Luis Bustamante, A., Molina López, J. M., García Herrero, J. (2017). Player: an open source tool to simulate complex maritime environments to evaluate data fusion performance. *Simulation modeling practice and theory*, 76, pp. 3-21.

DOI:https://doi.org/10.1016/j.simpat.2017.04.002

Player: An Open Source Tool to Simulate Complex Maritime Environments to Evaluate Data Fusion Performance

Alvaro Luis Bustamante, José Manuel Molina López, Jesús García Herrero

Avda. de la Universidad Carlos III, 22. Colmenarejo 28270. Spain

Abstract

In this paper it is presented a new open source tool for evaluating data fusion systems, mainly related to the maritime surveillance domain. This tool provides specific utilities for designing and simulating synthetic maritime environments to assists in the fusion development process, like designing vessels trajectories, placing different types of sensors, simulating vessels dynamics, or simulating sensors detections. This synthetic information can be used to feed a data fusion system to evaluate its response in a reproducible way under different conditions. This tool can be used to optimize the data fusion evaluation process, since testing a fusion system is a quite complex task, as the fusion output depends on the combination of multiple algorithms, configurations, measures, timing, and so on. Then, having reproducible synthetic environments can be quite useful when evaluating fusion results, system performance in dense scenarios, vessel trajectories with different dynamics, sensor coverages, and so on. This tool has been used with success for evaluating different fusion systems, and now it is presented as an open source tool, so it can be easily adapted to different environments, be used by other researchers, or extended by the community. This paper presents how it is built, the underlying algorithms, and presents some example use cases.

Keywords: Data Fusion, Maritime Surveillance, Simulation, Sensors, Evaluation, Performance

1. Introduction

Maritime Surveillance systems are commonly used for identifying and intercepting threats in seaports, coastal areas, maritime boundaries, maritime platforms, or important installations. Monitoring extensive environments, like the maritime, requires a wide deploy of sensors detecting targets of interest. Common sensors in this domain are the Automatic Identification System (AIS), radar, and Closed-circuit Television (CCTV) [1]. Although most modern systems can use UAVs (Unmanned Aerial Vehicles), and satellite imagery [2, 3] as additional data sources. It is common to obtain such distributed information in a common base station and use a Vessel Traffic Service (VTS) for evaluating possible threats in real-time. VTS systems normally include data fusion systems [4] to integrate sensor information in order to provide a more comprehensive representation of the environment state.

In such complex environments, it is quite important to ensure the proper operation of all the underlying components, specially the one related with the fusion system, as many of the services integrated in VTSs are normally built on top of the data fusion. A fusion system is a quite complex software development that includes multiple algorithms for data processing, association, filtering, combination, etc., that may work flawlessly in real-time to feed VTS information systems. So, before integrating a fusion system in a production environment it is required to test it extensively covering as many situations as possible.

However, testing a data fusion system can become a complex task, as it is not possible to cover every possible case due to the problem nature, which inherently contains infinite degrees of freedom. There are multiple variables that will condition the execution, like algorithm parameters, sensor observations, sensors noise, vessel dynamics, etc. This way, it is required to evaluate the most significant cases to ensure a proper operation under similar conditions, like vessels crossing so close, vessels moving in parallel, vessels moving along different sensors coverage keeping stable identifiers, vessels doing maneuvers at high speed, etc. It is not practical to evaluate this kind of special situations

using only information from a real-world deployment, as they will not happen quite often, exactly as required, or the information is not isolated to debug it easily.

Therefore, this paper describes a recently released open source tool to assist in the process of designing and simulating maritime surveillance scenarios, where the user can easily define sensors, and vessels trajectories by just clicking over a map representation. The description made by the user can be simulated in real-time to generate multi-sensor multi-target detections as they were generated in a real-environment. This simulation can be used to feed a fusion system in order evaluate its behavior under the specified conditions.

There are few alternatives, if any, specifically designed for the purpose of this tool. In [5] it is described a tool to simulate static trajectories with waypoints, making animations, visual effects, and events that can be visualized in the Google Earth tool with a generated KML (Keyhole Markup Language). However, this tool does not generate detections in real-time, cannot be connected to a fusion system, and does not allow design environments in the tool, as it is designed only for scenario representation purposes. There is a more complete approach defined in [6], that describes a fusion architecture that serves as a base for higher-level situation assessment algorithms. It includes a simulation module, but it is tightly coupled in the tool, so it cannot be used with different fusion architectures. Furthermore, none of these tools are available as open source.

There are other general-purpose alternatives that can be used to represent environments and its state, like the Google Earth tool [7], NASA World Wind [8], and in general, any Geographic Information System (GIS) [9]. Nevertheless, this kind of tools lacks all the specific layers covered by the Player tool, like the environment description, edition, simulation, animation, visualization, connection to fusion system, etc. Therefore, the Player tool intends to provide a complete and comprehensive suite for designing and simulating maritime surveillance scenarios for fusion system evaluation. This tool is released under a open source license, so any user can adapt it to their needs, or integrate in

commercial projects.

This paper is covering mainly the general Player tool architecture and the models used for simulating both sensors and vessels, while providing also some illustrative results for a better comprehension. The rest of the paper is organized as follows: Section 2 provides a background about data fusion systems and its complexity, and how the tool is integrated in such architectures. Section 3, and 4, describes the models used both for vessel and sensors simulations. Section 5 describes the final modules developed inside the tool to assist the user while designing synthetic environment. Finally, section 7 provides some results while using the tool to simulate different uses cases.

2. Background on Data Fusion in Maritime Surveillance

The core of maritime surveillance systems are Vessel Traffic Services (VTS), aimed to provide efficient transits and safe navigation for vessels [10, 11, 12, 13, 14]. To accomplish this task, technical means should be designed to enhance situational awareness and overcome the limitations of traditional methods such as direct sight and voice communications. There are diverse surveillance sources in maritime areas, which must be integrated to provide real-time decision support to operators. Data from cooperative sources (AIS transponders equipped in vessels) must be correlated with non-cooperative sensors, such as shore radar, high-frequency radars, or video (optical/infrared/satellites). This way, a VTS operator can obtain a more representative picture of the environment, which will support the decision making process in the surveillance task.

Then, a fusion system must process the different sources of information to provide a single fused output for each entity detected in the environment, like vessels. The output typically consists of a set of non-redundant tracks, called global tracks. Each global track represents a single entity in the environment that is normally composed of information describing vessel location and its cinematic: global track ID (unique for each vessel); last update timestamp; geodesic coordinates; speed and course over ground, etc. This type of systems

are normally modeled as a distributed system [15, 16], where a first layer contains processors for each data source (like sensors), and these mono-sensor tracks are then compared to decide if they can be correlated in the same entity. This process can be decomposed in several steps which are executed periodically in a fusion cycle [17, 18], which is represented in figure 1.

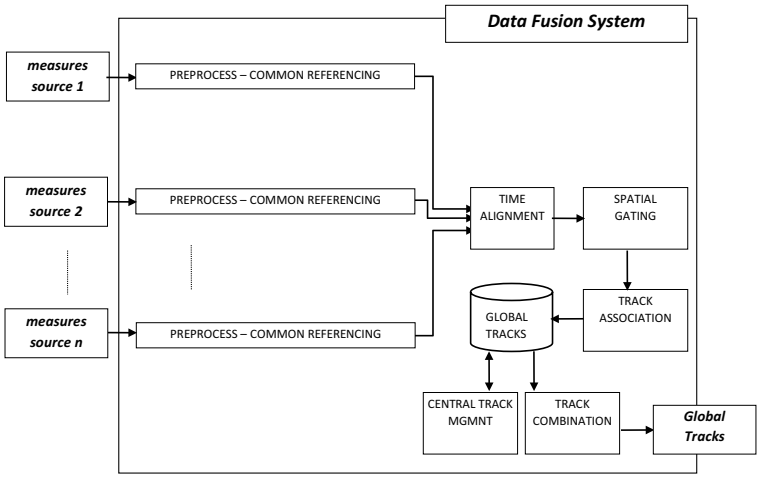

Figure 1: Distributed data fusion architecture.

However, there are many research needs on appropriate architectures and algorithms for multi-sensor fusion in maritime surveillance [4, 19, 20, 10]. The problem is challenging when dealing with large sensor networks and heterogeneous areas such as high-density zones crowded with diverse objects in motion. An important phase of analysis and adjustment of the system is usually necessary to refine the models and raise robust processes to ensure the reliability of the system in real conditions such as presence of inconsistent measures, sensors malfunction, dynamic behaviors, parameter changes, etc [21]. Thus, the proposed tool will assist to properly design and evaluate a data fusion system in maritime surveillance environments, at it will allow simulating multiple sources of information with multiple observable entities.

Figure 2 presents a general overview that describes how the proposed tool will interact with a fusion system, taking into account the fusion architecture, as described in figure 1. The tool will assist on designing synthetic entities like vessels, radars, AIS Stations, or cameras, to simulate them in real time. The result from the simulation, is then provided to the external fusion process which will process all input sources, similar as it where receiving measurements from real sensors. After the fusion cycle is completed, the fusion system will report the combined global track entities, that can be injected in-real time to the tool, to evaluate if generated entities correlates with the simulated ones. Hence, it is obtained a closed loop where the user can simulate entities and observe in real-time the results thrown by the fusion system.

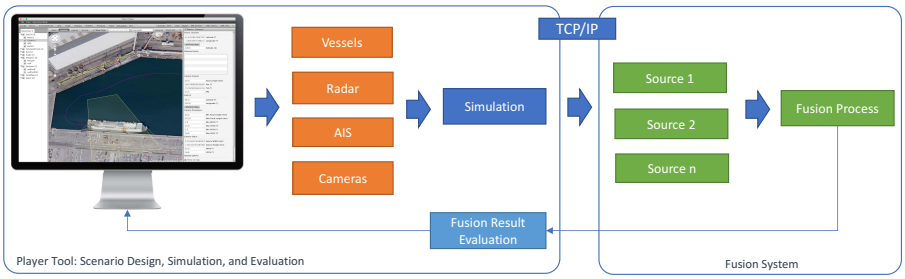

Figure 2: General Overview of the Simulation Tool with a Data Fusion System

3. Vessels Model

As described in the previous section, the Player tool allows designing and simulating different entities of the maritime surveillance environment, like vessels, radars, AIS stations, or cameras. This section describes how the vessel simulation is done inside the tool, which mainly consists on updating vessels locations based on the simulation time, the designed trajectory, and the vessel maneuvers.

The vessel trajectory is simulated in the Player tool according to the trajectory waypoints, taking into account the configured speed, and angular velocity.

It is interesting to mention that all the intermediary calculations to get the simulation working are done in geodesic coordinates to avoid translating the coordinates to a common reference plane and converting then back again to geodesic. This kind of approach is done in modern data fusion systems covering large areas in order to avoid transformation errors.

The simulator module has been designed taking into account that the system should be running in real-time to provide detections to the fusion system in evaluation, so it performs a continuous simulation according to each time step done inside the simulation. However, the simulation can be accelerated for designing the environments. In this way, all the simulation process is based on a variable simulation step (t_s) measured in seconds, that is used in the simulator equations. So it is easy to modify the simulation step in real-time to get a simulation running faster or slower. This is critical for designing long-time scenarios that can take several hours to complete. The complexity of the proposed simulation is $O(N)$, as each vessel is simulated independently, so the simulation time depends linearly on the number of simulated entities.

At first, the simulation relies on computing the movement between two waypoints, accelerating or decelerating the vessel properly to reach the destination waypoint at the specified speed following a constant bearing between two waypoints. The formulas described in the following description, are used for the constant bearing [22] simulation only, but the tool also supports navigating between points using great-circle paths [23]. The formulas are a little bit different for the great-circle navigation mode, mainly those related with calculating the travel distance (it will be usually shorter), the initial bearing for the vessel (as the bearing will change in every simulation step), and the destination point. Each vessel can specify its own navigation mode in the simulation, so it is possible to set constant bearing for those vessels traveling short paths, or apply the great-circle navigation for long distances. The modified simulation steps and the equations are available in the tool source code, but not described here for a better paper reading.

So, the vessel trajectory, based on the simulation step t_s, is summarized in

the algorithm 1. It basically consists on computing the new vessel location after the simulation step t_s, based on the current vessel location, the target waypoint location, the current vessel speed, and the target vessel speed.

Algorithm 1: Compute new Vessel location based on Target Waypoint, Location, and Speed

1 function NewVesselLocation $(v_l, v_w, v_s, w_s, t_S)$;

Input : Current Vessel Location v_l (geodetic coordinates in radians)

Input : Next Trajectory Location v_w (geodetic coordinates in radians)

Input : Current Vessel Speed v_s (m/s)

Input : Next Trajectory Speed w_s (m/s)

Input : Simulation Step t_s (seconds)

Output: New Vessel Location (geodetic coordinates in radians)

`// distance between current location and next waypoint`

2 w_d = distance between v_l and v_w using equation 1

`// bearing between current location and next waypoint`

3 w_α = bearing between v_l and v_w using equation 2

`// required acceleration to reach the next waypoint at the desired speed`

4 $a = ({w_s}^2 - {v_s}^2)/(2 \cdot w_d)$

`// compute traveled distance based on simulation step, computed acceleration, and current speed`

5 $d = v_s \cdot (t_s + (a \cdot {t_s}^2)/2$

`// compute new vessel location based on vessel location` v_l`, distance` d`, and bearing (`w_α`)`

6 v_l' = destination point based on v_l, d, and w_α using equation 3

7 return v_l'

$$\Delta\psi = \ln(\tan(\pi/4 + \varphi_2/2)/\tan(\pi/4 + \varphi_1/2))$$

$$\mathrm{q} = \Delta\varphi/\Delta\psi \text{ (or } \cos(\varphi_1) \text{ if } \Delta\psi = 0)$$

$$\mathrm{d} = \sqrt{(\Delta\varphi^2 + q^2 \cdot \Delta\lambda^2} \cdot R$$

Where ψ is latitude (rad), λ is longitude (rad), and $\Delta\lambda$ is taking shortest route ($\leq \pi$), and R is earth's radius (meters).

(1)

$$\Delta\psi = \ln(\tan(\pi/4 + \varphi_2/2)/\tan(\pi/4 + \varphi_1/2))$$

$$\theta = atan2(\Delta\lambda, \Delta\psi)$$

Where ψ is latitude (rad), λ is longitude (rad), and $\Delta\lambda$ is taking shortest route ($\leq \pi$).

(2)

$$\delta = d/R$$

$$\Delta\psi = \ln(\tan(\pi/4 + \varphi_2/2)/\tan(\pi/4 + \varphi_1/2))$$

$$\mathrm{q} = \Delta\varphi/\Delta\psi \text{ (or } \cos(\varphi_1) \text{ if } \Delta\psi = 0)$$

$$\Delta\lambda = \delta \cdot \sinh\theta/q$$

$$\varphi_2 = \varphi_1 + \delta \cdot \cos\theta$$

$$\lambda_2 = \lambda_1 + \Delta\lambda$$

Where ψ is latitude (rad), λ is longitude (rad), R is earth's radius (meters), and $\Delta\lambda$ is taking shortest route ($\leq \pi$).

(3)

The above description is only valid for calculating trajectories between two straight waypoints. However, following a path composed by multiple waypoints

requires some additional steps to detect when it is required to switch to the next waypoint destination, and also simulate the direction change. For simulating the transition between waypoints it is considered a constant speed and angular velocity, as defined by the user while editing the trajectory. This is quite similar to the actual Rate of Turn Indicator (ROTI) used by vessels to indicate the rate a ship is turning. The maximum ROTI is often given by the vessel type, its dimensions, and the maneouver stability, so it can be useful for defining turns according to the vessel type being simulated.

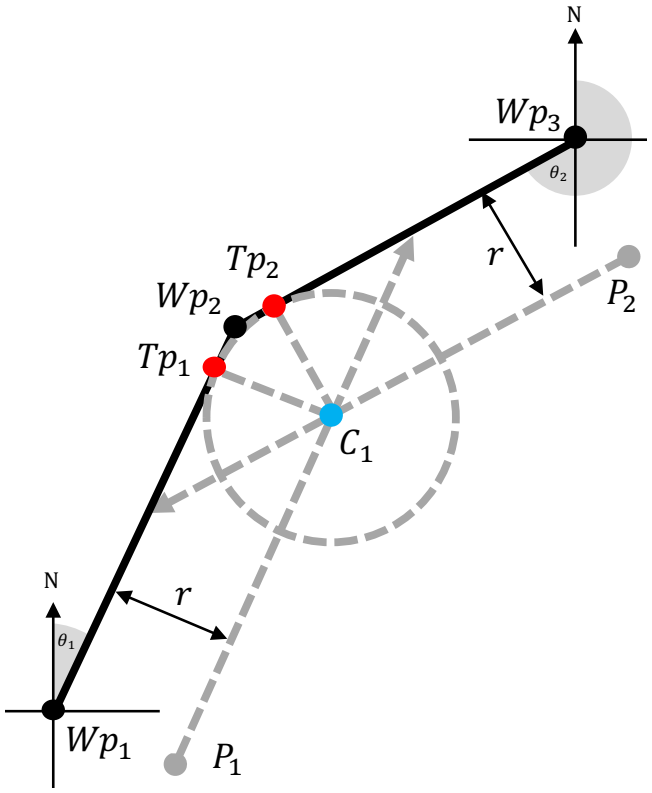

Figure 3: Turn points calculation using a geometric approach directly in geodesic coordinates taking into account the configured speed and angular velocity.

Therefore, each simulated vessel turning over a intermediary waypoint of the trajectory will describe a semi-circumference with a radius according to the configured speed, and angular velocity. The complexity here resides on

finding the points in the trajectory where the maneuver will start and end to appropriately describe the turn. This can be reduced to finding the intersection points between a circumference and two tangent lines. In this case, this is done over the geodetic coordinate system by using a geometric approach as described in algorithm 2, and illustrated in figure 3. This will result in two geodesic points T_{p1}, and T_{p2}, which are the points where the vessel should start and stop the turn maneuver. These points are marked in red in figure 3.

$$
\begin{gathered}
\delta_{12} = 2 \cdot \arcsin(\sqrt{\sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)}) \\
\theta_a = \arccos(\sin\varphi_2 - \sin\varphi_1 \cdot \cos\delta_{12} / \sin\delta_{12} \cdot \cos\varphi_1) \\
\theta_b = \arccos(\sin\varphi_1 - \sin\varphi_2 \cdot \cos\delta_{12} / \sin\delta_{12} \cdot \cos\varphi_2) \\
\text{if } \sin(\lambda_2 - \lambda_1) > 0 \\
\theta_{12} = \theta_a \\
\theta_{21} = 2\pi - \theta_b \\
\text{else} \\
\theta_{12} = 2\pi - \theta_a \\
\theta_{21} = \theta_b \\
\alpha_1 = (\theta_{13} - \theta_{12} + \pi)\%2\pi - \pi \\
\alpha_2 = (\theta_{21} - \theta_{23} + \pi)\%2\pi - \pi \\
\alpha_3 = \arccos(-\cos\alpha_1 \cdot \cos\alpha_2 + \sin\alpha_1 \cdot \sin\alpha_2 \cdot \cos\delta_{12}) \\
\delta_{13} = atan2(\sin\delta_{12} \cdot \sin\alpha_1 \cdot \sin\alpha_2, \cos\alpha_2 + \cos\alpha_1 \cdot \cos\alpha_3) \\
\varphi_3 = \arcsin(\sin\varphi_1 \cdot \cos\delta_{13} + \cos\varphi_1 \cdot \sin\delta_{13} \cdot \cos\theta_{13}) \\
\Delta\lambda_{13} = atan2(\sin\theta_{13} \cdot \sin\delta_{13} \cdot \cos\varphi_1, \cos\delta_{13} - \sin\varphi_1 \cdot \sin\varphi_3) \\
\lambda_3 = (\lambda_1 + \Delta\lambda_{13} + \pi)\%2\pi - \pi
\end{gathered}
$$

Where φ is latitude (rad), λ is longitude (rad), and θ is bearing (rad).

% represents the floating point module operation.

(4)

Algorithm 2: Compute turn points for a turn between three waypoints of the vessel trajectory

1 function TurnPoints $(W_{p1}, W_{p2}, W_{p3}, w_s, w_\omega)$;

Input : Waypoint Location W_{p1} (geodetic coordinates in radians)

Input : Waypoint Location W_{p2} (geodetic coordinates in radians)

Input : Waypoint Location W_{p3} (geodetic coordinates in radians)

Input : W_{p2} Waypoint Speed w_s (m/s)

Input : W_{p2} Waypoint Angular Speed w_ω (rad/s)

Output: Turn points in trajectory between waypoints T_{p1}, and T_{p2}

// calculate turn radius according w_s and w_ω

2 $r = w_s/w_\omega$

// Compute angles between waypoints

3 θ_1 = angle between W_{p1}, and W_{p2} using equation 2

4 θ_2 = angle between W_{p3} and W_{p2} using equation 2

// compute P_1 and P_2, as illustrated in figure 3, clockwise turn in this example (analogue for counterclockwise)

5 P_1 = destination point based on W_{p1}, r, and $\theta_1 + \pi/2$ using equation 3

6 P_2 = destination point based on W_{p2}, r, and $\theta_2 - \pi/2$ using equation 3

// compute intersection between parallel lines to find turn center

7 C_1 = intersection between $P_1\theta_1$, and $P_2\theta 2$ using equation 4

8 T_{p1} = intersection between $W_{p1}\theta_1$, and $C_1\theta_1 - \pi/2$ using equation 4

9 T_{p2} = intersection between $W_{p3}\theta_2$, and $C_1\theta_2 + \pi/2$ using equation 4

10 return T_{p1}, T_{p2}

There is a sample trajectory defined with the tool in figure 4. It can be observed a vessel that defines a trajectory composed by 5 waypoints. In the three intermediate waypoints there are circumferences that represents the rate of turn for the configured speed, which are calculated according to the above description. So, the simulator will take the whole trajectory and their waypoints to calculate all the turn points according to the speed and angular velocity of each one, like those shown as T_{p1} and T_{p2} in equation 3. This way, every intermediate waypoint will be associated with a pair of start and end turn points, which will be used for computing the vessel acceleration between waypoints. Once the vessel reaches a start turn point T_{p1}, it will follow a semicircular path with the calculated turn radius until it reaches T_{p2}, that defines the turn end. In this moment, the vessel will move to the next waypoint or turn start by following a straight line.

Figure 4: Sample trajectory edited in the simulator. For debug purposes, it is displayed the calculated circumference defining the turn trajectory, as well ass the computed start and end points.

4. Sensors Model

The Player tool also allows designing and simulating different sensors of the maritime surveillance environment, like radars, AIS stations, or cameras. This section describes how the different sensors are simulated inside the tool, which mainly consists on providing detections of the simulate vessels, depending on the current simulation step, vessel location, sensor period, etc. The following sections will describe the underlying models for radars, AIS stations, and cameras.

4.1. Radar Sensors

The radar simulation works by doing azimuthal changes as defined by the simulation step t_s, and the antenna rotation period. After each simulated antenna change, it is necessary to check whether the antenna orientation passed over a vessel in the radar coverage. This way, it is possible to provide accurate radar detection instants even for simulated vessels moving both clockwise or counterclockwise. This simulation is done as described in the following steps, which are also illustrated in figure 5.

1. Detect vessels in range by applying the distance between two geodesic points as described in equation 1.
2. Calculate the bearing from radar center to each target in range (t_θ) using the formula described in 2.
3. Taking initial radar azimuth θ_1, the radar angular velocity ω_r, and the simulation step t_s, calculate the target radar azimuth θ_2.
4. Report the position of every target with a bearing (t_θ) between θ_1 and θ_2. In figure 5, the only track reported for the simulation step t_s is t_1.

4.2. AIS Stations

AIS messages are encoded according to a standard recommendation [24] that defines around 27 different messages. These messages can contain different information according to the message type. For example, position report messages

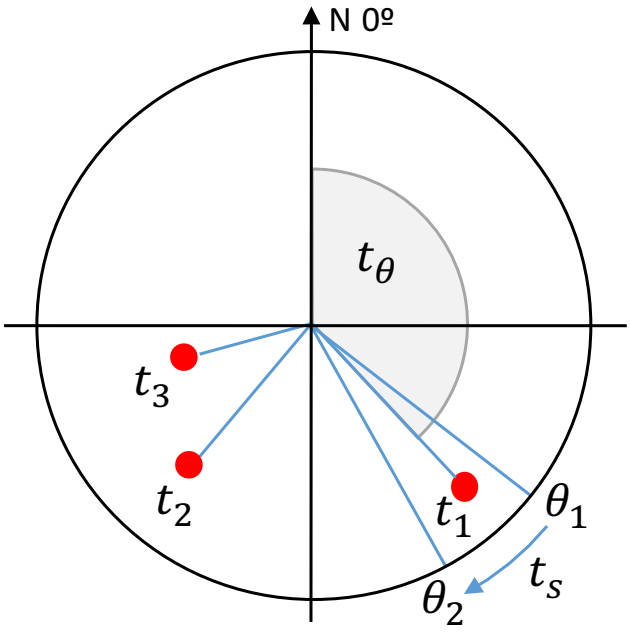

Figure 5: Radar detection simulation based on antenna rotation and bearing to the surrounding tracks in range.

contain dynamic information that can be changing constantly, like the vessel position, speed over ground, course, heading, rate of turn, and so on. This kind of messages are updated quite often to provide an accurate picture of the vessels moving in the sea.

In the other side, there are some messages with static and voyage related data, that contains information like vessel name, type of ship, vessel dimension, type of position fixing device, port destination, estimated time of arrival, etc. All those messages also includes the Maritime Mobile Service Identity (MMSI), that is an unique identifier for each vessel. Depending on the vessel speed and maneuver, the messages related with dynamic information are reported at different rates as specified in the AIS recommendation [24] summarized in table 1. In this way, a faster vessel will be reporting more frequently than the one that is anchored or moored.

So, in the Player tool, the configurable AIS stations are passive entities

Table 1: Class A shipborne mobile equipment reporting intervals

Parameter	**Description**
Ship at anchor or moored and not moving faster than 3 knots	3 min
Ship at anchor or moored and moving faster than 3 knots	10 s.
Ship 0-14 knots	10 s
Ship 0-14 knots and changing course	3 1/3 s
Ship 14-23 knots	6 s
Ship 14-23 knots and changing course	2 s
Ship $>$ 23 knots	2 s
Ship $>$ 23 knots and changing course	2 s

that are receiving messages from the surrounding vessels. Those vessels which incorporates an AIS transponder, will acquire the vessel location, speed, course, and MMSI to generate a dynamic message at a frequency as described in the AIS recommendation [24]. This message is then sent to all the AIS stations covering the vessel location.

4.3. Pan-Tilt-Zoom Cameras

As the same way the radars or AIS stations can provide information about detected targets, in the Player tool, the PTZ camera is also simulated to provide targets detections. So it is supposed that the simulated PTZ camera is using some kind of computer vision algorithm to discern the vessels present in the environment, and convert the 2D location to a real-world 3D position based on the camera location and calibration. This is done inside the simulator by calculating the projected surface FOV based on camera location, altitude, orientation, and sensor parameters.

The projected FOV is calculated in real-time based on sensor parameters, like sensor width (s_w), height (s_h), and focal length (f_l) (this is variable by changing

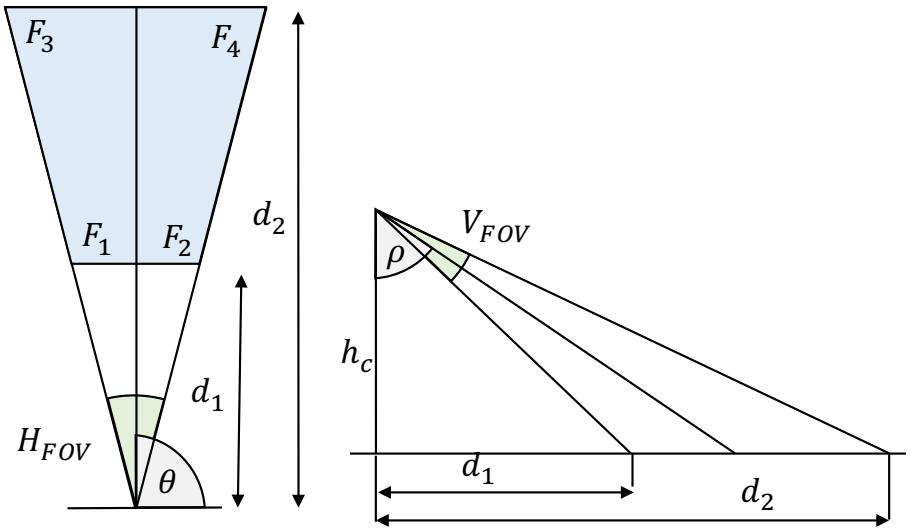

Figure 6: Illustration of FOV calculation based on pan (θ), tilt (ρ), and horizontal and vertical field of view H_{FOV}, V_{FOV}.

the zoom), that allows to determine the current horizontal FOV (H_{FOV}), and vertical FOV (V_{FOV}). With this information, and the camera altitule (h_c), current pan (θ), and tilt (ρ) angles, it is possible to calculate the FOV projection over the surface as described in the following steps. This procedure is also illustrated in figure 6.

1. Taking the current focal length (f_l), and sensor size (s_w, s_h), calculate the current horizontal and vertical FOV (H_{FOV}, V_{FOV}), as $H_{FOV} = 2 \cdot \arctan(s_w/(2 \cdot f_l))$, and $V_{FOV} = 2 \cdot \arctan(s_h/(2 \cdot f_l))$, as described in [25].
2. Given the V_{FOV}, h_c, and ρ, calculate the approximated maximum and minimum vision distance in the surface, as $d_1 = h_c \cdot \tan(\rho - V_{FOV}/2)$, and $d_2 = h_c \cdot \tan(\rho + V_{FOV}/2)$.
3. Calculate the WGS84 coordinates for the projected FOV (F_1, F_2, F_3, and F_4) using the formula described in 3, by using the camera location as origin, d_1, and d_2 as distances, and current pan angle ($\theta \pm H_{FOV}/2$) as bearings.

5. Implementation

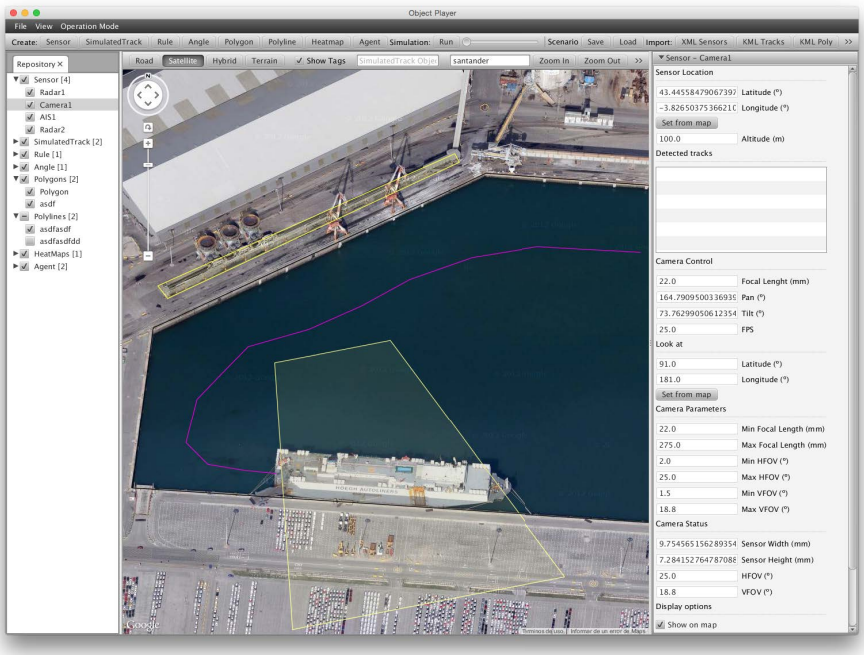

Figure 7: Player tool Overview. This screenshot represents the main tool window, where the user can define entities like radar, AIS, cameras, vessels, and manage the simulation process, and the connection with the fusion systems via TCP/IP. The underlying Geographic Information System (GIS) is based on Google Maps with a custom API layer for display and update the entities in real-time.

The published open source Tool is currently available at Github[1], which includes both sources and binaries. The tool is available for public access and also there is a YouTube Channel[2] of the GIAA research group, which includes some illustrative examples for a proper evaluation.

The project has been developed by using the latest JavaFX 2, which provides a portable way of creating interfaces. So the tool can be run in any operating

[1] https://github.com/alvarolb/player-tool
[2] https://www.youtube.com/user/GIAAUC3M

system like Windows, Mac OS, and Linux starting with Java 1.7, which is crucial to ensure the tool integration in any fusion development environment. In general terms, the interface is built over a WebView widget that embeds Google Maps as a GIS for displaying geographical entities. Google Maps is actually used in several research projects as Geographic Information System (GIS), like [26], as it provides a comprehensive Application Programming Interface (API) to display and update practically any geographic shape over the map. In this way, the Player tool interacts with the Google Maps API to display user simulated entities like sensors, vessels, trajectories, polygons, etc., over the map. The tool core is then responsible of computing all the geodesic coordinates as result of simulating vessels maneuvers, or sensors detections, to issue the required commands to the Google Maps API to update the environment state in real-time.

The general architecture of the tool follows the architecture outlined in figure 2, and the resulting main screen of the application is shown in figure 7. The following subsections will describe in more detail the different entities that can be configured in the Player tool, as part of the simulation subsystems presented in the architecture.

5.1. Vessels

One of the main features of the Player tool, is that the user can easily define several vessel entities that can describe trajectories or maneuvers over the simulated environment. This feature is quite convenient to simulate and evaluate the fusion response for entities that are approaching each other, or passing over Regions of Interest (ROI), describing strange or erratic movements, trying to navigate over complex paths near the littoral, and so on. There are multiple situations than can be of interest when developing a maritime surveillance system, and the fusion system must be prepared for almost any of them. Therefore, simulating this kind of situations is essential to ensure a proper operation.

Therefore, the user can place multiple virtual vessels in the environment with a starting position, and a set of additional parameters that can define

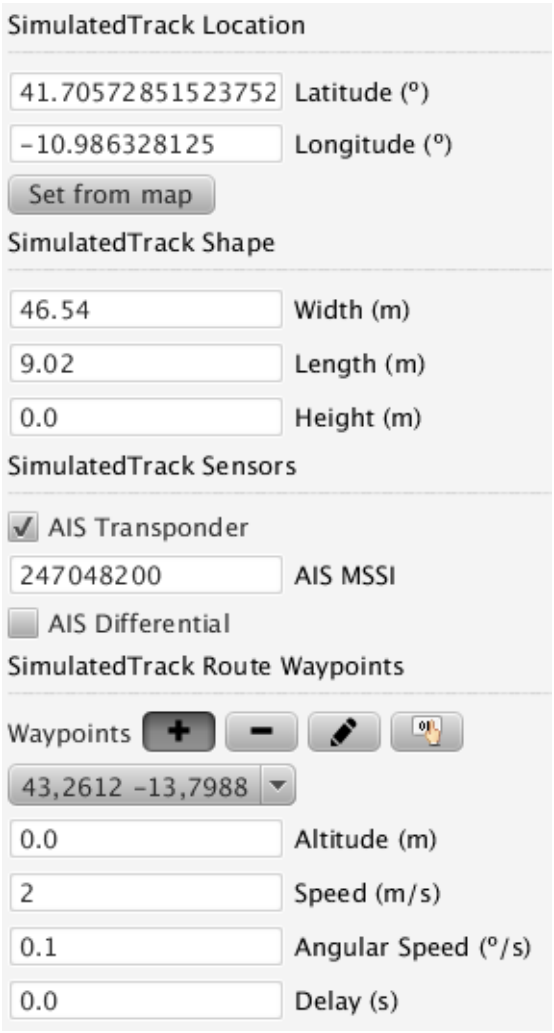

Figure 8: Configurable parameters for Simulated Vessels

the vessel dimensions or their equipped sensors. For each defined vessel in the environment it is possible to define a trajectory by clicking locations in the map to generate a set of waypoints. Aside of the waypoint location, it is possible to define the waypoint altitude (for further simulation of aerial environments), the speed, and the angular velocity to complete the turn to the next waypoint. There is an optional configurable delay for each waypoint if the target speed

is 0, so it is possible to simulate vessels stopping in a waypoint, and waiting a certain amount of time before traveling to the next waypoint. This is specially useful for the starting point, so it is easy to synchronize different trajectories by changing the initial timeout.

So, the Player tool is able to simulate vessel trajectories according to the requirements to satisfy, like target density, trajectories, behaviors, maneuvers, and so on. For each vessel, it is possible to configure different parameters that are described in figure 8. Notice how it is also possible to configure the MMSI for the Automatic Identification System (AIS), so it is possible to simulate vessels sharing the same MSSI, which happens quite often in real environments.

5.2. Radar

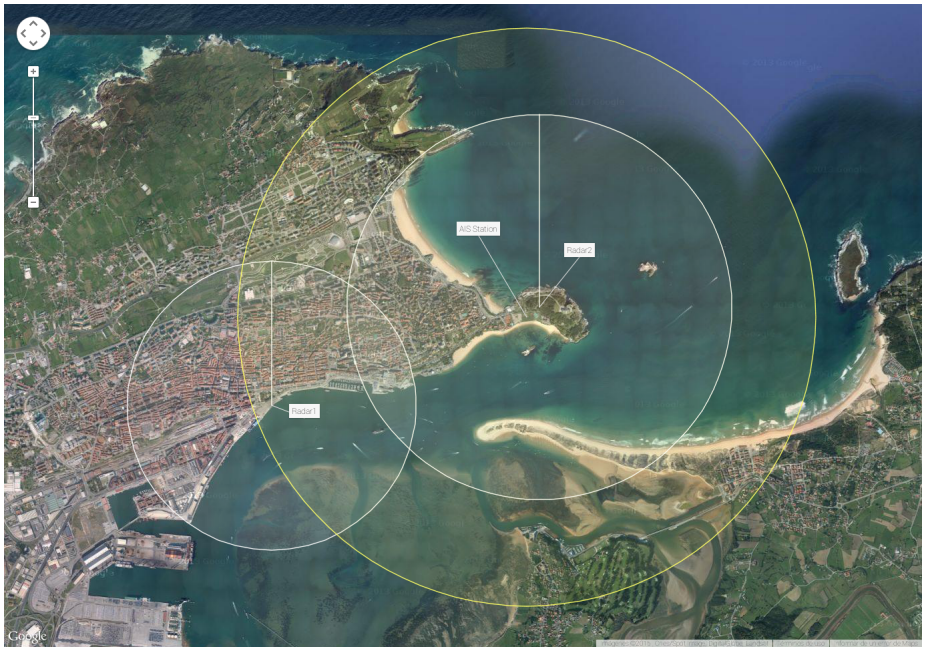

Figure 9: Example representation of multiple sensors deployed in the maritime environment. Each sensor is covering different zones, but also providing overlapping areas for testing the fusion system.

Simulating vessels is quite useful for describing interesting situations that

need to take attention over a data fusion implementation. However, vessels cannot be detected unless there are sensors deployed in the environment. In this way, for the maritime surveillance environment, the Player tool incorporates different kind of sensors that can be deployed in a similar way the vessels can be configured.

One of the most used sensor in maritime surveillance is the radar, that is usually placed over the coast covering different points of interest, like seaway channels, maritime ports, or any other critical areas. This kind of sensors scans the environment sending radio waves and looking for objects that reflects the energy. Due to the sensor nature, the radar scans the environment periodically according to a scan period, and other parameters like the pulse width. A radar also has a variable resolution both in azimuth and distance according to the distance to the target being monitored.

Simulating a whole radar sensor is completely out of the scope for this tool. A realistic radar simulation may provide a synthetic video feed of the scanned environment. However, in our simulated environment based on 2D projected maps, it is not possible to have a complete information about terrain orography, that is necessary for effectively simulate the terrain shielding. Also, creating a realistic simulation will include adding environmental conditions like rain or sea that generates clutter, shadows, and so on. Instead, this tool generates an output that is quite similar to the information provided by a radar extractor. The radar extractor analyzes the radar video feed to analyze plots, and track targets, which is done by using low-level data association and filtering algorithms.

In such a way, the expected output for the radar sensors being simulated with this tool, is a high-level detection made by each deployed sensor, according to the maximum radar range, and the azimuth location of the vessel according to the antenna rotation and radar position. It is also possible to simulate the detection precision described as an Area of Probability (AoP) according to the configured radar precision and the target detection range. But this tool is not actually simulating other factors like false alarms, azimuth resolution, noise, or occlusions between both static and dynamic entities. In this way, there is

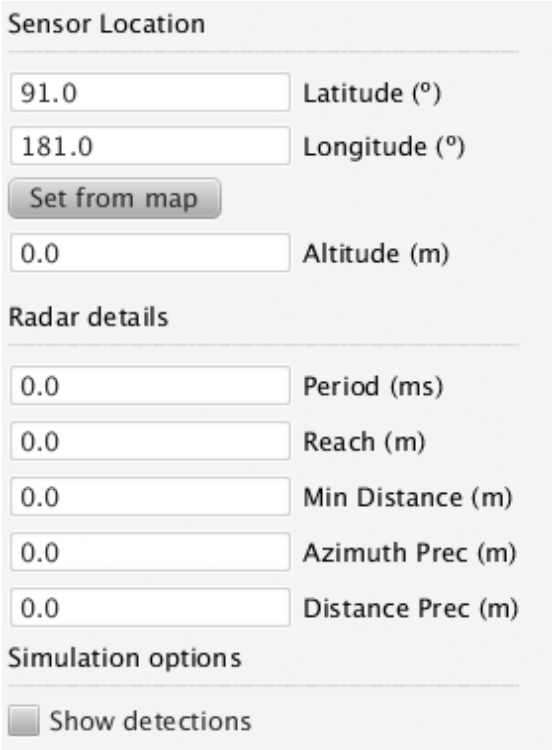

Figure 10: Configurable parameters for the Radar Sensor

margin for improvement in these topics, but they can be implemented in the future in the open source repository.

So, the user can place multiple radar sensors, like those shown in figure 9, that can be configured with different radar ranges, rotation periods, or resolutions. Then, once the simulation process starts, the simulated radars will start scanning the environment and providing detections for those tracks that are in their coverage. The configurable parameters for each radar sensor are described in figure 10.

5.3. AIS

The Automatic Identification System (AIS) is a maritime safety and vessel traffic system imposed by the International Maritime Organization (IMO). The onboard AIS system broadcasts position reports and short messages with infor-

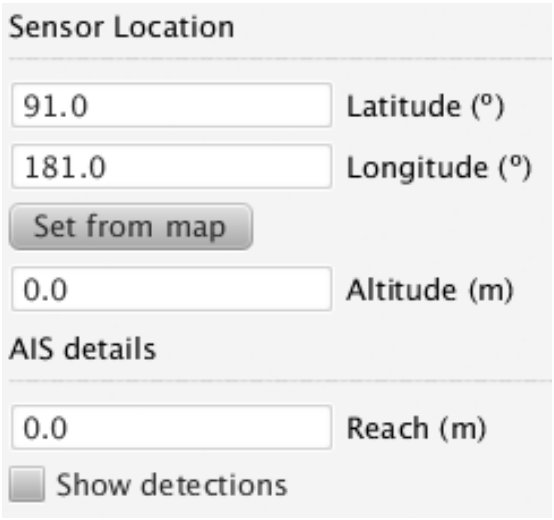

Figure 11: Configurable parameters for the AIS Sensor

mation about the ship and the voyage, using frequencies in the maritime VHF band. All messages are received by AIS stations that are normally deployed near to the coast.

The AIS stations are defined in the Player tool basically by their location in WGS84 coordinates, and a maximum range, as described in figure 11. So there are not additional parameters for this sensing system. In the figure 9 it is present an AIS station with two radar sensors.

5.4. Pant-Tilt-Zoom Cameras

A Pant-Tilt-Zoom (PTZ) camera is a sensor sensor widely used in surveillance systems. Such sensor allows the operator to change the Field Of View (FOV) to some region of interest to be monitored. These sensors are normally used in the maritime domain for monitoring specific procedures like vessel docking, vessel towing, or identifying possible threats. The PTZ cameras deployed in such kind of environments are specifically designed for these tasks, and provides a great vision range. Moreover, these sensors are also used in the research domain to provide vessels information by using computer vision techniques [27].

Figure 12: Example of simulated Pant-Tilt-Zoom cameras deployed in a maritime port. The polygons represents the calculated FOV according to camera pant, tilt, zoom and camera characteristics.

Such a feature opens the possibility to use this sensor as some kind of cooperative sensor that could provide additional information from targets of interest.

To simulate the camera sensor, it is necessary to configure the sensor size, and minimum and maximum focal length. Thus, it is possible to calculate the approximate projected FOV of the camera attending to the sensor parameters, current pan, tilt, and zoom. The result is a set of coordinates that can be represented as a closed polygon, like those available in figure 12. From this information it is relatively easy to calculate if a given point is inside the polygon or not, and then evaluate if the target is visible from the camera. This way, the cameras will work as an additional sensor in the environment, that will be able to provide target detections according to its current FOV. The reported targets will contain a local target identity, and its current location. The reporting frequency can be adjusted according to the camera Frames Per Second (FPS)

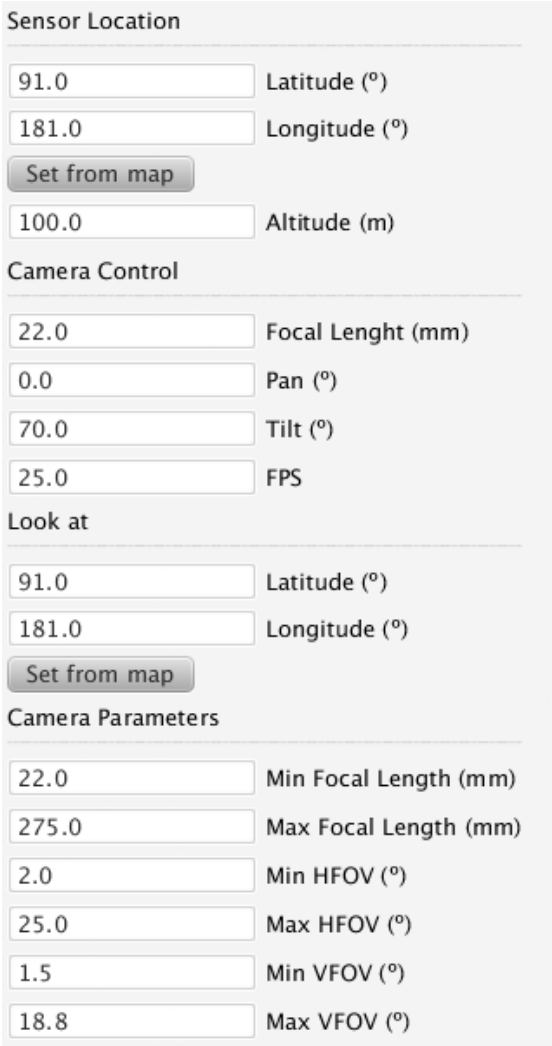

Figure 13: Configurable parameters for the Camera Sensor

that is a configurable parameter for each camera. These parameters are shown in 13.

5.5. Additional features

Aside from the simulation features described in detail, this tool also incorporate several additional features that can be useful when designing or evaluating scenarios. These features are mainly related with displaying elements over the

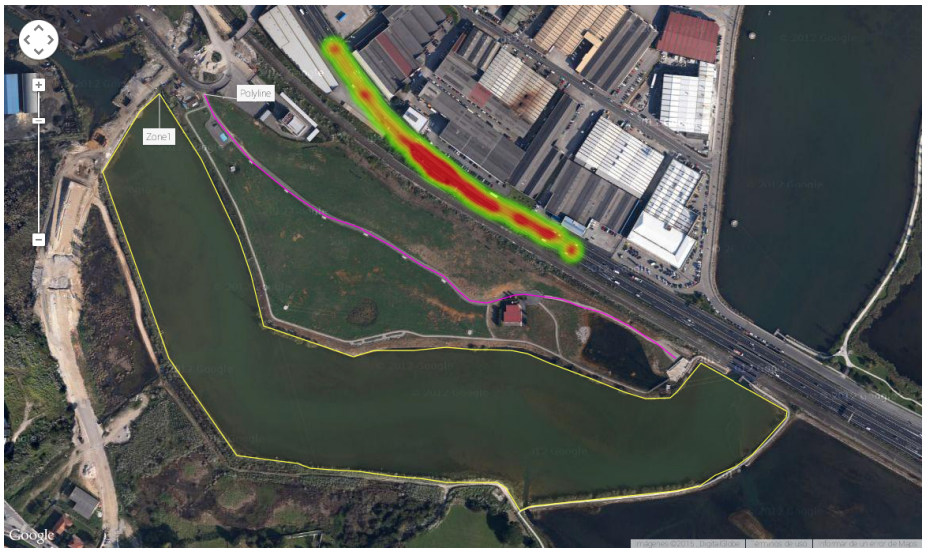

Figure 14: Sample representation of different entities available in the tool. In this example there are polygons, polylines, and heat maps.

map or calculating distances. Describing all those elements in detail is out of the scope of this paper, but it is worth to mention this kind of capabilities for a proper tool evaluation. The main additional features can be summarized in the following points, and some of them are illustrated in figure 14.

- Distance measurement: Measure distances between points, simulated tracks, tracks and sensors, and so on. Distances can be calculated as bearing-lines or using the great-circle path.

- Angle calculator: Sometimes it is quite useful to calculate angles over the map, so it is possible to define 3 geodesic points to calculate the angle between two lines.

- Polygon representation: It is possible to easily define polygons for describing areas or zones of interest just by clicking over the map. The tool also supports displaying polygon with holes.

- Polylines: This kind of geographic element is mainly used for describing

a vessel trajectory, but they can be also created apart of the vessel by clicking over the map.

- Heat Maps: A heat map is useful for displaying over the map the most transited areas. Such a way, it is possible to add heat map points in real-time to update the environment state.

- Real-time visualization: The Player tool defines a whole TCP/IP API for displaying, updating, and removing entities in real-time. For example, it is possible to display the output from the fusion system in real-time, and overlap the output with the generated simulation.

- Scenario multiplier: This capability allows building scenarios with a high number of detections. It takes a basic scenario and a multiplication area, and performs the generation of replicas of basic scenario in random locations within the multiplication area. The goal is providing high-load scenarios to test the fusion system in challenging conditions.

6. Results

This section provides some illustrative examples while using the tool for evaluating data fusion systems on maritime environments.

6.1. Crossing Vessels

In this example, it is shown a practical use case designed with the Player tool for evaluating a specific use case of a fusion engine. This particular one is related with two vessels approaching each other at an specific crossing distance. This type of scenario is useful for evaluating specific algorithms used in fusion systems, since there are two different entities so close that can interact between them. Simulating this use case is extremely useful as it is possible to easily adjust the crossing distance, crossing angles, relative crossing speed, course, etc. That kind of valuable situations and the flexibility obtained is quite difficult to

obtain from real environments, as it suppose to have real vessels doing different maneuvers, real AIS stations and Radars recording the information, then filtering, and so on.

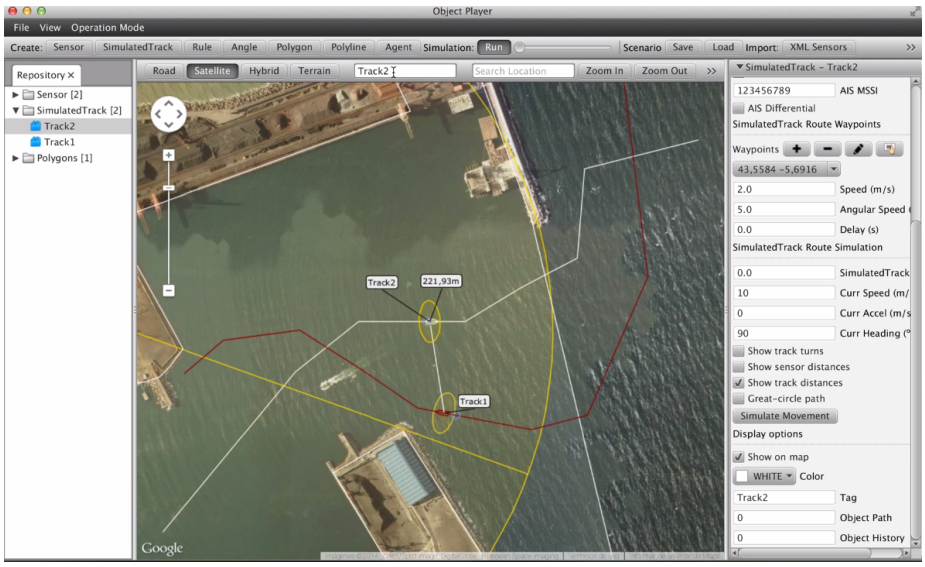

Figure 15: Example of designing scenarios for crossing vessel

In figure 15 it is presented an example scenario for evaluating such particular casuistic. It is also possible to see some of the features used in the simulator, like the distance between vessels that is updated in real-time, or the area of probability for each target detected by the radar sensor deployed in the scenario (in yellow). It is also available a video in YouTube[3] for a proper evaluation.

6.2. Cameras

Most of the modern VTS systems uses video surveillance systems for complement the information gathered by coastal sensors like radars or AIS stations. This kind of cameras are long-range cameras that can be used to zoom to some specific targets for view elements like IMO numbers, ship flags, cargo containers, and so on. With the Player tool it is possible to simulate those kind of

[3]https://www.youtube.com/watch?v=ytqfizjD-vU

Figure 16: PTZ Cameras simulation for monitoring vessels

long-range cameras also as another source of information. However, this feature can be also useful for the design and evaluation of autonomous monitoring algorithms automatically controlling PTZ cameras. These algorithms may work as autonomous entities to monitor the most relevant vessels according to different criteria, like distances to an specific area, crossing distances, velocity, rate of turn, and so on. As the simulation is happening in real time, it is possible to design algorithms receiving and evaluating the current simulated environment, to generate an output to modify simulated PTZ cameras state. This is possible as the Player tool provides an interface for managing the state of a PTZ camera in real time. Figure 16 illustrates the use of a PTZ camera to follow a simulated track.

This use case was exploited in [28] with a preliminary version of the Player tool to evaluate a multi-agent system with autonomous coordination to monitor different vessels of the environment. There is a video example of this integration in YouTube[4], and the 16

[4]https://www.youtube.com/watch?v=BrGGdKWwTRI

6.3. High Density Scenarios

Evaluating a data fusion engine is more than testing the low level algorithms or scenario particularities. In a real setup it is also quite important to evaluate the scalability of the system in order to assess its capabilities to ensure a proper operation QoS in high-load stress conditions. This kind of evaluation is practically impossible by using a real setup, as it would involve thousands of vessels sailing in the sensors coverage. So, the Payer tools offers some facilities to also help in the evaluation of high density scenarios. Starting with a base scenario with some sensors, and vessels, it is possible to replicate the setup by different orders of magnitude, so it is easier to test the fusion engine with hundred or thousand elements. These entities can be replicated by rotating or moving the trajectories from its origin inside a predefined area, by using the features for creating polygons as described in section 5.5. Hence, a fusion engine could have thousands of simulated trajectories with complex maneuvers and dynamics.

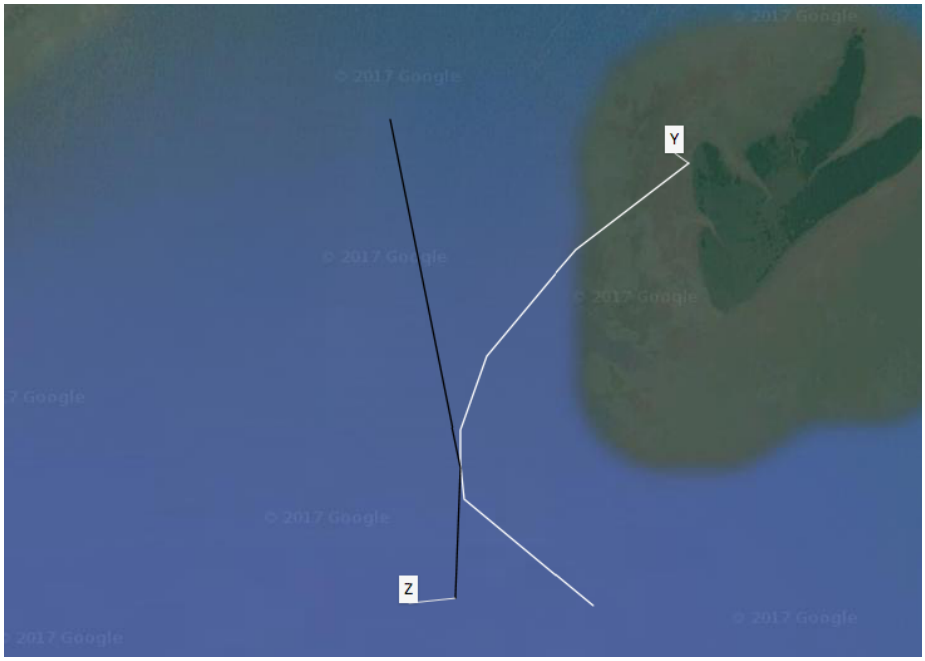

Figure 17: Example of high density simulated environments for evaluating fusion engine scalability.

Figure 17 represents a sample scenario where two different trajectories Y, and Z where designed for crossing at 50 meters of distance. After this sample trajectory is defined, then it can be later replicated along the whole scenario to test a data fusion under high loads. Figure 18 represents a simulated scenario with the mentioned crossing vessels that has been multiplied a thousand times. The new replicated entities are spread around a square area, without entering in the coast. This picture has been taken from a real Matlab Fusion engine that was integrated with the Player tool.

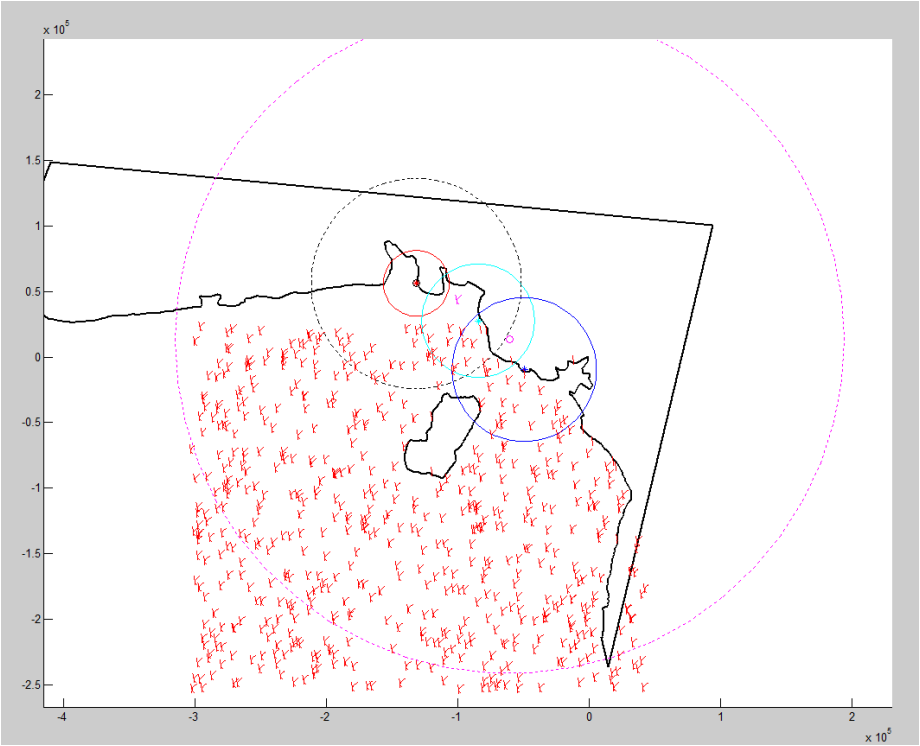

Figure 18: Example of high density simulated environments for evaluating fusion engine scalability.

6.4. Track Continuity

One of the most useful features of a data fusion system is providing a unique identifier for each vessel in the environment. It helps the operators in charge of

Figure 19: Example of simulating track continuity between different sensing areas.

the VTS, as it provides traceability for all detected entities, simplifying management processes. A fusion engine with this feature has to take into account that in this kind of environments there are different sensors covering different areas, and that the targets can move very long distances entering and exiting different sensor coverages. Therefore, it is required to evaluate fusion engines with vessels traveling along the whole scenario and sensors. There are multiple possible configurations in this use case, as a sensor covering other sensor coverage, sensors with partially overlapped fields of views, sensors not sharing any region, and so on. Evaluating such particular situations with a real setup is also quite complex, as it mainly depends on the hardware characteristics, the sensor locations, and the vessels traveling among different sensors. So, the Player tool results quite useful for this use case, as it is quite easy to define a sensor location, its range, its period, and also define vessels describing trajectories moving across the coverages of different sensors.

Figure 19 represents this specific use case, where there are deployed four different radar sensors along the coast with different coverage, some of them

with overlapping fields of view. Also there is one vessel describing a trajectory that pass along all the sensors.

7. Conclusions

This paper provides an overview of the main features of the open source Player tool, and mainly describes the general architecture, the main editable entities, the simulation models, and some practical use cases. This tool provides a basis for designing and simulating maritime surveillance entities that can be used to feed a data fusion system in real-time to evaluate its performance. This is quite useful for testing corner cases or specific situations that are not usually observable in the real-world. This tool provides an interface with a great usability, as the user can define entities like sensors, vessels, and trajectories directly over a map interface, to simulate its behavior just by clicking a button.

This tool has been extensively used and tested with several real data fusion projects to simulate and evaluate the performance of developed architectures. To cite a few related projects, in the work presented in [29], it was mainly used to validate a real fusion system developed to operate with EW (Electronic Warfare) and ISR (Intelligence, surveillance and reconnaissance) sensors on-board of patrol aircraft. In cite [28] was used for simulating and evaluating a collaborative multi-agent system controlling a set of PTZ cameras to monitor vessels of interest in a maritime surveillance environment. In [19] was used for defining and simulating multiple fusion scenarios, taking special attention the use of the tool for defining contextual zones and configurations. After testing the tool in different fusion-related domains, its release to the open source community can contribute to better develop multiple related projects.

Acknowledgment

This work was partially funded by projects MINECO TEC2014-57022-C2-2-R and TEC2012-37832-C02-01.

[1] N. Park, S. Cho, B.-D. Kim, B. Lee, D. Won, Security enhancement of user authentication scheme using ivef in vessel traffic service system, in: Computer Science and its Applications, Springer, 2012, pp. 699–705.

[2] G.-J. Duan, P.-F. Zhang, Research on application of uav for maritime supervision, Journal of Shipping and Ocean Engineering 4 (2014) 322–326.

[3] B. J. Daniel, A. P. Schaum, E. C. Allman, R. A. Leathers, T. V. Downes, Automatic ship detection from commercial multispectral satellite imagery, in: SPIE Defense, Security, and Sensing, International Society for Optics and Photonics, 2013, pp. 874312–874312.

[4] J. García, J. L. Guerrero Madrid, Á. Luis Bustamante, J. M. Molina, Robust sensor fusion in real maritime surveillance scenarios, 2010 10th International Conference on Information Fusion (FUSION).

[5] M. Vardjan, J. Porekar, Maritime surveillance scenario simulator, International Journal of Modeling and Optimization 2 (4) (2012) 449.

[6] Y. Fischer, A. Bauer, Object-oriented sensor data fusion for wide maritime surveillance, in: 2010 International Waterside Security Conference (WSS), IEEE, 2010, pp. 1–6.

[7] N. Chadil, A. Russameesawang, P. Keeratiwintakorn, Real-time tracking management system using gps, gprs and google earth, in: Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, Vol. 1, IEEE, 2008, pp. 393–396.

[8] M. Brovelli, P. Hogan, M. Minghini, G. Zamboni, The power of virtual globes for valorising cultural heritage and enabling sustainable tourism: Nasa world wind applications, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (2013) 115–120.

[9] S. Fotheringham, P. Rogerson, Spatial analysis and GIS, CRC Press, 2013.

[10] P. Braca, S. Maresca, R. Grasso, K. Bryan, J. Horstmann, Maritime surveillance with multiple over-the-horizon hfsw radars: An overview of recent experimentation, IEEE Aerospace and Electronic Systems Magazine 30 (12) (2015) 4–18.

[11] G. Battistello, M. Ulmke, F. Papi, M. Podt, Y. Boers, Assessment of vessel route information use in bayesian non-linear filtering, in: 2012 15th International Conference on Information Fusion (FUSION), IEEE, 2012, pp. 447–454.

[12] M. Vespe, M. Sciotti, G. Battistello, Multi-sensor autonomous tracking for maritime surveillance, in: 2008 International Conference on Radar, IEEE, 2008, pp. 525–530.

[13] A. Benavoli, L. Chisci, A. Farina, L. Timmoneri, G. Zappa, Knowledge-based system for multi-target tracking in a littoral environment, IEEE Transactions on Aerospace and Electronic Systems 42 (3).

[14] M. Guerriero, P. Willett, S. Coraluppi, C. Carthel, Radar/ais data fusion and sar tasking for maritime surveillance, in: 2008 11th International Conference on Information Fusion (FUSION), IEEE, 2008, pp. 1–5.

[15] C.-Y. Chong, Tracking and data fusion: A handbook of algorithms, IEEE Control Systems 32 (5) (2012) 114–116.

[16] X. Tian, Y. Bar-Shalom, Track-to-Track Fusion Architectures-A Review in Advances in Estimation, Navigation, and Spacecraft Control, Springer, 2015.

[17] D. L. Hall, J. Llinas, An introduction to multisensor data fusion, Proceedings of the IEEE 85 (1) (1997) 6–23.

[18] J. García, Á. Luis, J. M. Molina, Quality-of-service metrics for evaluating sensor fusion systems without ground truth, in: 2016 19th International Conference on Information Fusion (FUSION), IEEE, 2016, pp. 2251–2258.

[19] E. Marti, B. Gonzalez, A. Luis, J. Garcia, J. M. Molina, I. López García, Geographic context configuration in fusion algorithms for maritime surveillance, in: 2014 17th International Conference on Information Fusion (FUSION), IEEE, 2014, pp. 1–8.

[20] B. Khaleghi, A. Khamis, F. O. Karray, S. N. Razavi, Multisensor data fusion: A review of the state-of-the-art, 20013 16th International Conference on Information Fusion (FUSION) 14 (1) (2013) 28–44.

[21] H. Fargetton, J.-G. Siedler, Control of multi sensor system based on anomaly mapping and expert system, Sensor Data Fusion: Trends, Solutions, Applications, IEEE.

[22] E. Parsons, W. Morris, Edward wright and his work, Imago Mundi 3 (1) (1939) 61–71.

[23] W.-K. Tseng, H.-S. Lee, Navigation on a great ellipse, Journal of Marine Science and Technology 18 (3) (2010) 369–375.

[24] ITU, 1371-4,technical characteristics for an automatic identification system using time-division multiple access in the vhf maritime mobile band,, Electronic Publication, Geneva.

[25] E. McCollough, Photographic topography, Industry: A Monthly Magazine Devoted to Science, Engineering and Mechanic Arts (54) (1893) 65.

[26] M. S. Wilson, C. T. Miller, Using google maps web-application to create virtual plant maps for use as an online study tool in plant identification courses, HortTechnology 25 (2) (2015) 253–256.

[27] D. Bloisi, L. Iocchi, M. Fiorini, G. Graziano, Automatic maritime surveillance with visual target detection, in: International Defense and Homeland Security Simulation Workshop, 2011, pp. 141–145.

[28] A. L. Bustamante, J. M. Molina, M. A. Patricio, Information fusion as input source for improving multi-agent system autonomous decision-making in

maritime surveillance scenarios, in: Information Fusion (FUSION), 2014 17th International Conference on, IEEE, 2014, pp. 1–8.

[29] E. Marti, A. Luis, J. García, S. Oñate, C. Sanchez, S. González, Fusion of sensor data and intelligence in fits, in: 2013 16th International Conference on Information Fusion (FUSION), IEEE, 2013, pp. 342–349.