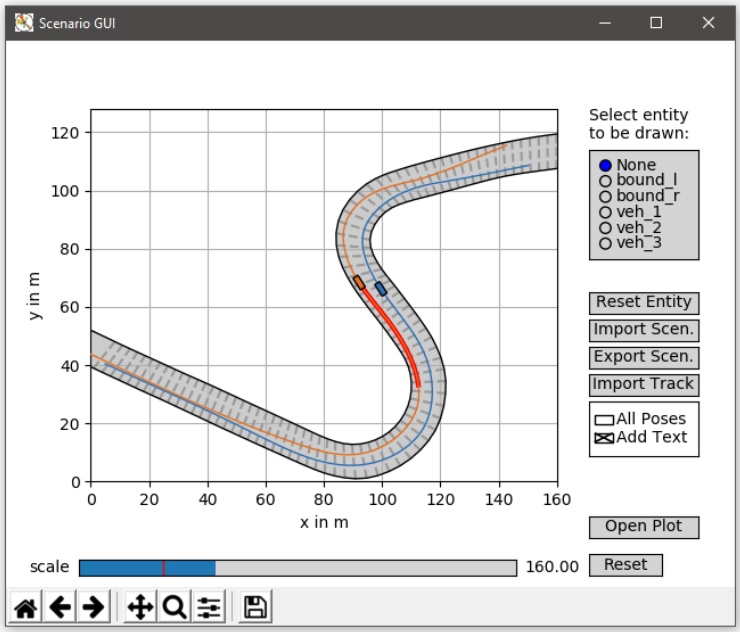
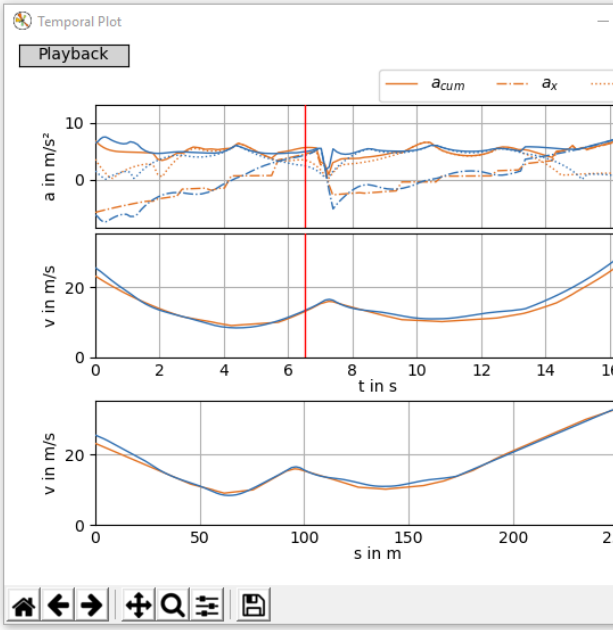
Scenario Architect

Release 0.0.2

Jul 12, 2021

Getting Started:





The Scenario Architect is a basic python tool to generate, import and export short concrete scenario intervals. These scenarios can then be used for safety assessment metric evaluations or other benchmarks.

CHAPTER 1

Repository Overview

The repository is composed of different components. Each of them is contained in a sub-folder.

Folder	Description
helper_funcs	This python module contains various helper functions used in several other functions, e.g. calculation of splines or velocity profiles.
params	This folder holds parameters configuring the Scenario Architect (e.g. export time increments, color styles, vehicle parameters for the automatic velocity-profile initialization)
interface_package	This folder holds some python script snippets, which come in handy, when working with the generated scenarios (e.g. extracting a specific time instance from the file).
sample_files	This folder holds a couple of sample scenarios generated with the Scenario Architect, as well as an sample track-file that can be imported with the Scenario Architect.

In the root folder is the *ScenarioGUI.py*-file located. This python class holds the main graphical user interface (GUI) used for the creation and modification of scenarios.

CHAPTER 2

Installation

This is a brief tutorial how to setup your computer to execute the Scenario Architect.

All the code was tested with **Python 3.7**, newer versions should also work. The code was tested for compliance with **Linux (Ubuntu)** and **Windows** machines.

Note: Reports from macOS users suggest to use **Python 3.8**, since the interactive features of the matplotlib library do not work properly under older versions.

Use the provided 'requirements.txt' in the root directory of the repository, in order to install all required modules.

```
pip3 install -r /path/to/requirements.txt
```

CHAPTER 3

Launching the Software

3.1 Launching the Scenario Architect GUI

In order to launch the main graphical editor, launch the *ScenarioGUI.py* script, located in the root-folder:

```
python3 ScenarioGUI.py
```

3.2 (Optional) Open a scenario file via command line

In order to launch the main graphical editor with a certain scenario file pre-loaded, launch the *ScenarioGUI.py* script appended by the path to the Scenario Architect archive file ('*.saa'):

```
python3 ScenarioGUI.py "c:\path\to\your\scenario-file.saa"
```

Hint: Replace the exemplary path by the absolute path to your Scenario Architect archive file.

3.3 (Optional) Associate scenario files with the Scenario Architect

On Windows machines, you can associate the Scenario Architect archive files ('*.saa') with the Scenario Architect GUI. That way, a double left-click with the pointing device on any stored scenario will open the scenario in the Scenario Architect.

In order to do so, create a new batch-file ('*.bat') at your desired location on your machine. The file should host the following content:

```
1 @ECHO OFF
2 SET "log_path=%*"
3 ECHO %log_path:\=/%
4 "C:\Python\Python37\python.exe" C:\scenario-architect\ScenarioGUI.py %log_path:\=/%
5 pause
```

Hint:

- Replace the first path in line 4 with the absolute path to your python installation.
- Replace the second path in line 4 with the absolute path to the 'ScenarioGUI.py'-file on your machine.

Afterwards, double left-click any scenario-file ('*.saa') with your mouse and select the created bash-file as default application to open this type of files.

3.4 Common problems

When launching directly from the PyCharm IDE make sure to **disable** 'Python Scientific'. Otherwise, the GUI will be displayed in form of a static image, not allowing for interaction.

CHAPTER 4

Basics

4.1 Selecting entities

Use the entity selector (highlighted red in Figure 1) to select a entity to be drawn or edited with the mouse. Afterwards, use single clicks in the main axis to draw any desired shape. End editing a entity by double-clicking or selecting any other entity in the entity selector. In order to remove all existing points of a certain entity, first select the entry in the entity selector and then click the button *Reset Entity*.

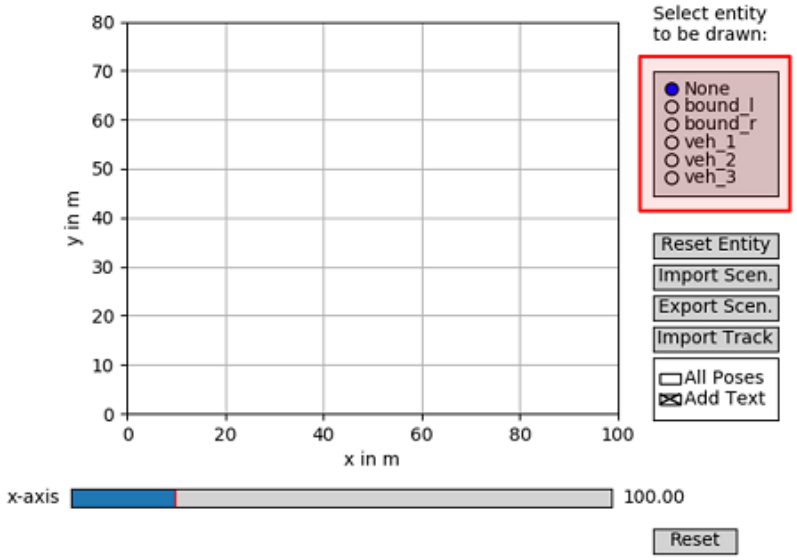


Figure 1: Scenario GUI window.

4.2 Drawing entities

Use these basic manipulation techniques to establish track bounds (left and right) as well as vehicle paths. A track is defined by a left and right bound, each holding pairwise coordinates (indicated by a dashed line between them). The overall procedure is demonstrated in the animation below.

Animation 1: Path and bound creation.

4.3 Manipulating existing entities

When hovering above an 'x'-tick of the highlighted data-points (selected via the entity selector) in the main window, you can left-click and drag the point to move it to a new position or right-click it to remove the point from the path. The procedures are visualized in the animation below.

Animation 2: Path modification - point dragging first, point deletion second.

CHAPTER 5

Import and Export

5.1 Export a scenario

The drawn scenario can be exported using the button *Export Scen.*. This process will generate a Scenario Architect archive file at the specified location. The generated file is a zip-like structure that holds the following data-files:

- Scenario dataset ('*.scn'): csv-flavoured file, holding all relevant data to be processed by any other software.
- Scenario pickle ('*.sas'): Pickle file, which can be directly imported by the GUI in order to modify any entity.
- Vehicle parameters ('*_ggv.csv' and '*_ax_max_machines.csv') holding associated vehicle parameters (when exporting, the currently loaded (from another '*.saa'-file) or in the 'params'-folder defined parameters are stored)

Hint: In order to export a scenario, at least the following entities have to be defined:

- Bounds (bound_l, bound_r)
- Trajectory of ego-vehicle (veh_1)

Hint: When generating a scenario, the ego vehicles trajectory (veh_1) is the characteristic one for the data export. Please also consider, that the ego vehicle holds a planned trajectory, which is exported to the scenario file. Therefore, the actually driven trajectory ends earlier (according to the specified trajectory preview horizon).

Other vehicles, holding a (temporally) longer trajectory compared to the ego vehicle, are cut at the position of the last ego-vehicle's data point.

Hint: It is possible to unzip the Scenario Architect archive ('*.saa') to obtain the individual files listed above. However, we recommend using the zipped structure in order to keep all files at hand.

5.2 Import a scenario

Use the button *Import Scen.* in order to import a Scenario Architect archive ('*.saa') or stored scene pickle ('*.sas').

Exemplary scenarios generated with the Scenario Architect can be found in the folder 'sample_files/scenario_1_vehicle' for the ego vehicle only and 'sample_files/scenario_n_vehicle' for multiple vehicles.

Hint: If a Scenario Architect archive ('*.saa') with stored vehicle parameters is loaded, the parameters are compared to the local parameters (in the 'params'-folder). When the parameter sets differ, the user is asked to select which of the two sets should be used.

5.3 Importing a track

The button *Import Track* allows to import stored track information (bounds). This function supports appropriate '*.csv' files (delimiter ';' or ',' - detected automatically). When importing a file, the user can specify the range of s-coordinates (counted from the first data point in the file) to be imported. If the start s-coordinate is larger than the end s-coordinate, the area crossing the start / end of the file is imported (in this case a closed track is assumed).

The headers in the first line of the csv-file (beginning with a '#') must at least hold one of the following list of entries:

- `x_m;y_m;w_tr_right_m;w_tr_left_m` (track-bounds will be imported)
- `x_ref_m;y_ref_m;width_right_m;width_left_m;x_normvec_m;y_normvec_m;alpha_mincurv_m;s_raceline_m` (track-bounds and ego-path will be imported)

A sample track of the first type is provided in the folder 'sample_files/sample_track'. Further supported tracks - extracted from real racetracks - can be found here in the folder 'tracks'.

CHAPTER 6

Scaling and Workspace Manipulation

Use the standard plot tools (pane, zoom) to change the zoom-level of the main axis

Furthermore, you can use the slider on the bottom of the 'Scenario GUI' window in order to zoom further out.

Hint: The default scale, as well as the maximum scaling value can be adjusted via configuration parameters. For further details, check the corresponding documentation (*Configuration*).

CHAPTER 7

Visualizing and Manipulating Temporal Information

Once you start drawing any path for one of the vehicles, the temporal information is automatically generated in parallel (see Animation 1). Thereby, the velocity profile is calculated in a way to drive as fast as possible on the drawn path. Any desired modifications can be inserted manually.

Animation 1: Path and bound creation.

7.1 Visualize a specific timestamp

When moving the cursor above the temporal window, the corresponding vehicle poses are highlighted in the main window, as shown in the animation below. That way it is possible to visualize the vehicle arrangement at any timestamp.

Animation 2: Visualize temporal states.

7.2 Modify the velocity profile of an entity

In order to modify the velocity of a certain vehicle manually, first select the vehicle in the 'Scenario GUI' window. The corresponding temporal plots in the 'Temporal Plot' window will be highlighted with a bold stroke. Furthermore, the spacial velocity profile will hold black dots that can be dragged up and down with the mouse in order to alter the velocity profile. When holding and dragging a point sideways it is possible to batch process multiple points in a linear manner. The velocity and acceleration profile is updated, once the mouse button is released. This process is demonstrated in the animation below.

Animation 3: Velocity profile manipulation.

7.3 Static visualization of temporal information

Use the checkboxes ‘All Poses’ and ‘Add Text’ in the main window to display poses of the vehicles with a fixed temporal spacing (default: every 1s - change this parameter in the config). Thereby, the ‘Add Text’ checkbox toggles the text description next to every pose (NOTE: Due to performance issues, the text is displayed only for vehicles not currently selected in the entity selector).

In order to plot the information displayed in the main window, it is possible to open with the button ‘Open Plot’ an external window with the plot axis only (all GUI elements are removed) - e.g. useful for presentations or publications.

CHAPTER 8

Working with the Automatically Generated Safety Rating

Once you start drawing any path for one of the vehicles, a safety rating for the given scenario is automatically generated in the temporal plot window.

On one hand, the safety w.r.t. a static environment (static safety) is evaluated, where intersections with the track limits, acceleration exceedance or kinematic constraint violations are considered for the score.

On the other hand, the safety w.r.t. a dynamic environment (dynamic safety) is determined. In this favor, Surrogate Safety Metrics (SSM) are evaluated along the course. Currently the Time to Collision (TTC) [dashed line in Fig. 1] and Difference of Space distance and Stopping distance (DSS) [solid line in Fig. 2] are implemented. In the config file, it is possible to specify which of the two should be used for the dynamic safety rating. The final rating is generated based on configurable theshold (e.g. <0m for the DSS) and a check for lateral overlaps in the lane-based coordinate system. Further details about the scores can be found in S. Mahmud et al. (S. M. S. Mahmud, L. Ferreira, Md. S. Hoque, and A. Tavassoli, "Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs," IATSS Research, vol. 41, no. 4, pp. 153–163, Dec. 2017, doi: 10/gf9kms.)

The resulting score for the static and dynamic safety along the course of the ego trajectory is visualized with green and orange bar segments. Furthermore, the score is also exported with every time-stamp.

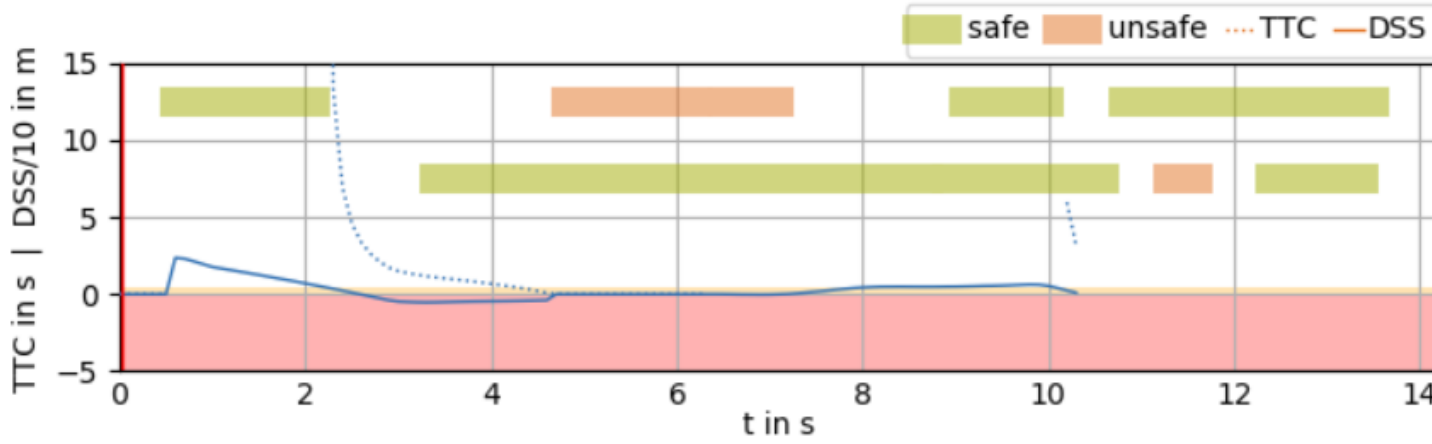


Figure 1: Safety rating based on surrogate safety metrics.

CHAPTER 9

Configuration

Vehicle dimensions, trajectory discretization and visualization cues (e.g. coloring) can be altered in the config file 'params/config.ini'. The config-file holds a detailed explanation for each of the parameters.

The dynamical behavior of the vehicle for the initially generated velocity profile can be adjusted with the files in the 'params/veh_dyn_info'-folder.

- The 'ax_max_machines.csv'-file describes the acceleration resources of the motor at certain velocity thresholds (values in between are interpolated linearly). The first column holds velocity values and the second column the possible acceleration at that velocity.
- The 'ggv.csv'-file describes the available friction based longitudinal and lateral acceleration at certain velocity thresholds (values in between are interpolated linearly). The first column holds velocity values, the second and third column the maximal longitudinal and lateral acceleration at that velocity.

CHAPTER 10

Using Exported Scenarios in Your Project

The generated '*.scn'-files in the Scenario Architect archive '*.saa' hold a CSV-flavoured listing of all relevant variables. The first two lines specify the surrounding boundaries of the track. The third line holds the headers for each data column (each separated with a semicolon). All subsequent lines hold each a snapshot of the scene at a certain time instance.

10.1 Retrieving data for a specific timestamp

An exemplary function to extract a certain time-stamp from any '*.saa' or '*.scn' file can be found in the 'interface_package/scenario_testing_tools'-folder. The function '*get_scene_timestample()*' ('get_scene_timestample.py') takes as input the path to a '*.saa' or '*.scn' file, paired with a timestamp and returns all relevant variables at the given timestamp. The function is set up in a way, that the given timestamp can be given in form of an integer as well as a floating point value. When an integer is given, the respective data recording in the file is returned. When a floating point value is given, the values at the time stamp (usually measured in seconds relative to the start of the scenario) is returned. When the given time instance is not present in the file directly, linear interpolation is used to generate the values between the neighboring occurrences.

10.2 Retrieving track / map information

A sample function to extract the track bounds from any '*.saa' or '*.scn' file is given with the function '*get_scene_track()*' in the 'interface_package/scenario_testing_tools/get_scene_track.py'-file (same folder as above). It takes the path to a '*.saa' or '*.scn' file as input and returns two numpy arrays, holding the coordinates of the left and right bound, respectively. The file also offers a second function '*get_scene_occupancy()*', which generates a basic occupancy grid with parameterizable resolution and framing. For further details, refer to the function's header documentation.

A function to generate a reference-line, matching normal-vectors and track-widths is given with the function '*generate_refline()*' ('interface_package/scenario_testing_tools/generate_refline.py'). The function takes the coordinate-arrays of the left and right bound (plus an option resolution for the resulting reference line).

10.3 Retrieving vehicle dynamic parameters

A function to extract the vehicle dynamic parameters (ggv and machine limitations) is given with the function '*get_scene_veh_param()*' in the 'interface_package/scenario_testing_tools/get_scene_veh_param.py'-file (same folder as above). It takes the path to a '*.saa' file as input and returns two numpy arrays, holding the ggv-diagram and maximum machine accelerations (details about the files in the /../using_the_GUI/content/config).

10.4 Implementation guide

In order to use these functions (functions within the 'interface_package/scenario_testing_tools'-folder) in any other repository, simply install the package with the command:

```
pip3 install scenario-testing-tools
```

In order to use the function, import the package and use the function as usual, e.g.:

```
import scenario_testing_tools as stt
```

In order to extract a certain timestamp, insert the following line (replace input parameters):

```
time, pos, heading, curv, vel, acc, ego_traj, object_array, time_f = \
    stt.get_scene_timesample.get_scene_timesample(file_path=YOUR_PATH, t_in=1.234)
```

In order to extract a certain timestamp with its expected safety rating [True, False, None], insert the following line (replace input parameters):

```
time, pos, heading, curv, vel, acc, ego_traj, ego_traj_em, object_list, time_f,␣
↪safety_dyn, safety_stat = \
    stt.get_scene_timesample.get_scene_timesample(file_path=YOUR_PATH, t_in=1.234,␣
↪append_safety=True)
```

Hint: In order to emulate real-time playback on a vehcile, keep track of the passed time and call the function above iteratively, e.g.:

```
import time
tic = time.time()

while True:
    return_tuple = stt.get_scene_timesample.get_scene_timesample(file_path=YOUR_PATH,
                                                                 t_in=time.time() -␣
↪tic)
```

In order to extract the bound-arrays use the following function:

```
bound_l, bound_r = stt.get_scene_track.get_scene_track(file_path=YOUR_PATH)
```

The bound-arrays can be used to retrieve an occupancy grid and / or a reference line with matching normal-vectors (e.g. useful for lane-based coordinate systems):

```
occ_grid, origin = stt.get_scene_track.get_scene_occupancy(bound_l=bound_l, bound_
↪r=bound_r)
```

(continues on next page)

(continued from previous page)

```
ref_line, normal_vectots, tw_left, tw_right = \
    stt.generate_refline.generate_refline(bound_l=<TODO>, bound_r=<TODO>)
```

Hint: All functions in this package are designed in a way to cope with Scenario Architect archives ('*.saa') as well as with scenario files ('*.scn').

CHAPTER 11

People Involved

11.1 Core Developer

- Tim Stahl

11.2 Acknowledgements

Regina Harrer contributed during her Project Thesis to the calculation of the Time to Collision (TTC) score.

CHAPTER 12

Contributions

[1] T. Stahl and J. Betz, "An Open-Source Scenario Architect for Autonomous Vehicles," in 2020 Fifteenth International Conference on Ecological Vehicles and Renewable Energies (EVER), 2020. (view pre-print)

If you find our work useful in your research, please consider citing:

```
@inproceedings{stahl2020a,
  title = {An Open-Source Scenario Architect for Autonomous Vehicles},
  booktitle = {2020 Fifteenth International Conference on Ecological Vehicles and␣
↪Renewable Energies (EVER)},
  author = {Stahl, Tim and Betz, Johannes},
  year = {2020}
}
```