Tackling Threatening behavior through a Semantic Approach

Claire Laudy, Simon Fossier, Johann Dreo

Tackling Threatening behavior through a Semantic Approach

Claire Laudy
Thales
Palaiseau, France
claire.laudy@thalesgroup.com

Simon Fossier
Thales
Palaiseau, France
simon.fossier@thalesgroup.com

Johann Dreo
Computational Systems Biomedicine lab.,
Department of Computational Biology,
Institut Pasteur, *Université Paris Cité*, Paris, France
johann.dreo@pasteur.fr

***Abstract*—We introduce a new approach to characterize and detect threatening behaviors in surveillance systems, without relying on history or expertise. This approach consists in simulating the worst-case attack plans, fusing their semantic descriptions and using the produced patterns to raise alerts in operational conditions. We demonstrate our set of tools on a simple scenario involving geolocated sensors looking for moving vehicles targeting a protected objective. We find that the system is able to recover well-grounded graph patterns defining detection rules which make sense in the operational context. We believe that our approach achieves a relevant compromise between data-based and expertise-based systems, and allows for a good balance between efficiency and understandability.**

***Index Terms*—Trajectory Fusion, Optimization, Motion Planning, Semantization, Threat Detection, Situation Awareness**

I. Introduction

Surveillance systems are typically composed of a set of sensors that track targets on the zone, in the objective of raising alerts when a "threatening" behavior is detected. These systems are typically designed by an expert who knows the operational context and the definition of targeted behaviors. Modern approaches may also make use of data captured by an existing system in order to *learn* the targeted behaviors, and possibly improve the system design iteratively [1]. These approaches require to be able to actually identify a *threatening* behavior in a database of situations. Once a set of threatening behaviors is identified, it can be used to train a classifier, which will be able to detect if a given situation is threatening or not [2]. However, most classification approaches would not be able to easily transfer their knowledge to an explainable form, such as a geometric definition of a behavior [1].

In this work, we propose an intermediate approach which substitutes the need for an explicit definition of a specific behavior with an implicit characterization based on the sole definition of the target's *objectives*. A simulator generates expected behaviors by optimizing the attainment of the threat's objective and these optimal simulations are interpreted as a base of probable behaviors to reach the objectives. This is close to *goal-based* analysis [3], but the simulator models the whole behavior, instead of letting an expert defining key points on the trajectories. These optimized situations are then discretized, semantic information is attached to them thanks to the use of a generic ontology, and common explicit patterns are finally extracted among the threatening behaviors, which constitutes an explicit basis for generating alerts.

We thus provide a new way to raise alerts on threatening behaviors, which may complement more classical rule-based [4] or learning-based approaches [5]. This is particularly important for situations where no sufficient observations or expertise is available, which is a common real-life occurrence in surveillance systems [6].

We rely on the assumption that a threatening target, that aims at traversing the system with a given objective, will try to avoid detection while being efficient (typically, fast) in its traversal [7]. For instance, a vehicle will try to traverse a monitored area as fast as possible, in order to reach the exit point which allows for an escape [8].

The objective is to help surveillance operators in intercepting a target before it reaches a goal or hides. Technically, this translates into: how to automatically trigger relevant alerts, when there is no historical records of nefarious activity, and no expert to explicitly design the alert system? This forces to think in terms of states and optimal behaviors: how can we design a system that anticipates the behaviors of the antagonist, while focusing on the surveillance sensors that are available, and that raises an alert when the measurements match a behavior pattern? Moreover, how can we predict an intercept state and/or a goal state in which the target may be present in the future, to be able to stop it?

The advantage of our approach is that it tackles the problem by answering the following questions:

- What are the possible plans an attacker would follow if they wanted to attack the system?
- What would be the common features between these possible plans?
- How to leverage these commonalities to catch the attacker?

Prior art only considers learning approaches on raw data from the sensors, explicit rules, or discretization of features in a human behavior video [1]. To our knowledge, this work is the first to compensate for the lack of real labeled data by using fully optimized simulation, and semantic graph fusion to raise alerts.

II. Coupling Optimization and Fusion to Parameterize an Alerting System

Our approach relies on the synergy of three elements:

- Simulation of threatening targets by a planning optimizer, which computes a set of possible plans of attacks on the system. A large set of simulations, involving different threat assumptions, provides a consequently large set of attack plans.
- Semantization of those plans, i.e. their transformation into a concept-centric description, using a given ontology, which leads to a large set of semantic graphs describing the plans of attack. This graph set is then fused to obtain a description of the most common elements that characterize threats, i.e. the elements that the alerting system should try to identify – in other words, patterns of attack.
- When the system is in use, these patterns of detection are correlated with the sensor measurements: the measurements first go through the same semantization process, then are correlated with the fused patterns. When this behavior matching method returns a high level of correlation, it is interpreted as a potential threat, which triggers the raising of an alert.

In this work, we consider a geolocated system, which threat vehicles traversing a given zone. Sensors are placed to detect moving threats in their assigned detection area, are usually characterized by a detection range and sensitivity, but can also have highly anisotropic capabilities by design or due to the terrain (for instance, range difference in plains vs. forests). On the attacker side, the threat can move more or less easily, depending on its location (e.g. move fast on a road with a car, move slowly while walking in the forest). The target behavior itself can be summarized by a trajectory, from a boundary of the system's domain to another boundary. During detection, there are remaining unknowns regarding the attacker's capabilities (e.g. the type of vehicle they will use), and it can be necessary to enumerate and work with several options (e.g. the attacker would use either a car or a truck).

The overall process can be summarized as follows:

1) Threats are simulated by a motion planning optimizer that computes attack trajectories on the system. A large set of simulations, involving different threat assumptions, provides a consequently large set of plans of attack. This is our base dataset for describing the possible threats.
2) These plans are then semantized with the support of an ontology (description of the different elements that appear in the domain), which leads to a large set of semantic graphs. This is a simpler representation of the threats, which has the advantage of being discrete and understandable in terms of concepts, zones, speeds, etc.
3) The set of graphs is then fused, so as to obtain the common and/or most threatening parts in the semantic graphs. Each fused graph forms a pattern of attack. This is our reference information to raise alerts.
4) When the system is in use, the sensors provide a characterisation of the threat behavior (e.g. a trajectory, a type of vehicle…). This behavior is passed through the same semantization process, leading to a set of real-time semantized detections, now expressed in the same formalism as the patterns of attack.
5) The detections are finally matched with the patterns of attack, in order to raise progressively increasing alerts as the measured semantic graph matches an increasing sub-part of a pattern of attack.

At first glance, it could be thought that most of these steps have implementations in the state of the art. However, when taking into account the input-output formalism of the successive steps, as well as the required association of these steps with the constraints associated to the problem, this creates a new implementation problem in itself and specific challenges. More specifically:

- 1) + 2) : The semantization of plans of attacks into conceptual graphs is highly unusual, since plans are usually processed at the event level or at the kinematics level.
- 3) + 4) + 5): Matching real-life detections with semantic graphs without expert transformation rules is a challenge in itself, made possible by the semantization module that couples the zone/subnetwork description with attack paths and transforms them into a semantized graph, allowing for graph matching algorithms to be applied on this coherent dataset.

A. Semantic Trajectory Modelling

One key aspect of our approach is the semantic modelling of the trajectories. We choose to model them using conceptual graphs. Conceptual graphs [9] are a family of formalisms for knowledge representation, made of ontological and factual knowledge. A CG is a bipartite graph representing factual knowledge referring to a vocabulary that represents the ontological knowledge.

A vocabulary is a 5-tuple $\mathcal{V} = (T_C, T_R, \sigma, I, \tau)$. T_C and T_R, that respectively correspond to concept and relation types, are two partially ordered disjoint finite sets, where ordering corresponds to generalisation. T_C contains a greatest element $\top$. Each relation type has an associated arity. A signature specifies the arity and the maximal concept type of each argument of a given relation type. σ maps to each relation type r its signature $\sigma(r)$, i.e. a tuple $(t_1, ...t_n)$ where each $t_i \in T_C$, thus specifying its arity, n, and the most general concept type t_i for each of its argument. I is a set of individual markers used to instantiate concept nodes. τ is a mapping from I to T_C that defines the type instantiated by each individual marker.

A CG is a bipartite labeled multigraph represented as a 4-tuple $G = (C, R, E, \mathrm{label})$ defined over such a vocabulary $\mathcal{V}$. C and R correspond to concept and relation nodes, E denotes the set of the edges connecting elements of C and R. label is a labelling function from C to $T_C \times I$ and from R to T_R. For any $r \in R$, $\mathrm{label}(r) = t_r \in T_R$ is the type of r and for

any $c \in C$, $\text{label}(c) = (t_c, i_c) \in T_C \times I$ where t_c is the type of c

The vocabulary we developed to model semantic trajectories is made of an ontology and a set of signatures associated with the relations types defined in the ontology. The ontology is depicted on Figure 1

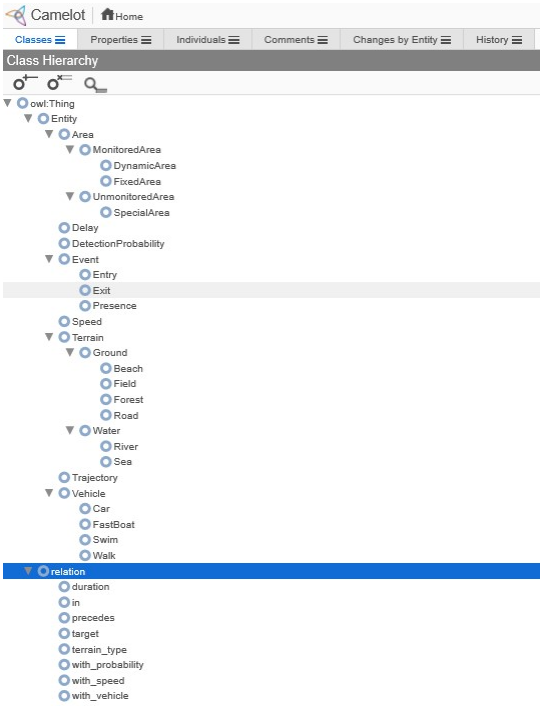

Fig. 1. Ontology for semantic trajectories modelling

We define the following relation signature in order to express the trajectories.

- $\text{in}(\mathit{Presence}, \mathit{Area}, \mathit{Duration})$,
- $\text{with\_speed}(\mathit{Presence}, \mathit{Speed})$,
- $\text{precedes}(\mathit{Presence}, \mathit{Presence}, \mathit{Delay})$.

An elementary unit of a trajectory (EUT) of a vehicle is defined by its presence in a given area for a given duration and with a given speed. A trajectory is defined as the succession of such EUTs, this succession being defined using the `precedes` relation defined above. Figure 2 depicts an example with a trajectory composed of a succession of *Presence* in the *radar2* zone and in the *uav1* zone.

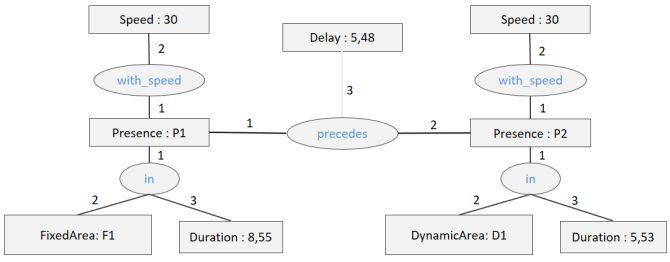

Fig. 2. Example of a semantized trajectory

B. Threat Planning Optimization

In this work, the objective of the threat model is to provide a set of paths which captures the worst case scenario, where the threat behaves optimally. This set $\Gamma^* \subset \Gamma$ encloses paths going from a "source" set of EUTs $\Upsilon \in \Omega$ (i.e. where the threat comes from), to a "targeted" set of EUTs $\Theta \in \partial\Omega$ (i.e. what the threat wants to reach by traversing the domain).

In this setting, the navigation of threats are a priori completely free, in the sense that it only has to satisfy a transport equation and thus satisfy locally the conservation of energy. More realistically, though, paths may be subject to momentum-related or environmental constraints, such as obstacles, radius of curvature (depending on the speed), bounded accelerations, bounded speed, etc. The model of navigation should thus be able to capture the characteristics of different vehicles. The following section introduces such models, and we refer the reader to [10] for more details.

1) Eikonal model: Solving such a constrained navigation problem is equivalent to solving the eikonal equation:

$$\begin{cases} |\nabla u(\boldsymbol{p})| = \frac{1}{c(\boldsymbol{p})} & \forall \boldsymbol{p} \in \Omega, \\ u(\partial\Omega) = 0 \end{cases} \tag{1}$$

for a given speed function $c : \overline{\Omega} \to]0, \infty[$.

The goal thus becomes to recover the value function $u : \Omega \to [0, \infty]$:

$$u(\boldsymbol{p}) := \inf \left\{ \text{length}_{\mathcal{F}}(\gamma) \middle| \gamma : [0,1] \to \overline{\Omega}, \gamma(0) \in \Upsilon, \gamma(1) \in \Theta \right\}$$

In the general sense, the path length should be measured as the sum of the length of (arbitrarily small) legs (i.e. sequence of two EUTs), with respect to a given metric $\mathcal{F} : \overline{\Omega} \times \mathbb{E}_d \to [0, \infty]$:

$$\text{length}_{\mathcal{F}}(\gamma) := \int_0^1 \mathcal{F}_{\gamma(t)}(\gamma(t)) \, \mathrm{d}t$$

$\mathcal{F}$ may be a conformal metric (locally proportional to the Euclidean one) of the form $\mathcal{F}(\boldsymbol{p}) = \|\boldsymbol{p}\|/c(\boldsymbol{p})$ (similar to equation 1).

The (generalized) eikonal equation then reads:

$$\mathcal{F}^*_{\boldsymbol{p}}(\mathrm{d}u(\boldsymbol{p})) = 1, \text{where } \mathcal{F}^*_{\boldsymbol{p}} := \sup\{\langle \hat{\boldsymbol{p}}, \vec{\boldsymbol{p}} \rangle \mid \mathcal{F}_{\boldsymbol{p}}(\vec{\boldsymbol{p}}) \leq 1\} \tag{2}$$

which allows to capture several anisotropic navigation constraints (e.g. radius of curvature, drift, detection probability, etc.) within $\mathcal{F}$, at each area of the domain.

It is worth noting that solving the eikonal equation 2 can be done efficiently in $\mathcal{O}(n \log n)$ by causal eikonal solvers, while also computing the vector field of optimal control $v : \mathbb{R} \to S^{n-1}$ at no additional cost, as:

$$\vec{\boldsymbol{v}}(\boldsymbol{p}) = \frac{-\nabla u(\boldsymbol{p})}{|\nabla u(\boldsymbol{p})|} \tag{3}$$

Integrating the vector field (3) from some sources $\boldsymbol{\psi} \in \Upsilon$ effectively recovers the optimal paths that reach the *nearest* —in the sense of the metric $\mathcal{F}$— targeted locations $\theta \in \Theta$. Note that the number of recovered paths is the same as the number of sources, effectively limiting the space of threat's behavior.

2) Sensor Network Traversal: Since we consider the problem of a threat traversing a domain on which a network of sensors is located, we define a "track" as a path linking a source point to the nearest entry in an area that the system is watching over. Sensors are modelled as areas where the system has a high probability of detecting (some kind of) vehicles. The cost metric $\mathcal{F}$ thus models the probability of detecting the threat for each point of the domain.

If $\mathcal{F}(\boldsymbol{p}) > 0 \, \forall \boldsymbol{p} \in \Omega$ (i.e. if there is a non-null probability of detecting a threat out of any sensor field of view), then the optimal paths resulting from integrating (3) are unique. Note that, in that case, if the threat does not traverse a sensor's field of view, then the problem falls back to a shortest path problem, hence producing realistic paths [11].

As an extension of this model, it is possible to set up a two players game model where a player optimizes the sensor network configuration against the player who optimizes the threat paths, without loss of generality (see [12] for an example).

In this work, we either use the HFM solver [13] to optimize threat tracks, or produce hand-made tracks with realistic shapes honoring equation 2.

C. Plans Semantization

The *semantization* consists in transforming a given threat plan into a graph representation following the ontology and relational structure defined in section II-A.

First, we use layers covering the navigation domain. Each layer is linked to a specific semantic meaning, such as the operational boundaries, the navigation speeds, or the probabilities of detection. Each layer is a partition of the domain, i.e. a set of non-overlapping layer subzones. Each layer subzone in those partitions is thus holding an atom of data with a semantic meaning ("detection = 50%", "boat speed = 12 kts", or even "no data").

The union of all layers produces a new partitioning of the domain in a set of non-overlapping areas of interest, each of which holds a *list* of data atoms, with one atom for each layer. An event is defined as an entry of a threat into an area of interest, or an exit from this area. The semantization then transforms a path traversing the navigation domain into a sequence of events.

In summary, the semantization process consists in:

1) computing the union of all layers,
2) discretizing the input threat track in a list of entry/exit events in/out of the traversed areas,
3) transforming this discrete sequence of events in a conceptual graph.

This results in tracks defined in the conceptual graph formalism, e.g., as "Presence of a SpeedBoat in MonitoredArea zone X1, with average speed 25, during a time period of 9, followed by a presence in zone Y2, with [etc.]".

The computational geometry functions are implemented with the CGAL library [14], using the kernel with exact predicates & exact constructions, and the 2D arrangements package [15].

D. Semantic Trajectory Fusion

The fusion process developed in the semantic information fusion module is a hybrid between the Apriori algorithm [16] and Taxogram [17].

Agrawal and Srikant [16] proposed the Apriori algorithm, for frequent item set mining and association rule learning from transactional databases. It operates on databases containing transactions (i.e. collections of items). Each transaction is seen as a set of items (an "itemset").

The algorithm discovers frequent item sets in the database in a "bottom up" approach. It extends already processed frequent item sets with new items. Thus, the algorithm first processes the frequent item sets of length 1 and then extends them one item at a time. The frequency condition is tested by processing the support of an item set in the database and comparing it to a threshold. The support of an item set is the number of times this item set appears in the database. Given a threshold ϵ, the Apriori algorithm identifies the item sets which are subsets of at least ϵ transactions in the database.

Taxogram is a taxonomy-superimposed graph-mining algorithm that can efficiently discover frequent graph structures in a database of taxonomy-superimposed graphs. Taxonomy-superimposed graph describes graphs which nodes take value in a taxonomy. It is very close to our conceptual graphs, the nodes of which take types in the taxonomy part of an ontology. However, in conceptual graphs, concept nodes are not only composed of types, but also value, that are not defined in the taxonomy part of the ontology.

Taxogram has two main properties. First, it performs a subgraph isomorphism test once per class of patterns, which are structurally isomorphic, but have different labels. Then, it reconciles standard graph mining methods with taxonomy-based graph mining and takes advantage of well-studied methods in the literature.

The Taxogram algorithm is composed of three stages:

1) relabelling nodes in the input database,
2) mining pattern classes/families and constructing associated occurrence indices, and
3) computing patterns and eliminating useless (i.e., over-generalized) patterns by post-processing occurrence indices.

In the Semantic Information fusion module, as in Taxogram, the graph data base is first generalised. Each concept of each graph of the database is generalised. That is to say that the type of concept is replaced by its most general super-type in the ontology that is also a subtype of Entity.

Once the graphs of the database are generalised, they are transformed into itemsets, in order to be processed by Apriori. Each relation of the graph, with its linked concept nodes is transformed into an item. The set of relations composing each track graph is thus transformed as an itemset. The Apriori algorithms processes a set of itemsets, searching for frequent patterns into these itemsets.

Once frequent generalised patterns are provided by Apriori, we use the approach described in [17] in order to process the

specific patterns.

E. Attacks Detection

The challenge of this final step is twofold: align the real trajectories with the frequent patterns extracted by Semantic Fusion, and define the explicit rules for triggering alerts.

Actual trajectories are provided by the different sensors on the monitored zone. Here, our base material is *system tracks*, which stem from the kinematic fusion of sets of *sensor tracks* associated with a single object detected by multiple sensors. For this paper, we make the assumption that there is a single system track per actual moving object, although our tests showed that, more often than not, an important preprocessing work is required, e.g. to manage track drops due to blind zones or objects closer than sensor resolution.

To align the system tracks with the semantic frequent patterns, we first use the same semantization tool as described in II-C. This ensures that the semantized version of the system trajectory follows the same ontology as the frequent patterns.

Then, we need to match zone transition sequences, i.e. the semantized track, with the frequent patterns which characterize the threat. Though seemingly straightforward, this matching is affected by:

- variability: the threat patterns provide hints about, e.g., expected speeds during a zone crossing, or time necessary to cross a zone, but the actual detections will never perfectly match these average/median/fused times. It is therefore not possible to use purely Boolean predicate-based matching systems, and we need to rely on probabilistic or threshold-parameterized rule systems.
- observability: threat models are based on motion planning in a fully modelled zone, but in actual contexts the threats are not guaranteed to be observable with 100% uptime. If a given zone is not or poorly covered by the detection tools available to the system, the track will either lack a portion of the plots that in this zone (best case) or even be discarded, leading to a new track created when newly re-detected in the next zone, with a new identifier. This is linked to the preprocessing mentioned earlier. Typically, a mitigation system has to be put in place, so as to both manage the inobservable zones and perform an hypothesis-based matching between partial tracks so as to recreate the full behavior of the threats.
- quantization/sampling side effects: the fact that the observations and the threats are described in a discrete paradigm results in classical time-space alignment issues, and some measure of interpolation is necessary.

After alignment, we define a set of metrics from the pattern/track correlation, which include: track vs. pattern speed alignment; zone traversal time alignment; presence/absence of missing segments in tracks vs. patterns.

Moreover, a dynamic alert system requires *timeliness*, i.e. the capability to raise alerts at the right time, with as much early notice as possible. By design, the threat patterns span the whole threat behavior, from early detection to threat realization. Therefore, we cannot wait until the whole threat pattern is matched by a trajectory to raise the alarm. This is managed by defining a level of completion of the patterns, i.e. a current position in the threat pattern (close or far to the objective).

The track is also complemented with contextual information, issued from the system identification capabilities. This information is either (1) controlled/measured – based on sensor measurements, e.g. video-extracted; (2) declarative – retrieved directly or indirectly from reports from the objects, e.g. AIS; (3) curated – aggregated in trustworthy databases with some human verification involved, e.g. ship registration databases.

Finally, all these metrics are merged into a rule system tuned with the stakeholders, to trigger alerts. These alerts are raised or dropped dynamically, and complemented with high-level indicators, such as a confidence level, an estimated time to completion, or hints on targets or geographical elements to be highlighted, based on the part of the frequent patterns which is still to happen.

III. Example

We applied the approach on several scenarios, with contexts ranging from a couple of departure and target points, to much bigger ones. We describe in the following section one of our middle size scenarios and present the results we obtained in terms of detection rules.

A. Scenario

The scenario depicted on Figure 3 takes place around the island of Salamine in the Saronic Gulf. The set of targets for the attackers, depicted in red on the figure, is situated in the inner part of the Eleusis Bay, along the coast. Possible departure points, depicted in yellow, are situated in the Saronic Gulf open sea, on the other side of the island.

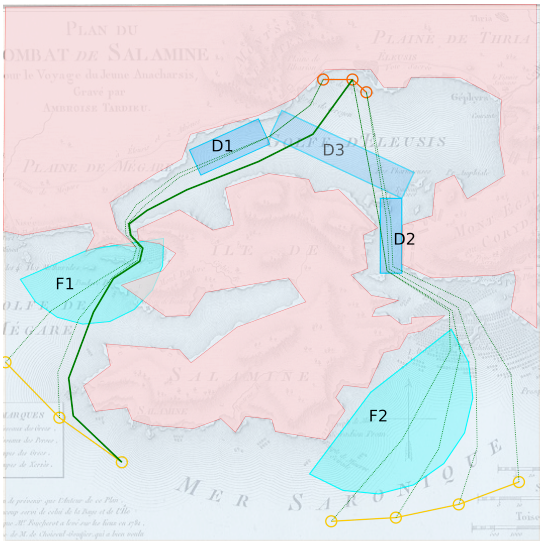

Fig. 3. Salamine Island Attack scenario

The area is too large to be fully covered by detection systems. Two fixed-position radars are located in the outwards coast, depicted in light blue circular sectors and referred

as F1 and F2 in the following. In addition, depicted in darker blue rectangles, two uncrewed aerial vehicles (UAV) provide dynamic detection systems, referred to as D1 and D2, and patrol within the gulf in predetermined flight patterns. Notably, it can be assumed that potential attackers are aware of the radars range and detection efficiency, since they are permanently installed, but not of the UAVs.

B. Threats simulation and tracks semantization

In order to keep the example visually interpretable, we generated only a few threatening tracks for this example. These tracks are depicted on figure 3 as the green lines between the possible departure and target points.

The tracks are then semantized so that they are transformed into semantic graphs before the fusion step. Figure 2 depicts the resulting semantic graph for the track depicted in plain line. This track first crosses the F1 detection area, then avoids the D1 area by bypassing it on the right and finally crosses the D3 detection area.

C. Track fusion

The fusion module aims at finding recurrent patterns across the different tracks, in order to generate detection rules from them. We want to detect with the same rule all tracks that have close trajectories; therefore, the patterns have to be underspecified, with regards to the input semantic tracks. However, as depicted on figure 2 each graph track has specific (and usually different) values for the duration of the presence of the vehicle in an area, speed of the vehicle and delay between the crossing of two successive areas. Therefore, the fusion algorithm needs to integrate a set of parametered fusion heuristics, which fuzzify the values by associating them to ranges of speed, duration and delay.

Second, the fusion algorithm needs to be parametered with the minimal *support* desired for each pattern graph (and thus detection rule), over the initial data set of input tracks. Here, a graph pattern is said to be *supported* by a set of input graphs if it *represents* these input graphs, i.e. the support of a graph pattern is the proportion of graphs in the input data set that are more specific than the graph pattern.

Obviously, we don't want to have too many rules, since it would reduce the interpretability of the detection rule set by operators. On the other hand, if rules are too generic and support too many threat configurations, the information they carry may be of little use to the operator: a rule being triggered would not give enough specific information on the behavior of the vehicle for an appropriate action to be taken.

For this example, if we target a minimum support of 50% for the graph pattern, we obtain only one frequent pattern, representing tracks that cross the D2 and then the D3 areas. This pattern is depicted on figure 4. None of the northern tracks are represented, and no difference is made between tracks coming from the east (not crossing F2) and the ones coming from the south (crossing F2).

When we lower the minimum support for graph patterns to 25%, two additional patterns are found (and depicted on figure 5) by the fusion algorithm. These patterns may then be translated into detection rules.

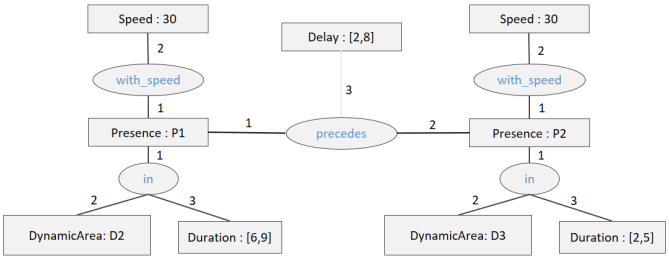

Fig. 4. Graph pattern with support of 50%

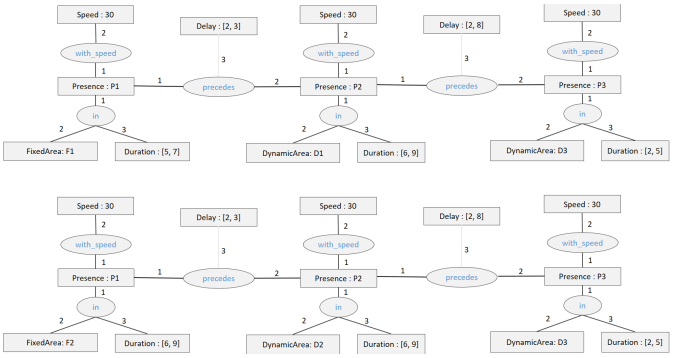

Fig. 5. Graph pattern with support of 25%

D. Detection rules

Semantization allows us to match the detected tracks with the graph patterns. The matching is checked for each threat and each threat pattern, reiterated over time, and is performed by following the sequence implied by the `Precedes` relation.

Each elementary unit of trajectory (EUT, linked by `precedes`) involves multiple values. Atomically, the comparison is straightforward when the values associated to the extracted concepts are fuzzified (here, `Duration` or `Delay` intervals) and imply a notion of membership of the semantized value to the fuzzified value. When the fused value is sharp (here, `Speed`), the comparison will require a heuristic membership criterion, typically a fuzzy or sharp threshold in the difference between values.

Then, at a given point in time, a potential threat will have matched a certain number of sequential EUTs. We check whether the threat pattern is matched from the beginning, if there are additional or missing EUTs, and how many EUTs in the pattern have been matched at that instant. From these criteria, we build, in link with the stakeholders, an aggregated threat level and a scale of operational alert levels.

Finally, during operation, threats will be continuously matched against patterns, their threat levels evaluated, and alerts raised with the appropriate levels, with the estimated time to completion, and with the estimated end zone (linkable to the targeted asset).

IV. Conclusion and perspectives

In this paper we presented an approach for building a surveillance system based on simulation of attacks rather than on the expertise of an operator defining specific threatening behaviors within specific contexts, or on a labeled dataset. The process we propose follows 5 steps:

1) the simulation of threatening behaviors (tracks generation),
2) the semantization of these tracks,
3) the fusion of the semantized tracks during the system preparation phase,
4) the semantization of tracks provided by sensors during operation,
5) the detection of threatening vehicles which behavior matches the behavior patterns found on step 3.

While we present here the use of the process with the aim of automatically generating behavior rules that are understandable by a human operator, a perspective of our work is to use it in order to configure the sensors as well. To do so, we can take advantage of the fact that threatening tracks are simulated. The idea is to add a system that will automatically optimize sensors positioning, in order to maximise the probability of detection of a vehicle (see [11], [12] for examples, albeit not involving semantic fusion). The process described here then remains unchanged: semantic behavior rules, associated with the best sensor positioning will be generated, and the final probability of raising an alert will be used as an objective function of a high-level optimization problem.

Furthermore, as the whole process would be automated, one may be able to find the best operational compromise between the probability of detection and the understandability (and thus usability) of detection rules.

More generally, the work described in this paper may also be applied on any topology that features a generic notion of "plan". We intend to apply it within the cybersecurity domain, i.e. the study and monitoring of actions in information systems networks. In such a use-case, the semantization of a plan resulting from an automated planning problem optimization may even bring more information to the following layers.

References

[1] K. K. Verma, B. M. Singh, and A. Dixit, "A review of supervised and unsupervised machine learning techniques for suspicious behavior recognition in intelligent surveillance system," *International Journal of Information Technology*, pp. 1–14, 2019.

[2] A. Joshi, N. Jagdale, R. Gandhi, and S. Chaudhari, "Smart surveillance system for detection of suspicious behaviour using machine learning," in *Intelligent Computing, Information and Control Systems*, A. P. Pandian, K. Ntalianis, and R. Palanisamy, Eds. Cham: Springer International Publishing, 2020, pp. 239–248.

[3] F. Tung, J. S. Zelek, and D. A. Clausi, "Goal-based trajectory analysis for unusual behaviour detection in intelligent surveillance," *Image and Vision Computing*, vol. 29, no. 4, pp. 230–240, 2011.

[4] W.-K. Lee, C.-F. Leong, W.-K. Lai, L.-K. Leow, and T.-H. Yap, "Archcam: Real time expert system for suspicious behaviour detection in atm site," *Expert Systems with Applications*, vol. 109, pp. 12–24, 2018.

[6] D. Martinez Torres, H. Loaiza Correa, and E. Caicedo Bravo, "Online learning of contexts for detecting suspicious behaviors in surveillance videos," *Image and Vision Computing*, vol. 89, pp. 197–210, 2019.

[7] J. Dreo, F. Desquilbet, F. Barbaresco, and J.-M. Mirebeau, "Netted multi-function radars positioning and modes selection by non-holonomic fast marching computation of highest threatening trajectories by cma-es optimization," in *2019 International Radar Conference (RADAR)*, 2019, pp. 1–6.

[8] C. Strode, "Optimising multistatic sensor locations using path planning and game theory," in *2011 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2011, pp. 9–16.

[9] M. Chein and M.-L. Mugnier, *A Graph-Based Approach to Knowledge Representation: Computational Foundations of Conceptual Graphs (Part. I)*. Springer, 10 2008.

[10] J.-M. Mirebeau, "Numerical schemes for anisotropic PDEs on cartesian grid domains," Habilitation à diriger des recherches, Université Paris-Sud XI, May 2018.

[11] Q. Renau, "Landscape-Aware Selection of Metaheuristics for the Optimization of Radar Networks," Theses, Institut Polytechnique de Paris, Jan. 2022.

[12] J.-M. Mirebeau and J. Dreo, "Automatic differentiation of non-holonomic fast marching for computing most threatening trajectories under sensors surveillance," in *Geometric Science of Information - Third International Conference, GSI 2017, Paris, France, November 7-9, 2017, Proceedings*, ser. Lecture Notes in Computer Science, F. Nielsen and F. Barbaresco, Eds., vol. 10589. Springer, 2017, pp. 791–800.

[13] J.-M. Mirebeau and J. Portegies, "Hamiltonian Fast Marching: A Numerical Solver for Anisotropic and Non-Holonomic Eikonal PDEs," *Image Processing On Line*, vol. 9, pp. 47–93, 2019.

[14] A. Fabri and S. Pion, "Cgal: The computational geometry algorithms library," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 538–539.

[15] R. Wein, E. Fogel, B. Zukerman, and D. Halperin, "Advanced programming techniques applied to CGAL's arrangement package," *Computational Geometry*, vol. 38, no. 1, pp. 37–63, 2007, special Issue on CGAL.

[16] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, p. 487–499.

[17] A. Cakmak and G. Ozsoyoglu, "Taxonomy-superimposed graph mining," in *Proceedings of the 11th Int.Conf. on Extending database technology: Advances in database technology*, 2008, pp. 217–228.

[5] A. Wiliem, V. Madasu, W. Boles, and P. Yarlagadda, "A suspicious behaviour detection using a context space model for smart surveillance systems," *Computer Vision and Image Understanding*, vol. 116, no. 2, pp. 194–209, 2012.