MODELING AND SIMULATION OF LAND AVOIDANCE BEHAVIOR BELONGING TO TACTICAL ENTITIES WITHIN A HIGH-FIDELITY SIMULATION ENVIRONMENT

Levent Senyurek([a]), Deniz Aldogan([b]), Mehmet Haklıdır([c])

TUBITAK BILGEM, Information Technologies Institute,
41470, Gebze-Kocaeli, TURKEY

[a]levent.senyurek@bte.tubitak.gov.tr, [b]deniz.aldogan@bte.tubitak.gov.tr, [c]mehmet.haklidir@bte.tubitak.gov.tr

ABSTRACT

The implementation of tactical entity land avoidance behavior integrated with realistic motion models is a crucial issue for a military simulation aiming high-fidelity. In this study, realistic surface platform land avoidance behavior has been implemented by utilizing a specific artificial intelligence path planning approach, namely, the vector field algorithm. The avoidance behavior is realized in conjunction with the automatic order execution behavior. A commercial simulation engine provided by the VR-Forces toolkit has been chosen for the implementation of computer-generated forces (CGF). Software modules and libraries for land avoidance as well as dynamic motion models and fuzzy controllers are all integrated into the component architecture of this CGF simulation engine. Results show that surface entities in the simulation exhibit realistic land avoidance behavior in parallel with order execution even in worst cases such as land crossing ordered paths and high ordered speeds.

Keywords: Land Avoidance, Vector Field Path Planning, Computer Generated Forces (CGF), Military Training Simulation

1. INTRODUCTION

Military training-critical simulation applications are utilized mainly because field exercises are too costly to be held for staff training, while high performance training results might only be obtained with high fidelity simulation.

In order to fulfill these high-fidelity constraints for military simulations, realistic CGF motion and control models should be implemented. Entities like surface platforms should overcome basic AI problems like land avoidance in addition to realistic dynamic motion and automatic order execution capability.

Land avoidance means handling the interaction of platforms with land realistically while the platforms continue to carry out their commanded tasks. Surface platforms should autonomously display avoidance from shallow waters with depth less than the platform draft. When it is impossible to stay away from land, such as when a platform positioned near to land is ordered to move directly towards land with maximum speed, land contact may not be prevented, which is but another realistic outcome.

This study aims to discuss the land avoidance behavior implemented in an on-going simulation system, which embodies a commercial CGF toolkit-integrated simulation of high-fidelity surface platforms. These platforms use complex motion equations and realistic fuzzy course and speed controllers which realize both mission execution and land avoidance behaviors.

For implementation, VR-Forces CGF unit has been chosen due to its easy customization and capability extension facilities. An HLA-compliant simulation environment, which uses RTI (Run-Time Infrastructure) for communication and coordination services, has been realized. HLA compliant simulations and currently offered HLA services are discussed in detail in Duman et al. (2003).

The VR-Forces based simulation components used in this study are front-end and back-end applications. The back-end application is where the actual CGF simulation takes place whereas the remote front-end GUI controls the back-end(s) existing in the whole simulation. The GUI application also provides features for scenario management and simulation execution control. The surface entities originally simulated by VR-Forces neither have movement models of appropriate fidelity nor exhibit realistic behavior such as land avoidance or fuzzy control. The details about the inner workings of VR-Forces architecture can be found in VR-Forces Developer's Guide (2006) or "VR-Forces The Complete Simulation Toolkit" article on the website belonging to the company named "MAK Technologies".

The software modules for the motion modeling and control of surface platforms have been implemented in the C++ programming language and have been integrated into the component architecture of VR-Forces CGF application (VR-Forces back-end) as composite objects.

The rest of the paper is divided into 4 sections. In Section 2, the land avoidance approach chosen in this study is given. Section 3 discusses the integration of land avoidance behavior into the simulated entity response. Section 4 explains the experimental setup and discusses the corresponding results. Finally, conclusions are given in Section 5.

2. LAND AVOIDANCE APPROACH

Automatic land avoidance problem is a specific type of path planning problem in the area of artificial intelligence (AI).

Among various methods in the literature, that attempt to solve the path planning problem, the vector field path planning algorithm, which is widely used in the field of autonomous robot control (Koren and Borenstein 1991; Khatib and Chatila 1995; Wolf, Robinson and Davies 2004), has been chosen as the most appropriate approach. It is mainly due to the fact that this algorithm is developed for predefined fields. Most of the calculations necessary for avoidance behavior can be executed off-line, increasing simulation performance.

A-star algorithm which is widely utilized for AI path planning problems is not suitable in these cases since high precision model platform behavior is required. Platform movement should highly depend on platform dynamics, so that a proper path cannot be dictated point by point.

VR-Forces CGF toolkit also provides land avoidance for entities by supplying information about whether land exists or not on a given trajectory vector. However, as platforms do not merely exhibit movement towards the bow direction, detection of land regions on one specific direction would not be sufficient enough. For the platforms to display sound land avoidance behavior where platforms not only avoid the coastal line, but also avoid regions shallower than their drafts, VR-Forces facility concerning just coast avoidance would not satisfy simulation needs. Vector field path planning is an appropriate solution considering these requirements.

The vector field path planning method identifies attraction and repulsion potentials and resultant commanded direction and speed for platforms are appeared to be the result of a weighted vector summation of these mostly conflicting potential force fields. Destination points or desired directions and commanded velocity values constitute the attraction potential for the platforms. On the other hand, a repulsion potential is generated by the obstacles on the field (namely the shallow waters and the coasts on the scenario map). Platforms behave according to the vector summation of these two potentials which is a function of the sea bottom, the current platform location and velocity. As a result, platforms execute their given orders while also avoiding land.

Repulsive normalized vectors for all possible platform coordinates are calculated beforehand with a certain resolution owing to the fact that all training fields are predefined. Afterwards, online land avoidance behavior is carried out by utilizing these previously computed data.

Land avoidance behavior is activated whenever the distance between platforms and shallow waters is less than 1 kilometer. This distance has been determined as an integer greater than the maximum stopping distance of all surface platforms. Avoidance vectors for all the points in the training fields are computed considering this maximum approach distance. Avoidance normalized vector calculation for each possible platform position is performed by taking into account all the land points in the square region of size 2 x 2 kilometers centered by the corresponding coordinate. The land points for surface platforms are defined as shallow waters with depth less than the average platform draft value (6m).

Avoidance value at a specific point is represented by a normalized avoidance potential vector (V_s). To obtain this vector, firstly, two dimensional gravity centers of the land regions inside the corresponding 2 x 2 kms. square is calculated. Then, the direction formed as a line between the gravity center and the possible platform coordinate is obtained. The value of the avoidance vector is inversely proportional to the distance between the two points. In other words, if this distance is minimum (0 km), then the value of the avoidance vector is 1, whereas if it is maximum (F2 km), then this value is calculated as 0. As a result the value of the normalized vector might be calculated with Equation 1.

$$| V_s | = (F2 - d)/ F2 \qquad (1)$$

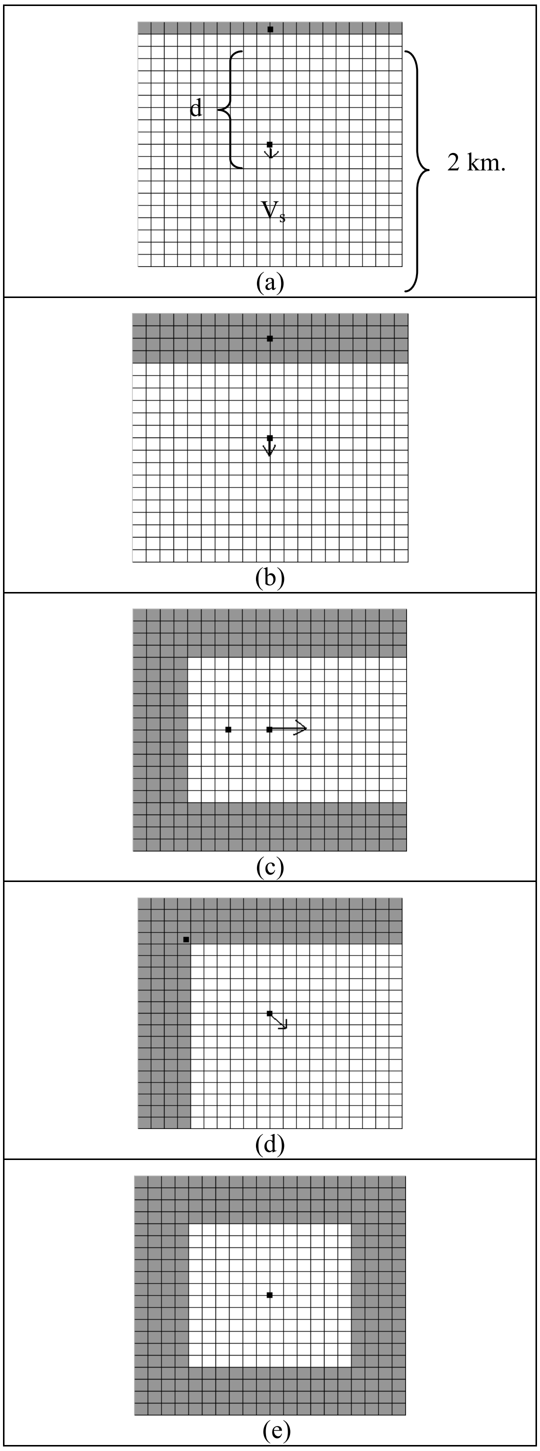
In the above equation, V_s is the normalized avoidance potential vector, while d is the distance between land gravity center and square center (platform location).

In Figure 1, approximate potential vectors calculated for different situations are visualized relative to each other. In the figure, the last scene displays the platform covered by land regions. In this special case, although the avoidance potential vector is at its maximum value, no avoidance is applied since the gravity center of land regions and the platform locations are the same and thus direction is undefined.

The next section begins with a brief overview of simulated surface platform components accomplishing commanded tasks. Then, the integration of the avoidance into the control mechanism of surface entities is explained.

3. INTEGRATION OF LAND AVOIDANCE BEHAVIOUR INTO THE SIMULATION MODELS

There are two main issues that must be taken into account to generate realistic navigation for surface platforms in military simulation systems: a high-fidelity motion model and a realistic control mechanism that provides the motion model with appropriate input parameters for completing commanded tasks. Surface platforms in the current system utilize a commercial toolkit independent software library that enables high-fidelity motion modeling. Details such as coordinate frames, motion equations, forces and moments acting on platforms along with simulation results are described in Haklidir, Aldogan and Tasdelen (2008). The software module for the motion model has been integrated into a specific VR-Forces component, namely an actuator component which is responsible for realizing the movement of the entity.



(a)

(b)

(c)

(d)

(e)

Figure 1: Representation of land avoidance potential vectors for different situations. Shaded regions represent land areas. The two marked points represent the gravity center of the land

region and the platform coordinates (central points).

In order to simulate quartermasters for the guidance of surface platforms, two distinct fuzzy logic controllers (FLC), namely heading and speed controllers, are utilized. These controllers obtain the desired heading and speed values corresponding to the current commanded task from the VR-Forces task controller. Afterwards, they calculate the necessary control parameters, i.e. rudder angle and shaft values, by utilizing one of the two different instances of a flexible fuzzy logic library (FLL). These two FLL instances are devoted to heading and speed control respectively and each of them are customized with a parameter and rule set in accordance with their specific type of control. A more detailed explanation as well as simulation results for FLC's can be found in Senyurek et al. (2008). After FLC's designate the values of the control parameters as a function of the inputted course and speed values and the current the state of the platform, platform's motion model receives these outcomes and calculates the next state according to the supplied control parameters.

The task control process deviates from its normal flow whenever there are land points within a certain range around the entity. In this case, current task related heading and speed values are first altered by land avoidance function and then are inputted to heading and speed FLCs as final desired heading and speed values. This process is accomplished in a way that permits the platform carry on its current task while causing it to exhibit adequate land avoidance behavior for fully simulating human expert behavior.

The task controllers in VR-Forces architecture are derived to encapsulate the land avoidance modeling along with fuzzy logic control just as VR-Forces actuator components have been derived to encapsulate motion models. Detailed information about the extension of the VR-Forces architecture and the integration of the software modules can be found in Aldogan et al. (2009).

Figure 2 displays the organizational structure of higher level task and avoidance controllers along with low level rudder and shaft controllers, each of which utilize low level FLL instances.

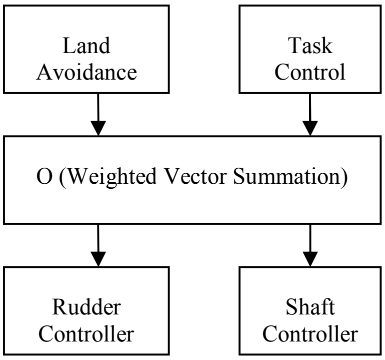


Figure 2: Organizational Structure of Controllers

The direction (L) and the speed (v_p) information dictated by the commanded task of the platform are united so as to form a task vector $V_g = [x_{vg}; y_{vg}]$ (eq. 2, eq. 3). This vector then undergoes vectorial summation with the scaled normalized avoidance potential vector (V_s). V_s is read from the database according to platform location and is scaled by an avoidance weight coefficient (k_s) (eq. 4). Direction and velocity information is retrieved from the summation vector (V_T) formed and is fed to rudder and shaft controllers as final demanded values.

$$|V_g| = v_p \qquad (2)$$

$$y_{vg}/\ x_{vg} = \tan(L) \qquad (3)$$

$$V_T = V_g + k_s.\ V_s \qquad (4)$$

Avoidance weight coefficient k_s should be directly proportional to the velocity of the platform (eq. 5). Since a platform that approaches towards land with maximum speed should apply maximum amount of avoidance, while a non-moving platform does not show any avoidance behavior at all.

$$k_s = k_1.\ v_p\ (t) \qquad (5)$$

A proper k_1 value might be obtained considering the worst-case scenario where the avoidance potential vector is at its maximum value ($d = 0$; $|V_s| = 1$). In such a case, automatic control must aim maximum velocity at the opposite direction in order to stop the platform that is moving towards the land region with maximum velocity (eq. 6). The symbol $V_p(t)$ refers to the velocity of the platform at time t.

$$V_T = - V_p (t) \qquad (6)$$

As the platform is headed directly to land, it can be assumed that V_s and $V_p(t)$ vectors have the same direction. Since V_s having maximum normal gain is a unit vector ($|V_s| = 1$), it can be rewritten as a function of $V_p(t)$ vector and platform's current speed(7).

$$V_s = -V_p(t)/v_p(t) \qquad (7)$$

The coefficient k1 can be calculated by placing k_s value calculated at eq. 5, V_T value calculated at eq. 6 and V_s value calculated at eq. 7 into eq. 4 while keeping the assumption that the commanded speed is achieved ($V_g = V_p(t)$) (eq. 8). Therefore, the avoidance weight coefficient (k_s) can be written in terms of platform's instant speed value $v_p(t)$ (eq. 9).

$$-V_p(t) = V_p(t) - k_1 . v_p(t) . V_p(t)/v_p(t) \text{ P } k_1=2 \qquad (8)$$

$$k_s = 2.v_p(t). \qquad (9)$$

By placing the k_s value calculated with eq. 9 into eq. 4, the final desired velocity vector for the current situation(V_T) can be formulated in terms of the velocity vector corresponding to the commanded task(V_g), the normalized avoidance vector at platform's location (V_s) and the instant magnitude of the velocity vector of the platform ($v_p(t)$) (eq. 10). The total velocity vector calculated in this way is decomposed into its magnitude and angle values and these values are inputted to the shaft and rudder controllers respectively.

$$V_T = V_g + 2.v_p(t).\ V_s. \qquad (10)$$

In order to maximize performance, the resultant vector calculation is performed only if there is land avoidance effect ($|V_s| > 0$) and the corresponding platform is moving ($v_p(t) > 0$).

4. EXPERIMENTS&DISCUSSION

Since the most fundamental aspect in this study is the land avoidance behavior of platforms, it must be verified that sea floor remains deeper than the draft during platform cruise.

The average draft has been set as 6 meters for surface platforms. The platforms shouldn't enter shallow waters while they approach their targets even when a route passing through land regions (islands etc.) is ordered.

In order to verify the land avoidance behavior of surface platforms, different platforms are commanded to move with different velocity and courses within the same test scenario. The sea floor depths of map points which the platforms are passing over are recorded into files during the scenario execution.

The test scenarios have been executed in the VR-Forces environment and recorded log files have been converted into figures by utilizing MATLAB 2007a. The map chosen for the test scenario, which contains sea depth information with 50m x 50 m resolution, covers a bay of Neverland with two small islands in the middle, one bigger island on the north and a cape on the south.

Figure 3 displays the initial state of the land avoidance test scenario. Surface platform 1 is commanded to move with a course of 80 degrees (wrt. north) so that its commanded route passes over the small islands. Surface platform 2 is ordered to move towards west so that the route given crosses the cape in the south. Surface Platform 3 is ordered to move to the waypoint named "Point 1", while the bigger island lies between the platform and the waypoint. Surface platform 4 is commanded to move directly towards south so that the route given passes over the smaller island in the middle of the map. Surface platform 5 is ordered to move along a route that passes between the islands. Surface platform 6 is ordered to move towards east, in other words, directly towards the land region. Surface platform 7 is expected to move along a route that is directed to west and that crosses the cape in the south. Finally, surface platform 8 is commanded to move to a waypoint named "Point 2", which is on shallow waters.

The initial heading values of all the platforms are in alignment with their ordered direction. Each platform is ordered to begin and continue movement with its maximum speed, which is dictated by the parameter set of its motion model and which is obviously the worst speed value for land avoidance performance.

Figure 4 shows the position and the orientation of entities approximately after 10 minutes of scenario execution. It can be observed that the platforms follow reasonable paths both considering land avoidance and order obedience. They head towards their goals by moving around islands and capes that cut their routes.

The platforms which are commanded to move directly towards the land regions with maximum speeds (Surface platforms 6 and 8) firstly slow down and maneuver but then crash to land. Figures 5, 8, 11, 14, 17, 20, 23 and 26 display the sea floor depth values of the points over which the platforms pass. It can be observed that during scenario execution, the floor depth remains more than 6 meters for all the platforms. The platforms that crash to land show a fixed sea floor depth of 6 meters after they run aground.

Figure 29 displays the followed trajectories of surface platforms. This figure along with the ones displaying bow direction and speed changes for each platform show that land avoidance is accomplished via acceptable course and speed deviations from the commanded values so that realistic behavior is maintained.

Surface Platform 2 slightly turns its bow direction towards north twice until it passes around the cape and tries to keep the commanded course of 270 degrees after each of these turns. Meanwhile, these maneuvers cause decreases in its speed. Once it passes around the cape, it manages to cruise with the commanded heading (270 degrees) and speed values (32 knots). Similarly, surface platforms 1, 3, 4, 5 and 7 also performed maneuvers to change their bow directions so that they can pass around land obstacles. Similarly, they decreased their speed during these maneuvers. After passing by land regions, they reached the commanded speed and bow direction values.

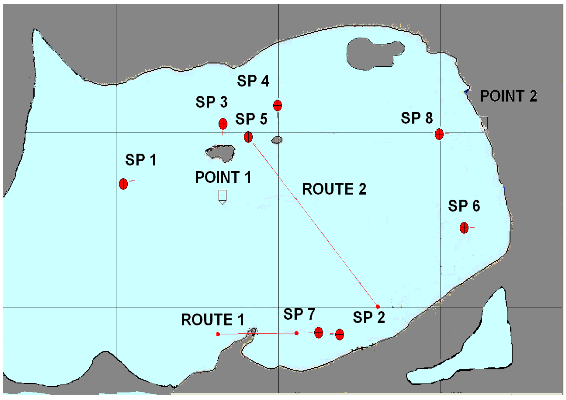


Figure 3. The initial state of the avoidance test scenario for surface platforms (SP refers to a Surface Platform)

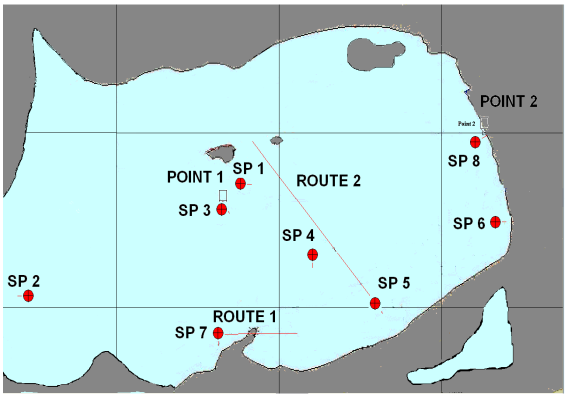


Figure 4. The final state of the land avoidance test scenario for surface platforms (SP refers to a Surface Platform)

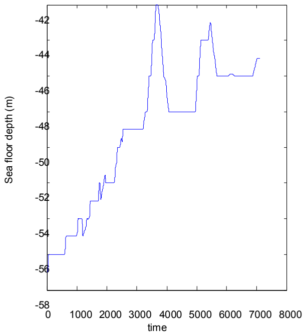


Figure 5. Sea floor depth for Surface Platform 1

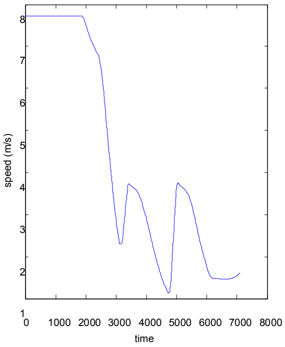


Figure 6. Speed change for Surface Platform 1

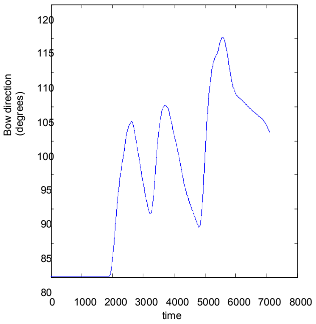


Figure 7. Bow direction change for Surface Platform 1

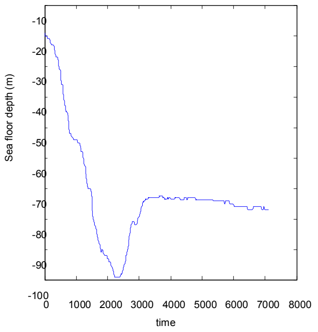


Figure 8. Sea floor depth for Surface Platform 2

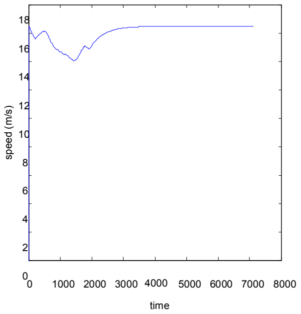


Figure 9. Speed change for Surface Platform 2

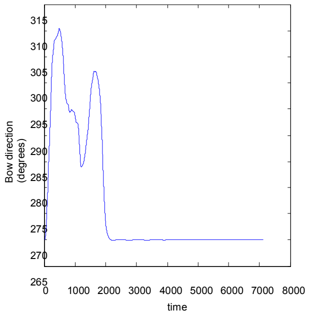


Figure 10. Bow direction change for Surface Platform 2

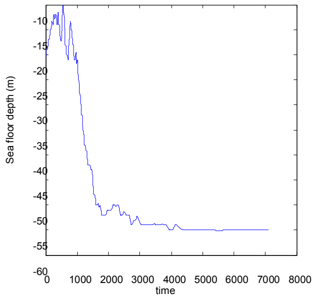


Figure 11. Sea floor depth for Surface Platform 3

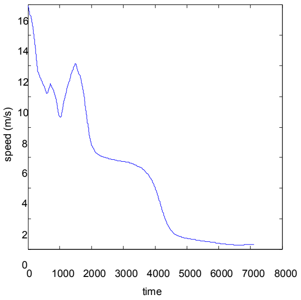


Figure 12. Speed change for Surface Platform 3

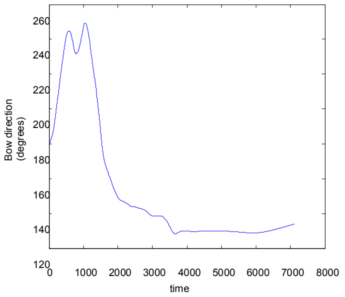


Figure 13. Bow direction change for Surface Platform 3

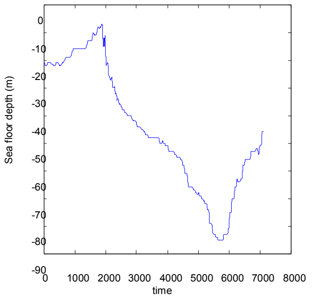


Figure 14. Sea floor depth for Surface Platform 4

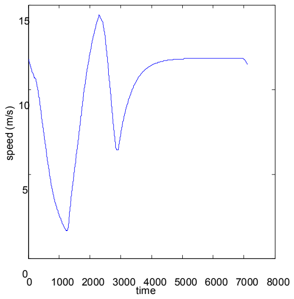


Figure 15. Speed change for Surface Platform 4

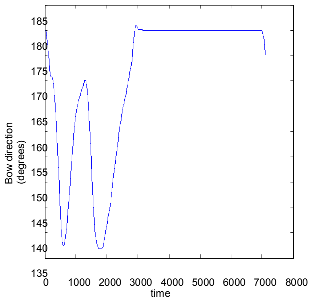


Figure 16. Bow direction change for Surface Platform 4

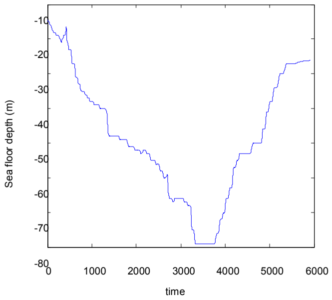


Figure 17. Sea floor depth for Surface Platform 5

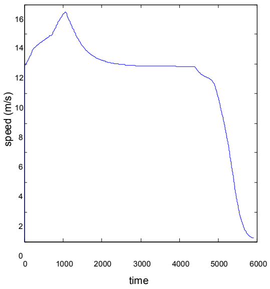


Figure 18. Speed change for Surface Platform 5

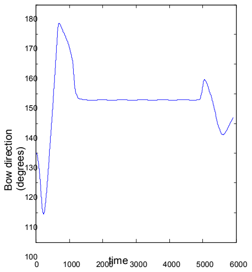


Figure 19. Bow direction change

for Surface Platform 5

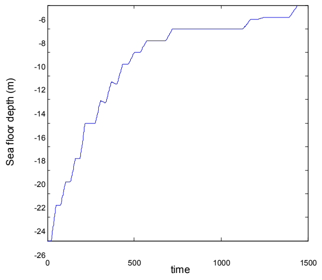


Figure 20. Sea floor depth for the Surface Platform 6

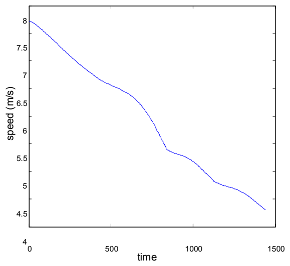


Figure 21. Speed change for Surface Platform 6

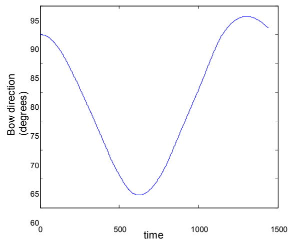


Figure 22. Bow direction change for Surface Platform 6

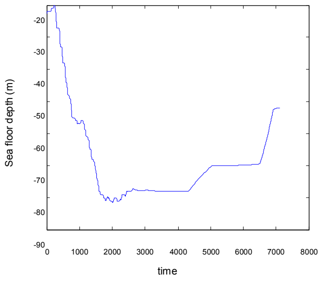


Figure 23. Sea floor depth for Surface Platform 7

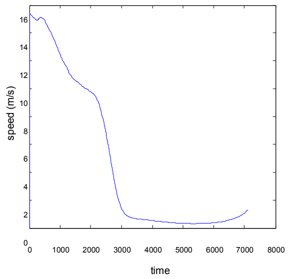


Figure 24. Speed change for Surface Platform 7

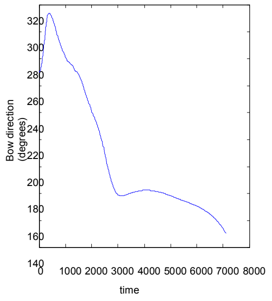


Figure 25. Bow direction change for Surface Platform 7

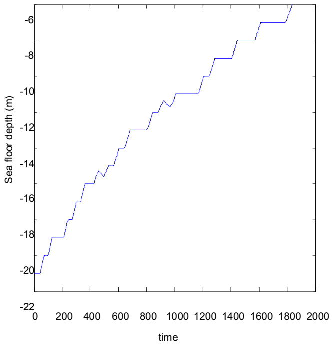


Figure 26. Sea floor depth for Surface Platform 8

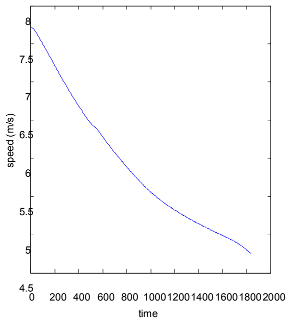


Figure 27. Speed change for Surface Platform 8

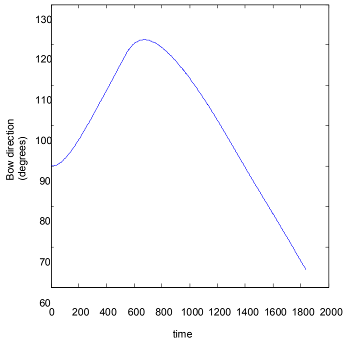


Figure 28. Bow direction change for Surface Platform 8

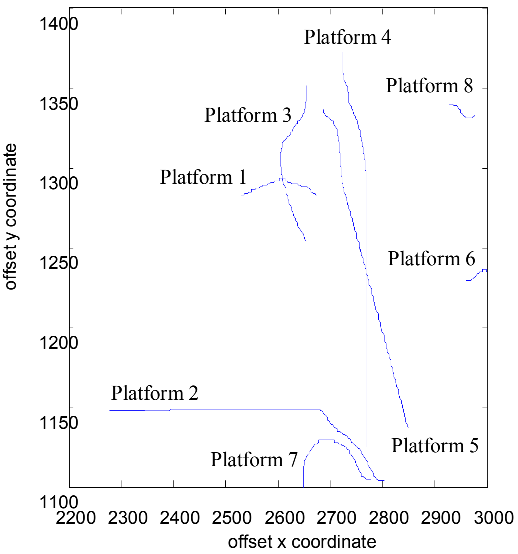


Figure 29: All followed trajectories belonging to the surface platforms during scenario execution

Since surface platforms 3, 5 and 7 are ordered to move to specific points on the map, namely to point 1 and to the end points of route 2, and route 1 respectively, during the last minutes of the scenario execution, their speed values approached to zero since they arrived at the commanded destination points. Surface platform 8, which began its movement at a point very close to the commanded waypoint and shallow waters, decelerated throughout its whole cruise time until it crashed to land. Likewise, surface platform 6 decelerated and changed its bow direction in order not to crash to land. These two platforms were positioned too close to shallow waters and were ordered to cruise towards land to assure that they run aground eventually. As a result, this special case of land interaction was also pointed out.

Data exhibiting platforms' behavioral details prove that the surface platform models in this study accomplish land avoidance behavior in a realistic way, via the simulation of a realistic quartermaster decision making process exploiting both fuzzy logic and vector field path planning.

5. CONCLUSIONS

This paper introduces an on-going study on a military training simulation system in which land avoidance behaviors are integrated into high fidelity motion models guided by fuzzy logic controllers. Thus, simulated entities accomplish both land avoidance and order execution while they preserve consistency with their dynamic models. In this way, it has been achieved to sufficiently mimic cruise control performed by platform captain and quartermaster. The whole design is integrated into the component architecture of VR-Forces, which provides a framework for developing Computer Generated Forces (CGF) applications. The overall integration has led the surface platforms of the VR-Forces application to reach high-fidelity which is vital for training-critical military simulations.

REFERENCES

Aldogan, D., Haklidir, M., Senyurek, L.H., Koksal, S., Eroglu, O., Akdemir, C., Franko, S., Tasdelen, I., Akgun, S., 2009. A General Approach to high fidelity modeling, simulation and control of tactical entities and implementation in a commercial computer generated forces toolkit. 21th European Modeling and Simulation Symposium, EMSS 2009, Tenerife, Spain.

Duman, I., SavaSan, H., Tahin, I., 2003. Bir Modelleme ve Simülasyon Standardı olarak HLA ve Simülasyon Uygulamalarında Kullanımı, (In Turkish), Turkish Navy Research Center Command Publications.

Haklidir, M., Aldogan, D., Tasdelen, I., 2008. High Fidelity Modeling and Simulation of Surface Platforms in a Computer Generated Forces Toolkit, SUMMERSIM 2008, Edinburgh, Scotland.

Khatib, M., Chatila, R., 1995. An Extended Potential Field Approach for Mobile Robot Sensor-Based Motions, Proceedings of the Intelligent Autonomous Systems, pp. 490-496, IAS-4, IOS Press, Karlsruhe Germany.

Koren, Y., Borenstein, J., 1991. Potential Field Methods Their Inherent Limitations for Mobile Robot Navigation, IEEE Conference on Robotics and Automation, pp. 1398-1404, Sacramento California.

Senyurek, L., Koksal, S., Genc, H.M., Aldogan, D., Haklidir, M., 2008. Implementation Of Fuzzy Control For Surface Platforms In A Computer Generated Forces Toolkit, International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA08), Vienna, Austria, 10-12 December.

VR-Forces Developer's Guide, MAK Technologies, 2006. Chapter 2, The VR-Forces Simulation Engine, Chapter 5, Creating New Components, Chapter 10, The VR-Forces Remote Control API.

VR-Forces The Complete Simulation Toolkit. MAK Technologies. Available from http://www.mak.com/products/vrforces.php [accessed 11 April 2011]

Wolf, J.C., Robinson, P. , and Davies, J.M., 2004. Vector Field Path Planning and Control of an Autonomous Robot in a Dynamic Environment," FIRA Robot World Congress.