, and the y-axis in the direction of the target velocity.
2) Calculate the constant along-trajectory acceleration and turn rate needed for the target to move from its position to the position of the measurement.
3) Find the probability of the target having this acceleration a and turn rate ω. We assume that both of these are Gaussian distributions with known means and standard deviations, so this probability is given by
$$p(a,\omega) = \frac{1}{\sigma_a \sigma_\omega 2\pi} e^{-\frac{1}{2}((\frac{a-\mu_a}{\sigma_a})^2 + (\frac{\omega^2-\mu_\omega}{\sigma_\omega})^2)} \quad (1)$$
4) Calculate the probability of the target being in the position of the measurement. This probability is given by
$$p(x,y) = \frac{p(a,\omega)}{det(\boldsymbol{J})} \quad (2)$$
where $\boldsymbol{J}$ is a Jacobian matrix:
$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial y}{\partial a} & \frac{\partial y}{\partial \omega} \\ \frac{\partial x}{\partial a} & \frac{\partial x}{\partial \omega} \end{bmatrix} \quad (3)$$

Having found the probability for this particular target-to-measurement association, it can now be compared that of other potential associations.

IV. CAT-NN

The CAT-NN distribution expands on CAT by including track state uncertainty. The goal is to generate a discrete probability distribution over the target position based on a

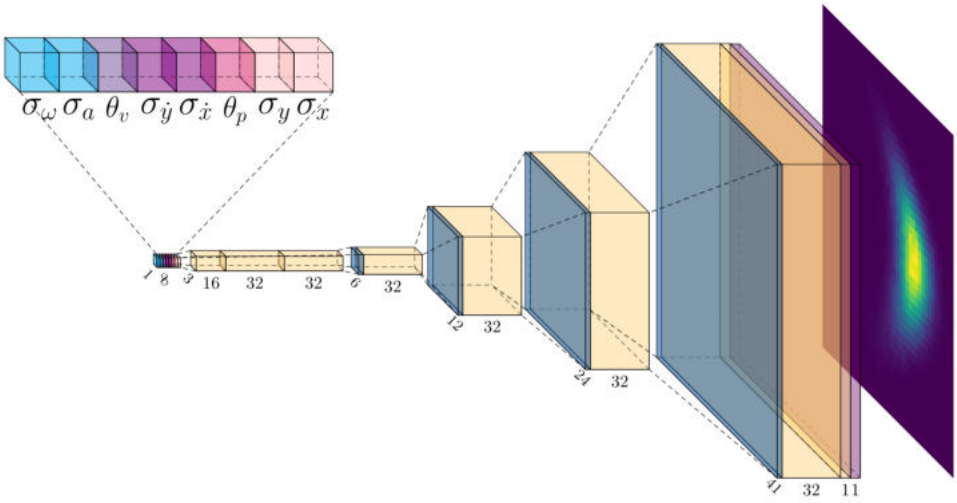


Fig. 5: Density network. A convolutional decoder network that takes 8 parameters and decodes them into a discrete probability density function. The 8 parameters are standard deviations for: position (σ_x, σ_y), velocity ($\sigma_{\dot{x}}, \sigma_{\dot{y}}$), acceleration ($\sigma_a$) and turn rate ($\sigma_\omega$), as well as the orientation of the position and velocity covariances with respect to the track velocity (θ_{pos} and θ_{vel}). We draw the output of each layer as boxes with corresponding dimensions. The output of the convolutional layers is drawn in yellow, resize/interpolation layers are drawn in blue and the final softmax layer is drawn in purple.

track state and motion model parameters. We can generate such distributions by simulating multiple potential trajectories. While this strategy could be used in a tracker directly, we found that training a neural network based on the simulated data gave a better performance, both in terms of accuracy and efficiency.

A. Data generation

The first step is to generate appropriate data, as this will define how the model works.

We generate training data based on our example scenario described in II-C, which means that the results can be rotated and re-scaled to fit a huge range of real-life situations. The data generation consists of two levels of sampling. The *scenario-level* sampling is where we sample potential targets with given uncertainty values. One sample represents a track with a corresponding motion model. The track can be represented with a position, orientation and velocity and a covariance matrix. The motion model is a probability distribution over acceleration and turn-rate. At this level, we only sample standard deviation values for the Gaussian distributions for a given target, as the mean values are already dictated by the scenario. Then there is the *target-level* sampling, where we sample concrete values from the Gaussian distributions from the scenario level. For each target, we sample 10^5 sets of position, velocity, acceleration, and turn rate. In Fig. 1b and Fig. 2b we have presented two such examples of target-level sampling, where the samples are forward propagated one timestep.

For each scenario-level sample, we can generate a discrete CAT distribution, that we can display as an image for two dimensions. To generate such an image we propagate the values from the target-level sampling from $T = 0$ to $T = 1$ with time steps of 0.01. We then generate a 2D histogram with 41x41 bins over the final 10^5 target positions and normalize the histogram to sum to one. We now have a set of parameters from the scenario-level sampling and a corresponding distribution image and such pairs are what we use to train our *density network*.

For the distribution parameters at the scenario level, we do a log-scaling of uniformly sampled values. We generally want a uniform sampling of the input space, but since the neural network easily learned to generate distributions for tracks with high uncertainty, we chose a log-sampling strategy where low uncertainty values are sampled with a higher frequency, resulting in faster model convergence.

B. Density network

We want a network where we can input target distribution parameters and get the corresponding distribution image. Our distributions are relatively simple, compared to other examples from the literature [25]. Therefore, we chose a simple architecture based on a feature pyramid network decoder [26] with 8 convolutional layers, and simple nearest-neighbor upscaling as illustrated in Fig. 5. In both Fig. 5 and Fig. 7b we have some examples of the trained network output based on unseen input data.

The input to the network is an 8-value parameterization representing a distribution generated by the *scenario level* sampling. The first 6 values represent the uncertainty of the track state. This is represented as the diagonal values of the positional and velocity uncertainty matrices in 2D cartesian coordinates and the rotation of the positional and velocity uncertainties. The next 2 values represent the parameters for the target motion model, with expected standard deviation in tangential acceleration and turn rate. For each input, we have

the corresponding distribution image as a target. We found that using a positional sine/cosine encoding such as [15], increased the convergence speed and gave a minuscule boost in performance.

The Cross-Entropy loss function is a perfect match for our application as we want the best possible overlap between the predicted and target distributions. As we want our network to output a probability distribution over position, we use softmax normalization over the image to ensure the network outputs a valid distribution.

We get the normalized probability estimate q with the softmax over the output image,

$$q(x_{uv}|\boldsymbol{\sigma}) = \frac{e^{x_{uv}}}{\sum_i^N \sum_j^M e^{x_{ij}}}, \tag{4}$$

where x_{ij} is the output value at index i, j. We then calculate the loss over the image,

$$L(p, q) = -\sum_u^N \sum_v^M p(x_{uv}|\boldsymbol{\sigma}) \log q(x_{uv}|\boldsymbol{\sigma}). \tag{5}$$

For each training step, we generated unique training samples. The final model was run through $6 \cdot 10^7$ samples, but we only gained an insignificant performance increase after $1 \cdot 10^7$ samples. We used the same procedure for generating training and test data, since every sample should be unique as long as we are mindful of the random seed. For our use case, we are not interested in extrapolating outside the training range, as we specifically set this range to cover our scenario. For large uncertainties the distributions tend towards Gaussian and our approach is less relevant, we therefore limited our input range to 0-3 in standard deviation and $-\pi$ to π for the rotational inputs. With some sporadic tests, we found that our model could slightly extrapolate outside the training range, but that the performance deteriorated with the distance from the intended range.

C. Scenario

For simplicity we limited our scenario to tracking in 2D Cartesian coordinates, with a *nearly constant velocity* model. There should, in principle, be no reason why our approach should not work for more complex scenarios. One possible obstacle could be limits to what kind of distribution a neural network can learn. These types of scenarios have been explored in other works [25]. The kind of training data that can be either collected or generated can certainly also limit the use of our approach for more complex scenarios. Perhaps the most prominent reason however to discard our approach, is that it provides no real benefit for scenarios where the positional uncertainty is larger than 2 times the expected target displacement between observations, as the distributions then tend towards Gaussian.

V. Comparison

CAT-NN is introduced as an alternative to the CAT distribution, which in turn was introduced as an alternative to the Gaussian predicted distribution. It is therefore natural to compare the CAT-NN with the CAT and Gaussian distributions. We will do this in two different ways.

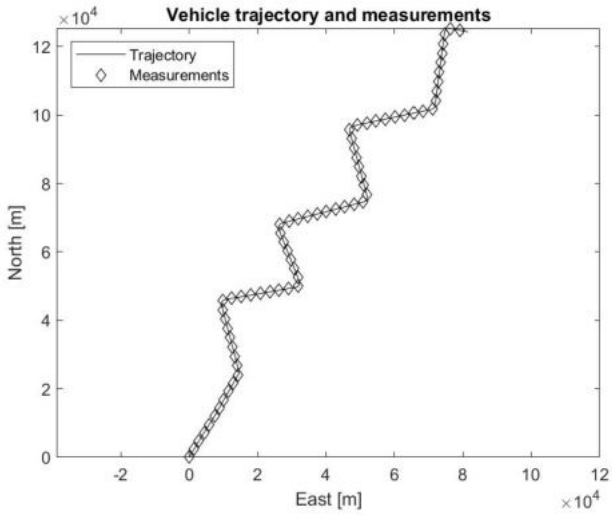


Fig. 6: The trajectory used for comparing CAT, CAT-NN and Gaussian predicted distributions. It is the simulated trajectory of a fighter jet.

The first comparison we make is comparing the CAT, CAT-NN and Gaussian distributions to the true distribution. We do this by calculating the Bhattacharyya coefficient, which is a metric for the similarity between two distributions. If they are identical the coefficient equals 1, while a coefficient of 0 means there is no overlap at all.

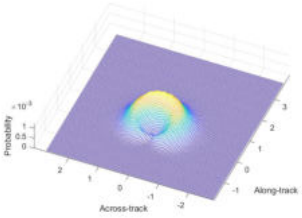
The second comparison we make is to apply the CAT-NN distribution and the Gaussian distribution to a simulated tracking scenario. We use the simulated trajectory of a fighter jet and the Stone Soup [27][1] tracking framework to generate detections and clutter.

A. Comparing with the true predicted distribution

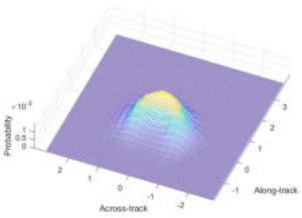
Gade et al. [12] demonstrate that the CAT distribution can be used in association and gating when there is no track uncertainty, and showed that it was a good match for the true distribution. As the CAT-NN was designed to include track uncertainty, we wish to examine how it compares to the CAT and Gaussian distributions for different magnitudes of track uncertainty.

Fig. 6 shows the trajectory used by Gade et al. [12]. We added track uncertainty to the track at each timestep and calculated the true distributions, as well as the CAT-NN, CAT, and Gaussian distributions. This is done for 10 different magnitudes of track uncertainty. An example of the different distributions is shown in Fig. 7. The track uncertainties are similar in x and y dimensions but with minor random variations for each timestep. The magnitude of the track uncertainties is of the order of e^i for track uncertainty i.

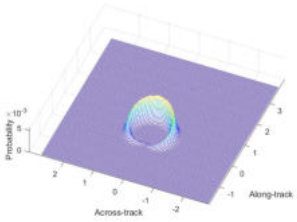
[1] https://github.com/dstl/Stone-Soup



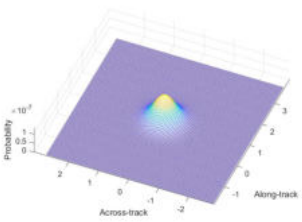
(a) True distribution



(b) CATNN distribution



(c) CAT distribution



(d) Gaussian distribution

Fig. 7: The true predicted distribution at a sample timestep from a simulation with significant track uncertainty, as well as the predicted CAT, CAT-NN and Gaussian distributions.

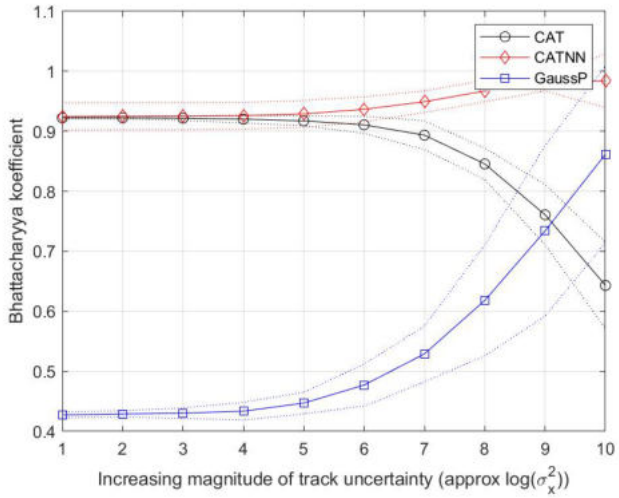


Fig. 8: The mean Bhattacharyya coefficient for the whole trajectory as a function of the magnitude of the track uncertainty.

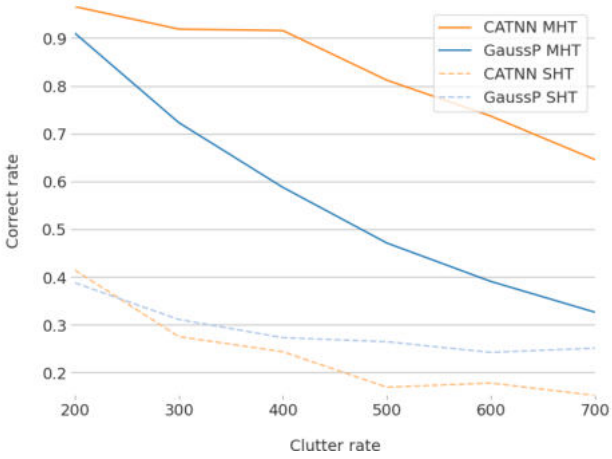


Fig. 9: Ratio of tracker updates that are inside the cutoff range of 2000m from the ground truth.

For the lowest track uncertainty ($i = 1$), we have that $\sigma_x^2 \approx e^1$, while for the highest track uncertainty we have $\sigma_x \approx e^{10}$, and similar for σ_y^2. The example distributions in Fig. 7 were created using the highest track uncertainty. The velocity covariances $\sigma_{v_x}^2$ and $\sigma_{v_y}^2$ are approximately one-quarter of the positional uncertainties.

The mean and standard deviation of the Bhattacharyya coefficient over the entire trajectory were calculated for CAT, CAT-NN, and Gaussian distributions, compared to the true distribution. This was done for each of the 10 magnitudes of track uncertainty. The result is shown in Fig. 8. The figure clearly shows that with low track uncertainty, the CAT distribution is close to the true distribution. However, its performance falls as the uncertainty increases. The performance of the Gaussian distribution is the opposite, gradually improving the Bhattacharyya coefficient with rising track uncertainty. The CAT-NN performs well over the entire span of uncertainties. It compares well to the CAT distribution for low track uncertainties and becomes more Gaussian-like as the uncertainty increases.

B. Single target tracking

We ran tests in a single target tracking scenario using the Stone Soup tracking framework, to verify that the CAT-NN distribution is more than a theoretical tool. Again, we use the trajectory shown in Fig. 6, and simulate measurements by adding noise to the trajectory and randomly sample clutter measurements, in line with a standard Stone Soup procedure.

We extended the PDAHypothesiser in StoneSoup to use our CAT-NN model. Then we first assembled a simple single-hypothesis tracker (SHT) using Probabilistic Data Association (PDA), and later a multi-hypothesis tracker (MHT), basing much of it on ready-made Stone Soup components. To focus the analysis on our extension, we only consider a single target tracking scenario, where the tracker is initiated with the true initial position and velocity. No new tracks are generated, nor is the track deleted. Our multi-hypothesis tracker pruned the track tree each timestep to retain the 5 best potential tracks.

For our CAT-NN model, we used a fixed normalized acceleration distribution of $\sigma_a = 0.3$ and turn-rate distribution $\sigma_\omega = \frac{\pi}{2}$. The magnitude of the track uncertainty due to the measurement uncertainty and the target dynamics was $log(\sigma^2) \approx 6$.

The simulations were carried out using clutter rates of $[200, 300, 400, 500, 600, 700]$, running the scenario 100 times for each clutter number. This allows us to calculate the average performance for each clutter rate.

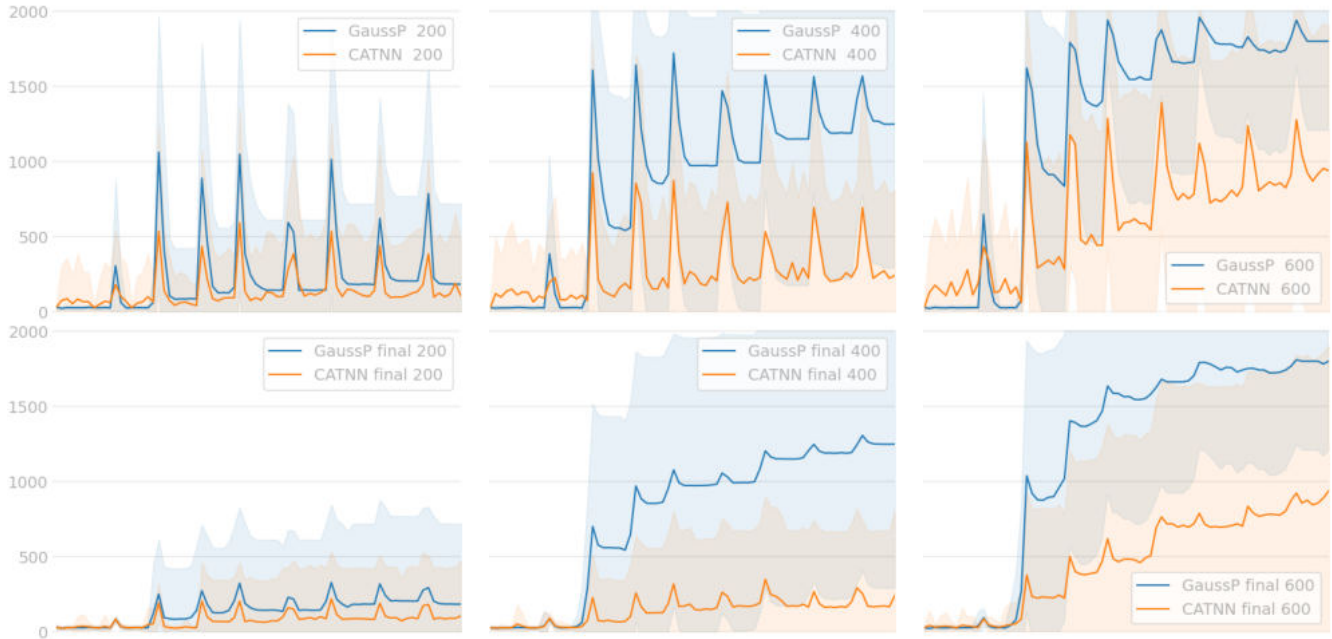


Fig. 10: Average OSPA distance for CAT-NN and Gaussian distributions (GaussP) with a cutoff distance of 2000m, using clutter numbers $[200, 400, 600]$. We filled in the standard deviation around the mean to express the variation. The top row evaluates the current real-time best hypothesis, while the bottom evaluates the trajectory of the highest-scoring track at the end.

A simple way to measure performance is to see how often the reported track position is close to the target. We define a cut-off distance of $dist_{cutt-off} = 2000$ meters and calculate the ratio of timesteps where the reported track position is inside the cutoff distance. In Fig. 9 we show how this ratio trails off when the clutter increases. The performance of the SHT tracker, using either CAT-NN or Gaussian distribution, was quite poor, and prompted us to create the MHT tracker. When running simulations we experienced that the SHT tracker was easily seduced by clutter, and was in general unable to recover. Interestingly, the SHT demonstrated worse performance when using the CAT-NN compared to Gaussian distribution for high clutter rates. While the CAT-NN generally matched less with clutter tracks, the clutter it did match had a more detrimental impact. A clutter match could result in more drastic turns, throwing the algorithm off track.

Fig. 9 also shows the performance of the MHT tracker using CAT-NN and Gaussian distributions. It is clear that in this scenario, MHT is far superior to SHT using PDA. It is also clear that the CAT-NN outperforms the Gaussian distribution.

To further investigate the performance of CAT-NN we calculated the average OSPA [28] score for different clutter rates, using our MHT. In our single target case with neither track initiation nor deletion, the OSPA score defaults to the root mean square error (RMSE), though with a cut-off distance. As the OSPA metric is included in the Stone Soup framework, we chose to use this rather than implement our own RMSE or other metric.

In Fig. 10 we see that the OSPA score is lower, and thus better when using the CAT-NN distribution. In the top row, we report real-time results, using the track position as reported each timestep. Here we see sharp spikes for both trackers, coinciding with the sharp turn of the target. In the sharp turns, both trackers can fall prey to track seduction, with the tracker giving a high association score to one or more clutter measurements that match well with the predicted distribution. However, the CAT-NN is unlikely to be completely thrown off track, since consecutive high-scoring clutter is rare due to the CAT-NN distribution being less expansive. For each spike in Fig. 10, the Gaussian model settles on a higher OSPA distance, indicating that some tracks fall off completely. The CAT-NN seems to be able to bounce back faster, as the spikes are less pronounced, and at a higher rate since the distance settles back at a lower level.

In the bottom row of Fig. 10, we show the batch results of the MHT, meaning we evaluate the final trajectory of the highest-scoring track at the end step. In this setting, the difference between CAT-NN and Gauss seems even more pronounced as the spikes from seduced tracks are diminished, highlighting how CAT-NN is more likely to maintain track.

VI. Conclusion

The CAT distribution introduced by Gade et al. [12] can be an improvement of the commonly used Gaussian distribution, for gating and association. The range of scenarios where CAT is an improvement is however limited since it does not include the track uncertainty. While including track uncertainty analytically can be complicated, we demonstrated that it can be fairly easily estimated by a convolutional neural network.

Simulations have shown that the CAT-NN distribution is a good match for the true predicted distribution for different magnitudes of track uncertainty for a target with a CAT motion model. In this it is superior to the CAT distribution, which is only applicable when the track uncertainty is really low, and Gaussian distribution which is only a good match when the uncertainty is relatively high.

Furthermore, the CAT-NN outperforms the Gaussian distribution when tracking a single target with a CAT motion model in a clutter-dense environment. Our simulations show that while MHT trackers using either distribution might momentarily struggle in sharp turns, the CAT-NN distribution gives the hypothesis with the true detection a higher score, and thus the tracker recovers much quicker.

The performance of the SHT compared to the MHT is worth emphasizing. We would expect the CAT-NN to perform better than the Gaussian distribution in our scenario, since it is a better match for the true predicted distribution. However, we experienced that the Gaussian distribution performed better when using an SHT, though both distributions performed poorly. It was only with the implementation of an MHT that the CAT-NN showed the promised improved performance.

As for further work, there are a few clear paths. The tracker used in our simulation used a linear 2D Kalman filter, with a state vector consisting of position and velocity. One might argue that since we are looking specifically at targets performing coordinated turns one might get greater performance from the tracker using an estimator that is designed for such movements. This might be an IMM-filter, or we might expand the state vector to include the turn rate. In the latter case, the turn rate would be an additional input to the neural network. This would mean creating new training and test data sets, but the fundamental idea would be the same. Indeed, the concept of using a neural network to estimate a predicted distribution can in theory be applied to any state vector or motion model.

REFERENCES

[1] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House, 1999.

[2] D. Musicki and R. Evans, "Joint integrated probabilistic data association: Jipda," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 1093–1099, 2004.

[3] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[4] B. N. Vo, B. T. Vo, and H. G. Hoang, "An Efficient Implementation of the Generalized Labeled Multi-Bernoulli Filter," *IEEE Transactions on Signal Processing*, 2017.

[5] P. C. Mahalanobis, "On tests and measures of group divergence i. theoretical formulae," *J. and Proc. Asiat. Soc of Bengal*, no. 26, pp. 541–588, 1930.

[6] M. Wang, Q. Wan, and Z. You, "A gate size estimation algorithm for data association filters," *Science in China, Series F: Information Sciences*, vol. 51, no. 4, pp. 425–432, 2008.

[7] Y. Duan, X. S. Tan, Z. G. Qu, H. Wang, and J. Wang, "Adaptive tracking gate design for maneuvering target in clutter environment," *Proceedings of the 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2017*, vol. 2018-Janua, pp. 1449–1455, 2018.

[8] X. Wang, S. Challa, and R. Evans, "Gating Techniques for Maneuvering Target Tracking in Clutter," no. 1957, 2002.

[9] T. Bailey, B. Upcroft, and H. Durrant-whyte, "Validation Gating for Non-Linear Non-Gaussian Target Tracking Computing a Validation Gate for Non-Gaussian PDFs Bearing-Only Tracking."

[10] S. K. Singh, M. Premalatha, and G. Nair, "Ellipsoidal Gating For an Airborn Track While Scan Radar," vol. 0, pp. 334–339, 1995.

[11] B. Gade, M. Kloster, and M. Aronsen, "Non-elliptical validation gate for maritime target tracking," in *2018 21st International Conference on Information Fusion (FUSION)*, 2018, pp. 1301–1308.

[12] B. H. H. Gade, C. N. Vooren, and M. Kloster, "Probability distribution for association of maneuvering vehicles," in *2019 22th International Conference on Information Fusion (FUSION)*, 2019, pp. 1–7.

[13] Q. Liu, J. Xu, R. Jiang, and W. H. Wong, "Density estimation using deep generative neural networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 118, no. 15, 2021.

[14] Y. Pan, K. S. Kuo, M. L. Rilee, and H. Yu, "Assessing Deep Neural Networks as Probability Estimators," in *Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021*, 2021.

[15] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12346 LNCS, 2020.

[16] P. Dai, R. Weng, W. Choi, C. Zhang, Z. He, and W. Ding, "Learning a Proposal Classifier for Multiple Object Tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021.

[17] G. Braso and L. Leal-Taixe, "Learning a Neural Solver for Multiple Object Tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020.

[18] O. Cetintas, G. Brasó, and L. Leal-Taixé, "Unifying Short and Long-Term Tracking with Graph Hierarchies," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2023-June, 2023.

[19] J. Y. Yu, M. O. Sghaier, and Z. Grabowiecka, "Deep learning approaches for AIS data association in the context of maritime domain awareness," in *Proceedings of 2020 23rd International Conference on Information Fusion, FUSION 2020*, 2020.

[20] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion Transformer with Global Intention Localization and Local Movement Refinement," in *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[21] J. Gu, C. Sun, and H. Zhao, "DenseTNT: End-to-end Trajectory Prediction from Dense Goal Sets," in *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

[22] S. Capobianco, L. M. Millefiori, N. Forti, P. Braca, and P. Willett, "Deep Learning Methods for Vessel Trajectory Prediction Based on Recurrent Neural Networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 6, 2021.

[23] N. Forti, L. M. Millefiori, P. Braca, and P. Willett, "Model-based Deep Learning for Maneuvering Target Tracking," in *2023 26th International Conference on Information Fusion, FUSION 2023*, 2023.

[24] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov, "Large Scale Interactive Motion Forecasting for Autonomous Driving: The WAYMO OPEN MOTION DATASET," in *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

[25] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[26] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017.

[27] D. Last, P. Thomas, S. Hiscocks, J. Barr, D. Kirkland, M. Rashid, S. B. Li, and L. Vladimirov, "Stone soup: announcement of beta release of an open-source framework for tracking and state estimation," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXVIII*, vol. 11018. SPIE, 2019, pp. 52–63.

[28] A. S. Rahmathullah, A. F. Garcia-Fernandez, and L. Svensson, "Generalized optimal sub-pattern assignment metric," in *20th International Conference on Information Fusion, Fusion 2017 - Proceedings*, 2017.