*Proceedings of the 2024 Winter Simulation Conference*
*H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.*

# DEVS-BASED SIMULATION ACCELERATION FOR AI TRAINING: UNMANNED SURFACE VEHICLE CASE

Juho Choi[1], Jang Won Bae[2], Il-Chul Moon[1,3]

[1]Dept. of Aerospace Eng., KAIST, Daejeon, REPUBLIC OF KOREA
[2]School of Industrial Management, Korea University of Technology and Education, Cheonan, REPUBLIC OF KOREA
[3]Dept. of Industrial and Systems Eng., KAIST, Daejeon, REPUBLIC OF KOREA

## ABSTRACT

As demand for USV usage increases, the development of simulators for AI training is becoming crucial. This paper introduces a DEVS-based simulation acceleration technique achieved through dynamic changes in the model structure, which is enabled by DSDEVS, while considering domain-specific characteristics. In the case study, the proposed method was applied to and evaluated using USV models. Specifically, the proposed method adapts the coupling structure of the USV maneuver model based on changes in bank angle during simulations; the coupling structure of the USV sensor model is modified according to the distance from enemy units. These changes reduce unnecessary event exchanges during simulation execution, thus increasing the speed of simulation execution. Furthermore, they can lead to the dynamic control of time advances in USV models, enables the improvement of simulation speed. The case study shows the proposed method effectively accelerates simulation execution, but it involves a trade-off with simulation accuracy.

## 1 INTRODUCTION

In recent years, the strategic deployment of Unmanned Surface Vehicles (USVs) in naval operations has increased, driven by advances in artificial intelligence (AI) (Zhao et al. 2021). As such unmanned systems take on increasingly complex tasks, the demand for advanced simulation tools to train and validate AI algorithms under diverse and challenging conditions is increasing (Adelani 2014). In this context, simulation plays a crucial role of generating varied and extensive data for learning algorithms, a process that simultaneously often incurs considerable execution time.

This demand underscores the necessity for an innovative approach of accelerating simulation execution time while adhering to a general modeling method. We considered that Discrete Event System Specification (DEVS) formalism (Zeigler et al. 2000), known for its robustness in modeling and simulating complex systems, offers a promising framework for addressing these challenges. To this end, This paper presents a simulation accelerating method that incorporates Dynamic Structured DEVS (DSDEVS) (Barros 1995) to reduce unnecessary message exchanges during simulation execution through dynamic structural changes of simulation models. It is also emphasized that these dynamic structural changes must adequately consider domain-specific knowledge.

The proposed method is realized and evaluated with a DEVS-based naval USV simulator. Specifically, within the context of USV operations, a sensing model in the USV keeps searching the enemy periodically, while the maneuver model continuously monitors for tactical movement. These operations are crucial in military tactics; however, they sometimes result in unnecessary event exchanges during simulation execution. These domain-specific considerations were represented as acceleration parameters, and these parameters mediated the dynamic structural changes in the USV simulation models by the virtue of DSDEVS semantics and eventually accelerated simulation execution.

In the case study, the experimental design incorporated the acceleration parameters to evaluate the efficiency of the proposed method. The present results demonstrate that the proposed method significantly speeds up the simulation execution (by up to 4.55 times). Furthermore, the meta-modeling analysis was conducted to identify more significant and robust parameters for the acceleration, and profiling of several cases was performed to examine the details of simulation performances. Through these investigations, we discovered that by setting the parameters values, the simulation results can exceed the boundaries, which means that there is a trade-off relationship between the acceleration and the accuracy of simulations. We note that it is crucial for users who employ a simulation model as a data generator for AI learning, and the proposed method is expected to be used to explore the balance between simulation speed and fidelity and to provide insights into the acceptable limits of information loss in accelerated simulation scenarios.

## 2 BACKGROUND

### 2.1 DEVS Formalism

DEVS (Discrete Event System Specification) is a formalism to develop discrete event models in a hierarchical and modular manner (Concepcion and Zeigler 1988) (Chow and Zeigler 1994). DEVS consists of two abstract models: atomic model and coupled model. The atomic and the coupled models describe the behaviors and the structure of a target system. DEVS models can be built with the combination of these model types in a hierarchical and modular manner. The formal specifications of the atomic (AM) and the coupled (CM) are as follows:

$$AM =< X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$$

$X$ = a set of input events,
$Y$ = a set of output events,
$S$ = a set of states,
$\delta_{ext} : Q \times X \rightarrow S, where\ Q = \{(s,e) | s \leq S, 0 \leq e \leq ta(s)\}$, an external transition function,
$\delta_{int} : S \rightarrow S$, an internal transition function,
$\lambda : S \rightarrow Y$, a output function,
$ta : S \rightarrow R_0^+ \cup \{\infty\}$, a time advance function

$$CM =< X, Y, M, EIC, EOC, IC, Select >$$

$X$ = a set of external input events,
$Y$ = a set of output events,
$M$ = a set of component models,
$EIC \subseteq X \times \cup_{m \in M} m.X$, where the $m.X$ is an input event of $m \in M$, external input coupling relations,
$EOC \subseteq \cup_{m \in M} m.Y \times Y$, where $m.Y$ is an output event of $m \in M$, external output coupling relations,
$IC \subseteq \cup_{m \in M} m.Y \times \cup_{n \in M} n.X$, where $m.Y$ and $n.X$ are an input and an output event of different component models ($m \neq n$), internal coupling relations,
$SELECT : 2^M - \emptyset \rightarrow M$, a tie-breaking function among components $M$,

### 2.2 Dynamic Structure DEVS Formalism

As an extension of DEVS, Dynamic Structure DEVS (DSDEVS) was proposed for allowing structural changes during simulation execution (Barros 1995). In other words, The DSDEVS formalism enables to describe dynamic structural changes, such as the addition and removal of coupling relations and components of DEVS coupled model, which is formally specified by introducing a new abstract model called DSDEVS network model(DSDEVN).

$$DSDEVN =< \chi, M_\chi >$$

$\chi$ = DSDEVS network executive,
$M_\chi = < X_\chi, Y_\chi, S_\chi, \delta_{ext_\chi}, \delta_{int_\chi}, \lambda_\chi, ta_\chi >$, model of $\chi$,
where $S_\chi = < X_\Delta, Y_\Delta, M_\Delta, EIC_\Delta, EOC_\Delta, IC_\Delta, Select_\Delta >$, the state of DSDEVS dynamic structure network $\Delta$

It should be noted that the specifications of $M_\chi$ are quite similar as those of the atomic and the coupled model in DEVS formalism. However, compared to them, $M_\chi$ holds the model structures ($\Delta$) in its state($S_\chi$), so varying one element of the state leads to the realization of the dynamic structural changes. The proposed method is theoretically based on this dynamic structural changes for accelerating simulation execution (Shang and Wainer 2006).

### 2.3 Literature Review on Accelerated Simulator for AI Training

Recently, there has been growing research interest in simulators for AI training. A research explored that web based GPU acceleration in embodied agent training workflow (Sisyukov et al. 2021). They used the simulation to supports RL training of agents. Training agents in real time environment is slow, hard to control, and expensive. Parallelized training with simulator bring the faster, safer, easier process.

In the region of computational resource scaling within simulation acceleration, recent research has primarily focused on reducing computation load at the simulator execution level, not a modeling structural level. An approach exists that Machine learning can be used to approximate the functions of DEVS models to reduce computational load (Saadawi et al. 2016). They approximated behaviors of DEVS coupled models and atomic models with a predictive model. The approximate equivalent predictive model developed by machine learning techniques predict the output of the DEVS model, and it can reduce computational load because a predictive model can replace a computation-intensive DEVS model. And the predictive model can store the metadata describing the DEVS model, it perform the significant performance in the repetitive computational situations. This acceleration method utilize the characteristic of DEVS,available to separates modeling and simulation execution, to deal with the simulation execution parts. So that, this method does not touch the modeling parts.

From another perspective, research on HW-based simulation acceleration is also active. Parallelization and Hardware optimization can accelerate massive models such as Approximate Bayesian Computation(ABC) (Kulkarni et al. 2020). They accelerate the simulation with parallel use of GPUs. Another research, stochastic simulation was accelerated by using massively parallel GPU (Köster et al. 2023). Another study on hardware-based simulation acceleration involves accelerating Monte Carlo integration of stochastic differential equations using GPUs in a CUDA programming environment (Spiechowicz et al. 2015). Additionally, research has focused on speeding up molecular modeling simulations for massively parallelized GPUs (Stone et al. 2010). And another research suggest that simulation of complex system using a hybrid system based on DEVS can be accelerated by multi core and GPU coupled architecture (Kim et al. 2018). These kind of methods actually does not reduce computational load, but Hardware structure based acceleration is worked. Based on the Parallel DEVS methodology,a technique named Multicore Acceleration of DEVS Systems suggested (Liu and Wainer 2012). The technique accelerate both memory-bound and compute-bound kernels using parallel DEVS simulation (Chow et al. 1994). There is another research of HW-based simulation acceleration method (Trabes et al. 2023). In this research, parallel DEVS simulations executed all task in parallel, and compute different cores on shared memory architectures.

In summary, although accelerating simulator on modeling structural level has not been widely used previously, this paper proposes a method for accelerating DEVS simulation on the modeling structural level.

## 3 DEVS-BASED SIMULATION ACCELERATION METHOD

The proposed method is based on two approaches. Firstly, it is to identify messages that are repeatedly transmitted according to the model specifications but are sometimes irrelevant in simulation progress. The acceleration can then be achieved by decoupling those unnecessary connections. Secondly, when the change in state transitions remains below a certain threshold, the acceleration can also be realized by reducing the number of calculating state transitions. This section elaborates on the two ideas using a naval USV model.

### 3.1 Coupling Structure of USV Model

This section presents USV models using DEVS formalism. Figure 1 depicts the coupling between models in a DEVS diagram, illustrating the overall model interactions. In this diagram, 'Blue' represents friendly Unmanned Surface Vehicles (USVs), while 'Red' represents enemy USVs. Through the DEVS coupling mechanism, these two models exchange information regarding their locations and the damage from gunfire. Utilizing an object-oriented simulator, each model is crafted as an independent and cohesive structure, allowing for a vast array of DEVS couplings to be efficiently implemented.

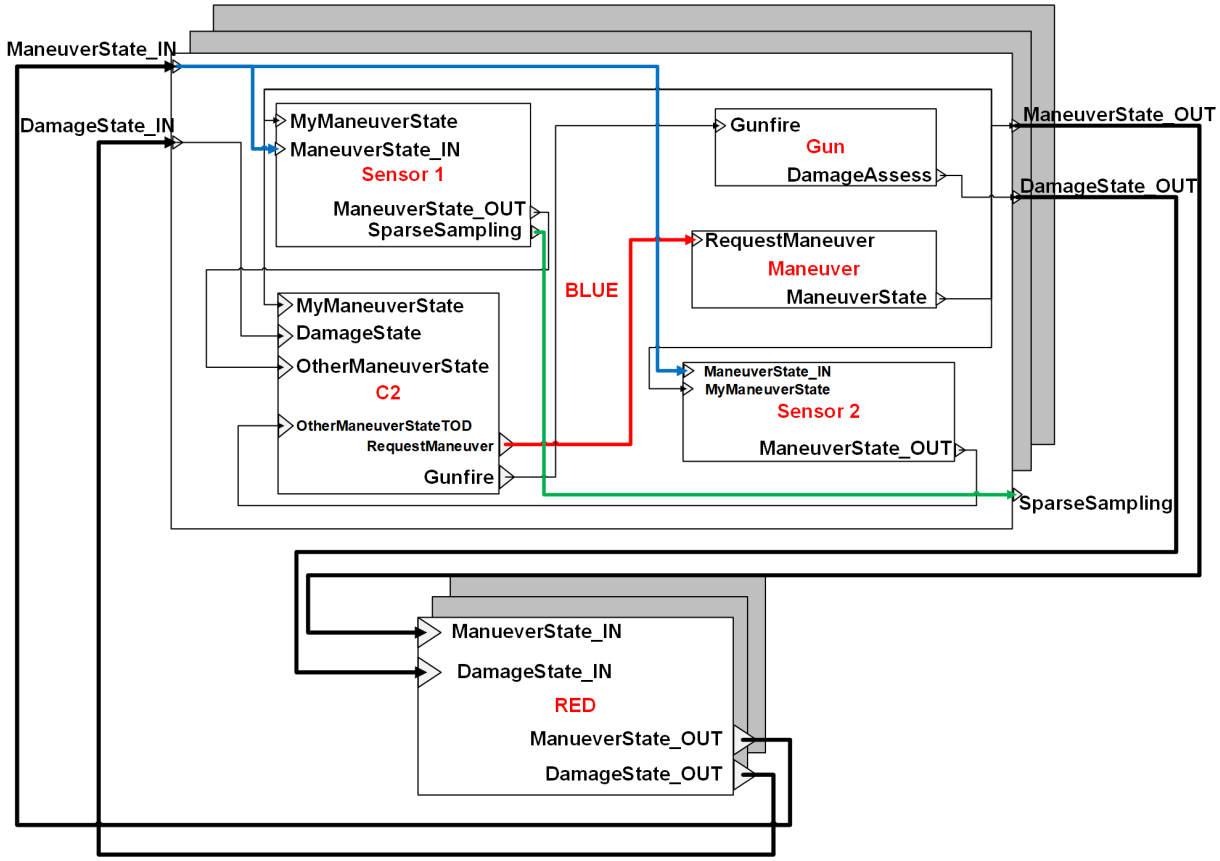


Figure 1: DEVS structure diagram of Unmanned Surface Vehicle(USV) coupled model.

Within this framework, the blue USV model is constructed as a DEVS coupled model comprising several components: two sensor models, a gun model for executing attack, a maneuver model to manage dynamics, such as position and velocity, and a C2 model responsible for control and strategic decision-making. The blue USV employs its sensor model to pinpoint the location of enemy red USVs and utilizes the maneuver model to update its own position and speed. Each model generates propagated events to inform C2 model

to decide movement and attacks by a tailored rule-based algorithm. The C2 decision becomes the output events delivered to the maneuver and gun models through DEVS coupling.

Figure 1 depicts a diagram of the DEVS Coupled model of the Blue USV (Praehofer and Pree 1993). The 'ManeuverState_IN' External Input Coupling (EIC) receives the location of the Red USV, which is then passed on to sensor models, activating the enemy detection functionality. The detected enemy location information is forwarded to the C2 model. Utilizing its own location received from the Maneuver model and the enemy's location from the sensor model, the C2 model determines the position of the enemy to pursue and passes this information to the Maneuver model to initiate the chase. Additionally, a firing command is sent to the GUN model's 'Gunfire' port, and the Gun model conveys damage assessment information to the Red that has been hit, through the 'Damagestate_OUT' External Output Coupling (EOC) of the Blue Coupled model.

### 3.2 Coupling Structure Changes for Acceleration

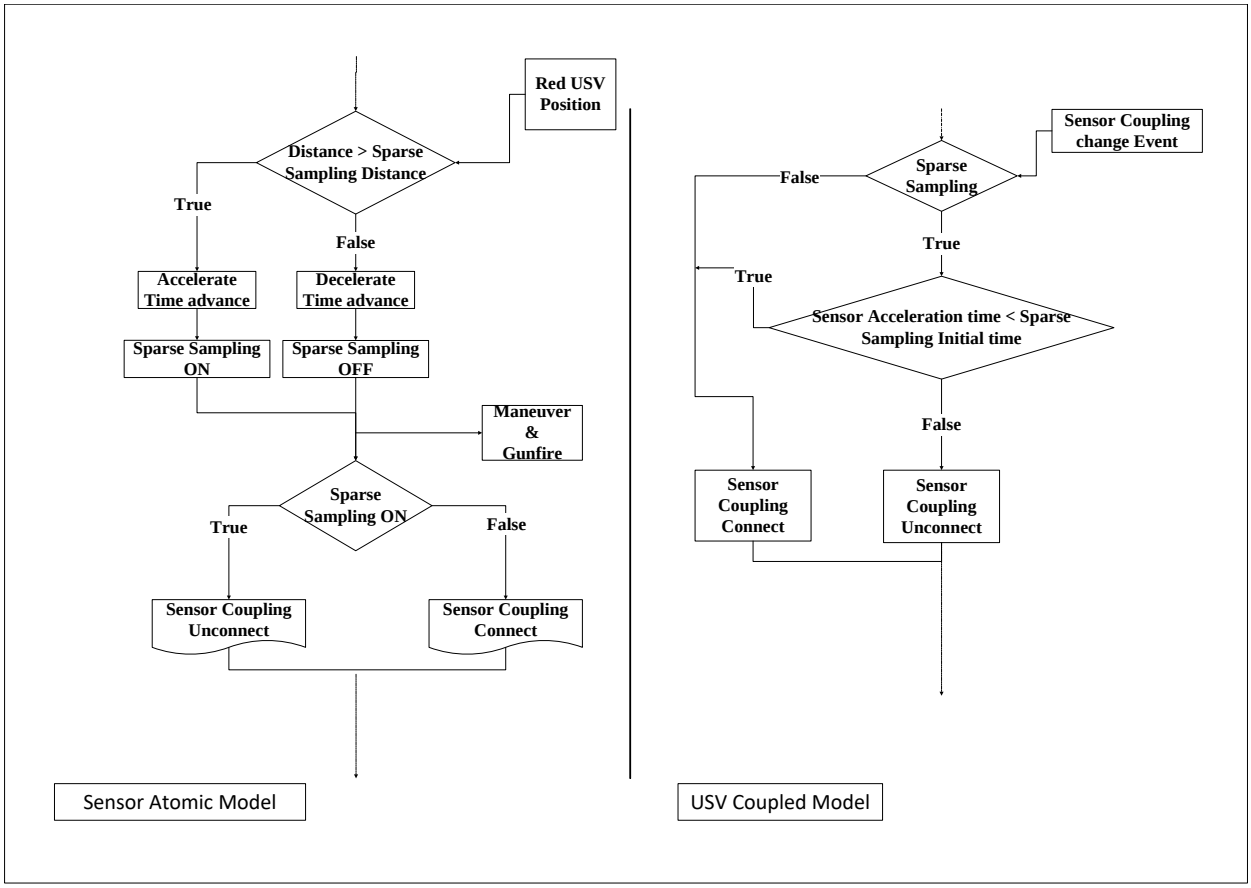


Figure 2: Flow chart of Sparse sensing.

Coupling between the 'SparseSampling' port and the sensor model is illustrated as a green colored coupling in Figure 1. This green coupling illustrates the coupling that are being dynamically created and removed by DSDEVS. When the conditions for Dynamic Structure are met within the threshold of sensor models, the sparse sampling is deactivated, and the Blue USV model induces changes in the structure of DEVS Coupling. On the other hand, when the condition claims that there are enough distance from the sensor model, the sparse sampling becomes activated, and the Blue USV model removes the blue colored coupling. We illustrate this coupling manipulation in Figure 2. The left side of the flow chart comes from

the sensor model to utilize the sparse sampling mechanism. The right side of the flow chart originated from the Blue USV coupled model.

**Sensor model** determines whether the Red position is close enough to be frequently sampled or not. We utilize a threshold to determine the deceleration or acceleration of this time advance. Once its determined, the sensor model time advance is changed to gain the computational efficiency, and the sensor model outputs the sensor coupling management request to the coupled model, Blue USV in this context. DSDEVS enables the coupled model to receive the input and to change its state and coupling structure.

**Blue USV model** receives the sparse sampling activation. However, it is necessary to restore the coupling mechanism to process the sparsely sampled detection event. Therefore, there are two conditional statements. The first condition exists to handle the sensor model request, and the second condition exists to reactive the sensor model by reconnecting the coupling structure dictated by the sparse sampling frequency.

Coupling between the C2 model and the **maneuver model** is illustrated as a red colored coupling in Fig 1. When the maneuver model received a message from the C2 model through the red coupling, it generates the target command to reach at the target maneuver received. Depending on the condition for target bank angle derived from target command, it is acceptable that the maneuver model can update maneuver sparsely. We illustrate this flows in Figure 3. Time advance will be accelerated when the target bank angle is less than a threshold, and decelerated when it is larger than that threshold.

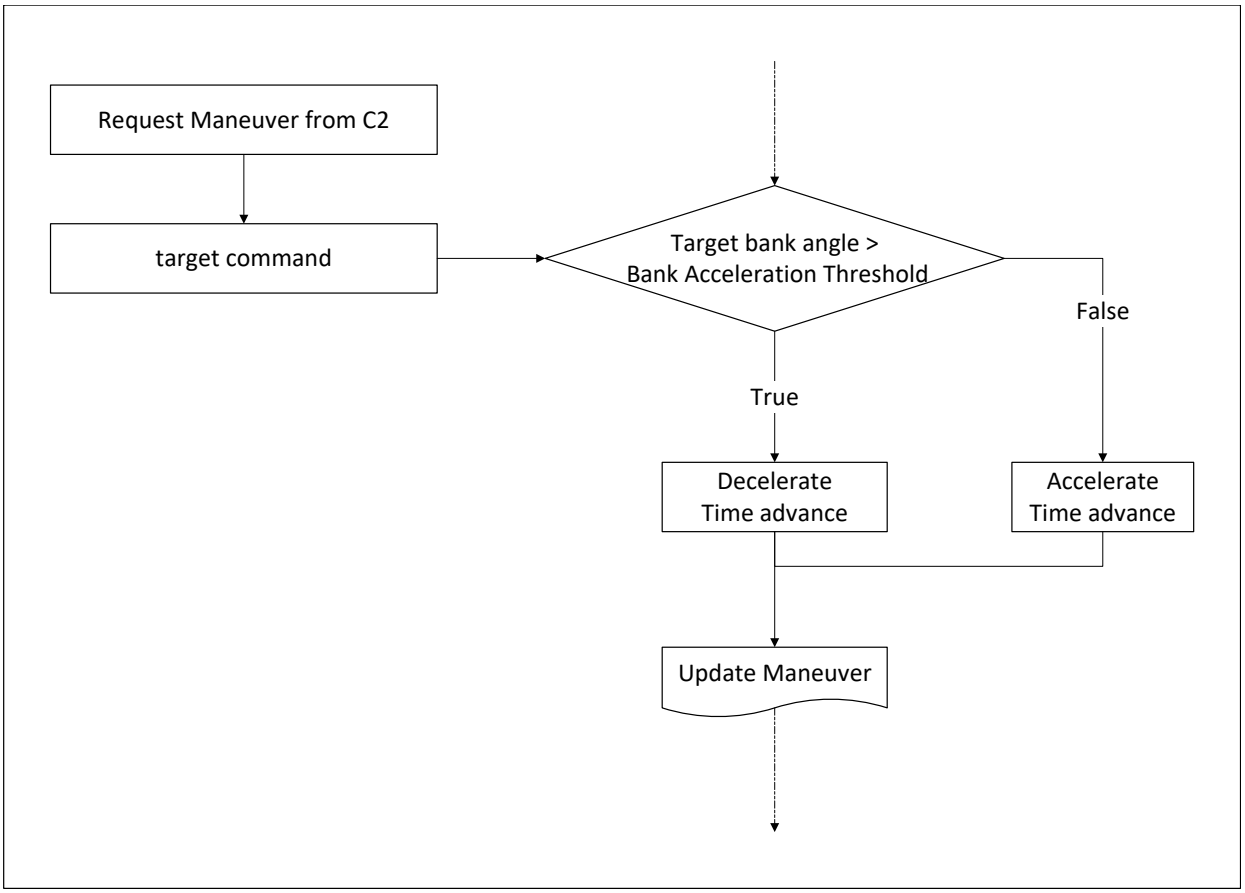


Figure 3: Flow chart of Maneuver update acceleration.

Given the above domain-specific knowledge, we can identify five parameters to accelerate the simulation model (see Table 1): Bank acceleration threshold (BAT) and acceleration time advance (ATA) are parameters

related to the maneuver model, Sparse sampling initial time (SSIT), sparse sampling distance (SSD) and scan acceleration time advance (SATA) related to the sensor model.

Table 1: Acceleration parameters.

| Acceleration parameters | Implement |
|---|---|
| Bank Acceleration Threshold (BAT) | Threshold of Bank Turn Command |
| Acceleration Time Advance (ATA) (sec) | Accelerated time advance case of maneuver update acceleration |
| Sparse Sampling Initial Time (SSIT) (sec) | Time of first detecting the enemy USV |
| Sparse Sampling Distance (SSD) (m) | Distance threshold of operating Sparse sensing |
| Scan Acceleration Time Advance (SATA) (sec) | Accelerated time advance activated when sparse sensing is activated |

## 4 CASE STUDY

### 4.1 Experimental Design

To evaluate the proposed method, a case study was conducted by establishing an experimental design with the identified acceleration parameters (see Table 2).

Table 2: Experimental design of USV Experiments with Acceleration Parameters.

| Acceleration parameters | Default | Variation Cases |
|---|---|---|
| BAT | 0.01 | 0.01, 0.05, 0.1 (3 cases) |
| ATA | 1 | 1, 5, 10 (3 cases) |
| SSIT | 100 | 100, 200, 300 (3 cases) |
| SSD | 5000 | 5000, 3000, 1000 (3 cases) |
| SATA | 1 | 1, 50, 100 (3 cases) |
| Total Cases | | $3\times3\times3\times3\times3=243$ |
| Total Experiment cases | 29 Experiments | $29\times243$ = 7047 |

This simulation experiment introduces the following performance measures (see Table 3) to quantify the extent of acceleration of the simulation and the deviation of the results from the default case.

Table 3: Performance measure of USV operation Experiments.

| Performance measures | | Implications |
|---|---|---|
| Acceleration performance | Runtime_Wall (sec) | Simulation execution time measured based on wall clock |
| Simulation accuracy | $D_{diff}(c)$ | Dead rate difference between $c$ and the default scenario. |
| | | It refers to Equation (1) and (2) |
| | MTDD | Mean Trajectory Discrepancy Distance. |
| | | It refers to Equation (4) |

Runtime_Wall is chosen to see how much the simulator faster than default case according to change acceleration parameters. It represents acceleration performance. However, the simulation acceleration may potentially impact simulation accuracy. The performance measures are analyzed based on the variations in acceleration parameters to evaluate the impact of each acceleration parameter on simulation acceleration and the effects on simulation outcomes. The ultimate goal is to maximize the acceleration of the simulation's

runtime while maintaining an acceptable level of result variation. On the other side, we also propose parameters to measure how much simulation acceleration affects the accuracy of the simulator. we introduce the 'Dead Rate Difference' and 'MTDD' (Mean Trajectory Discrepancy Distance)' as the simulation accuracy measure, indicators to compare how much the simulation results have changed compared to the Default simulator. The dead rate is a percentage that represents how much the Blue USV has eliminated Red USV before it completes its mission, i.e., reaches the goal. The dead rate $D(c)$ and the dead rate difference $D_{\text{Diff}}(c)$ of experiment case $c$ from default data is represented as follows:

$$D(c) = \frac{\text{Number of Red USVs eliminated by Blue USV in case } c}{\text{Total Number of Red USVs in case } c} \times 100(\%) \tag{1}$$

$$D_{\text{Diff}}(c) = D(c) - D(\text{Default}) \tag{2}$$

The Mean Trajectory Discrepancy Distance of case $c$ ($MTDD_c$) is defined as follows:

$$Pos_c(i,t) = (z_n(i,t), \quad z_e(i,t)) \tag{3}$$

$$z_n(i,t) = \text{Northern position of Blue i at time t}$$

$$z_e(i,t) = \text{Eastern position of Blue i at time t}$$

$$Pos_c(i,t) = \text{position of Blue } i \text{ at time } t \text{ under the experiment case } c. \tag{4}$$

$$MTDD_c = \sum_{t=0}^{Endtime} \sum_{i \in \text{Blue ID}} ||Pos_{\text{Default}}(i,st) - Pos_c(i,st)|| \tag{5}$$

The scale factor $s$ is the interval of time calculating positional differences. We use $s = 100$ at the experiment.

### 4.2 Results Analysis

Based on the experimental design and the performance measure, we conducted results analysis with the generated simulation results. In particular, we intend to investigate the relationship 1) between acceleration parameters and performance measures and 2) between the acceleration and the accuracy of simulations.

#### 4.2.1 Relationship Analysis between Acceleration Parameters and Performance measures

We investigated the impact of changes in acceleration parameters on performance measure. Figure 4 displays graphs of the relationship between each Acceleration parameter and Performance measure, represented through linear regression. As seen in the figures, ATA and SATA appear to have a significant impact on reducing runtime. However, an increase in SATA could negatively affect MTDD (Mean Trajectory Discrepancy Distance) and the Dead Rate Difference, while ATA has a positive impact on MTDD but a negative effect on the Dead Rate Difference. BAT and SSD do not significantly influence runtime but do affect simulation accuracy measure, and SSIT does not have a meaningful impact on any parameters.

Table 4 presents a linear regression analysis to identify the detailed correlations between acceleration parameters and performance parameters. Since the scales of each acceleration parameter is not the same scale, the analysis included standardization and was conducted with a significance level of 5%. The analysis results indicate that ATA and SATA have the most substantial impact on runtime, with SATA being the dominant factor. However, an increase in SATA also leads to an increase in $D_{diff}$ and MTDD. In the case of SSD, it influences the decrease in $D_{diff}$ and MTDD. It indicates that a larger value of SSD is beneficial for increasing simulation accuracy.

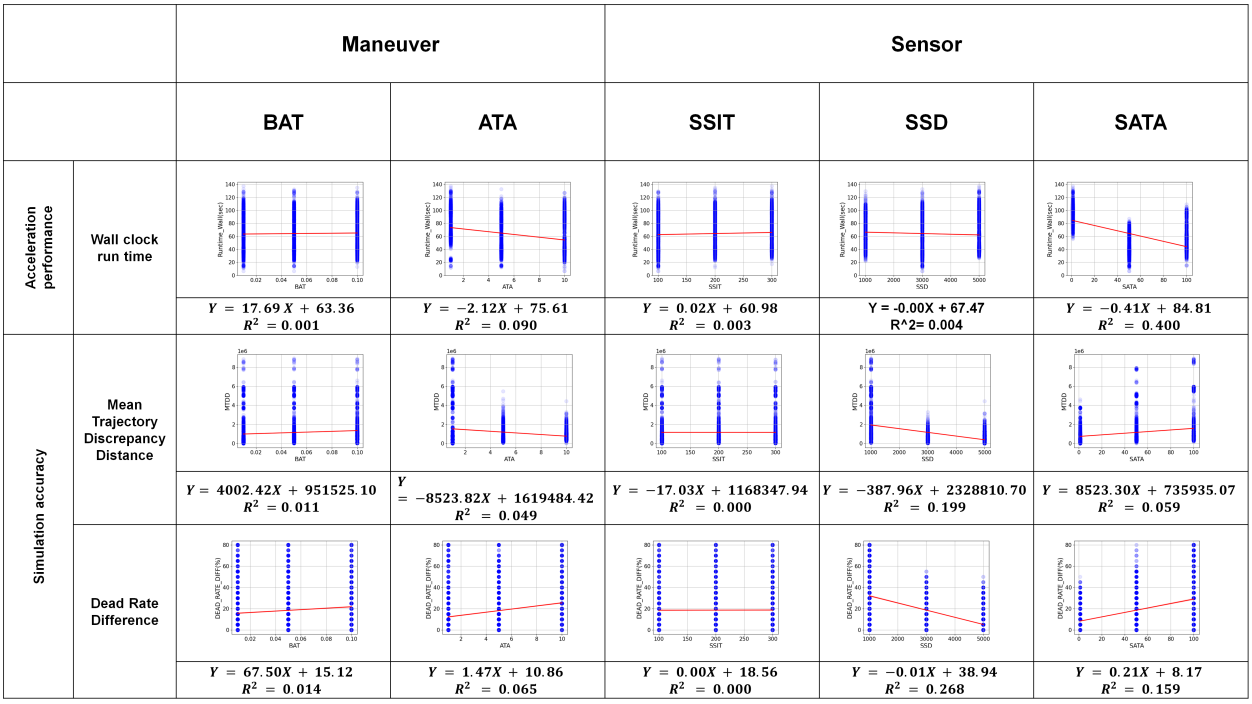


Figure 4: Linear regression analysis of the effect acceleration parameter on performance parameter.

Table 4: Regression analysis for the impact of acceleration parameters on performance parameters.
Note : The table reports the estimation results of regression analysis. The numbers under coefficient denote the coefficient values of the estimation on each values of performance parameters. Estimates with * denote statistical significance at the 5% level.

| | **Runtime** | | | **$D_{diff}(\%)$** | | | **MTDD** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Acceleration Parameter** | std.Coef | t | P-value | std.Coef | t | P-value | std.Coef | t | P-value |
| BAT | 0.025* | 2.962 | 0.003 | 0.117* | 13.967 | 0.000 | 0.104* | 10.553 | 0.000 |
| ATA | -0.300* | -35.509 | 0.000 | 0.255* | 30.487 | 0.000 | -0.221* | -22.477 | 0.000 |
| SSIT | 0.052* | 6.159 | 0.000 | 0.003 | 0.361 | 0.718 | -0.001 | -0.1 | 0.921 |
| SSD | -0.066* | -7.849 | 0.000 | -0.518* | -61.850 | 0.000 | -0.447* | -45.380 | 0.000 |
| SATA | -0.633* | -74.917 | 0.000 | 0.399* | 47.590 | 0.000 | 0.243* | 24.676 | 0.000 |
| R-Squared | | 0.498 | | | 0.506 | | | 0.318 | |
| Adj.R-Squared | | 0.497 | | | 0.506 | | | 0.318 | |

### 4.2.2 Detailed Analysis of Experimental Cases

We choose five scenarios for a more detailed analysis on acceleration performance measure. Table 5 shows the acceleration parameter setting of each scenario.

Left side of Figure 5 shows all 243 cases in the experiment design are plotted with the five acceleration scenarios highlighted in different colors. Right side of Figure 5 depicts the scatter plot between runtime and simulation accuracy measure, i.e., MTDD, according to the acceleration scenarios. We demonstrate that the accelerated cases 3 and 4 efficiently achieve both high acceleration performance and acceptable simulation accuracy with respect to MTDD. Accelerated case 5 appears to have best acceleration performance in the left side of Figure 5, Figure 6 shows the distributions of each Performance measures for five cases as defined in Table 5, from 29 experiments. Left side of Figure 6 shows how the runtime for accelerated cases be reduced. According to this, accelerated case 5 is the best acceleration scenario among five of them. However, middle of Figure 6 shows its MTDD values are significantly higher and more varied compared

Table 5: Acceleration scenario setting.

| | BAT | ATA | SSIT | SSD | SATA |
|---|---|---|---|---|---|
| Default | 0.01 | 1 | 100 | 5000 | 1 |
| Accelerated case 1 | 0.01 | 5 | 100 | 5000 | 1 |
| Accelerated case 2 | 0.01 | 1 | 100 | 5000 | 50 |
| Accelerated case 3 | 0.01 | 5 | 100 | 5000 | 50 |
| Accelerated case 4 | 0.01 | 10 | 100 | 5000 | 100 |
| Accelerated case 5 | 0.10 | 10 | 100 | 3000 | 50 |

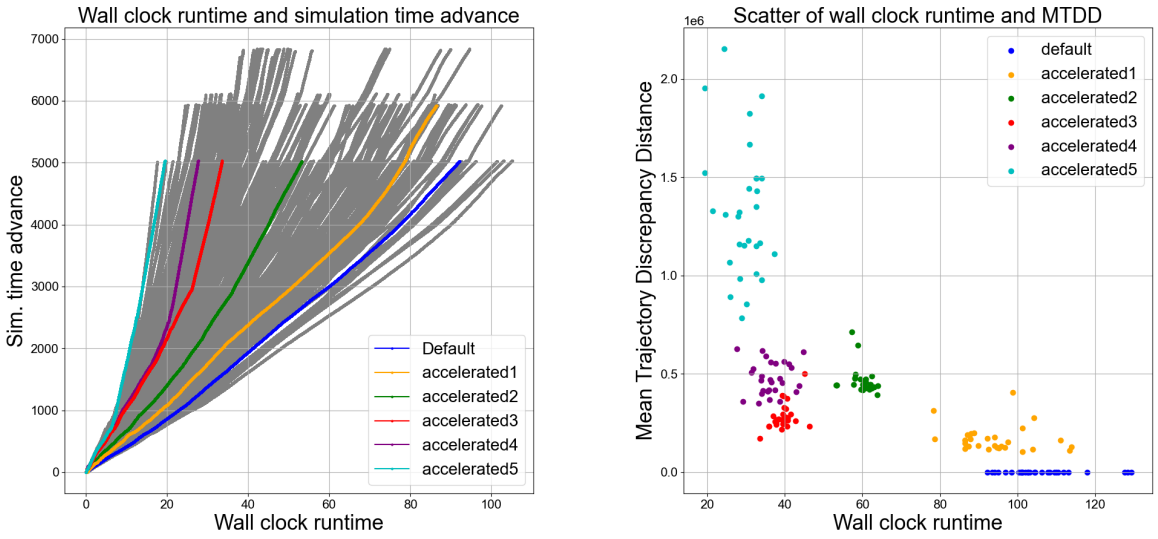


Figure 5: Left - wall clock runtime and simulation time advance / Right - scatter plot of acceleration performance measure (Runtime_Wall) and Simulation Accuracy measure (MTDD).

to other cases. Additionally, right side of Figure 6 also shows accelerated case 5 has a wide range of variance. Thus, we can say the accelerated scenario 5 has a significant decrease in simulation accuracy.

## 5 DISCUSSION

In this case study, we discovered the relationship between acceleration parameters and performance measures with linear regression analysis. To explore further details, we selected specific scenario cases for performance measure analysis. As the results, the increase in ATA and SATA have a significant relationship with runtime. We can see that the acceleration performance reaches up to 4.55 times compared to the default case. This method of simulation acceleration by reducing the computational load itself can play a significant role in generating data for AI training. Because, if the computing resources available for generating AI training data are limited, this software-oriented acceleration method would be very effective. It can also apply parallel with previously studied hardware-based acceleration methods, so it can be a large step on generation of AI training data.

The parameters BAT, SSIT, and SSD do not significantly affect simulation runtime but have the potential to influence the acceleration performance of ATA and SATA through scenario cases (please refer to the accelerated case 5). Future research could involve increasing the number of experimental cases, and using

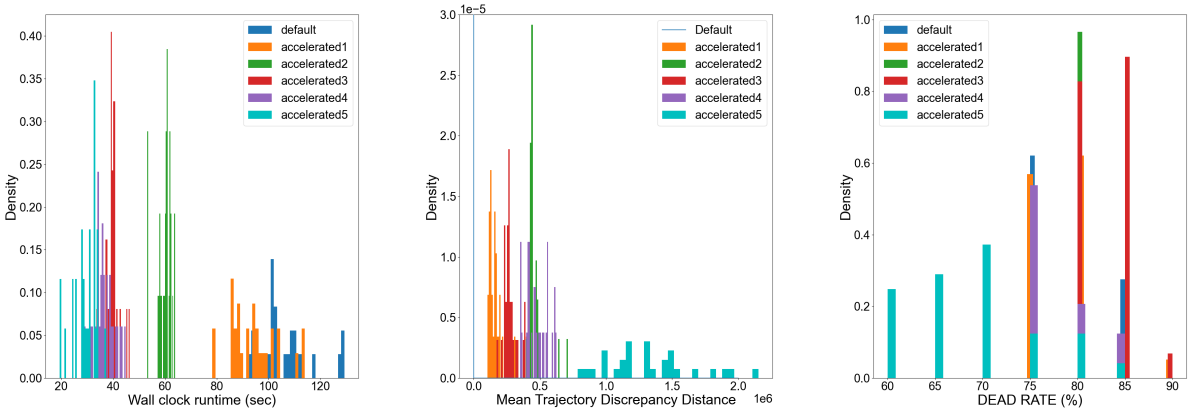


Figure 6: Performance measurements analysis of 5 Acceleration scenarios.

response surface analysis to analyze the interactions between each acceleration parameter to find the optimal point.

In addition to the USV model proposed in this paper, the acceleration method can be utilized if (1) model have coupling that transmits the repeated same message on specific condition and (2) state transition that repeatedly occurs with state changes remain below a certain threshold. When the system model meets one of the two conditions, it is worth considering acceleration method.

## 6 CONCLUSION

In this paper, we propose a DEVS-based simulation acceleration method that incorporates domain-specific knowledge and dynamic structural changes through DSDEVS formalism. The presented case study demonstrates how the proposed method was processed and implemented, and the experiment results show the efficiency of the proposed method. Furthermore, through the detailed analyses, a trade-off between the acceleration and the accuracy of simulations are empirically presented, which should be investigated more in the further research. It is expected that the proposed method be used to explore the balance between simulation speedup and fidelity and to provide insights into the acceptable limits of information loss in accelerated simulation scenarios.

## REFERENCES

Adelani, D. I. 2014. *A Devs-Based Ann Training and Prediction Platform*. Ph. D. thesis.

Barros, F. J. 1995. "Dynamic structure discrete event system specification: a new formalism for dynamic structure modeling and simulation". In *Proceedings of the 27th conference on Winter simulation*, 781–785.

Chow, A. and B. Zeigler. 1994. "Parallel DEVS: a parallel, hierarchical, modular modeling formalism". In *Proceedings of Winter Simulation Conference*, 716–722 https://doi.org/10.1109/WSC.1994.717419.

Chow, A., B. Zeigler, and D. H. Kim. 1994. "Abstract simulator for the parallel DEVS formalism". In *Fifth Annual Conference on AI, and Planning in High Autonomy Systems*, 157–163 https://doi.org/10.1109/AIHAS.1994.390488.

Concepcion, A. I. and B. P. Zeigler. 1988. "DEVS formalism: A framework for hierarchical model development". *IEEE Transactions on Software Engineering* 14(2):228–241.

Kim, S., J. Cho, and D. Park. 2018. "Accelerated DEVS Simulation Using Collaborative Computation on Multi-Cores and GPUs for Fire-Spreading IoT Sensing Applications". *Applied Sciences* 8(9) https://doi.org/10.3390/app8091466.

Köster, T., L. Herrmann, P. Andelfinger, and A. Uhrmacher. 2023. "GPU-Accelerated Simulation Ensembles of Stochastic Reaction Networks". In *Proceedings of the Winter Simulation Conference*, WSC '22, 2570–2581: IEEE Press.

Kulkarni, S., A. Tsyplikhin, M. M. Krell, and C. A. Moritz. 2020. "Accelerating Simulation-based Inference with Emerging AI Hardware". In *2020 International Conference on Rebooting Computing (ICRC)*, 126–132 https://doi.org/10.1109/ICRC2020.2020.00003.

Liu, Q. and G. Wainer. 2012. "Multicore acceleration of Discrete Event System Specification systems". *SIMULATION* 88(7):801–831 https://doi.org/10.1177/0037549711412237.

Praehofer, H. and D. Pree. 1993. "Visual modeling of DEVS-based multiformalism systems based on higraphs". In *Proceedings of the 25th conference on Winter simulation*, 595–603.

Saadawi, H., G. Wainer, and G. Pliego. 2016. "DEVS execution acceleration with machine learning". In *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*, 1–6 https://doi.org/10.23919/TMS.2016.7918816.

Shang, H. and G. Wainer. 2006. "A Simulation Algorithm for Dynamic Structure DEVS Modeling". In *Proceedings of the 2006 Winter Simulation Conference*, 815–822 https://doi.org/10.1109/WSC.2006.323163.

Sisyukov, A. N., V. Dobrynin, and O. S. Yulmetova. 2021. "Web Based GPU Acceleration in Embodied Agent Training Workflow". In *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, 686–689 https://doi.org/10.1109/ElConRus51938.2021.9396663.

Spiechowicz, J., M. Kostur, and L. Machura. 2015. "GPU accelerated Monte Carlo simulation of Brownian motors dynamics with CUDA". *Computer Physics Communications* 191:140–149 https://doi.org/10.1016/j.cpc.2015.01.021.

Stone, J. E., D. J. Hardy, I. S. Ufimtsev, and K. Schulten. 2010. "GPU-accelerated molecular modeling coming of age". *Journal of Molecular Graphics and Modelling* 29(2):116–125 https://doi.org/10.1016/j.jmgm.2010.06.010.

Trabes, G. G., G. A. Wainer, and V. Gil-Costa. 2023. "A Parallel Algorithm to Accelerate DEVS Simulations in Shared Memory Architectures". *IEEE Transactions on Parallel and Distributed Systems* 34(5):1609–1620 https://doi.org/10.1109/TPDS.2023.3256083.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation*. Academic press.

Zhao, Y., Y. Ma, and S. Hu. 2021. "USV Formation and Path-Following Control via Deep Reinforcement Learning With Random Braking". *IEEE Transactions on Neural Networks and Learning Systems* 32(12):5468–5478 https://doi.org/10.1109/TNNLS.2021.3068762.

## AUTHOR BIOGRAPHIES

**JUHO CHOI** is an master student in the Department of Aerospace Engineering at KAIST. His research interests include DEVS formalism and military M&S. His email address is 3747juho@kaist.ac.kr and his website is https://aai.kaist.ac.kr/bbs/board.php?bo_table=sub2_1&wr_id=19.

**JANG WON BAE** is an associate professor in the School of Industrial Management at Korea University of Technology and Education. His email address is jangwon_bae@koreatech.ac.kr.

**IL-CHUL MOON** is an associate professor in the Department of Industrial and System Engineering at KAIST. His email address is icmoon@kaist.ac.kr and his website is https://aai.kaist.ac.kr/bbs/board.php?bo_table=sub2_1&wr_id=3.