Sensors Simulation Environment for Sensor Data Fusion

Pavel Pohanka  
EW section / IT Department  
VOP-026 Šternberk, s.p.  
VTÚO Brno division  
Czech Republic  
pohanka@vtuo.cz

Juraj Hrabovský  
EW section / IT Department  
VOP-026 Šternberk, s.p.  
VTÚO Brno division  
Czech Republic  
hrabovsky@vtuo.cz

Michal Fiedler  
EW section / IT Department  
VOP-026 Šternberk, s.p.  
VTÚO Brno division  
Czech Republic  
fiedler@vtuo.cz

Abstract - There are efforts in the wide field of applications today to fuse heterogeneous data from sensors in order to provide outputs, which are more informative than the original data from particular sensors. Reliable data are essential to be able to fuse sensor data in sensor fusion system. Nevertheless, it is in some cases difficult or even impossible to build a real environment complete with deployed sensors in order to ensure reliable data for the research, development and testing of sensor data fusion systems. Implementation of a sensors simulation environment, which would generate synthetic sensor data (numerical and non-numerical) is a solution to fulfill such efforts. This article describes a proposal of open source technologies, components, interfaces and architecture of a sensors simulation environment, which is able to generate synthetic sensors data according to user-defined scenarios. The defined and described architecture allows standardized interconnection of sensor simulators with fusion systems within a communication net.

Keywords: synthetic sensor data, simulation, agents, data fusion, architecture, interface

1 Introduction

Data fusion relies on reliable sensor data. Such data can be generally obtained from real sensors that are situated in a real environment. However, this approach is not usually applicable in most cases. The proposed solution involves 3D environment where it is possible to create large areas (originally urban areas) and deploy various agents and sensors in order to simulate heterogeneous sensor output. Sensors simulation environment for sensor data fusion is able to provide mutually complementary sensor data for further processing (fusion) and can be used for various purposes from the simple sensor data generation to the fusion system evaluation. Standardized communication is an important aspect in sensor fusion systems in order to obtain sensor data. The majority of the current sensor fusion systems use their own solution for interconnection with sensor data resources. Our solution stressed the use of standards for sensor simulator outputs in order to provide seamless interconnection with other systems.

2 Related work

Sensor data fusion starts to be important in the recent years and various sensor data fusion testbeds has been developed. Some of them use their own sensor data generators and others use commercial simulation frameworks that can be utilized for generation of reliable sensor data. It does not exist ideal solution for the sensor simulation framework, because every fusion system is specific, but some solution can fulfill the functional requirements more than the others can.

- SceneGen is a scenario generation tool, which supports 3D visualization for creating scenarios, includes a number of sensor and target models and provides database support to manage different fusion and network algorithms. This tool was implemented and tested successfully for air tracking scenarios with radar and Ground Moving Target Indicator (GMTI) sensors by the group at SUNY at Buffalo. Unfortunately, there is no further information about this tool but [1], [2].

- The Multi-source Report-level Simulator (MRS) is a tool developed by Veridian Systems as part of its Model-adaptive Multi-source Track Fusion (MMTF) effort under DARPA’s Dynamic Tactical Targeting (DTT) program. MRS simulates ground vehicle tracks and generates simulated multisensor contact reports for Ground Moving Target Indicator (GMTI), HUMan INTelligence (HUMINT), IMagery INTelligence (IMINT), SIGnal INTelligence (SIGINT), Unattended Ground Sensors (UGS), and video sensors. It contains a spatial editor for creating ground tracks along which vehicles move over the terrain. Observation models characterize each sensor at the report level in terms of their operating characteristics (refresh rate, resolution, etc.), measurement errors and detection/classification performance. Contact reports are linked to ground truth data to facilitate the testing of track/fusion algorithms and the...
validation of associated performance models. More information can be found in [3].

• In [4] is proposed the architecture for Ground Target Identification Fusion System, which is composed of battlefield simulator, sensor simulator and data fusion system.

• FLExible Analysis, Modeling, and Exercise System (FLAMES®) [5] is a commercial military scenario generator (simulation framework) from Ternion company. This system is open and customizable simulation system that allows developing many different types of simulations. From the sensor point of view, it provides models for sensors and targets, which can be customized or rewrote in C and C++ language to provide more fidelity and resolution. Several data fusion projects use FLAMES for scenario generation and execution [5][6], [7], [8].

• Other commercial products that can be used for sensor data fusion are e.g. vsTasker [9] from VirtualSim, SLAMEM [10] from Tayon and VR-Forces [11] from MAK Technologies.

The disadvantage of non-commercial tools is usually the fact, that it is very difficult to obtain them and reuse them. The disadvantage of commercial tools could be its license cost and the need to customize them to be able to generate sensor data in the right format.

3 Design of system architecture

The overall system architecture was proposed after analysis of following requirements.

3.1 Simulation environment requirements

A sensor simulation environment for the sensor data fusion should fulfill the following requirements:

• Generation of synthetic sensor observation data and corresponding ground truth data from various sensors based on the scenario.

• Generation of environment ground truth data and contextual data in order to evaluate the fusion system.

• Sensor modeling – it is necessary to model each type of sensor to be able to simulate their functionalities.

• Agent behavior modeling – agents represent targets, which are detected by sensors. Realistic targets model is essential for sensor data fusion.

• Error modeling – this requirement is necessary for realistic simulation, because real sensors are not error free.

• Visualization – it is necessary for creation of the scenario, control of the simulation and the setting of its parameters.

• Standardized interfaces – are necessary for interoperability of cooperating systems.

3.2 System architecture

The Figure 1 depicts the overall architecture and data flows between particular parts of the sensor simulation environment. The Fusion System receives data from Virtual Sensor Platforms. These Virtual Sensor Platforms generate simulated sensor data (SSD) and ground-truth data (GTD) based on sensor simulator events (SSE) produced by a Sensor Simulator Engine. The Virtual Sensor Platform has two main parts – a Sensor Data Simulator and a Data Interface. The number of the Virtual Sensor Platforms is given by the scenario and controlled by the Simulation Management. The Sensor Simulator Engine consists of two main modules:

• Simulator Visualization module – consists of Scenario Visualization and Simulation visualization.

• Simulator Management module – consists of Scenario Management and Simulation Management.

The 3D Terrain is not part of the Sensor Simulator Engine, but creates and provides 3D environment for simulation.

The following text briefly describes the process from the scenario creation to the simulation execution. Blender [12] – open source, cross platform suite of tools for 3D creation – is used for the 3D environment model creation. Positions of buildings, roads, pavements and creation of navigation meshes for agent movements are specified during this phase. The environment model is exported to an XML format (Terrain DB). This model is loaded by the Scenario Management module, which cooperates with the Scenario Visualization module. Agents, sensors and other objects are added into the scenario together with all necessary information needed for scenario execution in the Simulation Management module. The scenario description file is created after that.

Before the simulation start, the processing of all objects within the scene and a pathfinding to calculate the paths among all waypoints within navigation mesh are performed. Agent modeling, sensor modeling and error modeling are performed and events for sensor data generation are generated after the start of the simulation.
3.3 Simulator data input

A scenario file or a 3D environmental model serves as an input data for the Simulator (Sensor Simulator Engine + Virtual Sensor Platform). The Simulator also needs some information from the Knowledge DB (KDB). A user intervention is needed for objects deployment and specification of objects parameters. These parameters are written in the scenario file, which contains all necessary information for loading the scene.

3.4 Simulator data output

The Simulator provides five different outputs, which can be exploited by the Fusion System. It provides Sensor Observation Data and related Sensor Observation Ground Truth Data, which represent sensor synthetic data output. Then it provides Scenario Ground Truth Data, which represent static information about objects in the scene and Dynamic Ground Truth Data, which represent dynamic information about objects in the scene. Finally, the simulator provides contextual information about 3D environment in the OGC standard CityGML [13] that is used for representation, storage and exchange of virtual 3D cities and landscape models. The Scenario Ground Truth Data go along with our own XML Schema (XSD)[14], [15], [16]. Sensor Observation Data and Sensor Observation Ground Truth Data can be in format according to specific XSD or Transducer Markup Language XSD [17][17].

The Sensor Observation Data contain information about time stamp, sensor identification, detected object position and detected object features.

The Scenario Ground Truth Data contain information about sensor position, orientation and field of view, features of agents and objects within the scene. The Dynamic GTD Data are sent when parameters of objects in the scenario are changed.

3.5 External data flows

The Figure 2 depicts data flows provided by the Sensor Simulator Engine and the Virtual Sensor Platform to the Fusion System and possibly to an Evaluation System. The Sensor Observation GTD Data (1b) and the Sensor Observation Data (2b). The Scenario Environmental Maps (5) contain a 3D environment and provide contextual information in the form of the CityGML format. Both Scenario GTD Data (3) and Dynamic GTD Data (4) represent GTD about agents, sensors and other objects.

3.6 Simulator development environment

Simulator is based on a modular, three-tier architecture. The Scenario Visualization and the Simulation Visualization stand within the presentation tier. This layer exploits Ogre Editor - Ogitor [18][13], which is a plug-in based WYSIWYG editor environment for an Object-Oriented Graphics Rendering Engine (OGRE) [19], and uses Qt toolkit [20] for graphical user interface (GUI). The Ogitor is released under the LGPL and the OUTL license policy. The OGRE is a 3D library for OpenGL and/or Direct3D and is released under MIT License. The Qt is released under the GNU LGPL, GPL or a commercial license. The Blender tool (GNU GPL License) is used for a 3D environmental model creation and a database management system (DBMS) PostgreSQL [21] (license similar to BSD and MIT) is used for databases creation and management. In the Figure 3, the Simulator application layout is depicted.

4 Scenario creation

This section describes necessary steps needed for scenario creation. Scenario creation is performed using the Scenario Visualization module and the Scenario Management module. The scenario description file is created at the end of this process. The Simulator uses the same user interface for both the Scenario Visualization and the Simulation Visualization. These two modules exploit the Ogitor, which was modified and improved by a large number of our plug-ins.

The scenario creation is performed in several steps. First of all, a terrain file from the Blender tool has to be loaded. This terrain file contains information about a static scene. This file includes buildings, streets, roads and other static objects. Information from the Knowledge DB is loaded...
and used for sensor and agent modeling. It is possible to add any needed objects, configure their parameters and specify various parameters specific for the simulation. The available objects for the deployment are described in the chapter 4.1 Simulation objects. The scenario can be saved and used anytime in the future.

4.1 Simulation objects

The following objects can be placed into the scene in order to create the scenario.

**Agent object:** HUMANs, ANIMALs, VEHICLEs, EVENTs and WEAPONs are considered as agents. Each type of agent possesses its own parameters. The agents can be controlled and semi-controlled. The goal of controlled agents is specifically determined by the scenario. The goal of semi-controlled agents is chosen randomly from a predefined set. There are several internal parameters assigned to the agents like – can be carried, can carry agents and is movable.

**Area object:** Represents significant area in the simulation. The area can be marked by non-stopping or no-entering flags.

**Communication & Communication Network object:** These objects allow simulating radio communication among agents within the communication group. Groups of agents are called communication networks and a particular agent can be part of several communication networks. The agents in the particular communication networks use the same frequency and modulation when transmitting to each other. There are active parts of communication in which agents transmit, and there are passive parts without transmission. The communication consists of one or more conversations that are divided into speeches. Each communication has several parameters to be set (start time, end time, half-duplex, full-duplex, frequency, modulation, network ID, channel utilization, etc.) and these parameters are stated in the communication object.

**Explosion object:** Represents the weapon agent with ability to detonate.

**Group object:** This object allows to group agents into the groups for various purposes.

**Scenario object:** The Scenario object allows setting important parameters for the scenario run. The main parameters are Scenario ID, Start Time, Random Seed, False Alarms Rates, number of automatically deployed agents etc. Scenario object has to be present in the scene to be executable.

**Sensor object:** This object allows sensors creation. Each sensor has position, direction, field of view, accuracy and frame rate parameter. There are eleven types of sensors and each type has specific input parameters for its modeling. The list of supported sensors is as follows:

- Acoustic (AC)
- Electro optical (EO)
- Communication intelligence (COMINT)
- Chemical, biological, radiological, nuclear and meteorological (CBRN/METEO)
- Hyper spectral (HSI)
- Identification friendly - foe (IFF)
- Laser range finder (LRF)
- Low-end (LE)
- Small ground radar (SGR)
- Synthetic aperture radar (SAR)

**Shooting object:** This object specifies an attacking and a defending group of agents and for each group it specifies a loss rates.

**Timeline object:** A timeline specifies which actions are performed in the specific time by particular agents. Among these actions belong Move, Pick, Drop and Execute. The main timeline has to be presented in each scene because it defines the main storyline of the scenario. The consecutive timelines are intended for additional storylines. These timelines are activated by triggers. Triggers define both space and time conditions for the activation of additional timelines. For instance, improvised explosive device (IED) will explode 10 seconds after Agent1 arrival at Waypoint 5.

**Waypoint object:** Waypoints represents significant locations in the scenario. Only the waypoints can be dedicated as goals for an agent movement. There are special kinds of waypoints through which agents can enter the scene or leave the scene.

**Track object:** Tracks are represented by waypoints and they determine agent movement. There are several types of tracks. There are tracks having the exact order of waypoints and tracks having a random order of waypoints.

**Trigger object:** Triggers evoke the activation of a planned timeline. A trigger can be based on a time (timed triggers) or a position (positional triggers). Timed triggers are triggered by absolute time and can be used either unrepeated or repeatedly. The positional trigger defines a distance (from the agent to the object or another agent, or the distance among the agents of the group), which can be lower or bigger than a defined threshold value. The both triggers can affect either a single agent or a group of agents.
5 Simulation

The following picture (Figure 4) depicts the main modules of the Simulation Management:

A kernel of the simulation process is the Simulation Management module, where a necessary logic for simulation run is implemented. This module controls the entire simulation process. Simulation management consists of a Simulation Core (SC) submodule and Sensor Data Generator (SDG) submodules. Each sensor in the simulated environment has its own instance of the SDG. The SC sends a filtered list of targets according to type of sensor to the SDG. The SDG determines the detectability of this target according to the target parameters and the location. Detectability of the target is computed within a detection function. The SDG module generates a sensor simulator event, which describes a recognized target. The SC executes and controls instances of Sensor Data Simulator (SDS) modules, which are connected to the particular SDG. The SDS converts the data to the defined format and sends them via the Data Interface module to the Fusion System. The SC exploits a pseudo random number generator to affect the simulation in a controlled way and therefore it is possible to rerun the simulation with the same results if necessary. Due to a simulator modular architecture, it is possible to add new functionalities easily.

5.1 Agent movement modeling

The pathfinding (Dijkstra algorithm) is used for finding several suitable paths inside the graph of the navigation mesh between significant points (waypoints) within the scenario. The randomly selected suitable path is fractalized and a poly-line is created. This path is further processed and used as a corridor for the agent movement. Each moveable agent tries to follow the path to reach a specified goal (specified by the SC). The steering behaviors algorithms control the agents to follow the path in a natural looking way [22]. Several behaviors have been implemented for individuals and groups:

- **Arrival** – this behavior is applied when the agent is stopping. This behavior affects the velocity of agents and that is why the movement looks natural.
- **Leader Following** – this behavior allows controlling the movement of groups.
- **Obstacle Avoidance** – agents avoid colliding with each other or with solid objects.
- **Path Following** – this behavior allows agents to move within the radius of a poly-line.
- **Wall Following** – this behavior allows agents to move parallel to the edges with a defined offset.

The Figure 5 depicts the agents (humans, vehicle) which head to the waypoints. The color poly-lines represent the computed paths.

5.2 Sensor modeling

In a 3D environment, all sensors are represented by their directional characteristics (Field of View - FoV). This means that the sensors with omnidirectional characteristics are visually represented by a sphere mesh and the others are represented by a pyramid mesh. In the Figure 6, the sensors with their FoV are depicted. Direct visibility between the sensor and the target is verified, when the agent is presented in the FoV of the sensor. If this verification succeeds, the detection function is performed and the event data structure with the sensor data is created. These data are mutually complementary, when the agent is observed by more than one sensor. Currently it is possible to generate observations from ten types of sensors (AC, EO, COMINT, CBRN/METEO, HSI, IFF, LRF, LE, SAR, SGR).
The detection function is very significant element in sensor modeling, because it contains the logic for the creation of the event data structure. Formula (1) represents declaration of the detection function (DetectFunction) method.

\[
\text{sensor simulator event} = \text{DetectFunction} ( \text{object data}, \text{environment data}, \text{input data})
\]

(1)

The Object data contains the ground truth information about the sensor and the target (type, position, parameters). The Environment data contains the information about the current scene conditions. This information affects the sensor measurement accuracy. The Input data are defined for each type of sensor and they contain the prior information about the sensor parameters (e.g. spectral range, spectral accuracy, resolution etc.). The Sensor simulator event is a list of data structures, which are sent to the SDG and contains distorted sensor data. The Pseudo random number generator is used within the detection function for the data distortion in order to be able to repeat the simulation with the same results.

5.3 Error modeling

There are several errors affecting sensor measurements and they fall under two categories. 
- The errors affecting the position and the features of detected objects within the Sensor Observation data and 
- the errors affecting object track identification.

The first category errors are modeled within the detect function. Object track identification errors are described further.

Track errors are represented by two types of false alarms – false positives and false negatives. The False positives alarms represent tracks that do not belong to any object within the scenario. This is modeled as phantom agents. The phantoms can represent any agent type. They appear in random moments close to the other agents and they move throughout the scene for a few seconds before they disappear. The number of the phantom agents is determined by a false positive rate defined within the scenario. They can be detected by only one sensor during this time. The False Negative alarms include track ID changes, incomplete tracks and track gaps. The Track ID change occurs when two agents are close to each other or when they overlap. In that case, the sensor can swap the target ID or assign a new target ID to the agent. The incomplete tracks are modeled by dropping out of some measurements. The probabilities for dropping out the measurement are determined by the scenario. The subsequent sensor measurements have different target IDs and this creates a perception that one agent disappeared and another one appeared. The track gaps are modeled by dropping out some measurements. In contrast to the incomplete tracks, subsequent measurements have the same Target ID. This creates a perception that there is a gap between the tracks. Errors are controlled by the Pseudo Random Generators.

5.4 Classification

A classification is evaluated within the detection function of each sensor. It affects the features assessment in the Sensor Observation Data. It is possible to classify the object because all parameters of the detected object are known (its bounding box, distance from sensor, object and sensor parameters). Two thresholds for the classification are used. First threshold represents maximum distance for classification. Second threshold represents distance for 100% classification. These thresholds are stated in the KDB for each sensor type and they are used for a probability function calculation for all known object types. Because the classified object is known, the appropriate probability function is used. Then an "Unknown" class will be created as a supplement to 100%. Then other possible classification classes are specified with their probability values within the "Unknown" class probability. For example, in the first step the object is determined as a Vehicle for 50% and as Unknown for 50%. In the second step the object is determined as Vehicle for 50%, as Unknown for 25%, as Human for 20% and as Animal for 5% (second step can be skipped). Classification is further effected by environmental conditions.

6 Data interchange

The simulator provides an interface, which allows sending the sensor simulator output to the sensor fusion system. The solution is based on information interoperability and standardization. The information interoperability can be described as an unambiguous exchange of information and services among automated information systems. Standardization is the main presumption for information interoperability creation. The standardization means a common agreement on what
information is exchanged, in what format, how the exchange is performed and under what conditions. Interoperability standards are a set of agreements that allow systems to co-operate with other systems by means of exchanging information.

There are various definitions and models of the interoperability levels. Our solution is based on the following interoperability layers definition:

- Physical communications layer (e.g. combat radio, Ethernet, etc.).
- Protocol layer (a transport protocol as e.g. TCP/IP, UDP/IP, etc.).
- Data structure layer (a transport format as e.g. TML, SensorML etc.)[17], [23].
- Semantic understanding of the data layer.

This solution uses TCP/IP or UDP/IP on protocol layer and Transducer Markup Language (TML) [17] is an XML-based system for capturing, characterizing and enabling the transport of raw sensor data. It has elements designed to make it a self-contained package, including not only the raw data (dynamic data) but also metadata (static information) which are necessary for processing (fuse) the information later. Even if the TML is mainly intended for raw data, TML’s transducer element is flexible enough to accommodate any type of data. TML packet with dynamic data can look like this:

```xml
<data clk="2011-06-04T14:41:31.01Z" ref="SENSOR_01_OBJECT_POS"> -35 0 70</data>
```

TML data are created by the SDS and can be sent via the Data Interface to the Fusion System (Figure 1) or the TML output data can be stored in the database and resent later. The Data Interface is a module providing the transfer/delivery mechanism for data. Currently it uses transport layer protocol TCP/IP or UDP/IP, but it can be extended easily to use various application protocols (e.g. HTTP, SOAP, etc.).

### 7 Applications

This sensors simulation environment is developing within Distributed and Adaptive multisensor FusioN Engine (DAFNE) project [24], which is the European Defence Agency (EDA) project. This project aims at designing and experimenting a real-time distributed multi sensor fusion engine that will combine data from heterogeneous sensors in order to generate reliable estimates about the entities and events in an urban warfare scenario. The DAFNE project focuses mainly on enhancing situational awareness during military operations by a fusion engine able to combine data from the sensors employed in an urban warfare scenario.

Further, the sensors simulation environment will be used in training and simulation center in order to simulate sensor networks for fighting in build up and open areas and for military operations in urban terrain.

Then, the sensors simulation environment will be used as a simulator for a system for communication, processing and storing sensor data streams [25], [26].

### 8 Conclusion

This paper describes a sensors simulation environment suitable for a sensor data fusion. This sensors simulation environment uses an open source technologies and due to its data interface, it can be interconnected with sensor fusion systems easily. Thanks to its universality, it can fulfill many purposes.

It can serve for a sensor simulation, a testing based on realistic observations, and it is possible to simulate various types of operations in order to generate reliable data for various types of sensor fusion systems. Due to its modular architecture, it can be easily extended by additional modules to provide other functionalities.

The simulation is now based on manually created terrains but there are plans to implement the functionality for loading external terrain data and improve the simulation in order to work on such surfaces. This involves some changes related to the scene creation phase and navimesh calculation. Improvements in reaction on dynamic changes inside the scene belong to future possible improvements as well.

### Acknowledgements

The authors would like to thank the European Defence Agency (EDA) for their financing of the DAFNE project and thus this research. Further we want to thank the partners in the DAFNE project: IDS, University of Udine, TNO, Warsaw University of Technology and IOSB for their collaboration and fruitful discussions.

### References


2021


2022