Simulation-Based Approach to Synthesizing Maritime Interaction Scenarios for Testing Autonomy

Benjamin E Hargis, Yiannis E Papelis
Old Dominion University
Norfolk, VA
bharg010@odu.edu, ypapelis@odu.edu

ABSTRACT

As the desire to equip Autonomous Surface Vessels with increasingly complex autonomy grows, development of effective testing methodologies for autonomy-based behaviors is of importance. Traditional simulation-based single event testing, however, lacks the ability to evaluate the performance of autonomy algorithms before in-situ testing takes place. This necessitates the use of long-run simulations which more accurately represents the intended operating timelines. One issue with designing events in long-run simulations with multiple consecutive events is that the state of the system under test is not known before runtime. Therefore, methods for designing interactions that are specific, yet versatile and flexible are required. This paper presents a method to deterministically synthesize maritime traffic interactions that can be presented to a system under test regardless of the state. Method development was motivated by three factors. First, the developed method needs to allow a targeted interaction design. Second, the method should accommodate closed-feedback testing approaches that can select testing situations based on prior performance of the SUT. Finally, the method must facilitate testing time compression by reducing and/or eliminating time during the simulation when the SUT is not being stressed by external factors. The method was validated via a simulation. Various scenarios were designed, both trivial and non-trivial and the resulting interaction data was recorded. The overall approach and results presented here validate the use of this method to synthesize varying intensity maritime traffic interactions. Results indicate that the approach can enable more robust evaluation of maritime autonomous algorithms.

ABOUT THE AUTHORS

Benjamin Hargis has a Bachelor of Science and Master of Science degrees in Mechanical Engineering from Tennessee Technological University with a focus in mobile robotics. He has 5 years of modeling and simulation experience. His graduate work focused on Newtonian dynamic modeling of skid-steer mobile robotics and simulation-based verification. As a Civil Servant at NASA, He has contributed significantly to the development of models to predict ballistic trajectories of payloads on a scaled model as well as developed a Monte Carlo simulation to determine performance characteristics of Inertial Transfer, a low technology readiness level concept for efficiently transporting payloads in space. His current research work is in simulation-based evaluation of autonomous algorithms and synthesis of simulation events during long-run simulations. He has worked on the N-ASTC/VMATE project and has contributed significantly to the development of intelligent behaviors to be used in autonomous algorithm evaluation.

Yiannis Papelis obtained a BS degree in electrical engineering with a minor in CS from Southern Illinois University, followed by Master's and Ph.D. degrees in Electrical & Computer Engineering at Purdue University and the University of Iowa respectively. He joined Old Dominion University in 2007 and is currently a Research Professor at the Virginia Modeling Analysis & Simulation Center where he serves at the Chief Technology Officer. Before joining ODU, Dr. Papelis' research focused on use of high-fidelity driving simulators for studying advanced automotive technologies, in-vehicle devices, and highway design. His research on the effectiveness of Electronic Stability Control was referenced by the Dept. of Transportation when making it mandatory to include ESC in all passenger vehicles built on or after 2012. Since joining ODU, Dr. Papelis has maintained an active research agenda in robotic autonomy, simulation, semi-immersive VR applications and serious games. Over his career he has been awarded research funding as principal and co-principal investigator in excess of $20 million, with his research funded by the US Navy, Dept. of Education, NASA, Air force laboratory as well as industry.

Simulation-Based Approach to Synthesizing Maritime Interaction Scenarios for Testing Autonomy

Benjamin E Hargis, Yiannis E Papelis
Old Dominion University
Norfolk, VA
bharg010@odu.edu, ypapelis@odu.edu

INTRODUCTION AND RELATED WORK

As the desire to equip Autonomous Surface Vessels (ASVs) with increasingly complex autonomy grows, development of effective testing methodologies for autonomy-based behaviors is of importance. This work focuses on scenario-based testing. There are two recognized approaches to generate logical scenarios, knowledge-driven and data-driven (Nalic, Mihalj, Bäumler, Lehmann, and Bernsteiner, 2020). The primary difference being that data-driven scenarios are the result of measurement data such as the canonical encounters derived from recorded flight data (Refai, Abramson, Lee, and Wu, 2019), while knowledge-driven scenarios are defined by experts (Zhao, Yao, Sun, Zhang, and Bai, 2018) or derived by relationships between scenario configurations and metrics (Mullins, Stankiewicz, Hawthorne, and Appler, 2017).

Traditional simulation-based, single event testing, however, lacks the ability to evaluate the performance of autonomy algorithms before in-situ testing takes place. Data-driven scenarios are useful for providing actual encounters that were observed, yet encounters for a single mission represent only a small percentage of the mission time and methods to use only those portions of the mission data are being developed (Refai, Abramson, Lee, and Wu, 2019).This illustrates that methods must facilitate testing time compression by reducing and/or eliminating time during the simulation when the system under test (SUT) is not being stressed by external factors. As both approaches are utilized in research, scenario design software should also allow a targeted interaction design which can leverage both knowledge-driven and data-driven approaches.

Exhaustive testing using either scenario generation approach is intractable due to the high dimensionality of the solution space (Koopman and Wagner, 2016). Methods such as performance boundary identification (Mullins, Stankiewicz, Hawthorne, and Appler, 2017) (Stankiewicz and Mullins, 2019) have been developed to determine scenarios of interest, namely those scenarios that lie near performance boundaries in the solution space. These methods require that testing software accommodate closed-feedback testing approaches that can select testing situations based on prior performance. Autonomous test event generation is a unique challenge (Huang, Wang, Lv, and Zhu, 2016) (Broy, 2006) which requires the judicious selection of test event parameters (Porres, Azimi, and Lilius, 2020). This becomes challenging when considering mission-length scenarios.

Mission-length scenarios are a series of events or interactions which are presented to a SUT during a single simulation run. These long-run simulations offer the ability to present a SUT with multiple test encounters without needing to stop and restart simulations as well as assist in the study of SUT emergent behaviors during missions which potentially last hours. One issue with designing events in long-run simulations with multiple consecutive events is that the state of the SUT is not known before runtime. One approach to this problem is to synthesize events using a method that does not rely on the SUT's current world frame position. Therefore, methods for designing interactions that are specific, yet versatile and flexible are required.

This paper presents a method to deterministically synthesize maritime traffic interactions that can be presented to a system under test regardless of the state of the SUT. This method can be utilized to generate multiple consecutive events during long-run simulations regardless of the SUT location with respect to the world frame. Test time compression is supported as this method can be integrated into a simulation to allow multiple designed encounters to take place during a single run. Further, this approach can be integrated into a closed feed-back testing approaches to allow autonomous test event generation and synthesis during a single simulation.

SIMULATION ENVIRONMENT

The overarching simulation framework hosting the event synthesis method is the US Navy's Autonomous Systems Test Capability (ASTC). ASTC is a larger project which aims to develop the necessary framework for modifying and integrating existing tools, and developing new tools as needed to provide simulation-based testing capabilities to ensure ASVs are safe before deployment. Within ASTC, the Virtual MAritime Testing Environment (VMATE) provides a high-fidelity simulation environment which includes all facilities necessary to run continuous time, multi-agent, deterministic simulations that can act as a stimulus for simulated ASVs. Sensors models, environment models, movement models, intelligent traffic models and autonomy behaviors can be rapidly integrated into a unified simulation environment that can run on a local machine or in a cloud computing environment. A detailed description of ASTC and VMATE is beyond the scope of this paper but there are two specific features that directly support the work described in this paper. The first feature is the ability to populate the simulation with traffic consisting of intelligent agents that as a basic behavior follow a high-level route but whose behavior can be overridden as necessary to synthesize events. The second feature is the inclusion of triggers, which are runtime coordinating agents that allow activation of the event synthesis methods based on a wide range of runtime conditions, including but not limited to simulation time and geographical proximity conditions. Within any VMATE simulation, a trigger can be setup to signal one of the autonomous entities to switch from its autonomous behavior and be controlled by the event synthesis code (effectively becoming the intruder) and which will target any other intelligent agent (which effectively become the SUT) and thus create repeatable scenarios.

EVENT SYNTHESIS METHOD

The basis of this approach is the application of Relative-Motion principles from Dynamics. These principles allow for event synthesis to be conducted in the SUT's local frame and then mapped or transformed to the world frame such that, from the perspective of the SUT, the presentation of event will be the same regardless of the SUT state. The event timeline has been discretized, by time, into three phases (Fig. 1). The first phase, orchestration, begins at time t_0 which marks the beginning of the event synthesis process. The second phase, implementation, marks the end of the orchestration phase at the start time, t_s, and indicates the time when the intruder, INT, begins the desired event trajectory. The final phase, post-event, begins following the event time, t_e. Although the interaction between the intruder and the SUT has already taken place, this phase will define the intruder's actions after the assigned event.

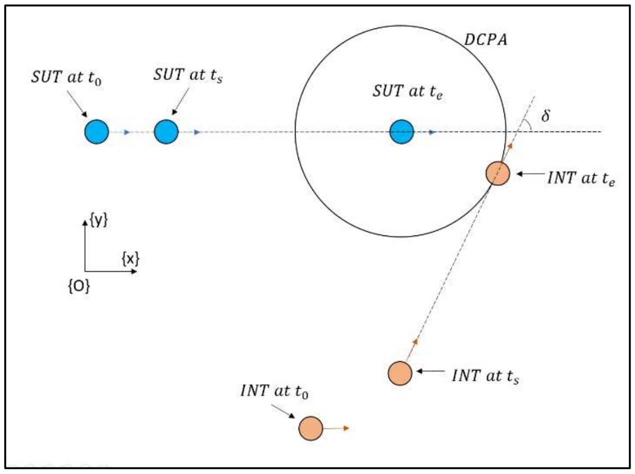


Figure 1: Graphical Representation of Event Timeline

Orchestration Phase

The purpose of the orchestration phase is determining the desired trajectory of the Intruder between t_s and t_e.The following synthesis approach uses the distance at closest point of approach (DCPA), Event Time, Setup Time, SUT state, intruder speed relative to the SUT, $v_{I/S}$, and heading difference, δ, to determine the trajectory of the intruder. First, determine the position and orientation of the intruder at the event time, t_e. Then, determine the position and orientation of the intruder at the setup time, t_s. The trajectory is assumed to be a constant velocity straight-line motion trajectory for $t_s \leq t \leq t_e$. Some options for positioning the intruder at time t_s include spawning the intruder at appropriate time with the desired state or dynamically tasking a preexisting ambient traffic entity.

The state variables used in this formulation are the position and velocity vectors $\vec{X}$ and $\vec{V}$, respectively. $\vec{X}$ is comprised of x and y cartesian coordinates as well as θ, the heading angle measured using right-hand coordinate system convention. $\vec{V}$ has elements of velocity along the x and y axes, $\dot{x}$ and $\dot{y}$ respectively, and the angular velocity about the z axis, $\dot{\theta}$. Capitalized subscripts used with these vectors indicate the referenced entity, S for SUT and I for intruder. The following equations also use varying reference frames. The reference frame is indicated in the subscript with $/S$ for the SUT frame and $/O$ for the origin or world frame. For example, $\vec{X}_{S/O}$ can is read as "the SUT position vector with respect to the Origin frame."

$$\vec{X}_{S/O} = \begin{bmatrix} x_{S/O} \\ y_{S/O} \\ \theta_{S/O} \end{bmatrix} \tag{1}$$

$$\vec{V}_{S/O} = \begin{bmatrix} \dot{x}_{S/O} \\ \dot{y}_{S/O} \\ \dot{\theta}_{S/O} \end{bmatrix} \tag{2}$$

First, predict the location of SUT at t_e in world frame coordinates assuming constant heading, $\dot{\theta}_{S/O} = 0$, and velocity:

$$\vec{X}_{S/O}(t_e) = \vec{X}_{S/O}(t_0) + \vec{V}_{S/O}(t_0)(t_e - t_0) \tag{3}$$

$$\vec{V}_{S/O}(t_e) = \vec{V}_{S/O}(t_0) \tag{4}$$

Next, determine desired intruder event velocity and heading with respect to the SUT frame.

$$\vec{V}_{I/S}(t_e) = (\vec{V}_{I/O}(t_e) - \vec{V}_{S/O}(t_e))^T \begin{bmatrix} \cos(\theta_{S/O}(t_e)) & -\sin(\theta_{S/O}(t_e)) \\ \sin(\theta_{S/O}(t_e)) & \cos(\theta_{S/O}(t_e)) \end{bmatrix} \tag{5}$$

Where,

$$\vec{V}_{I/O}(t_e) = \begin{bmatrix} v_{I/O}(t_e)\cos(\theta_{S/O}(t_0) + \delta) \\ v_{I/O}(t_e)\sin(\theta_{S/O}(t_0) + \delta) \\ 0 \end{bmatrix} \tag{6}$$

$$\gamma = \arctan 2(\dot{y}_{I/S}(t_e), \dot{x}_{I/S}(t_e)) \tag{7}$$

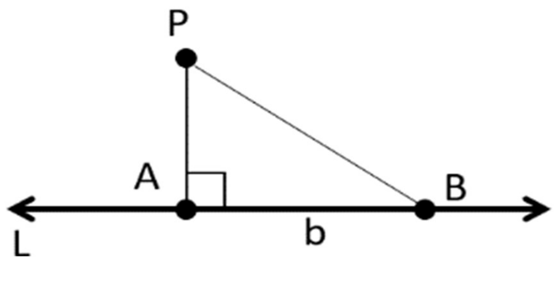
Then, calculate the position and orientation of the intruder at, t_e. The position is determined based on the calculated intruder trajectory and the theorem that the trajectory of the intruder at $t_e$is instantaneously tangent to a circle of radius DCPA which is centered on the SUT.

Theorem: The trajectory of the intruder at t_e is instantaneously tangent to a circle of radius DCPA which is centered on the SUT.

Definition: DCPA is the Distance at the closest point of approach between the SUT and intruder and occurs at the t_e.

Proof**:** This can be proven by using two previously proven theorems, Shortest Distance Theorem and Tangent Theorem.

The Shortest Distance Theorem proof shows that the shortest distance between a point P, and a line, L, is the perpendicular line form P to L (Sarig, 2021).



Proof By Contradiction: Let Line A be a point on line L such that PA is a perpendicular line to line L.

Figure 2: Shortest Distance Theorem

Proposition: There exists a point, B, on line L such that PA>PB. Using Pythagorean Theorem, $PB^2 = PA^2 + b^2$, where b is the distance between A and B. As b>0, PB will always be greater than PA, therefore the proposition that B exists is false. As such, the point A is the closest point on line L to point P.

The Tangent Theorem proof shows that if a line is perpendicular to a radius of a circle at a point on the circle, then the line is tangent to the circle (Erdos, 2013).

Proof By Contradiction: Let point A be a common point on line L and circle O such that PA is a perpendicular line to line L.

Proposition: There exists a point, B, that is common to line L and circle O other than A. Using Pythagorean Theorem, $PB^2 = PA^2 + b^2$, where b is the distance between A and B. As b>0, PB will always be greater than PA, therefore the proposition that B exists is false. As such, the point A is the only common point between line L and circle O. Since DCPA, by definition, is the closest point to the SUT, the arbitrarily small line segment of the intruder's trajectory containing the point at which the intruder achieves DCPA is perpendicular to the Line between the SUT and that point. As the trajectory segment is perpendicular at a point on a circle with radius DCPA, it is therefore tangential to that circle.

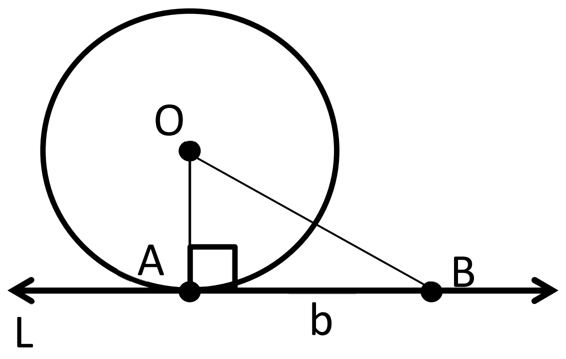


Figure 3: Tangent Theorem

Thus, the intruder position with respect to the SUT:

$$\vec{X}_{I/S}(t_e) = \begin{bmatrix} \cos(\gamma + \pi/2) \\ \sin(\gamma + \pi/2) \\ 0 \end{bmatrix} DCPA + \begin{bmatrix} 0 \\ 0 \\ -\delta \end{bmatrix} \quad (8)$$

Transform the intruder pose from the SUT frame to the world frame

$$\vec{X}_{I/O}(t_e) = \begin{bmatrix} \cos(\theta_{S/O}(t_e)) & \cos(\theta_{S/O}(t_e)) & 0 & x_{S/O}(t_e) \\ \cos(\theta_{S/O}(t_e)) & \cos(\theta_{S/O}(t_e)) & 0 & y_{S/O}(t_e) \\ 0 & 0 & 1 & \theta_{S/O}(t_e) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{I/S}(t_e) \\ y_{I/S}(t_e) \\ \theta_{I/S}(t_e) \\ 1 \end{bmatrix} \quad (9)$$

Finally, determine position of the intruder at t_s in world frame coordinates.

$$\vec{X}_{I/O}(t_s) = \vec{X}_{I/O}(t_e) - \vec{V}_{I/O}(t_e)(t_e - t_s) \quad (10)$$

$$\vec{X}_{I/O}(t_s) = \begin{bmatrix} \cos(\theta_{S/O}(t_s)) & -\sin(\theta_{S/O}(t_s)) & 0 & x_{S/O}(t_s) \\ \sin(\theta_{S/O}(t_s)) & \cos(\theta_{S/O}(t_s)) & 0 & y_{S/O}(t_s) \\ 0 & 0 & 1 & \theta_{S/O}(t_s) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{I/O}(t_s) \\ y_{I/O}(t_s) \\ \theta_{I/O}(t_s) \\ 1 \end{bmatrix} \tag{11}$$

Implementation phase

At time t_s, the intruder begins the trajectory calculated in the orchestration phase. Event complexity measures can be employed during this phase. For example, one method to increase the event complexity would be to define a percentage of $d_{I/S}$ that the intruder will actively pursue the desired event parameters even if the SUT were to maneuver in such a way as to render the SUT position prediction at time t_e invalid.

Intruder position with respect to the world frame for $t_s \leq t \leq t_e$.

$$\vec{X}_{I/S}(t) = \vec{X}_{I/S}(t_s) + \begin{bmatrix} \dfrac{x_{I/S}(t_e) - x_{I/S}(t_s)}{t_e - t_s} \\ \dfrac{x_{I/S}(t_e) - x_{I/S}(t_s)}{t_e - t_s} \\ 0 \\ \end{bmatrix} (t - t_s) \tag{12}$$

$$\vec{X}_{I/O}(t) = \begin{bmatrix} \cos(\theta_{S/O}(t)) & -\sin(\theta_{S/O}(t)) & 0 & x_{S/O}(t) \\ \sin(\theta_{S/O}(t)) & \cos(\theta_{S/O}(t)) & 0 & y_{S/O}(t) \\ 0 & 0 & 1 & \theta_{S/O}(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{I/S}(t) \\ y_{I/S}(t) \\ \theta_{I/S}(t) \\ 1 \end{bmatrix} \tag{13}$$

Post-event Phase

The post-event phase occurs following the event and represents the actions of the intruder. Potential options for the intruder actions include exiting a predefined area and de-spawning or returning to a previously assigned mission if the intruder was originally part of the ambient traffic. For this paper, the intruder is assumed to maintain heading and velocity after t_e.

VALIDATION

Validation of the method above was accomplished while striving to adhere to the Principles of Verification and Validation as outlined in (Tolk, 2012). While strict adherence would be a costly endeavor, the following identifies some of the ways in which this project observed these principles.

V&V Must be an integrated activity with model selection, development, and integration activities

The proofs concerning the relationship between the DCPA circle and the intruder trajectory are meant formally justify the state of the intruder at time, t_e. Informal methods were adopted to ensure appropriate validity of intermediate calculations. To validate the entire model, Dynamic testing was conducted using a simulation built in MATLAB. The results of this testing can be found in the Results section.

The intended purpose needs to be specified precisely

The purpose of the trajectory synthesis model is to deterministically synthesize maritime traffic interactions that can be presented to a system under test regardless of the state of the SUT. Further, the developed model must allow a targeted interaction design, closed-feedback testing, time compression.

Sufficient Knowledge, V&V tools, and V&V experts are available

The final two principles have been combined as efforts and justification for both are closely related. Background research, evidenced by the introduction, serves to provide fundamental domain knowledge. The author(s) have also been educated in the construction of simulations, Dynamics, and V&V methods.

RESULTS

The model was validated using a continuous simulation built in MATLAB 2021b. Two additional models were needed for implementation. The environment model is a frictionless, massless, plane located in a vacuum. The SUT is modeled as a point mass with motion described using rigid body kinematics. The intruder trajectory generation is modeled using the method described in the previous section. As evidenced by the method, the intruder is represented in the same manner as the SUT. The simulation evolved the appropriate states through time using Euler Integration with a time step of 0.1 seconds.

Figure 4 displays the result of a relatively trivial simulated event where the SUT has a non-zero velocity and a constant heading. Figure 5 displays the result of a relatively non-trivial simulated event where the SUT has a non-zero velocity and a change in heading during the event timeline. The input parameters used for each run are listed at in Table 1.

Table 1: Simulation Parameters for Presented Events

Figure	DCPA, m	T_s, s	T_e, s	δ	$v_{I/O}$, m/s
Fig. 4	-5.00^*	0.00	10.00	-90.00^*	2.00
Fig. 5	-5.00^*	0.00	10.00	-90.00^*	2.00
$*$ Signs of values in DCPA and δ dictate the direction of approach and fore or aft crossing.					

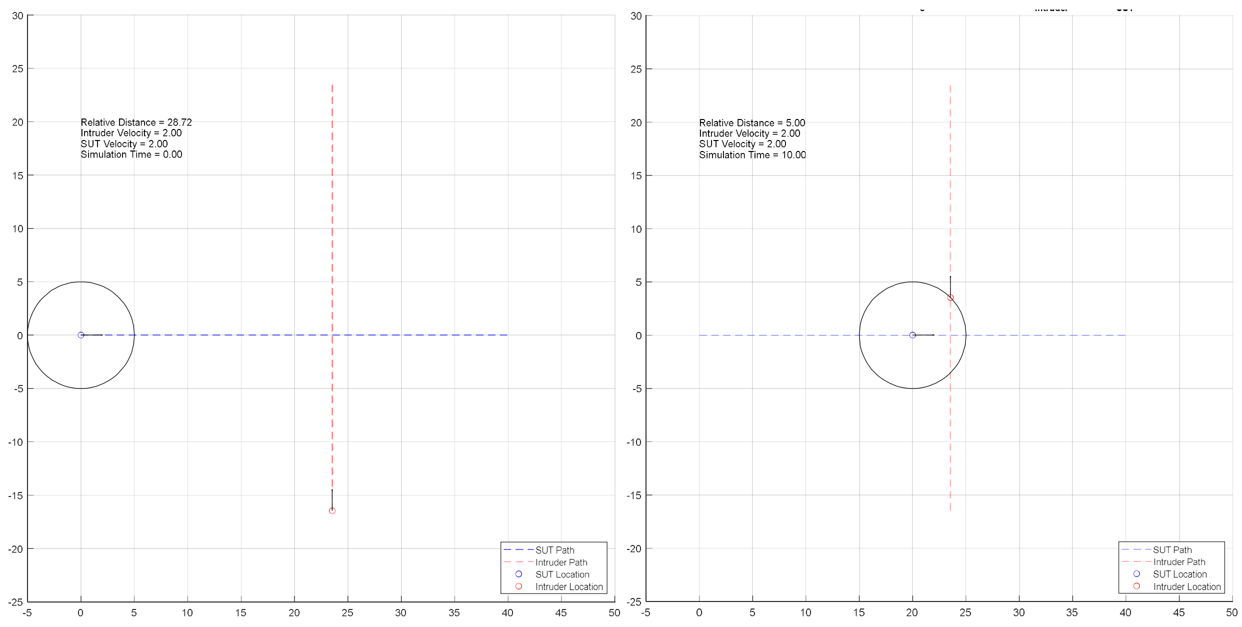


Figure 4: Constant Velocity and Heading Event at Start Time (left) and Event Time (right).

In both events, an ideal trajectory follower was able to follow the trajectory calculated and adhere to the simulation parameters. Per the results, event success was not affected by intermediate maneuvers of the SUT. Due to the limitations of static imagery, it may appear that the intruder's trajectory violates the DCPA circle but, in fact, does not.

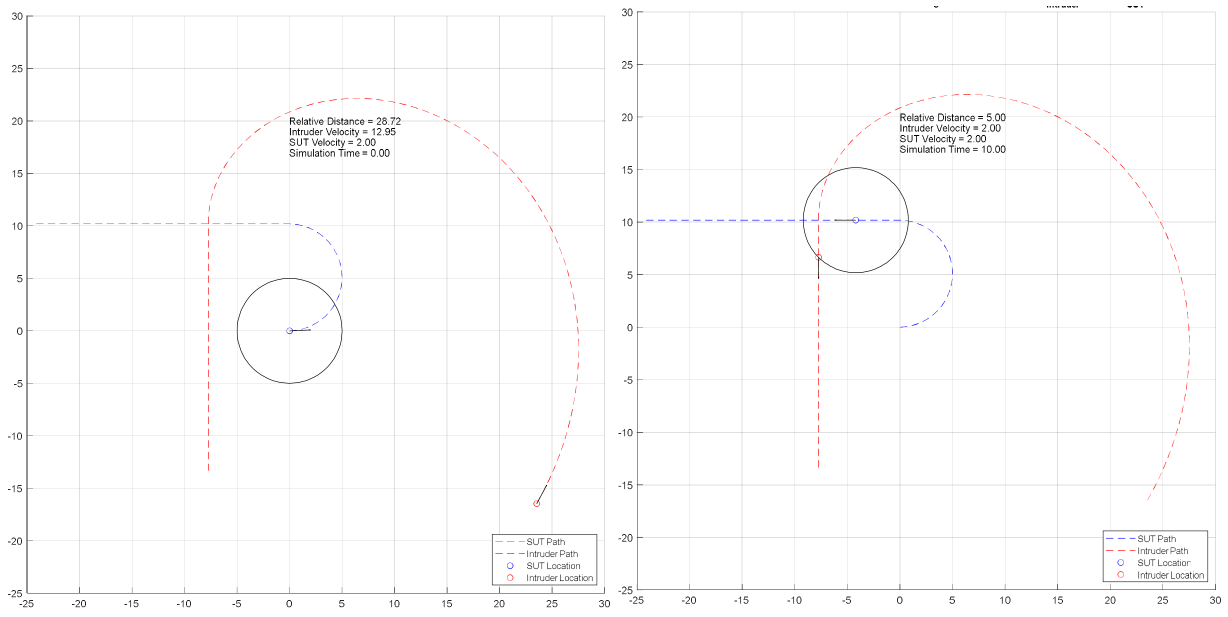


Figure 5: Constant Velocity and 180 Heading Change Event at Start Time (left) and Event Time (right).

CONCLUSION AND FUTURE WORK

This paper presented a method of event synthesis which has been demonstrated to allow designed encounters regardless of SUT state. The overall approach and results validate the use of this method to synthesize varying maritime traffic interactions. Results indicate that the approach can enable more robust evaluation of maritime autonomous algorithms. In the method's current form, the event design is observed if the constant SUT speed and straight-line motion is preserved at the event time. The trajectory generated is an ideal trajectory relative to the SUT position and orientation. After observing the results and reevaluating the data, it is possible to fully define the event with respect to the SUT. Currently, the intruder speed parameter is set with respect to the world frame. This was approach was decided early in the method development. If the speed were to be defined relative to the SUT the relative velocity of the intruder would be consistent through the entire event timeline and not just during the portions which conform to the straight-line motion and constant velocity assumption.

The next stage of development will implement the presented event synthesis method within the described simulation environment where a simulated vessel will follow the calculated trajectory using a kinematic motion model. Additionally, Future work will include the development of the complexity parameter along with an investigation of measurement noise on the calculated trajectory. The current concept of this complexity parameter is that it will define the upper limit of kinematic limitations with which an intruder will attempt to complete the calculated trajectory. Measurement noise and other non-deterministic elements used in the simulation environment will introduce errors in the intruder's conformance to synthesized trajectory, therefore their effects should be studied.

REFERENCES

Broy, M. (2006). Challenges in Automotive Software Engineering [PDF file]. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.472.5418&rep=rep1&type=pdf

Erdos, P. (2013). Tangent Theorem 1. Retrieved from http://proofsfromthebook.com/2013/06/19/tangent-theorem-1

Huang, W., Wang, K., Lv, Y., & Zhu, F. (2016). Autonomous Vehicles Testing Methods Review. *2016 IEEE 19th International Conference on Intelligent Transportation Systems*, 163–168. doi:10.1109/ITSC.2016.7795548.

Koopman, P. & Wagner, M. (2016). Challenges in Autonomous Vehicle Testing and Validation. *SAE International Journal of Transportation Safety*, 4(1), 15–24. doi:10.4271/2016-01-0128

Mullins, G. E., Stankiewicz, P. G., Hawthorne, R. C., & Appler, J. D. (2017). Delivering Test and Evaluation Tools for Autonomous Unmanned Vehicles to the Fleet [PDF file]. Retrieved from https://www.jhuapl.edu/Content/techdigest/pdf/V33-N04/33-04-Hawthorne.pdf

Nalic, D., Mihalj, T., Bäumler, M., Lehmann, M., & Bernsteiner, S. (2020). Scenario Based Testing of Automated Driving Systems: A Literature Survey. *FISTA Web Congress 2020,* FISTA Digital Library. Retrieved from https://www.fisita.com/library/fisita-world-congress/2020/f2020-acm-096

Porres, I., Azimi, S., & Lilius, J. (2020). Scenario-based Testing of a Ship Collision Avoidance System. *46th Euromicro Conference on Software Engineering and Advanced Applications,* 545–552. doi:10.1109/SEAA51224.2020.00090.

Refai, M., Abramson, M., Lee, S., & Wu, G. (2019). Encounter-Based Simulation Architecture for Detect-And-Avoid Modeling [PDF file]. Retrieved from https://ntrs.nasa.gov/api/citations/20190000084/downloads/20190000084.pdf

Sarig, I. (2021). Shortest Distance Theorem. Retrieved from https://geometryhelp.net/shortest-distance-theorem

Stankiewicz, P. G., & Mullins, G. E. (2019). Improving Evaluation Methodology for Autonomous Surface Vessel COLREGS Compliance. *OCEANS 2019 - Marseille*, 1–7. https://doi.org/10.1109/OCEANSE.2019.8867549

Tolk, A., (2012). *Engineering Principles of Combat Modeling and Distributed Simulation*. Hoboken, New Jersey: Wiley

Zhao, M., Yao, X., Sun, J., Zhang, S., & Bai, J. (2018). GIS-Based Simulation Methodology for Evaluating Ship Encounters Probability to Improve Maritime Traffic Safety. *IEEE Transactions on Intelligent Transportation Systems*, 20(1), 323-337. doi:10.1109/TITS.2018.2812601