Fielding Artificial Intelligence techniques for radar object behavior analysis on High Performance Computing (HPC) hardware

Richard Stottler[1]
Stottler Henke Associates, Inc., San Mateo, CA,94402

The ultimate goal of this effort is to improve the ability to determine suspicious radar track behavior and suspicious areas to focus attention on. We are developing a framework for representing the knowledge and human-quality reasoning required to process large quantities of radar data, transforming it into a form that can be efficiently executed in real time on an HPC system, and then efficiently executing it given the actual dynamic tactical situation. In addition to being scalable up to a large number of pixels and objects, it should also be adaptable both in the long term to different HPC configurations and in real time to different dynamic computational loads.

Nomenclature

BTN	=	Behavior Transition Network
CBR	=	Case-Based Reasoning
CPU	=	Central Processing Unit
FL	=	Fuzzy Logic
FLOPS	=	Floating Point Operations per second
GMTI	=	Probabilistic Road Map
GPU	=	Graphics Processing Unit
HPC	=	High Performance Computing
MHR	=	Multiple Hypothesis Reasoning
MIC	=	Many Independent Core
SAR	=	Synthetic Aperture Radar
SIMD	=	Single Instruction Multiple Data
TMS	=	Truth Maintenance System

I. Motivation And Problem Description

Modern warfare for the United States military is characterized by asymmetric forces and tactics, often operating in urban environments to conceal their activities. Cities feature complex structures, dense populations, an enormous amount of activity, and many different organizations operating within them. In response to these tactical realities, the US Air Force has developed ever-more-capable sensors with finer and finer resolution, broader areas of coverage, increased persistence, and multiple types of data being available. Of course, these improved sensors produce far more data than previous generations. It is easy to imagine radar systems producing frames requiring over 5 Gigabits of throughput for fine spatial resolution and to track thousands of objects. Clearly this is too much data for humans to process and an automated system is necessary to cue a human operator's interest (and/or other sensors or sensor modes) as to specific areas or objects to focus additional information gathering and threat assessment on.

But such an automated system should bring significant intelligence to bear on the problem. When expert human operators have more time and a much smaller area of responsibility (and therefore far less data), they can bring significant knowledge and experience to bear, including knowledge of the road system (intersections, stoplights, signs, traffic laws, etc.), knowledge of traffic patterns based on time of day, day of week, and season and other patterns of normalcy, knowledge of significant locations, and the ability to pay attention to a few possible threat

[1] President, 1670 South Amphlett Blvd., Suite 310, San Mateo, CA, 94402, AIAA Member.

vehicles, analyzing their behavior to determine how much of a threat they represent either to friendly forces or to protected high-value landmarks. What is required is a way to automate this level of human-quality reasoning.

However, processing that quantity of data in real time is not possible with conventional hardware. High Performance Computing (HPC) is required, which introduces its own issues. Successful HPC systems are heterogeneous, being comprised of conventional X86 CPUs augmented with various types of accelerators such as Graphical Processing Units (GPUs), Many Independent Core (MIC) products, etc. Each of these has its own strengths and weaknesses as to what type of computations it is best for and within each type there are variations that have to be considered. Trying to directly implement cognitive, knowledge-based reasoning on an HPC system would be extremely difficult. What's needed is a framework for representing the knowledge, heuristics, and the human-quality reasoning required to process large quantities of radar and/or other sensor data, transforming it into a form that can be efficiently executed in real time on an HPC system to process very large quantities of data, and then efficiently executing it given the actual dynamic tactical situation, which leads to varying number of objects and varying different types of processing and the resulting real-time varying processing load on each HPC component. In addition to being scalable up to a large number of pixels and objects, it should also be adaptable both in the long term to different HPC configurations and in real time to different dynamic computational load levels.

In real time, there should be an intelligent examination of each tracked vehicle's behavior and other information available to determine its likely intent and possibly hostile plans. This analysis takes at least two forms. One is a determination of whether, given the time of day, day of week, and the vehicle's type, location, speed, and relatively recent history, its behavior is abnormal. For example, is it typical for a large truck to turn that direction at that intersection at that time and day of the week? The second form of analysis is of whether the behavior conforms to known hostile tactics, either learned or predefined.

With enough personnel and enough time (perhaps, in the extreme, one dedicated to each track), the behavior of every track could be analyzed and suspicions raised appropriately for proper situational awareness. Of course, that quantity of manpower will never exist. There are far too many tracks and far too few personnel to pay close attention to all of them and to remember their previous histories. What is required is for the human expertise described above to be implemented in software and effectively replicated for every track. This is the realm of artificial intelligence.

In addition to the tactical problem described above, there is also a technical one. To analyze the behavior of each vehicle requires that the proper radar tracks associated with each vehicle in different frames be properly associated with each other. This problem arises if two or more tracks become close enough that, given the 1 second between frames, it is not obvious which is which. If the tracks have been previously tracked (perhaps all the way from their point of origin (e.g., a house), which fairly uniquely identifies them), they should now be treated as ambiguous, where each can be hypothesized to be either of the original two. Knowledge-based correlation (as described in the System Description Section) can normally be used to resolve these types of situations quickly. This data available includes dimension estimates for the vehicle, rough shape, and the amplitudes of and relationships among the different magnitude scatterers. These will depend on the vehicle, aspect, and how it is currently configured and loaded. Normally, between frames, little time has transpired to allow any of these factors to have changed. However, if more time has passed (because of occlusions, tunnels, more time between frames due to processing limitations, etc.), the aspect may have changed substantially. So the association of radar tracks from different frames should be performed using an intelligent approach that considers all the available information.

II. System Description

A. Overview

The required reasoning is represented in the HPC AI Framework (HAIF) and is translated into forms most appropriate for execution by the different types of HPC components, including highly parallelizing the computations. During real-time sensor data processing and execution, HAIF monitors the processing performance and available capacity of each HPC component and the number of objects and tactical cognitive processing required and makes real-time decisions as to what processing is most important to be done (normally in terms of geographical areas and tracked objects) and optimally schedules it on the available HPC resources. Monitoring the available capacity allows other, external applications to coexist on the HPC system. Note that the translation process may produce more than one executable representation for the same small cognitive inferencing component, one for each type of hardware that it could possibly be executed on. This gives the real-time processing scheduler more options to more effectively balance the processing load, if needed.

The required processing includes vision processing of the pixel data to extract objects and their features, knowledge-based correlation to correctly associate objects in one frame with the same object in other frames,

knowledge-based analysis of the object's behavior, including comparison to past measures of normalcy, and recommendations based on this analysis. Obviously, the better the ability to correctly associate objects in consecutive frames, the longer period of time that objects can be reliably tracked, and the longer the tracking time, the more information is available for analysis. So if association performance can be substantially improved, the performance of the entire system will be enhanced.

B. Knowledge-Based Radar Track Correlation

To improve correct frame-to-frame object association performance, a significant amount of a priori information, knowledge, and inference techniques should be utilized. First and foremost is the road system, including intersections, traffic lights, stop signs, and speed limits. A priori maps can be augmented with information learned over time such as parking and turn locations beyond the known intersections. Kinematic limits on the vehicles are also very useful. Effectively no street legal vehicles can accelerate or decelerate faster than 1 G (32 ft/s2), which means that using a vehicle's velocity from one frame to predict where it will be in the next frame, 1 second later, will be within 16 feet of correct ($d = \frac{1}{2} a t^2$), ignoring other errors and noise. Vehicles almost always stay on the road network, which means that the Doppler velocity can be translated into the actual velocity vector, using this assumption. Furthermore, if the assumption is incorrect (i.e., that the vehicle is leaving the road), then that will become evident in the next few frames. Also, given the kinematic limit, the 1 second between frames is not nearly long enough for a vehicle to accelerate, pass another vehicle, and decelerate so that two vehicles cannot swap their relative positions without an indication beforehand and afterward. Vehicles lined up at stop signs and stop lights cannot pass through each other. Performing the correlation of vehicles between frames in the order that the vehicles are queued up at an intersection allows a very rapid, one pass correct association of the vehicles.

(Using the Doppler velocity with the road system to estimate the velocity vector only breaks down when the relevant stretch of road is nearly orthogonal to the object's relative position vector (from the perspective of the radar system). This should only occur for a small percentage of the road segments and the aircraft can position itself to ensure that it will not occur for important areas (since in any particular small (especially urban) area, roads are normally laid out in an orthogonal grid).)

In addition to the detection, position, and Doppler information, the radar will also provide 1-foot pixels. The rough size, shape, and scatterer information derivable for each vehicle from these pixels provides supplemental correlation information and/or confirmation that the correct objects were associated using the other means described above. (Frame to frame, the aspect will change very little, except when making a turn, and this will be indicated as a possibility by a slow speed when approaching a potential turn location and confirmed after the fact by the location of the object in a later frame.)

C. Considerations for Real-Time Scheduling of HPC Resources

Given the very large number of radar pixels and objects to be tracked; the additional information, knowledge, and heuristics that should be intelligently applied to this large amount of data; that a HPC architecture will be used to apply the knowledge and heuristics and process the data; and that this HPC architecture will likely consist of many specialized processors, some of which will have different capabilities than others, efficiently scheduling which processing should be performed, when, and on which specific processors and/or cores, is not a trivial problem.

Certain objects and areas will likely be considered more significant than others. And "significance" is a concept with many dimensions. One is time. Even if a vehicle is considered likely to be hostile, the time by which action needs to be taken, either because in that time the vehicle can cause harm or it can get away, will still vary. (E.g., a very hostile vehicle, in the middle of a long desert road, may be important, but there may be plenty of time to do something about it.) Another is the magnitude of the capability and intent to do harm, how important the vehicle is or how much damage can it inflict. E.g., a large truck filled with enemy troops or explosives may have large magnitude as could a small car that happens to have in it the leader of a terrorist organization or its lead bomb maker. A final dimension is probability: how likely the importance dimension is to be correct. E.g., there may be indications that a truck is filled with explosives and is under the control of a hostile organization, but the probability (based on the observed behavior and other evidence) may be high or low. All three factors have to be considered when allocating computing resources to the tracking and analysis of such a vehicle, and how each will be used will depend on the degree to which the HPC hardware is currently being overloaded (or not). For example, if HPC resources are very tight, such that many areas and objects cannot be fully processed, then objects with a low probability of suspicion will likely not be processed. But if more HPC capacity is available, then perhaps processing even low-probability, high-magnitude objects makes sense in order to minimize potential damage.

Other additional information, useful when determining the suspiciousness of an object, the importance of an area, or normalcy of traffic patterns, includes significant locations such as schools, hospitals, houses of worship,

government buildings, military installations, notable landmarks, and commercial areas. Also useful are areas that can be categorized (by ethnicity or socioeconomic status, for example) such as neighborhoods, towns, or areas controlled, infiltrated, or sympathetic to specific organizations.

D. Useful Heuristics for Analyzing radar tracks overlaid onto a road network

Generally it makes sense to assume that benign vehicles will move from their departure point to a destination point in the most time-efficient or distance-efficient manner, making turns as needed that are not correlated with any other vehicles. (There are some exceptions, such as when a wedding party all proceeds together from the place of the wedding to a reception location.) Using this fact and knowledge of specific significant locations or areas, vehicle behaviors can be automatically analyzed in isolation and in comparison with other vehicles, including friendly patrols and other protected assets. Behavior that is suspicious requires greater scrutiny, which could affect the processing schedule or the schedule of ID assets (such as other sensors, UAVs, USVs, manned helicopters, or even to provide a "heads up" to a human at an upcoming (from the vehicle's perspective) checkpoint) to take a closer look. The concept is fairly simple. Basically, non-tactical vehicles behave in predictable ways depending on what they are doing—commuting to work, going shopping, making deliveries, etc. So whenever a vehicle makes a turn, a quick check is done to see if it might be reacting to any other object in the vicinity or if it is contrary to its obvious type and recent behavior. Several correlated turns with the same net effect (e.g., following or avoiding) invite suspicion. Additionally, currently tracked vehicles' current locations, speed, direction, points of origin, type, etc. could be quickly checked against historic norms for the same time of day, day of week, and season to see how abnormal it is. Obviously, abnormality tends to evoke suspicion.

After radar tracks have been correctly associated to form a single, reasonably reliable, coherent path (of any length) for each vehicle, a two-pronged approach can be used to analyze the entire (possibly short) history of that path. The first prong, looking for deviations from normal behavior, relies on a database of past track path history.

Normal, benign (non-hostile) human activity has daily, weekly, and seasonal rhythms. These can be exploited to establish normal traffic patterns and deviations from the norm in several different ways. Given the vehicle gross type (e.g., sedan, pickup truck, van, large truck, etc.), location, speed, heading, turn intersection, direction if it just turned, time of day, day of week, and month, similar previous tracks can be retrieved (using Case-Based Reasoning (CBR)) and statistics calculated as to how normal these parameters are. If a large number of vehicles of this type are often heading in a similar direction and speed on this road segment or at this intersection during this time of day and week, this would constitute normal behavior. (Knowing the neighborhood where the vehicle originated would significantly improve the precision of this step, hence the desire to improve frame-to-frame association performance.) If however this combination of parameters is relatively rare, the vehicle should be considered suspicious. Similar reasoning can be applied to groups of vehicles close together and following the same path (like a convoy) over a short period of time. Based on a current group, past similar groups can be retrieved and examined to determine if any parameters from the current group are abnormal, such as the size of the group. In cases where individual vehicles have been tracked from their originating address (which when combined with their type fairly uniquely identifies them), past paths involving this same vehicle can be retrieved. What is this vehicle normally doing at this time and day of the week? Has it been in this location heading in this direction before? Again, deviations should engender suspicion. The degree to which a vehicle or group of vehicles is suspicious can be reported in its own right as well as being folded into the analysis, looking for specific evidence of hostile intent or specific known enemy tactics.

The second prong involves analyzing the behavior of the full history of each vehicle's path and correlations between the vehicle's turns and path and turns and paths of other vehicles. Correlations between a vehicle's turns and the turns of friendly units or ID assets (such as a helicopter or UAV) may be indications of hostile intent and should at least be labeled suspicious. Correlations to look for include following and avoiding. Often attacks of several vehicles are initiated from the same location and time. They may proceed together in convoy, at least for some time. Some vehicles may trail a moving target while others speed ahead on parallel roads to set up an ambush.

E. Efficiently Mapping Computations onto HPC Components

The main HPC component types are CPUs, GPUs, and MICs, and each has its own strengths and weaknesses and is more or less appropriate for different types of computations. And of course within each category, specific products will have individual differences that impact these issues. For example, GPUs can deliver a huge number of Floating Point Operations per second (FLOPS) for highly parallel computations executing in a Single Program, Multiple Data (SPMD) style. This makes them ideal for processing large numbers of pixels in either computer vision or computer graphics applications. However, they are less suitable for branching logic (loops, conditions, if-thens, case statements, etc.), which would be common for the type of high-level reasoning discussed above for both object

behavior analysis and some of the frame-to-frame object association. CPUs and MIC products, since they are essentially many independent general purpose processors, have probably the most general applicability, though are not necessarily the fastest at any single type of processing. They could perhaps be the most useful "filling in the gaps" when processing of a certain type exceeds the available capacity for the type of processor that is most suited for that type of processing. And of course, since all the technologies are parallel in nature, parallelizing the computation is an important aspect, though the form of this parallelization differs for each type of HPC component. (E.g., effective use of MIC products really only requires that the computation be broken down into enough threads, whereas more effort is required to make best use of GPUs.)

There are a number of AI and other techniques that will also be used in HAIF. Fuzzy Logic (FL) and behavior transition networks (BTNs), which were originally developed to simulate real-time decision making in tactical situations, are used to analyze track behavior. These FL rules and BTNs typically utilize various vector operations applied to track velocities, relative positions, and other vectors to determine intercepting paths, relative velocities, traveling along typical routes, or turning off them. A track's path history is broken into road segments connected by turns. The recognition that a segment has ended and a turn initiated triggers much of the analysis processing. When a turn completes, an analysis is conducted relative to recent turns of other tracks to determine whether the new segment's velocity, relative to the velocity of the track before the maneuver in relation to another track that recently maneuvered, tends to maintain a nearby parallel course, maintain an intercept, maintain a particular distance, or maintain an avoidance path where that avoidance was degraded by the other track's earlier turn. Multiple turns by a track correlated in the same way to another track or group of tracks (e.g., multiple turns of a single vehicle all correlated to turns of friendly units that all tended to restore avoidance of the friendly tracks) increase the level of suspicion.

Some ambiguities may be left after the knowledge-based correlation step. Keeping track of these different hypothetical correlations and quickly determining the ramifications when one or more of the most likely ones has to be reversed falls in the realm of Multiple Hypothesis Reasoning (MHR), also called Truth Maintenance Systems (TMSs). Truth maintenance software already exists that is ready to be translated to HPC hardware and applied to this application to manage the multiple possible (parallel) hypotheses resulting from ambiguities in use-type classification (especially as relate to consistency with the track's behavior (e.g., a track with movement patterns resembling a delivery truck that has been otherwise classified as either a delivery-type truck or an SUV), multiple explanations for an observed behavior, or multiple possible correlations for one vehicle track with multiple others).

The off-line component of HAIF includes a knowledge editing environment that provides facilities for entering various kinds of tactical reasoning knowledge that will be applied to the processed radar system data. This information is the basic input to the Compiling and Translating process, which transforms the knowledge into a form that can be rapidly executed on large quantities of data on the HPC system.

A major component of the representation framework is anticipated to be BTNs. We have already developed a translation mechanism from the graphical BTNs to C++ code. This can be adapted slightly to produce code that existing specialized compilers can use to produce GPU-optimized executables. The BTNs are already replicated for each object they apply to so they are already in highly parallelized form.

Parallelization is an important aspect of efficient HPC utilization as already alluded to. To divide up the knowledge-based correlation and behavior processing of objects requires dividing up the objects into non-interacting (or minimally interacting) sets. This is most easily done based on proximity to the same intersections. Each intersection represents the center of an area that can be approximately completely processed separately from all other intersections/areas. (Intersections that are very close together should be grouped together.) Objects should be grouped based on the intersection they are closest to with a bias toward favoring intersections in front of their direction of travel. This division of objects will minimize interactions of objects across areas during all phases of the processing. For example, pixel processing (though not a focus of this effort), is limited in terms of interactions to the small number of pixels that make up an object. Similarly, frame-to-frame object correlation inherently involves objects very close together, geographically. There is always the possibility that nearby objects could be on opposite sides of any area border, but objects tend to be closest when they are stopped at an intersection, so centering the area on the intersection will inherently tend to keep the objects likely to be close together in the same area. This same reasoning applies to behavior analysis (i.e., that objects that are influencing the behavior of each other must be usually fairly near to each other).

Shown below is the ultimate real-time HAIF architecture. It includes an HPC hardware system consisting of 0 to many GPUs, CPUs, and MIC chips. The HPC hardware (HW) includes or is interfaced to high-speed memory, which also receives data from the radar system. This memory also stores any other data that must be input into the HPC components, such as the knowledge transformed into executable form off-line, as described above. Frames of radar data are processed in parallel and pipeline fashion as shown below. First, computer vision algorithms process

the pixels into objects and extract feature data for each object. Second, Knowledge-Based Radar Track Correlation uses all available data to associate radar tracks from different frames in order to assemble a coherent path for each tracked object as described in Section 1.3. It will likely utilize multiple frames of data. Specifically, when associating objects between two adjacent (in time) frames it may also access frames 1, 2, and 3 seconds before or after the two frames in question. This way it can easily verify that its assumptions held a short time into the future (such as a track travelling on a road and not leaving it in the near future or evidence of starting to decelerate a few seconds before turning).

Remaining ambiguities are passed to Multi-Hypothesis Reasoning (not shown but used by both Frame-to-frame Correlation and Behavior Analysis) to be resolved when possible with additional data. Path analysis examines the known paths of vehicles and compares them to known and learned routes and areas (i.e., parking lots) and to the paths of other vehicles. It also associates tracks together that appear to be cooperating and estimates the hostility of each track, based on intentions inferable from the platform's path. Multi-Hypothesis Reasoning manages the creation of hypotheses resulting from ambiguous correlations and contradictory path analyses and performs the process of elimination reasoning as additional data are received in order to resolve the ambiguity in other, related tracks.

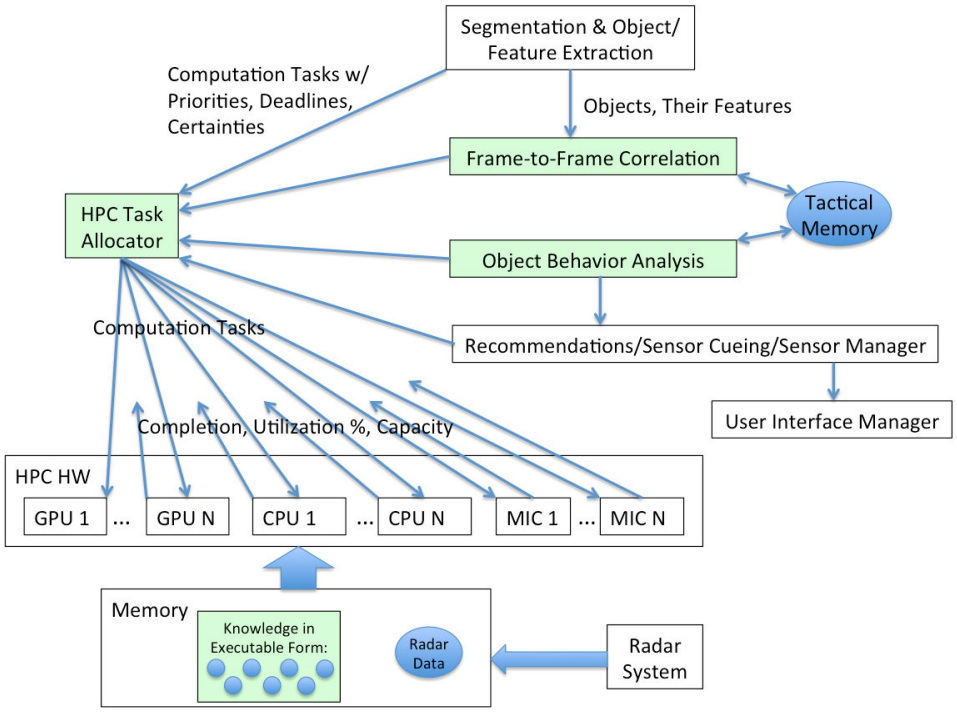

Figure 1. Real-Time, Online Architecture

Patterns and Retrieval (not shown, a component of Behavior Analysis) determines whether the track is exhibiting typical or atypical behavior for similar platforms at similar times in similar locations and uses that estimation to provide a level of suspicion. Recommendations/Sensor Cueing/Sensor Manager makes decisions about what should be done given the list of suspicious objects and their attributes. Possibilities include increased processing of the objects and the areas they are in, highlighting them to a human operator, tasking additional sensors, etc. For recommendations involving human operators, a user interface manager acts as switchboard to get the right information to the correct operator and to moderate the interaction with each user.

All the processing tasks (just an abstract description of them, not the specific data, which would already reside in the high-speed memory for efficient access by the HPC components) that could potentially be executed, with benefit, on the HPC hardware are sent to the HPC Task Allocator from the modules listed above and it makes the real-time decisions as to which HPC resource should perform each task to operate efficiently, get the most important processing done, and meet required deadlines.

F. Path Analysis

Path Analysis exploits the fact that most civilian vehicles proceed in direct routes, using the most direct path from their departure to their destination at the speed of surrounding traffic or, if little traffic exists, near the speed limit (or learned typical speed for that road segment and time of day). Generally turns are infrequent and predictable. They also do not generally react to the movements of other vehicles (other than to queue up at intersections or avoid collisions). Unusual turns inherently invite further inspection in the form of looking for correlations between each turn and turns and/or locations of other nearby vehicles and ground units. Path Analysis looks at all aspects of the vehicle's path. It typically uses fuzzy logic rules to do this. For example, consider the following fuzzy logic

statements:

If Track is Following Friendly-Unit
Then
{
Hostility = inversely proportional to distance between them;
Magnitude = Size of Track's Vehicle + Size of other vehicles in Track's ConvoyGroup (if any)
}

Following is a fuzzy number between 0 and 1 and is the fraction of the Friendly-Unit's path that Track has also been on. In other words, if Track has traversed 90% of the same path as Friendly-Unit, then Following is 90% and Hostility will be 90% of the inverse of the distance between the vehicles.

Other fuzzy logic rules check if each track is in the right kind of area or along the right route for its type, where "right" is established from the previous traffic and usage patterns. Other relationships between tracks besides Following are Heading-Toward and avoidance. These tend to indicate a hostile relationship. Some others involve possible cooperation between platforms. These relationships are convoying, rendezvous, same point of origin or same stopping location, and having a hostile relationship (i.e., attacking, following, avoidance, etc.) with the same other vehicle or unit. Getting hostility information about one track of a cooperating group allows some of the same information to be inferred for the other tracks in the group, albeit with lower certainty. The certainty is scaled to the degree of certainty in the relationship. For example, two tracks originating from the same location, proceeding together in convoy for a relatively long time have a high degree of cooperating certainty. If one is later found to have hostile intent, the other will too and with the same high degree of certainty the system has in the relationship. A final consideration is that a track hiding behind something where vehicles do not normally go is likely trying to engage in deception, which makes it inherently suspicious.

G. Multiple Hypothesis Reasoning (MHR)

MHR keeps track of the multiple hypotheses associated with each track and their relationships to the hypotheses for other tracks. In particular, it is responsible for resolving identities from ambiguities left over from the radar track association process. For each track, it keeps track of the possible predecessors and gradually weeds these predecessors out, depending on which are logically consistent with the current hypotheses. When only one direct predecessor is left, it sets that predecessor as the associated older track and propagates the change to other tracks at the same level. (Resolving one track may allow others to be resolved because the just-associated old track is no longer a viable hypothesis for another track.)

Uncertainty is represented by a certainty value on the hypothesis between zero and one, where zero means the hypothesis is definitely false, and one means the hypothesis is definitely true. When objects in one frame fail to associate with certainty with a single object in the next frame, the identity of the new frame objects is ambiguous. Suppose tracks T1 and T2 were old tracks that failed to positively associate with any tracks in the new frame and that there are two new tracks, T3 and T4, which have not been associated with any old tracks so that T3 and T4 are T1 and T2 but that it is unclear whether T3 matches T1 or T2, and the same holds for T4. Besides the identity of T3 and T4, we may be concerned with hypotheses about various attributes of the tracks, such as level of hostility and the vehicle type and/or its neighborhood (or possibly even address) of origination. T3 and T4 will have the hypotheses from T1 and T2 until new information disambiguates their identities or updates the hypotheses. The certainty module maintains this history of possible associations and updates the track hypotheses. Based on new information about T3, for example from a newer frame, the certainty module will attempt to disambiguate whether T3 matches T1 or T2. If so, then T4 will also be updated to have the correct predecessor track.

H. Patterns and Retrieval

If the airborne radar is operating in the same urban area for a long period, traffic patterns can be used to help the reasoning process. There are four mechanisms for this. The first is to, given a particular track of possible interest, retrieve similar past tracks and calculate various statistics on those similar tracks as a basis for making predictions about the current one. Similarity of tracks includes information immediately sensed by the radar, such as location, velocity, size, and rough vehicle type, but also includes additional information that may also be available. If the vehicle has been tracked since its current operation started, there will be a specific location (e.g., an address or a specific latitude/longitude precise enough to identify a specific building or parking location) and/or an area (such as a specific neighborhood or commercial district) where it started. If this is the first operation of the day for the vehicle or if it has been continuously tracked (even while parked), then the address and/or location where the vehicle spent the night will also be available. This address/location along with the vehicle type/size can be used over weeks to automatically learn and classify the usage type of the vehicle as observed by the radar system. Usage types could

be defined by the workweek and weekend usage pattern. For example, a typical usage pattern (which we might call "commuter") a high percentage of the time during the workweek starts in the morning with a trip from the home address in a residential section to the same work location every workday; ends with a trip from the work location to the home address; and includes small local (errand-running) trips on the weekends at outside of work hours. Other typical usage patterns would exist for different types of commercial vehicles (taxis, delivery trucks for a single business, government and commercial mail package delivery trucks, long-haul trucking, etc.) and different types of personal usage. Note the size and rough vehicle type can be extracted from the radar data. In the case of a vehicle moving on a road, the long dimension is oriented along the velocity of the vehicle, which is the same as the direction of the road, which is known, and since the location of the airborne radar system is precisely known, the aspect of the vehicle is also precisely known, meaning that, for most angles, the vehicle's length, height, and width can be calculated along with a rough classification as to its type (e.g., motorcycle or other very small vehicle, small car, medium-sized car, SUV/large car/van, pickup truck, commercial-type delivery-type truck/van, 18 wheeler, construction equipment (e.g., bulldozer, backhoe, dump truck, etc.), etc.). This additional information is optional to the processing. If it exists it will be utilized to more precisely retrieve patterns and make predictions. If it is not available, the same mechanisms apply; they will just tend to be less sensitive to subtleties. For example, if a vehicle is spotted at 3 AM at high speed on a section of road where this is unusual (based on retrieved 3 AM data for this section of road), and no other information about the vehicle exists, it will still be flagged as suspicious. But if it is 3 PM, and no other information exists, it is likely that retrieved patterns will show nothing unusual. However, if it is a large vehicle and this is unusual for this location and direction then that could be flagged. Or if the vehicle's nighttime location is a neighborhood from which vehicles rarely come to the current location, this could be flagged. Or if the vehicle is large (which by itself is not unusual), and its usage type is commercial delivery (which by itself is not unusual), and the vehicle type is a large delivery truck (as expected), and the location of its home address is a specific commercial district where the businesses are typically owned and frequented by a specific ethnic group (and this, by itself, is not unusual either) but the combination of all these factors together IS unusual, then this could be flagged as suspicious, if all this information were available.

The second mechanism is to make use of groups of vehicles. There are two kinds of groups. One is just the mix of vehicles and speeds heading in the same direction on a specific segment of road. The second kind of group is a set of vehicles that have been tracked together for some time and appear to be a convoy. In either case, a current group can be compared to retrieved past similar groups. The degree to which it conforms to past patterns of activity can be used to determine a level of suspicion that is appropriate. For example, if a specific segment of a specific road contains a very unusual percentage of some type or size of vehicle, then that could be flagged as suspicious. Or if a large convoy of personal-type cars often leaves a specific location (perhaps a church) on Saturdays, then that might be typical (for weddings, say) but might be flagged on a Wednesday or in the middle of the night.

The third mechanism is to short circuit the retrieval and statistical processing of past similar groups by having the software look at patterns in the groups and then create pattern objects that correspond to found patterns. Current groups can then be matched directly to the pattern objects. A fourth mechanism, if possible based on the attributes of the current radar track under consideration (probably because it currently has been tracked since its departure from its "home"), is to retrieve the specific vehicle's previous patterns of activity.

In general the philosophy is that any time a track or group fits a pattern of activity, it is useful to help predict intent and behavior where the prediction is based on the statistically most common data of the past tracks that fit the pattern. When a track or group defies a pattern, it is useful for warning that something atypical (and therefore suspicious) is occurring. Typical examples of defying a pattern are: a vehicle being at an unusual location for the current day/time and its vehicle type, size, and/or neighborhood of origin or making an unusual turn; similar criteria for a group; and that a group's number or composition is unusual (again, given the location and day/time).

To determine if a group or track is unusual, a similarity search can be applied to whatever information exists for the track or group (location, heading, speed, rough vehicle type, size, area of origin, etc.). The statistics for the retrieved past tracks or groups can be computed and compared to the current track to determine level of typicality and therefore suspicion.

Group retrieval primarily addresses volume of traffic and convoy composition/timing. When a current group is created, similar past groups are retrieved and statistics calculated for the retrieved groups. Similarity is based on location (or point of origin), heading, time of day, average speed, etc. If the current group conforms to the retrieved groups, those statistics can be used to compare to the current group's composition. If the current group has a similar number and similar types of vehicles on the same day of the week at the same time of the day, then the current group is very likely benign (or as hostile as these types of groups statistically tend to be). However, if the current group has far more vehicles or very different types of vehicles, then this is a cause for suspicion perhaps that an attacking group is hiding in the normal traffic. Similarly, if the tracks in the past group almost always followed the same route

to roughly the same location, but several of the tracks in the current group suddenly turned in an usual direction then this would be a cause for additional suspicion.

Pattern objects are created by analyzing the past groups, off-line. A pattern is a series of similar groups that repeats with high probability. For example, every weekday, if morning rush hour traffic would create similar road segment groups at the same road segments (in terms of vehicle composition, speed, etc.) then pattern objects would be created to capture these facts for each road segment. A current group could then be compared to the pattern and predictions and warnings made, similarly to the process described in the paragraph above.

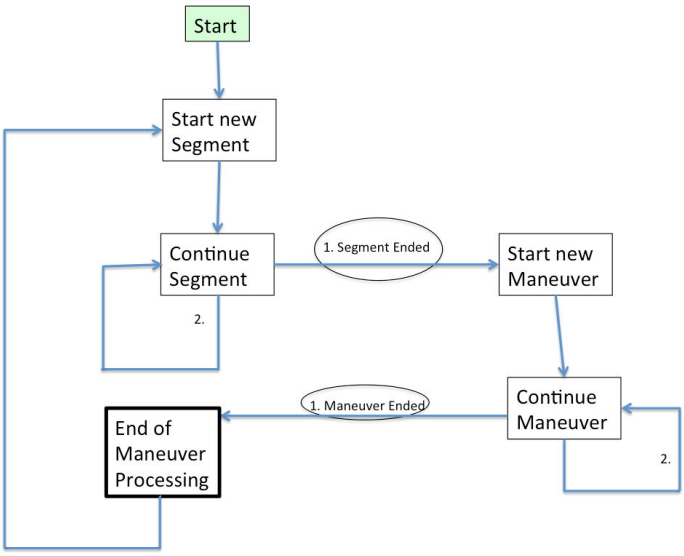

Figure 2. BTN to Segment Each Vehicle's Trajectory

I. Behavior Transition Networks (BTNs)

BTNs are used to create intelligent behaviors by dividing the behavior hierarchically into tasks connected with transitions. They are very similar to Finite State Machines. The current task executes until one of its outgoing transitions becomes true, then control transitions to the task indicated by the true transitions arrow. Tasks with a heavy outline are themselves BTNs that are further expanded. Different Tasks and BTNs can communicate with each other using a variety of means. Generally small groups of BTNs cooperate to fulfill some role and are replicated many times, once each for the objects that they will operate on. This makes them highly parallelizable.

Two example BTNs are shown below. Each would be replicated for each tracked vehicle. Each begins in the green start state and immediately transitions to the next state. In the BTN immediately below, Segments grow until they end (because a turn or stop away from an intersection is detected) and then a new turn or maneuver is begun and continues until the maneuver ends because the vehicle is proceeding straight on a road again. Whenever a maneuver ends, end of maneuver processing is initiated, which is its own BTN which is shown further below, expanded out.

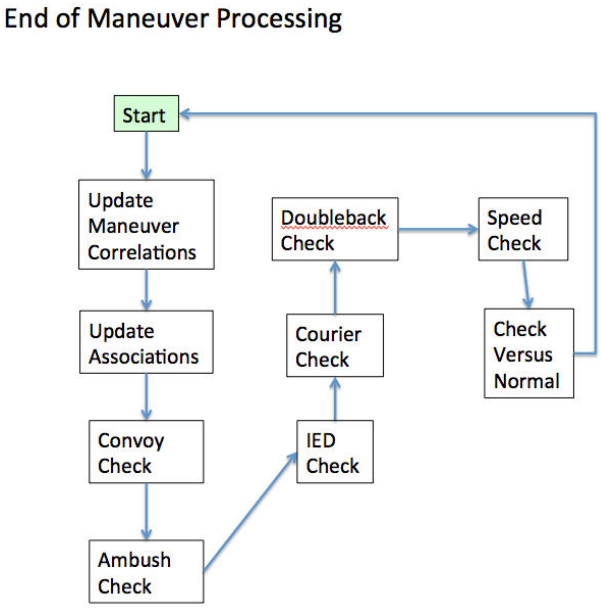

Figure 3. End of Maneuver Processing BTN

J. Truth Maintenance

Truth Maintenance refers to techniques relating to keeping track of multiple competing hypothesis, multiple sets of consistent hypotheses (sometimes called "worlds"), and the dependencies between hypotheses. Truth maintenance systems use nonmonotonic logic, where facts are not just added to the logical structure but also retracted. (The fact that the number of facts sometimes decreases, i.e., does not monotonically increase, leads to the name.) Very general truth maintenance systems are computationally intensive. However, for the radar tracking problem, we utilize the fact that dependencies between hypotheses are only of two kinds. The most obvious is the exclusivity principle—a track can only be one of the competing hypotheses for a track. The second type of dependency between hypotheses is based on the fact that if, because of an ambiguous correlation, there are two new tracks that must each be one of two old tracks but which-is-which is unknown and if one of the new tracks becomes known, then the other new track must be its remaining hypothesis (process of elimination reasoning). The system also makes use of the fact that there will be a relatively small number of these leftover ambiguous situations active at any one time. It is quite easy to create situations that are too complex for a human to

reason through in real-time but for which the logic draws conclusions instantly. In tracking applications, typically only the hypotheses for small groups of tracks interact so that each small group can be processed in parallel.

K. Case-Based Reasoning (CBR)

Case-Based Reasoning is the field of AI that attempts to solve a current problem by retrieving a previously encountered similar problem and adapting that problem's solution to the current situation. In this domain, CBR is used to estimate hostility (in order to focus attention) based on similar, retrieved, past tracks and groups of tracks. By storing past tracks and additional associated information (when available), previous tracks can be retrieved, when faced with similar circumstances, and these can be used as a basis for understanding what typical behavior for such tracks is and to make statistics-based predictions of the hostility and threat level of the current tracks. Reference 1 describes how this retrieval can be performed essentially instantaneously (constant time), if the appropriate index has been indexed. The algorithms have executed on casebases with millions of cases. While this only represents perhaps a single day's worth of cases (10,000 objects, each representing a new case about every six minutes), the retrieval time doesn't grow with the size of the case base, and very similar or identical sets of cases can be merged into groups and patterns. CBR can be performed in parallel for each track or group since there is no interaction.

L. Fuzzy Logic (FL)

One type of rule representation that seems well suited to tactical problems, especially sensor data interpretation, is Fuzzy Logic. In FL, rules are captured that reference qualitative, inexact, or fuzzy values such as High, Low, and Medium. For example, a fuzzy rule might state:

If there are two Tracks who have made the same set of N turns and their relative positions are Close
Then Assume the Tracks are in the same Group (e.g., a convoy)

An FL system operates on quantitative data—such as sensor data; and through a process called fuzzification, it converts that data to a set of qualitative values with associated fuzzy membership values. The rules referencing these qualitative values are each fired to the degree indicated by the membership values. The results of several competing rules are then combined, making fuzzy assignments of qualitative values. These can be used directly or defuzzified into quantitative data.

M. Intelligent Scheduling

The HPC Task Allocator, which optimally distributes the parallel computing tasks to the HPC resources, will be assembled in an existing intelligent scheduling system architecture, which provides for customization of each decision point in the scheduling processing and provides for the functionality common across different scheduling systems. For example, it handles resource usage profiles and resource type requirements, provides for pluggable resource/time window selection methods and satisfaction of temporal, spatial, and arbitrary constraints, and includes the notion of scheduling cycles, an evolving schedule, and the need for a separate preprocessing module.

Scheduling systems take as input a description of the scheduling problem including the tasks to be scheduled, the resources available, and the constraints. However, mere transmittal of the task to the scheduler does not ensure that the required resources can be found to execute it. Whether enough resources exist to schedule all requested tasks is based on the number of resources available (and the optimality of the scheduling algorithm, of course). Associated with each task are its resource requirements and temporal, spatial, and other constraints. For this problem, a "task" will be a required computation and the separate ways it can be accomplished. These separate options would each require resources in the form of one or more HPC components for some length of time or for some number of FLOPs (or other measure of needed computation). A typical temporal constraint, for example, is that one particular task must be complete before another can begin (e.g., object and feature extraction from the pixels for a specific frame must be accomplished before each of the objects can be associated with objects from the last frame) or that a specific task must be completed by a certain time (e.g., the level of suspicion on a specific track must be determined by behavior analysis of the path of that track within 60 seconds). Constraints may exist associated with tasks, resources or both. Examples include a constraint that a particular task must use a specific resource, that a specific resource is unavailable (or partly unavailable due to it being used by another application), or that certain tasks should always (or never) occur at the same time. The scheduler conceptually takes as input the tasks, HPC resources, and constraints, and produces, as output, the assignments to resources of the tasks for specific time windows that meet the constraints for execution on the HPC hardware. Aurora has proven itself adept, in numerous domains, at applying human-quality reasoning to difficult scheduling problems extremely rapidly. It mimics human scheduling cognitive processes in software but executes those processes far faster than a human could. High-quality scheduling

involves optimization of various criteria under resource and time limitations and a large number of constraints and considerations.

III. Prototype Description

Select portions of the above system were prototyped to prove the concept and to collect rough estimates of the type of acceleration that could be expected from different HPC configurations. Owing to the data that was available for the initial study, and with consideration as to which capabilities would be of most immediate benefit in an operational system, the prototype focused on intelligently analyzing the behavior of radar tracks. The data available was simulated correlated GMTI data of vehicles moving fairly randomly in a 10 km by 10 km area of Dayton, Ohio. The road system the vehicles moved on was an accurate recreation of Dayton's roads down to the most specific, detailed, local streets. The simulated data consisted of various simulation runs with 500, 1000, 2000, and 5000 vehicles and about 500 seconds in length each with location and velocity data twice each second. A relatively dense area was also extracted from the 5000 vehicle case to create a 525 vehicle scenario.

The prototype analyzed this movement data in simulated real-time against an open source data set representing the road network of Dayton, determining whether each vehicle was on a road or not and which one. As shown in Figure 2, as the vehicle data comes in, the prototype first divides the vehicle trajectories into "segments" and "turns" where "turns" effectively represent decisions by the vehicle operator. These decisions are turns from one road to another, turning off the road into a known off-road location, like a parking lot, or unknown off-road location, or stopping in the middle of a road segment, without apparent reason.

Three specific behaviors are looked for. One is convoying, that two or more vehicles are travelling in relative proximity for an extended period of time. Another is following, that one vehicle is making a high percentage of the same turns (from the same road onto the same cross-street and in the same direction) as another, at a later point in time. This requires looking for correlations of turns between each vehicle to all the other vehicles within the same vicinity (an $O(N^2)$ computation). The final behavior looked for is called the IED behavior and consists of two parts. The first looks for a vehicle stopping in the middle of a road segment (i.e., not near an intersection) for a few minutes. This by itself is inherently suspicious. If that vehicle (or one associated with it, such as having been in a convoy relationship previously) then proceeds to a different location that provides an overlook opportunity (line of sight) to the initial stopping location, suspicions of an IED behavior are further increased.

The Open Source BTN graphical editing and runtime execution tool, SimBionic, was used to represent the BTNs for the intelligent examination of the vehicle behavior. Figures 3 and 4 represent the main BTNs that segmented each trajectory and called the individual behavior analysis BTNs. Then there were three BTNs that looked for each of the three behaviors described above. Processing 500 seconds of the 525 vehicle scenario on a standard desktop processor, without trying to optimize the code, took 2.5 hours of processing time (about 9000 seconds or about 18x slower than real-time). Simply setting appropriate compiler flags reduced this time to 28 minutes (1700 seconds or 3.4x slower than real-time). Analysis of the computation requirements of different parts of the software revealed that an effective parallelization strategy primarily related to distributing the calculations relating to separate vehicles to separate cores. As long as the number of cores that could be efficiently used was less than the number of vehicles, this would provide a parallelization speed-up proportional to the number of cores (divided by the processing speed slow-down of the individual cores, of course).

Three HPC platforms were targeting with each port building on the previous. First, we ported to multi-core CPUs with OpenMP. This platform typically has between 16 and 32 cores operating at between 2.4 and 3.2 GHz. This simple parallelization revealed which data structures needed to be duplicated or protected by locks so that they can be shared between threads. Speedup depended on minimizing the number of synchronization points between threads. As a bonus, the OpenMP parallelized code worked without modification on the Intel Xeon Phi (aka MIC). The MIC has approximately 60 cores at about 1GHz, each supporting at most four threads, for a maximum 240 OpenMP threads (240 entities processed at once). The MIC can run OpenMP codes at a very high thread count so running on the MIC gave us a good idea about the code's scalability. Once OpenMP parallelization was complete, we planned to perform a CUDA parallelization targeting NVIDIA GPUs. We had several GPUs we could test with, including an NVIDIA Kepler K20 with 2496 cores at about 0.7 GHz. Past experience indicates that an irregular code like the prototype would see a 4x speedup compared to the CPU parallelization. Speedup would depend on distributing the application's data in memory in a way that most effectively uses the GPU's multi-tiered memory hierarchy.

IV. Results

The first HPC implementation resulted in a 12x speedup and thus executed significantly faster than real-time (500 seconds of data processed in 115 seconds). The MIC implementation actually executed slower than the desktop (2700 seconds). This was because each vehicle is processed with the same logic but that logic quickly branches in separate directions so that SIMD processing was not possible preventing effective use of the MIC's Vector Processing Unit. Furthermore, few of the computations were actually arthimetic, the analysis better characterized as symbolic reasoning. For these reasons, translation to the GPU platform was not attempted.

Of course this represented a fraction of the desired analysis. Implementing additional intelligent analysis BTNs to look for more behaviors would require more processing, but not linearly more as a higher and higher percentage of the calculations already being performed can be reused. With these factor considered, it is likely that the first HPC configuration could at least keep up with the required set of BTNs. That would leave pixel processing, frame-to-frame correlation, and comparison to previous normal traffic to be processed on additional HPC resources. These would be more applicable to the MICs and GPUs, so the fuller set of functions represents a better mix of processing types.

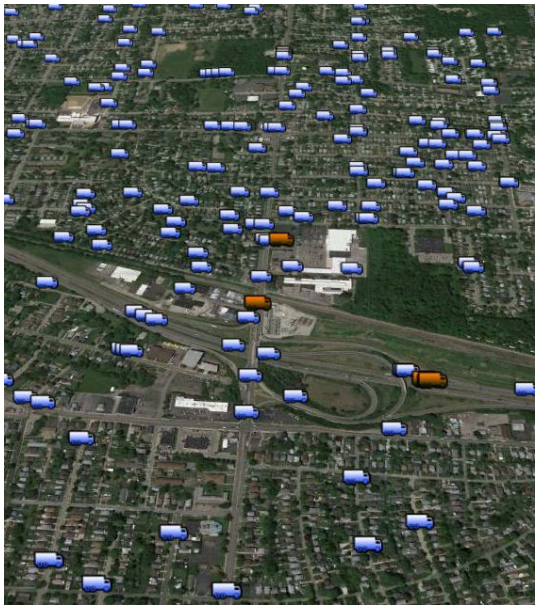

Figure 4. Two Convoys

As expected, the convoy analysis found a handful of vehicle pairs out of the 525 vehicle scenario that happened to stay close together. Two pairs are highlighted in brown in the figure to the left.

As expected given the random turns, there were no cases in the scenario data of one vehicle following a large percentage of the same turns as another. Therefore, independently of the programming team, an example of one vehicle following another was inserted into one of the scenario files. This was found with a high score as shown to the right. The follower is highlighted yellow and the "followee" is in green.

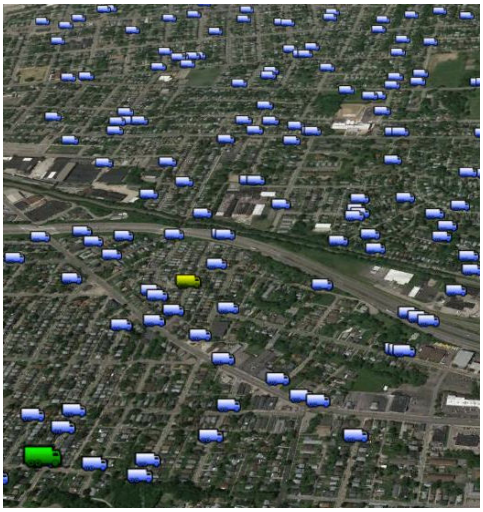

Figure 5. Following

Similarly, the simulated data was generated in such a way that no vehicle stopped in the middle of a road segment so a vehicle with this event was inserted in the data. The vehicle then takes a roundabout route to an overlook location. The first vehicle stop lasted 2 minutes and is shown below in red. The later overlook by the same vehicle is also shown below in the last figure in red. The second figure shows an intermediate point on the vehicles route between the IED plant location and the overlook position.

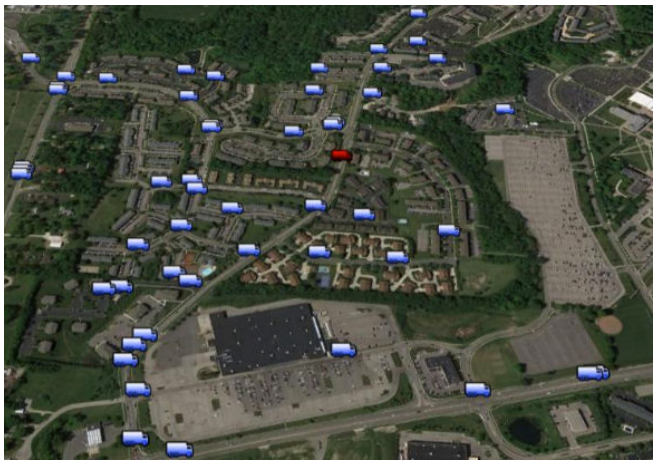

Figure 6. Planting IED

Figure 7. Driving to Overlook

In all three of the above cases, no human being spotted the patterns from the real-time vehicle displays. It was effectively impossible in real-time for a human being to spot any of these types of suspicious activity, which very effectively illustrated the benefit of automatic processing. Interestingly even the vehicle stopped away from an intersection was difficult to notice.

V. Future Work

Because of the large diversity of different aspects of this work, there are a large number of additional research avenues to pursue. Perhaps the most obvious is to implement additional intelligent vehicle behavior analysis BTNs. By implementing a higher percentage of the requirements, more reliable estimates of total HPC processing required could be developed. Furthermore, this would provide additional data into another aspect only begun to be addressed, the automatic parallelization of the intelligent processing. In the initial effort, the parallelization process was effectively performed by hand, but did point to the likely benefits and feasibility of parallelizing the computations across the different vehicles. So far, how this process could be automated is clear and straightforward, for the BTNs implemented to date, since it is based on individual vehicles. Additionally some minor restrictions on the use of SimBionic are also required and would need to be enforced by minor changes to the SimBionic's open source graphical editor and runtime. Whether this would continue to hold for the full set of BTNs remains to be seen.

Figure 8. Overlook

Learning normal traffic patterns requires past data in order to establish normalcy for different times of the day, different days of the week, and during different times of the year. Using publicly available traffic data, we plan to investigate the usefulness of the four CBR techniques described in the Patterns and Retrieval Section (retrieval based on specific vehicle type, group, learned patterns, and specific vehicle ID) to detect interesting deviations from normal behavior. The real-time retrieval component should be highly parallelizable based on the individual vehicles and individual groups. I.e., we will be able to retrieve similar past data for current vehicles and current groups in parallel with each other. Past work[1] has already shown than a index can be built off-line to make individual similarly retrievals very fast. We believe that the process of building these indices off-line could be radically accelerated with HPC hardware. Incrementing the index for individual vehicles and groups can be performed in parallel across vehicles and groups and across different instances of time. Furthermore, if required, different parts of the index corresponding to different geographic locations, times, velocities, and/or combinations of symbolic feature values can be assigned to different cores and performed in parallel.

In the next phase of the effort, using simulated radar GMTI we will investigate the ability to improve tracking performance (average correct tracking time per vehicle) based on the knowledge-based, frame-to-frame object correlation described in the System Description section. To make this practical in near real-time will require substantial parallelization. Unfortunately unlike all of the computations described so far, parallelizing the computations across the set of objects is not entirely straightforward. This is because multiple objects in one frame may compete for association with multiple objects in an earlier frame; the computations are not independent of each other. Fortunately, the large majority of objects can possibly associate with exactly one object in a previous frame, and for these cases, the computations can be parallelized. (Even more fortunate is the fact that the calculations to determine this are exactly the same for all objects and so can be easily parallelized on Single Instruction Multiple Data (SIMD) GPUs). The small fraction of objects that don't fall into this category will have to be processed on the same CPU, or perhaps parallelized into non-overlapping sets of competing objects.

Other possible future research endeavors include correlating the GMTI tracks with SAR data. Radar GMTI tracks inherently will break down at very slow speeds or when stopped. Meanwhile SAR images provide good pictures for stationary objects but moving objects will not be visible. When a GMTI radar track stops for a long enough period of time, a SAR image of it should exist. Correlating these events in the different products would provide significant additional data on these objects. Similarly, fusing the GMTI track data with other intelligence products such as optical or IR Wide Area Motion Imagery (WAMI) would provide significant benefits. It is likely that these calculations could be parallelized based on geographic location.

It is not our intention to pursue research efforts directed toward processing the raw and low-level radar data to produce the individual GMTI tracks or SAR images. However, it is likely that such processing would share the same HPC hardware as the intelligent processing described throughout this paper and therefore our approach must allow for it. This relates to the Intelligent Scheduling described in the System Description Section where real-time scheduling of processing tasks on HPC resources is performed based on the current tactical situation. In particular, all of the intelligent processing must be allocated to the various diverse CPUs, GPUs, and MICs based on their current availability (i.e., some radar signal processing outside the scope described here may be occurring on some of the HPC elements) and the priorities and deadlines of the processing tasks, themselves based on the tactical situations. Although simple priority-based and deadline-based scheduling schemes have been shown to be severely suboptimal in many domains, they may be reasonable for this domain. A few of these should be implemented and compared to more intelligent (but still computationally efficient) approaches such as bottleneck avoidance[2].

VIII. Conclusions

This work showed the effectiveness, feasibility and benefits of intelligently processing radar track data to automatically spot suspicious behavior and to perform those calculations on massive data sets in faster than real-time on HPC hardware. Mechanisms to automatically perform the parallelization and porting from a single thread to massively parallel the computations on HPC resources were indicated. The overall design for a complete system was described along with the results from a prototyping and HPC fielding effort. Several avenues for future work were presented.

References

[1]Stottler, R., Henke, A., and King, J., "Rapid Retrieval Algorithms for Case-based Reasoning," Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Vol. 1, Morgan Kauffman Publishers, Inc., San Mateo, CA, 1989, pp. 233-237

[2]Stottler, R., Mahan, K., and Jensen, R., "Bottleneck Avoidance Techniques for Automated Satellite Communication Scheduling," Proceedings of the Infotech@Aerospace 2011 Conference, AIAA, Reston, VA, 2011, pp. 2518-2526