TECHNISCHE UNIVERSITÄT DRESDEN

Performant Simulation of Inland Ship Traffic: PERIST

A documentation for using the PERSIST framework to model and assess inland vessel traffic

Niklas Paulig
Fabian Hart

May 10, 2023

Disclaimer

This documentation is provided ”as is” and to the best of our knowledge, contains accurate and reliable information. However, it is important to note that this documentation is a work in progress and may not be complete or up-to-date. We make no representations or warranties of any kind, express or implied, about the completeness, accuracy, reliability, suitability, or availability with respect to the documentation or the information contained within it. Any reliance you place on this documentation is therefore strictly at your own risk. We reserve the right to make changes to this documentation at any time without prior notice.

Copyright

Contact

Niklas Paulig
Institut für Wirstschaft und Verkehr
Würzburger Straße 35, 01187 Dresden

Contact: niklas.paulig@tu-dresden.de

Changelog

v1.0.1	2022-06-20	Enabled logging for the entire simulation process. The data of individual vessels will be logged into a separate file, located in the simlogs folder.
v1.0.2	2033-06-23	Bug fixes.
v1.0.3	2022-08-24	Vessels are now modeled individually by a class. All ships share the same river object. Drivable area per vessel is pre-computed for a discrete range of squat values. Ship draught mandatory in configuration file.

Table of Contents

1 Introduction

1.1 About PERSIST

Traffic modeling is an important component for the safety and economical development of inland waterways. It enables investigating system behavior and development under static and dynamic environmental conditions, such as decline in water levels, changing number of traffic participants or changing behavior of individuals.

As part of a research collaboration, a micro traffic simulation model (TSM) called Performant Simulation of Inland Ship Traffic (PERSIST) was developed. The model uses agents to control individual inland vessels, which were trained through a machine learning approach to select appropriate trajectories and make decisions on ship-ship interactions while considering hydraulic and traffic constraints. PERSIST provides an excellent foundation for advanced applications, including traffic forecasting and analysis related to shipping traffic automation. [1]

[1] The mentioned applications are a non-exhaustive snapshot at the time of writing, as PERSIST is currently under development.

In short, PERSIST allows you run a traffic simulation by specifying a configuration file, which contains all relevant information about the simulation scenario, such as the number of vessels, their starting positions, desired powers, masses, dimensions, and the waterway. The simulation is then run for a specified number of time steps, and the resulting trajectories are stored in a log file. The log file can then be used to analyze the simulation results.

1.2 Structure of this documentation

This documentation is intended to provide a comprehensive overview of the PERSIST framework, and is therefore split into two parts.

Part one serves as a Practitioner's Guide, providing a detailed instruction manual on how to install and use the PERSIST framework and its functions. We will explain the functionality of the most important modules and provide examples on how to use, and adapt them.[2] We will take a look at the configuration files and explain how to set up a simulation scenario. Finally, we will provide a detailed description of the output visualizations and logs and how to interpret them.

[2] Most code examples in this documentation are written in Python, so a basic understanding of the language is required.

Part two serves as a methodological reference, providing a detailed description of the underlying concepts and methods used in the PERSIST framework. We will explain the algorithmic foundation as well as the training and testing procedures for the machine learning models used in PERSIST.

2 Practitioner's Guide

2.1 Installation

Prerequisites PERSIST is written in Python 3.9 and due to the use of type hinting with built-in types it currently does not support backwards compatibility. You can install the latest version of Python from the official website via this link.

After obtaining a working Python installation, you can obtain a copy of the PERSIST framework either from the official GitHub repository via this link... or by downloading the latest release from the official website via this link... .

After downloading and/or unpacking the PERSIST framework to a folder of your choice, we recommend installing it into a virtual environment.

2.1.1 Setting up a virtual environment

> **Note:** This section is optional. If you do not want to use a virtual environment, you can skip this section and proceed to the next section.

A virtual environment is a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages. Different virtual environments can use different Python versions and can have different sets of installed packages. This allows you to have multiple versions of Python and multiple versions of the same package installed on the same system without conflicts.

You can create a virtual environment by running the following command in your terminal:[3]

```
python -m venv .persist
```

This will create a new folder called *.persist* in your current

[3] If not stated otherwise, all command-line instructions in this documentation are written for Linux-based systems. If you are using a Windows system, we recommend using the Windows Subsystem for Linux to run Linux commands.

working directory. To activate the virtual environment, run the following command:

```
source .persist/bin/activate
```

You can deactivate the virtual environment at any time by running the following command:

```
deactivate
```

2.1.2 Installing PERSIST

After activating the virtual environment, make sure you are in the root directory of the PERSIST framework and run the following command:[4]

```
pip install .
```

In case you want to install PERSIST in development mode, i.e. you do not have to reinstall the package after manual changes in the code, run the following command instead:

```
pip install -e .
```

To verify that the installation was successful, you can run:

```
persist
```

which should print a welcome message together with the current version of PERSIST.

Congrationlations - you have successfully installed PERSIST!

[4] PERSISTs dependency management is handled via poetry. If you do not have poetry installed, you can install it by following the instructions on the official website via this link ↗.

2.2 Configuration files

Basic information To prepare and execute simulations, PERSIST uses two configuration files in the `.toml`[5] file format.

The first file, called `rivers.toml` contains names and data paths of all rivers that are available for simulation. This file is always located under `./data/rivers/rivers.toml` and should only be changed if you want to add or remove rivers from the simulation.

[5] TOML (Tom's Obvious, Minimal Language) is a configuration file format designed to be easy to read and write, yet unambiguous and precise. It aims to be a minimal configuration file format that can be easily parsed and manipulated by both humans and machines.

The second file will be the main configuration file of the simulation, and can be named arbitrarily, as long as it has the `.toml` file extension. This file contains all relevant information about the simulation scenario, such as the number of vessels, their starting positions, desired powers, masses, dimensions, and the waterway specifications. This file has no specific location, although we recommend storing it in the `./configs/` folder.

We will now explain the structure of both files in detail.

2.2.1 rivers.toml

As per release, the `rivers.toml` file contains the following information:

```
[lower_rhine]
path = ”lr”

[middle_rhine]
path = ”mr”
```

The name in brackets is the name of the river, which is used to reference the river in the main configuration file. The `path` key contains the path to the river data folder, which is located in the `./data/rivers/` folder. The path is relative to the `./data/rivers/` folder, so in this case the `lower_rhine` river data folder is located at `./data/rivers/lr/`.

For information about adding new rivers to the simulation, please refer to the *Adding new rivers* section.

2.2.2 Main configuration file

The main configuration file contains all relevant information about the simulation scenario. The following example shows the structure of the `demo.toml` file which comes with the PERSIST framework:

```
[River]
river_name = ”lower_rhine”
GPW = 26
BPD = 20

[Ships]
lengths = [100, 100, 100, 100, 100, 100 ]
widths = [10, 10, 10, 10, 10, 10 ]
masses = [100_000, 100_000, 100_000, 100_000, 100_000, 100_000 ]
draughts = [2.8, 2.8, 2.8, 2.8, 2.8, 2.8 ]
x_locations = [17_900, 18_300, 18_800, 25_100, 25_600, 26_300 ]
y_locations = [190, 190, 190, 310, 310, 310 ]
overtaking_levels = [1, 1, 1, 1, 1, 2 ]
```

```
directions = [ 1, 1, 1, -1, -1, -1 ]
desired_powers = [ 700_000, 600_000, 300_000, 300_000, 600_000,
    700_000 ]

[Visualization]
visualize = false
freeze = true
utm_transform = true
followed_vessel = 5
zoom = 1_500

[Simulation]
speed_up = 1
n_iters = 2_000

[Data]
river_path = "data/rivers"
onnx_path = "data/onnx_nets"
logging_root = "simlogs"
```

The file is divided into several sections, each of which will be explained in detail.

[River] The `River` section contains information about the river that is to be simulated. The `river_name` key contains the name of the river as specified in the `rivers.toml` file. The `GPW` and `BPD` keys contain information about the grid specification used by the specified river file.

GPW (Grid Points per Width) specify how many grid points are used to discretize the width of the river. BPD (Base Point Distance) specifies the distance between two grid points in the horizontal and vertical irection (See Figure 1 on the right).

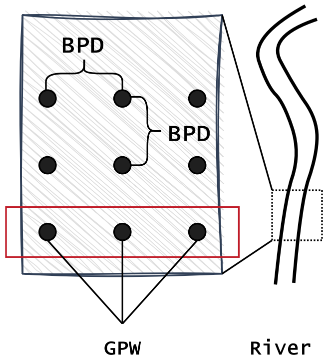

Figure 1: GPW and BPD specification.

> **Note:** For the simulation to work, the GPW and BPD values must match the values used to generate the river data file. If you are using one of the rivers that come with the PERSIST framework, you do not need to change these values.

In our example we are using the `lower_rhine` river, which has a GPW of 26 and a BPD of 20, meaning that the river is discretized into 26 grid points in the width direction, and that the distance between two grid points is 20 meters.

[Ships] The `Ships` section contains information about the vessels that are to be simulated. The `lengths`, `widths`, `masses`, and `draughts` keys contain the lengths, widths, masses, and draughts of the vessels in meters, kilograms, and meters, respectively.

The `x_locations` and `y_locations` keys contain the starting positions of the vessels in meters, given a flat projection of the river onto the x-y plane as seen in Figure 2. In the flat projection, the river starts at the origin and flows downstream for increasing x values.

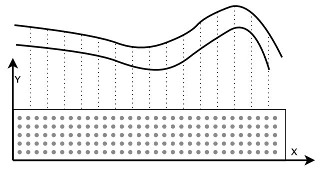

Figure 2: Flat projection of the river onto the x-y plane.

The `overtaking_levels` key contains the overtaking levels of the vessels. A vessel can and will only overtake vessel with overtaking levels that are lower than its own. For example, a vessel with an overtaking level of 2 will overtake vessels with overtaking levels of 1 and 0, but not vessels with overtaking levels of 2 or higher. An overtaking level of 0 is equivalent to not overtaking at all.

The `directions` key contains the directions in which the vessels are to be simulated. A value of 1 means that the vessel will advance downstream, and a value of -1 means that the vessel will advance upsteam.

The `desired_powers` key contains the desired powers of the vessels in watts. The maximum power of the vessel depends on the type of vessel simulated. Currently only a single vessel type (Europaschiff 100x10m) can be simulated, which has a maximum power of 1,000,000 watts. [6]

[6] For future versions, this specification may be replaced by directly specifying the vessel type.

[Visualization] The `Visualization` section contains information about the visualization of the simulation.

> **Note:** Due to its implementation in *matplotlib*, the visualization module of PERSIST is very slow and thus intended to be a visual debugging tool only. Logging is disabled when the visualization is enabled.

The `visualize` key specifies whether the simulation should be visualized or not.

The `freeze` key specifies whether the simulation should be frozen at the beginning of the simulation. Freezing locks the camera at the average position of every vessel at the specified zoom level.[7]

[7] This feature is experimental and may not work as intended. If you cannot see the vessels in the visualization, try changing the zoom level to a higher value.

The `utm_transform` key specifies whether the visualization should be transformed into the UTM coordinate system. If set to `false`, the visualization will be in the flat projection as seen in Figure 2.

The `followed_vessel` key specifies which vessel should be followed by the visualization. The first vessel has an index of 0, the second vessel has an index of 1, and so on.

The `zoom` key specifies the zoom level of the visualization in meters of radius around the followed vessel (see Figure 3).

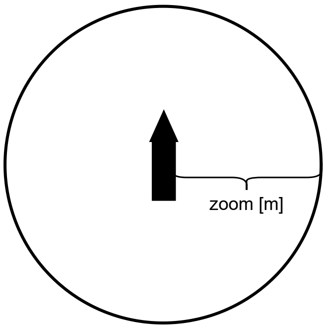

Figure 3: Visualization zoom level.

[Simulation] The `Simulation` section contains information about the simulation itself.

The `speed_up` key specifies the speed up factor of the simulation. A value of 1 means that the positions of the vessels are updated once per second. A value of 2 means that the positions of the vessels are updated every two seconds, and so on.

The `n_iters` key specifies the number of iterations that the simulation should run for.[8]

[8] Currently, there is no option for the simulation to run in sandbox mode (indefinitely), as vessels leaving the river are not yet handled.

[Data] The `Data` section contains information about the storage folders of the simulation.

The `river_path` key specifies the path to the river data folder. This is also where the `rivers.toml` file must be located.

The `onnx_path` key specifies the path to the exported neural network model. Unless you retrained the neural networks and saved them to a different location, you do not need to change this value.

The `logging_root` key decides where all the logfiles of this configuration will be saved to. If the folder does not exist, it will be created. The logfiles will be saved in a subfolder named after the current date, time and number of vessels simulated. This will be discussed in more detail in Section ....

2.3 Logging

Text-based logging PERSIST features a comprehensive logging system for every stage of the simulation. Upon starting the simulation, a new folder will be created in the `logging_root` folder specified in the configuration file. This folder will be named after the current date, time and number of vessels simulated.

> **Note:** The logging folder will be named `YYYY-MM-DD-HHMMSS_rivername_nvessels`

Inside this folder, a logfile will be created for every vessel simulated. The logfile contains information about the position, speed, power, and overtaking level of the vessel at each time step. The logfiles are saved in the csv format, which can be opened in any spreadsheet software such as Microsoft Excel or LibreOffice Calc.

Additionally to the vessel logfiles, a copy of the configuration file, and all print outs to stdout (saved in run.log) will be saved in the logging folder.

Graphical logging For every completed simulation, PERSIST will generate a summary graph of the simulation. This graph contains a visual representation of the speeds, locations and overtaking levels of all vessels over the course of the simulation.

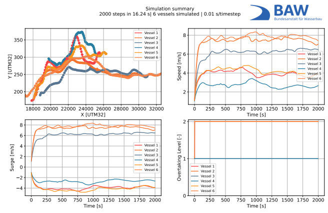

Figure 4: Graphical logging example.

We encourage you to adapt the summary plot to your needs. The function is called in the main routine at

./src/persist/main.py:129

and is implemented using the matplotlib library.[9]

[9] For example, it may be useful to pass the river object to the summary plotter, so that the river can be plotted in the background alongside the positions of the vessels.

2.4 Running PERSIST

To run PERSIST, simply call it from the command line:

```
$ persist -c <config_file>
```

where <config_file> is the path to the configuration file.[10]

[10] To avoid file location issues, we recommend to always use absolute paths.

If you want to run a demo simulation you call call PERSIST with the --demo flag:

```
$ persist --demo
```

> **Note:** Due to the lack of testing facilites, the calling of PERSIST as a self-contained script could not be tested on Windows machines. If you encounter any problems, please try running PERSIST as a Python module instead. Call it via:
>
> ```
> $ python -m persist -c <config_file>
> ```

Appendices

A Appendix Section

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam auctor mi risus, quis tempor libero hendrerit at. Duis hendrerit placerat quam et semper. Nam ultricies metus vehicula arcu viverra, vel ullamcorper justo elementum. Pellentesque vel mi ac lectus cursus posuere et nec ex. Fusce quis mauris egestas lacus commodo venenatis. Ut at arcu lectus. Donec et urna nunc. Morbi eu nisl cursus sapien eleifend tincidunt quis quis est. Donec ut orci ex. Praesent ligula enim, ullamcorper non lorem a, ultrices volutpat dolor. Nullam at imperdiet urna. Pellentesque nec velit eget euismod pretium.

B Appendix Section

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam auctor mi risus, quis tempor libero hendrerit at. Duis hendrerit placerat quam et semper. Nam ultricies metus vehicula arcu viverra, vel ullamcorper justo elementum. Pellentesque vel mi ac lectus cursus posuere et nec ex. Fusce quis mauris egestas lacus commodo venenatis. Ut at arcu lectus. Donec et urna nunc. Morbi eu nisl cursus sapien eleifend tincidunt quis quis est. Donec ut orci ex. Praesent ligula enim, ullamcorper non lorem a, ultrices volutpat dolor. Nullam at imperdiet urna. Pellentesque nec velit eget euismod pretium.

C Appendix Section

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam auctor mi risus, quis tempor libero hendrerit at. Duis hendrerit placerat quam et semper. Nam ultricies metus vehicula arcu viverra, vel ullamcorper justo elementum. Pellentesque vel mi ac lectus cursus posuere et nec ex. Fusce quis mauris egestas lacus commodo venenatis. Ut at arcu lectus. Donec et urna nunc. Morbi eu nisl cursus sapien eleifend tincidunt quis quis est. Donec ut orci ex. Praesent ligula enim, ullamcorper non lorem a, ultrices volutpat dolor. Nullam at imperdiet urna. Pellentesque nec velit eget euismod pretium.