DOI 10.5162/ETTC2024/A6.2

Multiplatform sensor fusion
Drawing a common tactical scenario

Miguel Arévalo Nogales[1]
[1] Airbus Defence and Space, John Lennon Av, Getafe Madrid, Spain
miguel.arevalo@airbus.com

Abstract:

Recent years have proven the importance of sharing sensor data in complex scenarios. The platforms in play are wildly heterogeneous, from small, unmanned vehicles to aircrafts, land units or vessels. This paper explains the concept of Common Operational Picture, a tactical scenario where multiple participants share sensory data to draw a single vision, improving data quality and reliability through data fusion. The main topic to be covered will be the data fusion and integration techniques used to enhance quality and reliability and the machine learning methods used for decision making with the data being stored.

The root of the problem lies in the origin of data, as it is provided, processed, and stored by different participants and later on shared with others. This pipeline brings some challenges such as different timing systems, sensor hysteresis, and connectivity problems which burden data quality. Artificial intelligence and machine learning offer several solutions in this regard: bandwidth can be reduce using dimensionality reduction techniques, data anomalies, outliers, can be detected and analyzed using support vector machines and a greater situational awareness can be achieved.

Key words: Sensor, machine learning, data fusion, common operational picture, connected environment.

Introduction

Recent years have marked a huge improvement in sensors and networking, making information ubiquitous and cheap. This fact, combined with the improvement in calculus capabilities and the advent of single board computers means large amounts of information can be read, processed, and shared, easily and conveniently, between many devices.

Some specific area of interest is a situation where vehicles, manned or unmanned, possibly helped by other actors (ground stations, satellites), participate in a joint operational environment, e.g. a rescue or surveillance mission. Sharing a common view of the scenario, fostering collaboration, and enhancing their situational awareness is a very welcomed feature provided by communication advances. This scenario is what is commonly known as the common operational picture.

The objective of this paper is to study how machine learning techniques can be applied to sensor fusion and the enhancements this methodology provides to a common operational picture that all these actors are trying to depict together.

The paper will analyze the problems, algorithms and opportunities that must be faced when trying to work a common scenario where sensors are not managed by a single entity but transmitted over networks and shared between many participants.

Working on a Common Operational Picture

When we are talking about sensor fusion we should have a clear objective, sensor data is being merged and processed to provide a Common Operational Picture to all the participants in a joint operation.

The objective is to share data and create additional sensors that provide a better view of the scenario being operated.

Normal scenarios, such as monitoring disputed areas, mean several participants collaborate and bring data together. Navy ships can act in the field with sonar, radar and thermal cameras, satellites can provide images and reconnaissance aircrafts can provide radar (ISAR/SAR), sonobuoys (and acoustic processing), additional imagery (FLIR, etc.), as well as IFF interrogators and/or AIS-SART. These sensors are normally managed through the tactical system of the participants where they are displayed to an operator. Datalinks can

provide some sort of interconnection between participants so data is shared, but if not properly classified we can find the information to be overwhelming and misleading. The need for some classification and data fusion algorithms is clear[1].

STANAG 4676 and nomenclature

Since the objective of the project is to collaborate in a common operational picture it is necessary to define a standard language for the participants to talk. NATO defines a series of standardization agreements or STANAGs, these agreements cover many topics, but we will focus on STANAG 4676[2] which is the Intelligence Surveillance Reconnaissance Tracking Standard. This agreement specifies a data model which can be used among all participants.

Before getting into the model itself it is important to know the basic nomenclature:

- NITS: Operation being performed.
- Tracker: A tracker is an active actor participating in the operation. It generates information using its sensors.
- Product: Products are a collection of sensors, they can be, but are not limited to, actors in the scenario, for example, a system that provides several sensors, is a product, but an actor can have many products.
- Sensor: Each and everyone of the measurement units in a product, a product may have many of them (e.g. a multicamera pod).
- Track: Information of an object provided by a sensor, it normally focuses on position, speed, and acceleration but it is not limited to them.

STANAG4676 data model is quite complex, supporting not only ID, position, and speed of different objects, but also the source of this measurements, and more complex features such as coordinates transformations, the basic model diagram can be seen in Fig 2.

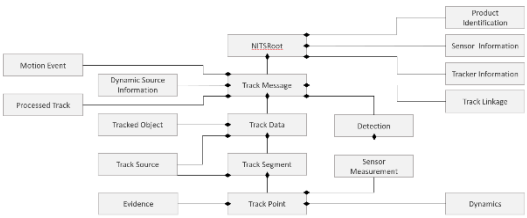

Fig.1.STANAG4676 model. First level.

The model can be understood as a tree, NITS is a common entity, shared by all participants, that hosts all the information: sensor definition, trackers, messages, etc. This entity can be used to add new data or to forward current information to other participants.

All participating actors (trackers) register themselves, providing tracker, product, and sensor information.

Messages are the basic entity containing transmitted data, a single NITS can contain many messages, or a single one if historical information is not transmitted. Messages are linked to several other entities, motion evens, track data or their source.

Track Data is the primary entity containing a track, it is defined by a set of tracked objects, tied to a track source (tracker, product and / or sensors) and linked to a segment which is made of a set of track points.

Track Points are the key to understanding the model, a track point is tied to a detection, comes from a sensor measurement or a sensor composition, and is linked to a set of evidence. Tracked object position is stored in a dedicated entity itself linked to a track point: Dynamics. This entity can store position, speed, and acceleration, in addition to the coordinate system used by the data.

Composed sensors are supported by means of transformations, so they can be traced back to originating sensors. The transformation itself can be also described.

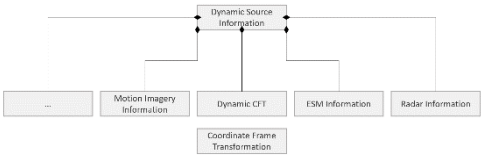

Fig.2.STANAG4676 Source - Sensor relationship.

The reason for choosing STANAG 4676 as the data model lies in the features it supports: data transformation, track stitching, confidence definition, track merging, etc. It fits nicely with the concept of sensor fusion.

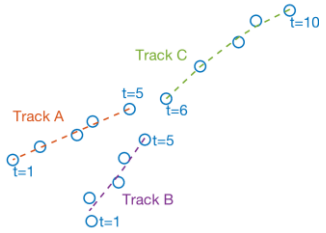

Fig.3.STANAG4676 Track merge example.

Algorithm description

Algorithms used for the project are not exclusive to sensor fusion, but generic machine learning ones.

For the sake of better understanding how the algorithms are being used for sensor fusion a brief section describing the methods being used has been included.

Principal component analysis

Principal component analysis (PCA)[3] is a technique used to describe a data set in terms of new, uncorrelated variables (eigenvectors). These variables are ordered by the amount of variance, from the original dataset, that they are describing (eigenvalues). The original idea was to reduce the dimensionality of the data.

The whole process can be reversed losing some of the original information, this way we are reducing the bandwidth being used but we keep most relevant data.

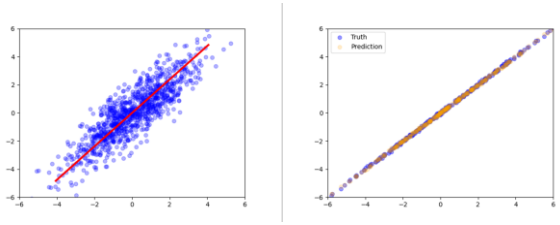

Fig.5.Original dataset, PCA and reconstruction.

PCA can be reconstructed using equation (1).

$$PCA_R = PC_{score} * Eigenvector + Mean \quad (1)$$

There is another relevant use case as hinted by Fig. 5 which is reducing the amount of noise in a dataset.

Autoencoders

Autoencoders [4][5] are an unsupervised type of neural network used to learn codings (features) of unlabelled data. This can be used to compress data and later on decompress (reconstruct) data to its original form.

They are composed of three parts:

- Encoder. It is a layer, or set of layers in case of multi-layered autoencoders, used to compress (encode) the input data into a reduced set of features.
- Latent space. It is a position on the space describing, using a different set of variables, the original position coming from the input space. Since normally latent space is chosen to have a reduced dimension to input space the dimensionality is reduced.
- Decoder. Layer, or set of layers, used to decompress (decode) the latent space representation of some data into another representation into the original data space. Since the dimension of the latent space is normally smaller thant the original data space reconstruction is lossy.

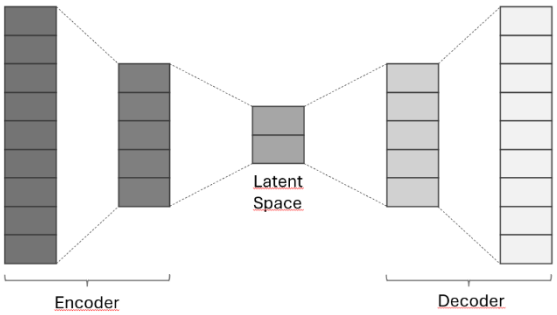

Fig.6.Autoencoders.

Support Vector Machines (SVM)

This supervised learning method is a max-margin model used for classification or regression. The original idea was to find a vector which divided a set of points in two different groups, hence classifying new points based on the original points that were used for training and obtaining the vector.

SVM have evolved into new techniques such as kernel support vector machines (KSVM), which transform the original two-dimensional space into a higher dimensionality one using a kernel trick.

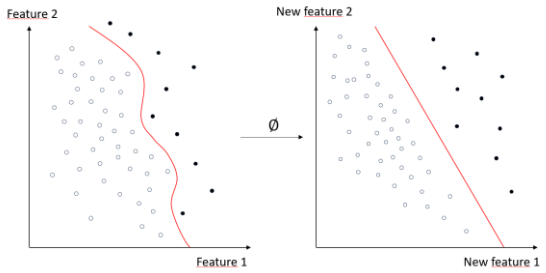

Fig.7.Kernel function.

SVMs robustness to outliers can be used to separate them and curate datasets [6][7].

K-Nearest Neighbor (k-NN)

Not to be confused with k-Means, k-NN [8][9] is a supervised learning algorithm used for classification or regression.

When used for data classification a point is assigned a class based on the classes of its neighbors, the most frequent class is assigned to the point, k is the number of neighbors to be considered.

If used for regression the output is the property (measurement) value of the object, for example

if the sensors measuring the property failed. The value is the average of the nearest k neighbors.

A normal modification of the algorithm is the addition of weights to the neighbors. This way averages consider a factor that modifies the relevance of the neighbor (the weight) into the result. This weight can depend on many things, distance to the original data, neighbor index (closer ones weigh more than farther ones, but distance is not considered), etc.

Introduction to sensor fusion

Sensor (or data) fusion is the process of merging several measurements to get a better understanding of the status of a system.

Previous definition is intentionally broad to cover very different use cases, let's try to clarify it by making it more precise.

A system can be many things, a human being, a set of moving objects or a computer cluster connected to a network. The status is dependent on the system, for a set of moving objects it makes sense to be interested in their relative position whereas for a computer cluster it may be a single measurement providing its health / resource usage.

Finally, sensors can be any kind of information coming from the system, bandwidth, resource usage or latency if speaking from a computer environment, blood pressure and heart rate in case of a medical monitoring system, or position, speed and direction in the case of a moving object. These measurements are normally registered over a time frame, providing some sort of guidance or evolution.

Sensor fusion can be divided based on the stage or level where the fusion is performed.

- Low-level Fusion (a. in Fig. 2): Raw sensory data is fused together to get a more accurate measurement. It is the most common type of sensor fusion; we are normally merging homogeneous sensors which may differ in type but provide the same kind of measurement (e.g. taking several samples of a fingerprint scan and merging them together to get a single value or using several images coming from different source to identify an object).
- Mid-level Fusion or Feature Level Fusion (b. in Fig. 2): Multiple heterogeneous measurements are fused together into a single entity, for example biometrics sources such as fingerprints, iris scanner and face / image recognition are brought together to identify a person. This is done by detecting correlated measurements and merging them into a single dimension.
- High-level fusion or decision level fusion (c. in Fig. 2): As its name implies, decision level fusion is aimed at making decisions: selecting a single hypothesis from a group of hypotheses based on the information provided by a set of sensors, which may be identical or not. For example, in a car, several sensors, some of them already fused (external temperature, speed, route, weight) can be merged to decide if the driver has to be notified to change route or stop in a gas station / electric vehicle charging station because he is not going to make it to destination.

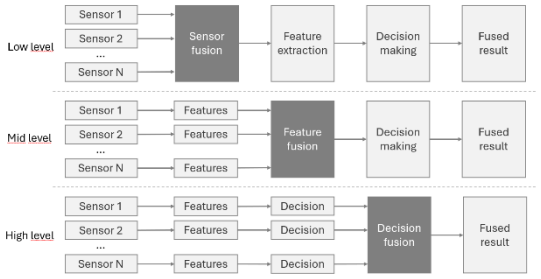

Fig.8.Sensor fusion classification by levels.

Sensor fusion is also divided in two main categories depending on where it occurs:

- Centralized: Participants simply forward the information to a central processing entity which merges the information, if clients are not relevant actors they might not be notified back with the results.
- Decentralized: All actors can perform some level of data fusion and have some degree of autonomy in decision-making. This is normally the case on edge / fog computing environments.

Data fusion: bandwith usage reduction

Although the focus of the paper, and the techniques being used, is mid and high-level data fusion, there is a specific use case that has to do with low level fusion: bandwidth enhancement in fog / edge computing.

There is a perennial problem in connected environments, which is sharing data between participants. Bandwidth is normally limited, so the amount of data that can be shared is scarce, this hits precision and decision making.

A straightforward application of machine learning techniques is dimensionality reduction using PCA analysis / autoencoders: by reducing the dimensions of data we can minimize used bandwidth at the cost of precision.

Autoencoders can detect relevant features from a dataset and compress it into a latent space, this space's dimensionality is lower than the

initial space, improving bandwidth usage. Once the information is received it can be decompressed and reconstructed into a similar space to the original one, the resulting space is not equal since we are losing some fidelity, but the tradeoff is the amount of data being sent is significantly reduced.

Another method of bandwidth reduction being studied is the creation of an adaptive rate using sensor data [10]. The basic idea is to adapt the transmission rate to the data being read so we can make better use of the available bandwidth.

To properly adapt the rate, we should be first doing a PCA analysis of the sensors, this determines which ones are more relevant to the use case.

The idea is to classify sensors into groups using PCA. These groups are defined as most relevant sensors to the scenario being studied, for example in a COP where we are receiving positional data from hundreds of tracks, we may discard most information from all tracks in an area to focus / improve data transmission of tracks in a relevant area, in this way information from secondary tracks is consuming less bandwidth. Doing this classification at sensor level means that we can focus on specific measurements (e.g. position) relegating others (e.g. speed) to a secondary role / transmission.

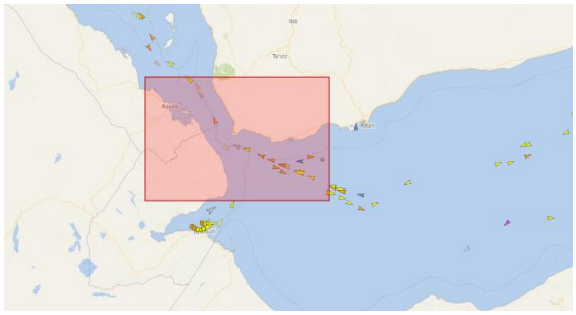

Fig.9.Example: priority area for track transmission.

Once sensors are classified, the value difference from the sensor to previous measurement is compared to a threshold, so even non relevant values are sent when the variance is large enough (or a certain time has passed).

This method is similar to a QoS in networks but done based on what sensor information is more relevant to our use case.

There are several caveats with these techniques:

- We are using compression techniques that are reducing the fidelity of the data being sent (lossy compression), this means we may be missing some relevant information.
- We are saving on bandwidth at the cost of computing power, which is not always a good tradeoff.

Feature fusion: confidence definition

There is a straightforward application to mid-level fusion for STANAG 4676: obtaining the uncertainty of a measurement. Sensors do not know the real position or speed of an object being tracked, in fact STANAG 4676 model has a placeholder to define the confidence of a measurement, but how can it be obtained? A method has been developed to provide an expected value for this field.

The algorithm relies on comparing previous measurements from the same set of sensors and comparing them with a set of well-known data, incorporating contour conditions such as tracker heading, distance to tracked object, etc.

For the use case under analysis, we are basing our training in AIS data, there are several reasons to do this.

- If data comes from some well-known participants the quality of the data is very high, as the participant is already making corrections to it by means of low-level data fusion.
- AIS data directly comes tagged with its originator, so we have no uncertainty on the originator of the trace and the is no obscuring.
- AIS data is transmitted through the network and there are historical databases that can be checked and used. This eases the training of the models.

To create new, data-quality, sensors the first step was to make a simple comparison between AIS data and sensor data from historical records. This comparison provides, given the sensor type and contour conditions an estimation of the expected error (certainty) for any given sensor and measurement.

At this stage the influence of the variables in the error is not known, this is what the model is going to analyze and provide to us.

Once we have the dataset a multi-variable regression algorithm is trained (k-NN was used but others are under evaluation). This model is trained for a single error / certainty, if we want to distinguish between different types of measurement errors (e.g. speed, position) we should train different models.

These steps (model training) are done before the operation (real object tracking) is performed.

During the operation the sensor suite from the tracker is generating sensor data. The measurements can be validated if the object being tracked is also providing AIS data or not, if, as is usually the case in disputed environments or with hostile tracks.

In any case, the next step in the algorithm is to enhance the sensor dataset with contour conditions. These conditions are expected to be the same ones used during the training: tracker position and heading, time and day, meteorological, etc.

If the object being tracked is also providing AIS data, we can use all this information to further train the model, moreover, this data will be given higher priority since it is providing closer to reality contour conditions (meteorological, etc.). Certainty is calculated directly and not through the model.

If, on the other hand, the tracked object does not provide AIS data we can use the model to calculate the certainty of the measurements. To calculate this value, we simple use the model to predict an expected certainty.

This method has validated some logical assumptions for example, it has been found that images taken against the sun (so sun position is relevant) provide worse detection quality than the ones being taken in normal conditions. The same can be said about lightning conditions, when the attitude of the camera is not adequate, or the object being tracked is too far away.

The additional sensor information (certainty) can be then fed to image recognition algorithms to enhance them or take into consideration when processing the image.

Abnormal track detection

Decision level data fusion may be the core reason to have a COP where several participants (trackers) collaborate and bring information together. Merging all this information produces better informed decisions and a level situational awareness not attainable when working alone.

One of the most important areas of analysis in a COP is detecting abnormal behaviors. This behavior may be due to technical problems (e.g. The Dali colliding with the Francis Scott Key Bridge) or deliberate as in unsafe driver's behavior [11].

STANAG 4676 supports several sensors to classify the track, there is already plenty of work doing with abnormal behavior identification using cameras [12], but project's development has focused on typical sensor information such as position, heading or speed.

The basic idea of the algorithms being implemented is to provide operators a measurement on the level of suspiciousness of certain tracked object.

The first approach to identify if a tracked object had an abnormal behavior was to use data classification algorithms.

A one-class SVM [13] was trained with AIS data available online. This type of SVM uses a kernel function to map the input data into a higher dimension space, this way point distance is larger, and the algorithm can distinguish between different points.

Since naval routes are well defined it is easy to identify if a ship is deviating from a typical route.

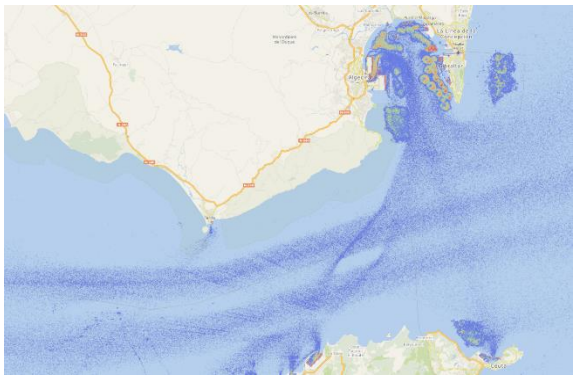

Fig.10.Historical AIS data in the Gibraltar strait area.

This approach is not without its problems:

- Only X/Y position is considered, no speed, acceleration or heading is taken into account for the decision criteria.
- Points are seen as independent variables, they are not linked in a time series, strange routes within the valid area are considered normal by the algorithm.

These problems highlight that working with trajectories (tracks) has some additional challenges. First, tracks are not a disjointed set of points, they have a direction, this variable needs to be considered since it has an impact on what the normal behavior is for a given trajectory. Secondly, and linked to previous problem, trajectories are made of a set of ordered points, this poses a problem for typical ML algorithms that are focused on non-ordered sets.

The first problem can be solved by having two different instances of the algorithm trained with different datasets, one for each direction or lane.

The second problem on the other hand means that basic classification algorithms may not be able to detect abnormalities in trajectories, and additional data conditioning techniques are needed to properly analyze data.

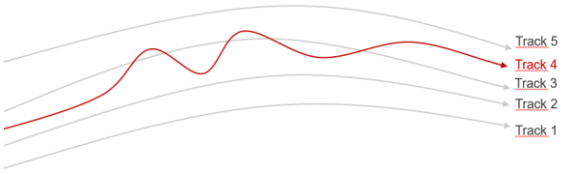

Fig.11.Abnormal track within trajectory boundaries.

A typical approach to solve the timing problem is to divide the track into different tiles or windows. Partitioning the data into fixed size windows means that every point in the trajectory P_i (2) is encoded into a fixed size attribute vector V_P (3), along with other relevant variables for the point.

$$P_i = \{X, Y, V^x, V^y \dots |\overrightarrow{XY}|\} \quad (2)$$

$$V_P = \{P_0 \dots P_N\} \quad (3)$$

All windows must have the same number of points, for these trajectories must be resampled, and each point has to be composed of the same set of variables, creating matrixes (4) for each window.

$$\begin{bmatrix} X_0 & Y_0 & V_0^x & V_0^y & \cdots & |\overrightarrow{XY_0}| \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ X_N & Y_N & V_N^x & V_N^y & \cdots & |\overrightarrow{XY_N}| \end{bmatrix} \quad (4)$$

These matrixes are defining the behavior of a track in any given tile but even more so, they can be compared to extract common values and patterns among tiles.

This windowing is based on Li et al. [14] work, which focused on hierarchical feature classification using a rule-based classifier, this was the core of their moving objects anomaly detection algorithm (ROAM). Other examples include Lee et al. [15] who also used trajectory partitioning for their outlier detection algorithm.

Once data is partitioned, we can analyze each window separately using traditional machine learning algorithms.

The workflow is quite basic: a classification algorithm is trained using the basic information for a window (V_P), we have as many instances of the algorithm as windows, which is interesting as it highlights abnormalities per zones.

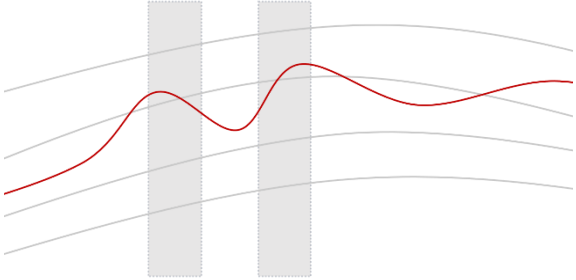

Fig.12.Windowed data where we can find anomalies.

The track data is then measured by the tracker, using the windowing criteria a window is selected and the associated instance is fed with the data. The algorithm outputs if the data is within boundaries or is an outlier.

Once the theory has been presented, we can discuss the algorithms are being used and how they are being trained.

First thing to do is define a partitioning criterion for the data, in our specific use case, and since we are focused on trajectories, we are using geospatial positioning for windowing. The area under analysis is divided into a tile matrix, the size of each tile must have into account the responsiveness of the objects being tracked.

Now complete trajectories are created, ensuring the track being analyzed has the right dimension (all tiles must have the same number of sample), for this we resample the track using linear interpolation. We can think of this procedure as a resampling in the space domain instead of time. It is in this step that additional parameters are added to the dataset, e.g. distance to other objects.

Now a set of tiles are selected based on the position / trajectory of the tracked object, this defines the set of trained models that will be used for analyzing the behavior.

Once divided and assigned to tiles a similar approach to the first algorithm can be used: a one-class SVM. A matrix of tile objects, each one tied to a OneClassSVM object is kept in memory. This SVMs have been trained using AIS data and reproduce what a normal behavior should be, considering not only position but speed, direction, distance to nearest track, etc.

In real time a measurement is made and buffered, if the data is enough to reconstruct a trajectory in the given tile, then the proper instance is invoked with the track's data for that tile, this way the algorithm can predict if the measurement is an outlier (behavior is not normal) or not.

Several things must be considered when training the algorithm:

- Tile size has a huge impact on the precision of the decisions provided by the ML algorithm.
- Kernel function and gamma must be tuned to match the type of variables being measured.

To put both problems into context, slow cruising ships or transports, very slow moving and with highly limited routes allow for larger tiles and smaller gamma, highly maneuverable objects,

on the other hand require smaller tiles and are much harder to fine tune.

Both these algorithms offer several advantages on their own, but the real value is obtained when combining both.

Simple naval route comparison is fast and does not need windowing, it provides a fast glimpse on how probable is to find a track in certain area and what the typical values for his speed and position would be. If combined with a windowing method additional, time related information is obtained: track direction, speed, acceleration and if the trajectory is normal or not. This information takes longer to be processed and is not readily available because of the windowing.

Conclusions

To summarize the work done, we have seen how sensor data can be collected and transmitted between different participants in a scenario, depicting a common operational picture that improves situational awareness for all participants.

The studied scenario has been one dealing with surveillance and reconnaissance with different actors, naval, aerial, etc. but can be extended to other situations such as cars driving in a highway, provided they have V2V connectivity, and collaborate to depict a common picture.

Once the scenario, the roles and communications have been defined we have studied what machine learning methodologies can be applied to sensor data, going beyond low-level and focusing more on mid-level and high-level data fusion.

Mid and high-level data fusion can benefit from machine learning in several ways. Mid-level data fusion can create new sensors, such as data quality sensors, based on original sensor types, position, or speed. We can go further and create a transmission rate sensor using trained information.

High level data fusion revolves around decision making, and in this aspect machine learning is also useful. Models can be trained to determine the normality of a behavior, detecting inconsistencies in the measurements, inconsistencies that do not necessarily have to do with invalid data but with abnormal behaviors that reveal that one of the monitored participants is having a problem. New sensors can be built measuring the level of certainty on the abnormality of the behavior.

Two types of machine learning algorithms have proven to be suited to this kind of sensor creation:

- Classification algorithms.
- Anomaly detection algorithms: Support Vector Machines & Principal Component Analysis.

Nevertheless, the usage of machine learning is not without its problems. There are two problems that are inherent to machine learning:

- Training. Most of the algorithms used for machine learning require the training of the system, this means many of the applications are only meaningful if applied to a dataset within the training margins, which is not always possible, for example anomaly detection for cargo boat trajectories is normally tied to the area where the model was trained, moving to another area changes the model and means some retraining is necessary.
- Overfitting. This is the result of trying to get exceptionally meaningful results, we may end up in fitting our model so much that is only relevant for the training dataset.

Finally, there is a specific problem for our use case, it has to do with how data is used. One of the use cases for the project was using data fusion in disputed areas. The problem is if the rivals know the algorithm being used and its results, in which case they can adapt their techniques and behavior to go unnoticed, effectively cheating the system.

References

[1] R. Nowak, R. Biedrzycki and J. Misiurewicz, Machine learning methods in data fusion systems, 2012 13th International Radar Symposium, Warsaw, Poland, 2012, pp. 400-405, doi: http://dx.doi.org/10.1109/IRS.2012.6233354

[2] G.Van Nederveen, NATO intelligence surveillance reconnaissance tracking standard NATO STANAG 4676, IET Seminar on Terget Tracking and Data Fusion: Algorithms and Applications, 2008, doi: https://doi.org/10.1049/ic:20080063

[3] Shlens, Jonathon. A tutorial on principal component analysis, doi: https://doi.org/10.48550/arXiv.1404.1100

[4] David Charte, Francisco Charte, Salvador García, María J. del Jesus, Francisco Herrera, A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines, Information Fusion, Volume 44, 2018, https://doi.org/10.1016/j.inffus.2017.12.007.

[5] Chen S, Guo W. Auto-Encoders in Deep Learning—A Review with New Perspectives. Mathematics. 2023; 11(8):1777, doi: https://doi.org/10.3390/math11081777

[6] Marta Baldomero-Naranjo, Luisa I. Martínez-Merino, Antonio M. Rodríguez-Chía, A robust

SVM-based approach with feature selection and outliers detection for classification problems, Expert Systems with Applications, Volume 178, 2021, 115017, ISSN 0957-4174, doi: https://doi.org/10.1016/j.eswa.2021.115017.

[7] Tsyurmasto, Peter & Zabarankin, Michael & Uryasev, Stan. (2014). Value-at-risk support vector machine: Stability to outliers. Journal of Combinatorial Optimization. 28, doi: 10.1007/s10878-013-9678-9.

[8] K. Taunk, S. De, S. Verma and A. Swetapadma, A Brief Review of Nearest Neighbor Algorithm for Learning and Classification, 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 1255-1260, doi: 10.1109/ICCS45141.2019.9065747.

[9] Guo, Gongde & Wang, Hui & Bell, David & Bi, Yaxin & Greer, Kieran. (2003). KNN Model-Based Approach in Classification. Lect Notes Comput Sci. 2888. 986-996, doi: 10.1007/978-3-540-39964-3_62.

[10] C. Z. Liu and M. Kavakli, Data-aware QoE-QoS management, 2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA), Hefei, China, 2016, pp. 1818-1823, doi: 10.1109/ICIEA.2016.7603882.

[11] Lattanzi E, Castellucci G, Freschi V. Improving Machine Learning Identification of Unsafe Driver Behavior by Means of Sensor Fusion. Applied Sciences. 2020; 10(18):6417. doi: https://doi.org/10.3390/app10186417

[12] R. Srinath, J. Vrindavanam, V. P. Vasudev, S. Supreeth, H. Raj and A. Kesarwani, A Machine Learning Approach for Localization of Suspicious Objects using Multiple Cameras, 2020 IEEE International Conference for Innovation in Technology (INOCON), Bangluru, India, 2020, doi: 10.1109/INOCON50539.2020.9298364.

[13] Y. Wang, J. Wong and A. Miner, "Anomaly intrusion detection using one class SVM," Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004., West Point, NY, USA, 2004, pp. 358-364, doi: 10.1109/IAW.2004.1437839.

[14] Li, Xiaolei & Han, Jiawei & Kim, Sangkyum & Gonzalez, Hector. (2007). ROAM: Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets, doi: 10.1137/1.9781611972771.25.

[15] Lee, Jae-Gil & Han, Jiawei & Li, Xiaolei. (2008). Trajectory Outlier Detection: A Partition-and-Detect Framework. 140-149. doi: 10.1109/ICDE.2008.4497422.