Modeling, Identification and Control, Vol. 45, No. 4, 2024, pp. 127–136, ISSN 1890–1328

Multi-Agent Maritime Traffic Simulator

Luka Grgičević [1]

[1] *Department of ICT and Natural Sciences, Norwegian University of Science and Technology, N-6025 Ålesund, Norway. E-mail: luka.grgicevic@ntnu.no*

Abstract

This simulator is designed to support research on centralised game-theoretical algorithms for maritime traffic management. It supports an arbitrary number of vessels and land masses import. Vessels are modelled as agents whose motion is governed by the kinematic equations and the land masses are polygon shape files. In the simulator, each vessel has access to the reward oracle, which evaluates the agents' strategies by taking into account the risk of collision and grounding, the level of compliance with the traffic rules and the operational efficiency. A game-theoretical model predictive control then generates optimal trajectories for every traffic participant simultaneously. Vessels are engaged in repeated competitive polymatrix games, whose equilibria solutions are a series of waypoints, meant to be broadcast as navigational decision support by the Vessel Traffic Services. We convey the agents' and functions' modelling principle implemented in NetLogo and present the overall simulator structure and scope.

Keywords: Centralised guidance system, NetLogo, Maritime traffic, Vessel Traffic Services

1 Introduction

Autonomous vessels have a growing presence in maritime traffic and the solutions that facilitate seamless inclusion of autonomous vessels into conventional traffic will be primarily verified through simulations. For example, Det Norske Veritas (DNV), a global quality assurance and risk management company, is actively involved in developing and verifying the simulators for multi-vessel collision avoidance (Minos et al., 2024). Software used for algorithm verification, such as SeaCharts (Blindheim and Johansen, 2022), Oceanscape (Fagerhaug et al., 2025), Autoferry Gemini (Vasstein et al., 2020), Marine Systems Simulator (Tristan and Fossen, 2009) and Python Vehicle Simulator (Fossen, 2021), to name a few, result from projects in which researchers from the Norwegian University of Science and Technology play a pivotal role. Some of the projects are research-based innovation centre, SFI AutoShip (2020), Autonomous all-electric passenger ferries for urban water transport Autoferry (2018), Sensor fusion and collision avoidance for autonomous surface vehicles Autosea (2016), The Safe Maritime Autonomous Technology SAFEMATE (2022), Online risk management and risk control for autonomous ships ORCAS (2018) and Breaching the boundaries of safety and intelligence in autonomous systems with risk-based rationality BREACH (2024). Meanwhile, the International Maritime Organisation (IMO), with the help of the flag countries' maritime authorities, continues to tailor the Maritime Autonomous Surface Ship (MASS) code (IMO, 2024). In summary, we witness efforts to develop standardised norms for autonomous vessels' operations, including the technical aspects, such as situational awareness and decision-making, and the regulatory requirements, such as COLREG (IMO, 1972).

1.1 Related work

Agent-based simulations are a powerful tool for modeling interactions among multiple objects or agents, capturing complex, dynamic systems with individual behaviors. Their performance benefits significantly from parallelization, as distributing agents or sub-tasks across processors reduces computation time for large-

doi:10.4173/mic.2024.4.2

scale simulations. Efficient memory usage and well-designed primitive functions are critical to handle the diverse and intensive data processing demands of these simulations. Practical overview of the existing agent-based simulators is provided in (Pal et al., 2020) and (Railsback et al., 2006).

For maritime purposes, agents could be vessels (of different vendors, class an other properties), land masses, pieces of the environment grid (such as patches of sea or the seabed), aids to navigation objects, other dynamic and static obstacles etc. Agent-based simulators for vessel guidance algorithms verification are scarce in the existing literature. For example, research in (Xiao et al., 2013) and (Xiao et al., 2012) uses NetLogo as modelling tool for maritime traffic safety assessment in waterways. Moreover, in the same simulator they used multiple vessel guidance models based on artificial potential fields to show the similarity with the Automatic Identification System data. Another example of usage of NetLogo in maritime traffic simulations is (Vaněk et al., 2013), where researchers developed AgentC, a platform to identify the maritime piracy and mitigate it through safe corridors design. Lastly, a related research study in (Krasowski et al., 2025) relies on a multi-agent simulator with built-in rule-adaptive waypoint engine and low-level model predictive controller. This is a scalable platform with the possibility to include Automatic Identification System (AIS) data. This research facilitates training agents using reinforcement learning. We extend this research by deploying a comprehensive vessel guidance risk assessment (reward framework), and a game-theoretical mechanism to observe strategic interactions among agents.

1.2 Scope and contributions

A majority of custom-made maritime simulators present in the research are oriented towards the control systems and deploy dynamical models that describe interactions of the rigid body with air and fluid. For the purpose of vessel guidance, there are numerous manoeuvring models that approximate the underlying physical properties to a bigger or smaller extent. Such dynamical system requires model of situational awareness that serve as input to the low-level vessel control and the decision making algorithm that guides the vessel. Scaling the simulators to handle multiple instances of such systems recquires significant memory consumption and computational capacity.

The use case should be the starting point in defining the scope of the simulator in order to achieve efficiency and the intended accuracy of the simulations. This simulator is built specifically to observe multi-vessel encounters that operate in high-density traffic in open or near-shore waters. In other words, the simulator was developed to enable research on guidance decision support that could be further extended to centralised traffic management. Such navigational assistance is meant to be provided by the Vessel Traffic Services (VTS). In this setting, the decision support extends beyond optimising individual cost functions, to a game-theoretical setup in which vessels' strategic interactions become interesting.

The main contribution of this article is to show how NetLogo, a multi-agent programmable modelling environment Wilensky (1999), can be utilised for vessel guidance algorithm verification. We made some vessel model and the environment simplifications to capture only the most important system dynamics. Each vessel is an agent that moves on the flat plane as a velocity vector that changes position, heading and speed. The heading and speed change according to the kinematic model described initially in (Grgičević et al., 2024a). Since the kinematic model replaces the vessels' speed and heading autopilots we assume the environmental forces do not significantly influence the vessels' ability to follow the reference trajectories. Therefore, the wind, wave and water currents are not modelled. Every agent has a strategy set, the available heading and speed change setpoints, whose entries are intended reference (setpoint) values to its kinematic model. Using this set, each agent generates offspring, a set of agents of different *breed*, one for each strategy. Land masses are represented as polygons whose vertices are static agents with known coordinates. The behaviour of vessels is defined by the game-theoretical model predictive control guidance algorithm, a real-time applicable solution, in detail described in (Grgičević et al., 2024b).

1.3 Outline

In Section 2, the simulator is described from the birds-eye perspective. In Section 3, we described how to represent a vessel and its strategies with agents of two breeds and in Section 4 we describe the callable custom functions that are used to gather agents' information, construct knowledge arrays and contribute building the complete game environment. Additional functions that serve to solve the game using the evolutionary dynamics are also described. The games' solutions dictate the vessels' trajectories in the simulation. In Section 5, we describe the available tuning parameters. Lastly, in Section 6, we elaborate on possible extensions in the simulator and some simulation results, and in Section 7 the concluding remarks are provided.

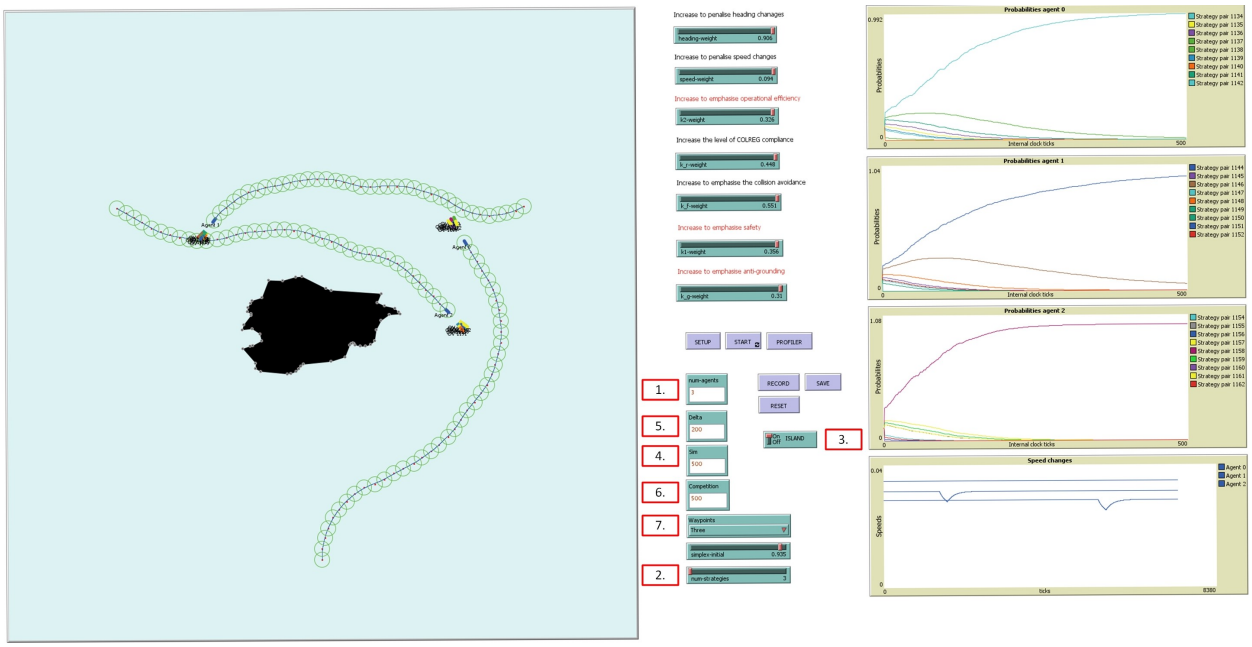

Figure 1: The main interface

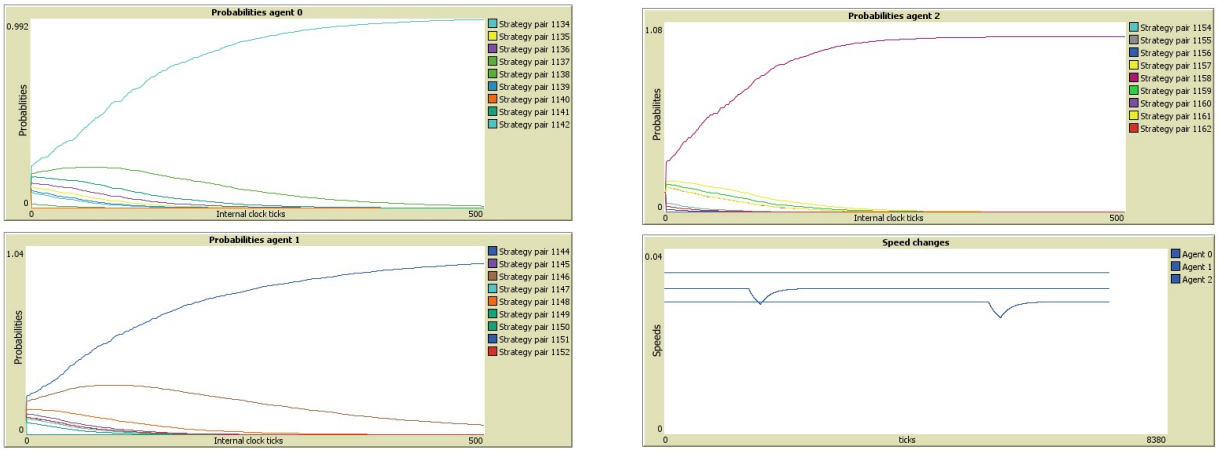

Figure 2: Enlarged plots from the main interface. Convergence of the mixed strategies probablity distribution for each parent agent and their speeds. Note: the map is not to scale, so the speeds are in patch/s.

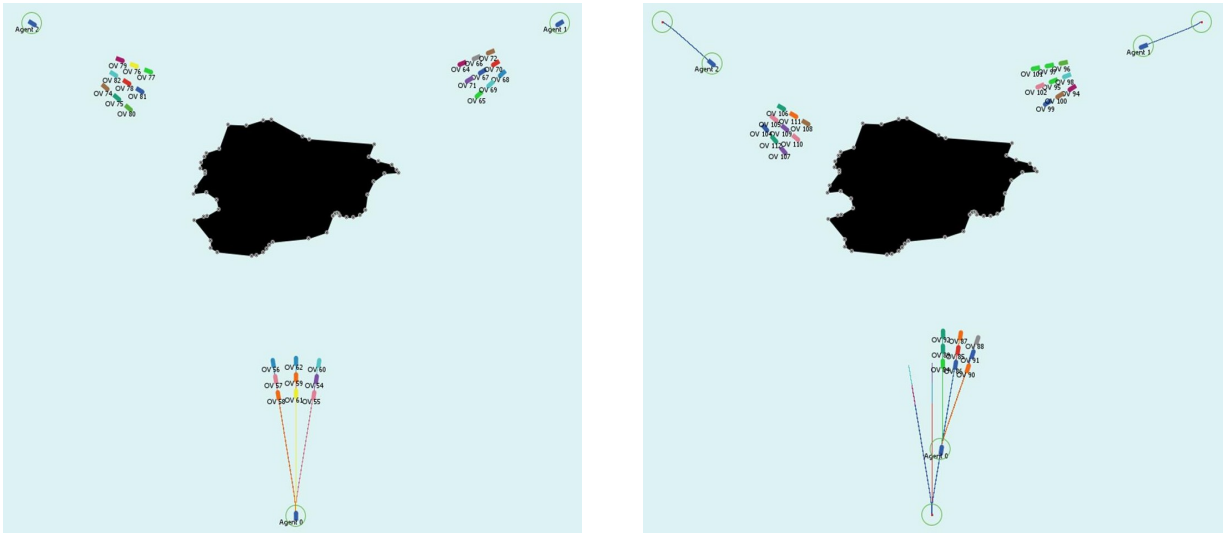

Figure 3: One control message has been generated and broadcast to the three vessels (in blue). Suggested green waypoints are being sent as decision support.

2 Simulator overview

The user can specify an arbitrary number of vessels, with their current speed over ground, heading and position, and include land maps. The simulator will generate a trajectory for each vessel in the shape of a series of waypoints with a constant speed on each path leg.

NetLogo environment supports four types of agents: turtles, patches, links and the observer (Wilensky and Rand, 2015). Each type can be of a different breed, meaning separate private variables could be linked to it. Agents of the same type share the available primitive methods that are abundant in NetLogo. Additional supported extensions could be added in NetLogo to get access to additional functions needed. Turtles are used for vessels and the observer is used to set up the simulator, run it in an infinite loop until the interrupt is made and generate the plots. In Figure 1, the main interface could be depicted. On the left, the map with the provisional number of light blue patches in width, agents with the corresponding trajectories and an island in black could be depicted. If the real dimensions are needed, each patch has a size in pixels of the screen, which makes it possible to set the adequate ratio and individual agents' sizes. In the middle, there are buttons, sliders and input fields for starting and stopping the simulations, and for choosing the parameters for the simulation. In Figure 2, the plots represent the agents' preferences for choosing one of the available strategies, a convergence of the initially mixed strategy to an equilibrium, and the three agents' speeds in the bottom right.

3 Agents

There are two breeds of the vessel agent type, the parent agents (vessels) and the offspring agents (strategies) that inherit all the private variables of their respective parents. Offspring agents are performing motion as if the parent executed all the available strategies at once. The available strategies are respective heading and speed changes that are given as setpoints to vessels' autopilots modelled with corresponding kinematic equations. Each offspring agent performs additional computation, maintains knowledge arrays and contributes to building the game setup using the reward framework. For this purpose, a changing subset of offspring agents will be referred to as *targets*. Information and the knowledge arrays that agents store as private variables are given in appendix A.

After the polymatrix games are constructed and before each consecutive strategy selection round the offspring agents are deleted. The new collection of offspring agents is created every δ time, marking the control messages' broadcast time. The intuitive algorithm timeline is depicted in Figure 4. The Δ is a state evolution time, the strategies prediction horizon, and t is a time for each parent agent's strategies to settle to pure Nash equilibrium (PNE) (Nash, 1950). In Figure 3 on the left, offspring are generated, they evolve their states for Δ and compete during PNE search for t time. This competition is depicted in Figure 2. In Figure 4 on the right, PNE is selected and broadcast to local traffic in the shape of speed and heading changes. The next control message has been sent after time δ has passed. At this time, when the broadcast control mes-

sage is received and implemented, the location of the vessels could be depicted by the second green circular waypoint.

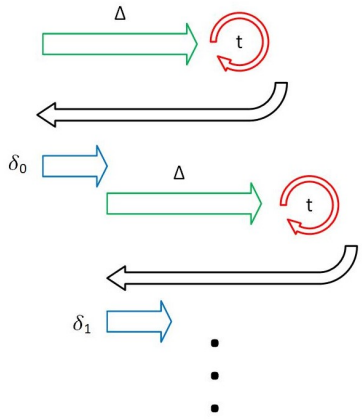

Figure 4: The guidance algorithm scheduling. Green arrows represent a lifetime of offspring agents, and blue the periods during which parent agents (vessels) move.

4 Functions

Agents can perform computations using the many available primitives (methods), such as distances or bearings to other agents or patches. The additional custom-designed functions in this simulator could be grouped as follows:

- State evolution.
- Knowledge representation.
- Reward framework.
- Game solution.

The functions grouped as the state evolution are essentially kinematic equations solvers. The input to the state evolution function group is the velocity vector of a parent agent. Based on the desired number of strategies and their desired shape, such as the distance between the heading changes and the difference between the speed changes, an arbitrary number of strategies is generated, inheriting the velocity vector from its parent. The transition to the given heading and the given speed is obtained using the Euler integration. For details, the reader is referred to Grgičević et al. (2024a). If the evolution time Δ is shorter than the near steady state time is reached, the offspring agents inherit the speeds from their parents, as shown in Figure 2.

The functions grouped as the knowledge representation are used by the offspring agents. Each agent, at the end of the state evolution, obtains velocity vector information from all the other strategies and constructs the knowledge arrays. Those arrays, such as lists of distances and times to the closest points of approach, attention and scaling lists etc, are necessary to obtain the strategy performance computed in the reward framework function group.

The reward framework consists of several functions used to obtain the offspring agents' performance. In essence, it combines balances between the functions related to vessel safety and operational efficiency that minimise the cost of achieving the anti-collision and anti-grounding manoeuvres while complying with the traffic rules. By calling the reward oracle each offspring agent can obtain a scalar or reward vector for any perturbation of strategies of other parent agents.

Those reward entries are then used to build a normal form game tensor, or a succinct representation, a polymatrix game. The game solution function group is used to generate a collection of asymmetric bimatrix games and search for a solution (PNE), utilising evolutionary dynamics.

The state evolution (green), knowledge representation (brown), reward framework (blue) and game solution (purple) are listed in appendix B. Other functions, marked in black, are used to schedule the simulation, generate the initial conditions, monitor the performance, analyse the algorithm exception speed and generate videos from the simulations.

The simulator execution can be conveyed through pseudo-code, initially introduced in Grgičević et al. (2024b). The simplified overview is given in Algorithm 1.

5 Parameters

In the middle of Figure 1, the user set up the most important initial conditions, marked with the corresponding numbers in red boxes:

1. Number of vessels in the local traffic.
2. The number of strategies available to each vessel. For example, nine strategies would correspond to three heading change setpoints and three speed change setpoints.
3. Switch to include the imported map.
4. The time of the offspring agents' state evolution.
5. The time between the waypoints, the control messages.
6. The convergence time during the strategy selection (search for PNE).

Algorithm 1 One control message (PNE) generation

if Message time δ_{k} **then**
  READ traffic participants' states (U, ψ)
  ASSIGN an agent for each n parent agent
  ASSIGN an agent for each strategy
  for each Offpring agent **do**
    while $\Delta\,! = \Delta_{\mathrm{final}}$ **do**
      COMPUTE state evolution
    end while
  end for
  for each Offspring agent **do**
    COMPUTE agents' knowledge arrays
  end for
  DEFINE fully connected graph of parent agents
  for each Parent agent **do**
    for each Outward edge **do**
      DEFINE other parent agents' strategies
      COMPUTE reward framework
      CONSTRUCT $n-1$ bimatrix games
    end for
  end for
  while $t\,! = t_{\mathrm{final}}$ **do**
    GET random edge on a graph
    OBTAIN an asymmetric game on that edge
    UPDATE the agents' population structure
    INCREMENT the internal clock
  end while
  DEFINE the winning n strategy pairs (PNE)
  BROADCAST the control message
end if

7. The vessels' initial positions, speeds and headings.

Moreover, the first seven parameters (k_{r}, k_{f}, k_{s}, k_{ψ}, k_{g}, k_1 and k_2) that could be depicted by the weighting sliders from zero to one, are used to tune the guidance algorithm reward framework. Parameter modification will result in different trajectories for each vessel.

6 Discussions

In Figure 5, emerges an interesting pattern of trajectories in the traffic of ten vessels, initially passing through the centre of the map. The local traffic can be arbitrarily scaled, vessels with different manoeuvrability can be included, maps in different formats can be included using the NetLogo GIS Extension (Russell and Edelson, 2012), and the guidance algorithm can be differently tuned.

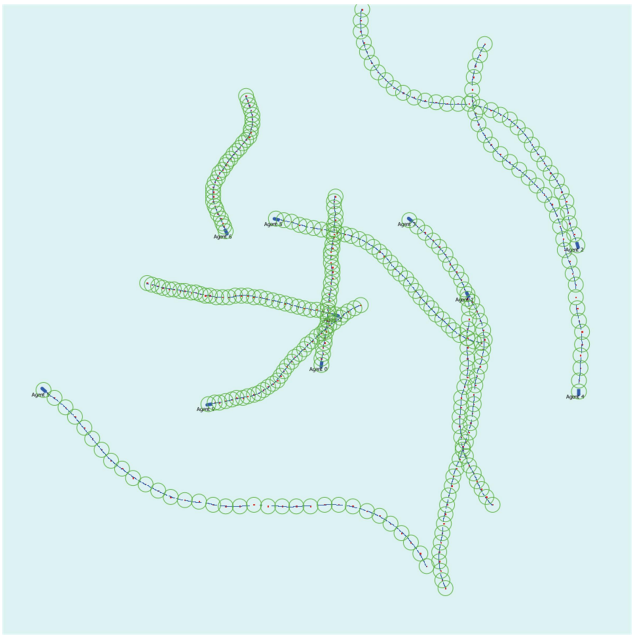

Figure 5: Ten vessels with different nominal speeds receiving and implementing the guidance advice.

Practical extension to the simulator may include handling new incoming vessels that are not initially on the map, as well as outgoing vessels. Including a real-time AIS data stream to generate decision support in the shape of several waypoints in the future would also be beneficial to increase the simulator's technology readiness level. NetLogo is a practical programming tool for modelling high-density maritime traffic facilitating game-theoretical algorithm implementation in which strategies are represented with agents.

7 Conclusions

The article describes in detail an agent-based simulator for maritime traffic management. The simulator shows how an arbitrary traffic of active players and static obstacles can be modelled. The tunable game-theoretical guidance algorithm built in, finds shortest and fastest trajectories for each vessel that, at the same time, avoid collisions and grounding, and show compliance with the COLREG. We conveyed how to model the agents and described the functions using which they represent the environment, evaluate it, and select the appropriate strategies for any traffic situation.The solution is envisioned to be a step towards the real-time application of navigational assistance in VTS, which will serve as maritime management providers through route suggestion and decision support.

8 Acknowledgment and declarations

The author expresses gratitude to his supervisory team: Erlend M. Coates, Robin T. Bye and Ottar L. Osen, affiliated with the Department of ICT and Natural Sciences at the Norwegian University of Science and Technology (NTNU) in Ålesund, and Thor I. Fossen, affiliated with the Department of Engineering Cybernetics at NTNU in Trondheim for leading the research project and fruitful discussions over the past years, invaluable guidance and critical reflections that elevated the research in both depth and width.

This research is funded by the Doctoral Programme for Integrated Research Activities to Unlock a Potential of Top-level Researchers in Digital Transformation for Sustainability (PERSEUS), the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement number 101034240. This research is also funded by SFI AutoShip, an 8 -year research-based innovation centre focusing on safe and sustainable autonomous ship operations, through the Research Council of Norway, project number 309230. The author have no competing interests to declare that are relevant to the content of this article.

References

Autoferry. Autonomous all-electric passenger ferries for urban water transport. 2018. URL https://www.ntnu.edu/autoferry. Accessed: 2024-12-17.

Autosea. Sensor fusion and collision avoidance for autonomous surface vehicles. 2016. URL https://www.ntnu.edu/autosea. Accessed: 2024-12-17.

AutoShip. Centre for Research-based Innovation (SFI). 2020. URL https://www.ntnu.edu/sfi-autoship. Accessed: 2024-12-17.

Blindheim, S. and Johansen, T. A. Electronic Navigational Charts for Visualization, Simulation, and Autonomous Ship Control. *IEEE Access*, 2022. 10:3716–3737. doi:10.1109/ACCESS.2021.3139767.

BREACH. Breaching the boundaries of safety and intelligence in autonomous systems with risk-based rationality. 2024. URL https://www.ntnu.edu/imt/erc-advanced-grant-breach. Accessed: 2024-12-17.

Fagerhaug, E. S., Bye, R. T., Osen, O. L., and Hatledal, L. I. Oceanscape: A graph-based framework for autonomous coastal navigation. *Ocean Engineering*, 2025. 320:120230. doi:10.1016/j.oceaneng.2024.120230.

Fossen, T. I. *Handbook of Marine Craft Hydrodynamics and Motion Control, 2nd Edition*. John Wiley and Sons, 2021.

Grgičević, L., Coates, E. M., Bye, R. T., Fossen, T. I., and Osen, O. L. Towards Decision Support in Vessel Guidance Using Multi-Agent Modelling. *2024 European Control Conference (ECC)*, 2024a. pages 1131–1138. doi:10.23919/ECC64448.2024.10591229.

Grgičević, L., Coates, E. M., Fossen, T. I., Bye, R. T., and Osen, O. L. Centralised Decision Support in Maritime Vessel Traffic Services: A Polymatrix Game Solution. 2024b. doi:10.13140/RG.2.2.35437.09447. Preprint at http://dx.doi.org/10.13140/RG.2.2.35437.09447.

IMO. International Maritime Organization, International Regulations for Preventing Collisions at Sea (COLREG). 1972.

IMO. International Maritime Organisation, Maritime Safety Committee, 108th session (MSC 108). 2024. Accessed: 13/12/2024.

Krasowski, H., Schärdinger, S., Arcak, M., and Althoff, M. Intelligent Sailing Model for Open Sea Navigation. 2025. doi:10.48550/arXiv.2501.04988.

Minos, H., Arne, P. T., and Claas, R. Maritime schema. 2024. URL https://github.com/dnv-opensource/maritime-schema. Accessed: 13/12/2024.

Nash, J. *Non-cooperative Games*. Ph.D. thesis, Princeton University, 1950.

ORCAS. Online risk management and risk control for autonomous ships. 2018. URL https://www.ntnu.edu/imt/orcas. Accessed: 2024-12-17.

Pal, C.-V., Leon, F., Paprzycki, M., and Ganzha, M. A Review of Platforms for the Development of Agent Systems. *Inf.*, 2020. 14:348. doi:10.48550/arXiv.2007.08961.

Railsback, S. F., Lytinen, S. L., and Jackson, S. K. Agent-based Simulation Platforms: Review and Development Recommendations. *SIMULATION*, 2006. 82(9):609–623. doi:10.1177/0037549706073695.

Russell, E. and Edelson, D. Gis extension. https://github.com/NetLogo/GIS-Extension, 2012.

SAFEMATE. The SAFE Maritime Autonomous Technology research project. 2022. URL https://www.dnv.com/expert-story/maritime-impact/safemate-showing-the-way-for-autonomous-shipping. Accessed: 2024-12-17.

Tristan, P. and Fossen, T. I. A Matlab Toolbox for Parametric Identification of Radiation-Force Models of Ships and Offshore Structures. *Modeling, Identification and Control*, 2009. 30. doi:10.4173/mic.2009.1.1.

Vaněk, O., Jakob, M., Hrstka, O., and Pěchouček, M. Agent-based model of maritime traffic in piracy-affected waters. *Transportation Research Part C: Emerging Technologies*, 2013. 36:157–176. doi:10.1016/j.trc.2013.08.009.

Vasstein, K., Brekke, E., Mester, R., and Eide, E. Autoferry gemini: A real-time simulation platform for electromagnetic radiation sensors on autonomous ships. *IOP Conference Series: Materials Science and Engineering*, 2020. 929(1). doi:10.1088/1757-899X/929/1/012032.

Wilensky, U. NetLogo, Center for Connected Learning and Computer-Based Modeling. 1999. Northwestern University, Evanston, IL.

Wilensky, U. and Rand, W. *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. MIT Press, 2015.

Xiao, F., Ligteringen, H., van Gulijk, C., and Ale, B. Nautical traffic simulation with multi-agent system for safety. *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 2013. pages 1245–1252. doi:10.1109/ITSC.2013.6728402.

Xiao, F., Ligteringen, H., van Gulijk, C., and Ale, B. Artificial Force Fields for Multi-agent Simulations of Maritime Traffic: A Case Study of Chinese Waterway. *Procedia Engineering*, 2012. 45:807–814. doi:10.1016/j.proeng.2012.08.243. 2012 International Symposium on Safety Science and Technology.

Appendix A

Variables marked with asterix are inherited and maintained by the offspring agents.

Parent agents (vessels)	Offspring agents (strategies)	Globals
τ* - time constant in speed change	$point_y$ - variable, line integration of function R - North coordinates	OV_{wp} - list of waypoints
ζ* - damping parameter in speed	$point_x$ - variable, line integration of function R - East coordinates	OV_h - list of headings
own_x* - East coordinates	y_{mc} - copy list, construction of F and F_{ga}	D_{WP} - VTS decision support scope
own_y* - Northe coordinates	x_{mc} - copy list, construction of F and F_{ga}	creation - strategies creation boolean
τ_r* - time constant in heading change	σ_{ac} - copy list, construction of F and F_{ga}	selection - control message trigger boolean
ζ_r* - damping parameter in heading	$tcpa_{ac}$ - copy list, construction of A	i, j - increments
ψ* - current heading	$dcpa_{ac}$ - copy list, construction of A	Δt - sampling time
ψ_{-1}* - heading 1 Δt ago	$dcpa_a$ - list, distance to closest point of approach (CPA) of each target	δ - time between the control messages
ψ_{-2}* - heading 2 Δt ago	W - list, attention to targets	δ_{sim} - time saver for δ
U* - current speed	E - list, rewards operational efficiency	offset - plotting variable
v_E* - speed decomposed in East	R - list, rewards traffic rules	stop - simulation end boolean
v_N* - speed decomposed in North	F_{ga} - list, rewards grounding risk	agents - list of agents' identifications
U_{-1}* - speed 1 Δt ago	F - list, rewards collision risk	clear - map clearing boolean
U_{-2}* - speed 2 Δt ago	$targets_a$ - list of active targets, identifications	$\Delta\psi$ - equiangular offset in heading strategies
U_{ch}* - current speed setpoint	$tcpa_{a\text{-}1}$ - list, time to CPA of each target 1 Δt ago	ΔU - offset between the speed strategies
WP - list of control messages, waypoints	WP_a - local traffic exit waypoint	ΔU - offset between the speed strategies
U_{prev} - speed before the new strategy selection	L_a - list of active targets' lengths	m_ψ - number of available heading strategies
U_{nom} - nominal speed	σ_a - list, global collision risk metric	m - number of available strategy pairs
ψ_{nom} - current nominal heading	parent - parent agent identification	m_g - lists of agents' strategy pairs
D* - list of distances to targets	y_m - list, North coordinate used in F	obstacle - static land mass polygons shape files
D_{-1}* - D list 1 Δt ago	dcpa - variable, distance to CPA	
D_{tr}* - circular safety distance	x_m - list, East coordinate used in F	
L* - vessel length	$tcpa_a$ - list, time to CPA of each target	
A* - boolean list of approaching vessels	tcpa - variable, time to CPA	
ψ_a - heading change setpoint list	σ - variable, global collision risk metric	
U_a - speed change setpoint list		
games - tensor of games		
strategies - probability distribution over strategies		

Appendix B

Described functions grouped with colour.

Functions	Description
SETUP	Initialization function in which the user may customise the vessels' manoeuvrability, add new charts, change the offsets between the strategies etc.
GO	This function runs indefinitely, governing the simulator workflow and managing the plots.
REPLICATOR	Game solutions are series of PNE that are sought using the evolutionary replicator dynamics applied on polymatrix games, fully connected graphs of bimatrix games on edges with parent agents on vertices.
RANGE	Used for generating the probability distributions.
ENERGY	Computes operational efficiency of each strategy, penalising the deviations from nominal speed and heading.
INTEGRATE	Line integration of global collision risk metric, called by the offspring agents.
COLREG	Cost of violating the traffic rules.
NOLRMALISE	Min-max scaling of individual costs in the reward framework.
ROTATION	Function used by COLREG that computes the rotation matrix, a relative position between two offspring agents.
PERT-x	Function used by COLREG that computes the bivariate PERT distribution, longitudinal direction.
PERT-x	Function used by COLREG that computes the bivariate PERT distribution, perpendicular direction.
CPA	Computes lists of distances and times to the closest approach to each target vessel, as well as boolean approach lists and parameters' lists used in the reward framework.
DCPA	Computes distance to the closest point of approach, given the two velocity vectors. Used in CPA.
ATTENTION	Attention to targets mechanism, used to scale individual contributions in the reward framework
HEADINGS	Genrates strategy sets in heading change setpoints of arbitrary size and shape.
SPEEDS	Genrates strategy sets in speed change setpoints of arbitrary size and shape.
SPEED-TF	Integrates kinematic equation in speed change, first-order system.
HEADING-TF	Integrates kinematic equation in heading change, critically damped second order system.
EDGE	Computes local traffic exit waypoint coordinates for each vessel.
BINOMIAL	Generates initial conditions in strategy probability distribution before the PNE search.
BINS	Used in BINOMIAL to generate a binomial probability distribution for arbitrary startegy set size.
DISTANCES	Used in algorithm safety analysis, monitoring the distances between the vessels during the simulation run.
RECORDER	Used in algorithm analysis, generating videos from simulations.
PROFILER	Used in algorithm performance analysis and optimisation, evaluating algorithm run time.