Interactive Visual Analysis of Anomalies in Simulations of Energy Flow

Fabrizia Bando-Bechtold*
Kresimir Matkovic (supervisor)†

VRVis Research Center, Vienna, Austria

Abstract

At a time when the demand for alternatives to gas and oil is steadily increasing, the automotive industry faces the challenge of designing vehicle systems, which are as efficient as possible. Vehicle system simulation is used for concept analysis, subsystem design, and virtual component integration. The high-level simulation model accurately depicts the energy flow between different vehicle parts over an extended period of time. The the resulting data is presented as a time-dependent edge-weighted graph. The analysis and exploration of such data is a tedious task.

We propose a novel approach for exploring and analyzing energy flow data at different levels of detail to assist engineers and guide them to the events of interest. We employ automatic anomaly detection techniques and propose intuitive navigation to time steps of interest. We also propose expandable cards or labels that depict current and overall data. Users can interactively and dynamically choose how much information is displayed on the labels by changing their level of detail. We provide four levels of detail, each giving more information than the previous level. The first level depicts the current amount of energy; the second level shows sparklines for the energy flow over the entire time interval; the third level shows a timeline of anomaly occurrences; level four shows the sparkline zoomed around the current time frame; finally, level five depicts expands the label to a large display of the previous four levels of detail of the flow data for the selected element.

The new approach is implemented as an interactive web application. We are currently evaluating it with domain experts. Since the initial feedback is very positive, we expect rapid adoption of the newly proposed approach by automotive industry professionals.

Keywords: Visual Analytics, Anomaly Detection, Flow Graph Visualization

*bechtold@vrvis.at
†matkovic@vrvis.at

1 Introduction

Creating and designing a vehicle is a difficult task in and of itself. Consumer concerns about the environment and the high operating costs of traditional fuel-powered vehicles exacerbate the problem even further. Engineers working in the automotive industry must therefore design systems for cars that are both cost-effective and environmentally responsible.

Once a system design is finished and before an actual vehicle is built, engineers want to know how the potential system performs during a drive. They want to collect and analyse data extracted from drives which are longer or shorter and mimic different geographic and environmental conditions. Driving on a hilly road or through snow, for instance, is different from driving on a highway in good weather. The car behaves differently in a traffic jam than it does on a smooth ride. The energy flowing between the system's components is calculated over time for each of these various settings. The analysis problem for hybrid cars becomes even more difficult, because some system components, like the battery, can be both sources and sinks of energy flow during a single driving simulation. The potentially large cardinality of the set of parameters alone makes the analysis task intimidating. Engineers want to collect this data for various driving cycles, which is not feasible to do from real life test drives. Therefore, in the design phase of a vehicle, when engineers carefully examine potential system component layouts, simulation plays a significant role.

A simulation is a technique that mimics a real-world system or process and tracks its evolution [2][17]. It is typically run on a computer and is based on a model - often mathematical - displaying key characteristics of a process. Simulations are frequently used when a real-world process, such as monitoring car system components during a test drive, is too complex and expensive to build or provide analytical solutions for. Computer simulation is also used in vehicle engineering to improve the aerodynamic properties of a component [12], the turbulent combustion system [20], the influence of tail structure on the rear field [13].

While simulations are frequently used in research and design, analyzing such data can be time-consuming and challenging, particularly when dealing with multivari-

ate time-dependent data. As a result, linear analysis of such time-dependent data takes a long time and is labor-intensive.

We propose an analysis system, that can reduce the engineers workload by conducting automatic anomaly detection on the data and visually guiding them to incidents, which require more thorough investigation. The goal of anomaly detection is to identify events that differ from the norm [15][16]. Anomaly detection is often accomplished by employing AI and training a model on a particular test data set. Our approach employs anomaly detection techniques, which are only dependent on the input data. This has the advantage of being applicable to any given time-series without the need for humans to identify anomalous occurrences beforehand. We use anomaly detection to direct the engineers' attention to specific spots in the simulated test drive. The engineer might focus on the time frame around the occurrence to assess whether or not the system is behaving strangely.

Research on the effectiveness of visualizations shows, that visualizations exploit a humans' cognitive capabilities and therefore increase the users performance for spotting anomalous behavior [14]. Visual Analytics, especially when guided, [6][18] supports humans in identifying patterns and points of interest, discover previously unknown connections, gaining fresh insight and making data-driven decisions. Interactivity can further enhance the users performance.

Interactive visualization has become a well-established tool to support engineers in the exploration and analysis of complex data. We, therefore, propose an interactive visualization approach to further support engineers in the analysis of energy flow data. In a collaboration with domain experts, we designed a novel visual analysis system to support engineers in their work flow. The new approach unifies automatic anomaly detection and interactive visualization to guide engineers to events of interest.

Our proposed system can be generalized as a flow graph. It consists of nodes, which represent the various car's components, and edges, which represent the energy flowing between the components. In order to cope with the complexity of the data, we abstract and give an overview of the data in carefully designed visualization elements. Additionally we provide the user with the option to increase the displayed information, by preparing various visualizations showing different details and only displaying them on demand. The detected anomalies are displayed in a timeline, giving a summary of the anomalies.

The summarized main contributions of this paper are:

1. interactive visualization approach for the analysis of energy flow data,
2. augmented and extended flow graph visualization
3. anomaly detection to guide users to incidents of interest.

2 Related Work

Our proposed application is closely related to the existing software **AVL CRUISE™ M**[1][7] developed by **AVL List GmbH ("AVL")**. The software is used to design a car's system and then simualte diverse test runs. After the simulation, engineers can analyse the behaviour of the system by watching a replay, which shows a simple graph consisting of all the car's components as nodes and the flow of energy between them as edges. The current energy value of a component is showcased as two scalar values next to the component as incoming flow and outgoing flow, which is also visually encoded by the thickness of the edge. The dynamic flow is further depicted by huge arrows moving along the edge, with the arrow head facing the direction of the flow, which can change during the drive. Due to the redundant information incoming and outgoing labels, the screen gets easily cluttered. We plan to solve this by encoding all information in one label and decreasing the size and quantity of the arrows. The visualization itself is static and the user is unable to query further information. We plan to implement diverse interactions such as various playback modes, and interaction with and selection of the components. The biggest limitation of the original software, though, is the linear playback mode without any indication of where and when during the simulation incidents of interest happened. For this we plan extend the visualization with automatic anomaly detection and displaying them on a time line. This allows the user to only watch the important parts of the visualization.

Matković et al. [11] introduce diverse visualization techniques to process monitoring, namely history encoding, multi-instruments and level of detail. Additionally to displaying the current value, the authors encode values of the near past into their visualization (history encoding) by adapting a bar chart. Instead of encoding the current value as a bar, which takes up a whole column or row, they encode the value as a heavily saturated line. Like this, they can add more data points by adding lines and gradually decreasing the saturation for points further in the past. This feature could be useful for our application to show the recent change in energy flow to give the user an idea of how the values changed, especially around an anomaly. Matković et al. display several data values within one virtual instrument, to easily compare related values and quickly spot divergences (multi-instruments). Another advantage of this approach is that it saves screen space, allowing for showcasing a lot of information in a condensed and easy to grasp manner. We make use of this feature in our anomaly timeline (see Section 5) by displaying different types of anomalies as a stacked one-dimensional scatter chart. Focus and context approaches

[1] https://www.avl.com/cruise-m

coarsely represent the entire information in the available screen space and give different levels of detail for different degrees of interest. Users can see information with the highest degree of interest in the highest level of detail, and areas of low interest can be depicted as only a scalar value. We employ this focus and context approach for displaying condensed information of the flow of energy for each component. Users can manually set the level of detail for each component and see different visualizations, depending on the selected level.

Cambridge Intelligence developed a `JavaScript`-based toolkit for scalable timeline visualizations that reveal patterns in time data, mostly aimed at cyber security and fraud detection analysis. The main feature is a scalable timeline, which highlights events occurring between two entities. Its goal is to guide analysts in detecting patterns and abnormal behaviour in events. The graph consists of rows, each row dedicated to an entity. An event is encoded as a (directed) line between the two rows of the respective entities on a low zoom level and an aggreated heat map on a higher. The software can be extended to show a graph or network of the underlying data. Since the view of the graph and the timeline are linked, users can easily query subsets of the data. While the graph visualization seems powerful enough, it does not allow dynamic playback, or encoding aggregated information or statistics into the graph itself. The timeline is a powerful tool to investigate events, but it uses a lot of screen space. As our main focus is on the visualization of the graph, and only enhancing it with a timeline, this does not seem feasible. Though the aggregated timeline could be useful for future extensions of the **Energy Flow Explorer**. [5]

3 Energy Flow Data and Analysis Requirements

During a drive, a car is subjected to many different external influences. Temperature, weather, traffic and such affect the performance of the car's components and the whole system's behaviour. Testing and analysing the behaviour in the real world is not desirable due to unpredictable weather and difficult reproducibility. Furthermore, for the automotive industry simulations are vital to save costs and reduce time to market their products.

A driving simulation imitates the drive with a hybrid car from point A to point B under specific environmental circumstances, or with a given set of system parameters. For the simulation of our test data the software **AVL CRUISE™ M**[2][7] developed by **AVL List GmbH ("AVL")**[3] was used. It is a "multi-disciplinary vehicle system simulation tool", which allows for "Powertrain Concept Analysis","Control Function Development and Calibration", "Vehicle Simulation on Testbeds" as well as "Sub-system Analysis" [9]. The latter can be used to perform detailed design layouts and optimizations of subsystems. It serves as the basis for our proposed analysis tool, the **Energy Flow Explorer**.

The **AVL CRUISE™ M** software allows to simulate the flow of energy between the components of the hybrid car and is able to replay and show the amount of flow between components. Figure 1 depicts the simulation player of the software, where the "Mass flow [kg/s]" is distributed within a small test-set containing eight components: an energy source (Boundary 1) and a target (Boundary 2). The figure shows the direction and the amount of flow entering and leaving a component on a small label attached to each component. The view only allows for linear playback of the simulation and analysing the simulation at a specific point in time. The user has to watch the whole animation to analyse the simulation and is provided with only very limited information; no data aggregation or statistics are performed. The visualization also gives no indication of abnormal behaviour or the change of energy flow throughout the whole simulation.

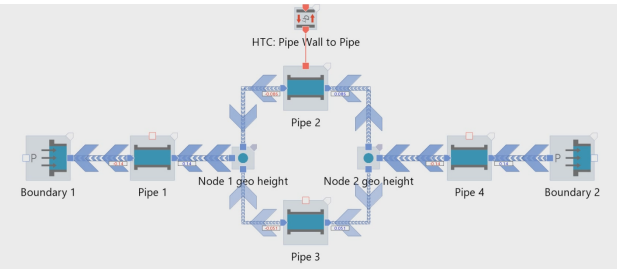

Figure 1: A frame captured from the replay of a small simulation using the software **AVL CRUISE™ M**. It shows the direction and amount of the energy flowing between components at the time of the capture.

The **Energy Flow Explorer** is an application with the goal to solve the previously stated limitations and provide better support for engineers from the automotive industry in their analysis work.

Before designing the architecture of the system, we thoroughly examined the analysis process together with domain experts and abstracted the following requirements:

- **R1 - Automatic Data Processing**: load multiple simulation results from different sources
- **R2 - Data Aggregation**: support different types of anomaly detection and data aggregation
- **R3 - Model Structure**: store system information in simple and easy to query format
- **R4 - Simulation Player**: show energy flow between components throughout simulation
- **R5 - Improved "Data-Ink Ratio"** [18]: improve data density and encoding in the visualization

[2]https://www.avl.com/cruise-m
[3]www.avl.com

- **R6 - Anomaly Visualization**: show anomalous behaviour in multiple levels of detail
- **R7 - Multi-Resolution Data Graphs** [18]: show different levels of detail on the data with overview first and more detail on demand

To conform to these tasks, the designed system has to fulfill the following criteria: read input data of different formats; have the possibility to perform data analysis techniques and data aggregation; perform queries on the data and the model itself; display various non-standard visualizations and provide user interaction. Since the system should support powerful data analysis as well as user interaction, a `client-server` architecture following the request-response model is employed:

1. *Data Handling & Analysis*: a python server is responsible for reading the source data, generating an easy-to-query model data structure, carrying out data analysis and responding to user queries (Requirements T1-T3).
2. *Data Visualization & User Interaction*: a JavaScript-based web-application displays the processed data in an interactive dashboard, where the user is able to freely manipulate and explore the data (Requirements T4-T7).

Section 4 describes the first part of the system *Data Handling & Analysis*, whereas the second part - *Data Visualization* - is described in section 5.

4 Data Handling & Analysis

Our application requires several types of input data. First, a graph model consisting of information about each component, such as physical properties, meta data and ports (connection between two components). Second, the used parameters for the simulation, which is required for selecting subsets in the data when comparing multiple simulations (see Section 6 - Future Work). Third, the actual flow data for each component for the whole simulation.6 - Future Work).)
We chose `Python`[4] due to its flexibility and vast data manipulation libraries. We employ a `Flask`[5] server as it enables easy creation of websites *Data Handling & Analysis*: Once the user selects a simulation case-set, the server is responsible for parsing the model data and generating the component-flow model before performing anomaly detection on each component and sending the processed data to the user.
Python provides multiple libraries for anomaly detection. We used PyCaret [1], which provides a flexible and extensive framework for anomaly detection of time-series. Here it is possible to choose from a multitude of machine learning based anomaly detection algorithms. We chose `Isolation Forest` or `IForest` [10], an unsupervised model. Using randomly chosen characteristics, an Isolation Forest processes randomly sub-sampled data in a tree structure. As they required more cuttings to separate, samples that travel further into the tree are less likely to include anomalies and samples that end up on shorter branches tend to be anomalies. The second approach for anomaly detection is a simple `Min-Max Threshold` model, where the user can set a percentile, at which data points below or above are marked as anomalous. Lastly, we chose to use the `Modified Z Score` [8]. A z-score in statistics indicates how many standard deviations a result deviates from the mean. However, unusually big or tiny data values can have an impact on z-scores, so using a modified z-score is a more reliable method of identifying outliers, as it is based on the median rather than the mean.

Engineers desire a system where they can analyze multiple simulation runs for the same component system but with different parameter settings at the same time and get a feeling for the special features of the system as well as its peculiarities and hidden correlations between components. These aspects were already considered for the design of the system architecture, but not yet fully realized at the publishing date of this paper.

The python server is split into the following components to perform these tasks:

- *DataFileParser*:
  The results generated by the **AVL CRUISE™ M** come in different formats - ranging from `.csv` and `.xml` to their own file format `.gid` - and vary greatly in their structure. The *DataFileParser* provides all necessary functionality of requirement **R1 - Automatic Data Processing** to retrieve data from all provided raw data files. Additionally it provides methods to persist and load the generated models and data to skip time-consuming processes such as performing anomaly detection - a resource-heavy task - in future analysis sessions of the same case-set.
- *CaseSetManager*:
  This manager is responsible for invoking all operations happening on the server. It functions as an organiser for the selected case-sets by loading the model data and passing it to the *ComponentManager*, where it is rearranged into a easy-to-query structure. It then gives the order to aggregate the data with the selected parameters and finally arranges all relevant data and information into `.json` format to send to the client.
- *ComponentManager*:
  The systems model consists of different units: `Components`, `Ports`, and `Connections` with additional information on the flow direction (OUTgoing, IN-coming or NEUTRAL).

[4] www.python.org/
[5] flask.palletsprojects.com

The *ComponentManger* stores all this information and provides methods to perform diverse queries on the components, such as selecting only `Components` associated with a specific flow, and re-structure the query result in different formats needed by the client and hence fulfilling requirement **R3 - Model Structure** (see Section 3 - 3).

- *DataAggregation*:
  To visualize more information than just the simulation results, the data needs to be processed and aggregated. *DataAggregation* is responsible for reading every component's data and storing the wrangled time-series data. It performs the above mentioned anomaly detection algorithms, which can be extended to any other outlier detection method. Additionally, data aggregation methods are available to calculate parameters such as the minimum, cumulative sum or integral, though these are mostly used for the exploration of multiple simulation ensembles (see section 6 - Future Work). This part of the system fulfills requirement **R2 - Data Aggregation** (see Section 3).

Once all the required data has been compiled, the server sends the data in `.JSON` format as a response to the client. The server then waits for requests from the client, and responds adequately with data updates or setting changes.

5 Data Visualization & User Interaction

The main motivation for our proposed application is to improve the existing simulation player used by domain experts to guide them to interesting incidents in the mimicked drive. Therefore, the minimum criteria for our visualization is to show the system component layout with its connections and an animation of the calculated results displayed with each component (requirement **R4 - Simulation Player** of Section 3). The original animation is linear, where users can change the playback speed. It also tends to clutter due to redundant information and spacious visualizations. To ease the analysis process domain experts desire more succinct information at one glance (requirement **R5 - Improved "Data-Ink Ratio"** of Section 3) and a visual feedback on where events of interest occurred during the simulation (requirement **R6 - Anomaly Visualization** of Section 3). They desire an overview of the simulation, but at the same time more detailed information in certain areas (requirement **R7 - Multi-Resolution Data Graphs** of Section 3). Lastly, the new application should be interactive, enabling the user to focus on specific areas, if desired.

For the client we opted for a web application, as it has the advantage of not requiring the user to download any specific software, is platform independent and many versatile data visualization libraries exist. We used `HTML` in combination with vanilla `JavaScript` and `CSS`. Additionally, we use `d3.js`, "a JavaScript library for manipulating documents based on data", as it is written in `JavaScript` and allows web developers "to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document" [4].

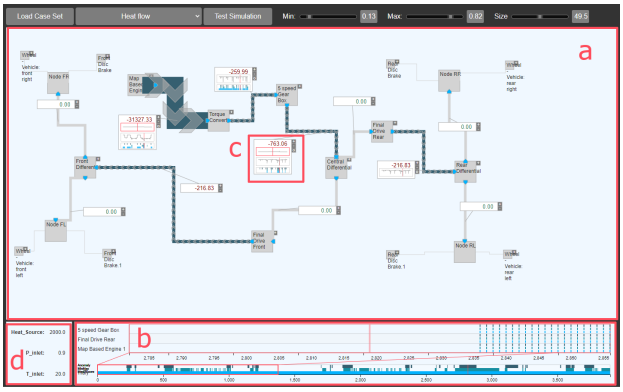

Figure 2: The proposed web application - **Energy Flow Explorer** with its four main windows. a) the *Simulation Player* shows the system layout, flows and *Data Label's*. b) the *Anomaly Timeline* depicts detected anomalies of the simulation. c) *Data Labels* give detailed information on demand. d) the *Parameter View* shows the applied parameters for the simulation.

To comply with all requirements we split the application into four visualization parts as indicated by the red rectangles in Figure 2: **a)** `Simulation Player`, **b)** `Anomaly Timeline`, **c)** `Data Label` and **d** `Parameter Table`.

First, the `Simulation Player` - the biggest visualization in Figure 2 **a** - shows the layout of used components in the selected simulation. Only the relevant components are shown as gray boxes, as not all components may be affected by the selected energy flow. Figure 2 **a**, e.g., shows the **Energy Flow Explorer**, where a model with 30 components was loaded but only 20 are visible, as only those have *'Heat Flow'*, the select flow parameter. Users can display the inactive components on demand. All components are connected by orthogonal flow lines, forming a planar graph drawing. A flow is defined as a connection between one component with outgoing flow and one with incoming flow, and displays the data from the outgoing component. The simulation software calculated a scalar value for each flow at every point in time during the simulated drive. These scalar values are reflected in the thickness of the flow lines; the thickness is interpolated between the minimum and maximum flow thickness, which can be set by the user. Although thickness is a visual encoding with low discernability of small changes ([3][19]), we use it here to give a visual feedback on the current en-

ergy distribution throughout the system, as size in general is a good visualizer for the quantitative data. In Figure 2 **a**, it is immediately visible, that there is a lot of energy flowing between two components in the upper left part, while only low amount of energy flows between the rest of the components. Here it is less important how big the change is between two frames, than how big the current value is. Additionally, the color of the flow is *gray*, if the current value is zero and *navy blue* otherwise. As the direction of the flow can change during the simulation, the current flow direction is indicated through animated half-transparent arrows along the flow line. These features realize requirement **R5 - Improved "Data-Ink Ratio"** (see Section 3).

The user can choose to animate the simulation, which complies with requirement **R4 - Simulation Player** (see Section 3). This updates the current flow value and consequently the size, color, and arrow position and direction

The `Anomaly Timeline` in Figure 2 **b** is dedicated to showing the points of interest, or anomalies, and is called `Anomaly Timeline`. As the name indicates, it shows at which point in time an anomaly occurred. The `Anomaly Timeline` has three rows, as we performed three different anomaly detection techniques (IForest anomaly detection, Min-Max Threshold, Modified Z Score), each dedicated to one technique. An anomalous point is indicated by a line and positioned relative to the point in time when it occurred in the simulation, with the left-most point translating to the start and the right-most to the end of the simulation. The `Anomaly Timeline` gives an overview of the anomalies of all components, as can be seen in Figure 6. If the user wants to see the anomalies of a specific component, they can add the component to the timeline. The associated anomalies will then appear on top of the summary timeline. In Figure 2 **b** three components have been added. This component-based timeline shows not every anomaly of a component, but only those within a specified time frame, i.e. x frames before and after the current time stamp, creating a zoomed anomaly timeline. A zoom-window linked to the time frame appears on the summary timeline as soon as one component has been added to the zoomed timeline (see Figure 3). The `Anomaly Timeline` complies to requirements **R6 - Anomaly Visualization** and **R7 - Multi-Resolution Data Graphs** (see Section 3) by showing more details of anomalies when requested. The user can move the zoom-window along the timeline, as well as change the size. A mouse-over effect reveals the anomaly type, component and time-stamp of an anomaly. The mouse-over visually links the referenced component by highlighting the corresponding `Data Label` in red for a short time.

A `Data Label` is an interactive and responsive visualization inherent to the **Energy Flow Explorer**. Each flow has a `Data Label`, which is an expandable graph, showing diverse information of the flow at multiple levels of detail (LOD) (see in Figure 2 **c**). The label always shows the scalar value associated with the current time stamp (see the first `Data Label` in Figure 4). The scalar value is either green or red, depending on if the difference in change since the last time stamp was positive or negative. The user can change the level of detail of the `Data Label` by clicking on the **[+]** button or **[-]** button on the right side. Showing the scalar value is level of detail one (LOD 1). LOD 2 shows a *Sparkline* of the whole simulation results for the associated component. This gives the user an overview of the components behaviour. Increasing the level of detail to 3 reveals an `Anomaly Timeline`, showing the anomalies exhibited by the component. A final increase in LOD shows a zoomed sparkline, centered around the current time stamp, with the same time frame as the zoomed timeline in visualization **b** - `Anomaly Timeline`. When playing the simulation, this time frame synchronizes with the zoom window and therefore stays always up-to-date. The right-most graph in Figure 4 depicts a `Data Label` at LOD 4, showing the scalar value, the zoomed sparkline, overview sparkline and anomalies. To not clutter the whole visualization, these `Data Labels` are kept small to give an overview, which can impede the analysis. To counter that, the user can enlarge a `Data Label` so that it gives more room and details to the graphs. Figure 5 shows the enlarged view of component *'Map Based Engine'*. In the future we plan to provide a pin feature, where the user can select maxed `Data Labels` and pin a smaller version of them to the side of the explorer, similar to "sticky notes". Future work might also include an extension of displayed information, such as multiple spark lines, which show additional statistical features e.g. the first derivative.

The user can choose to change the LOD for selected components, or globally set the LOD in the settings menu. In Figure 6 the global LOD level has been set to 4, the maximum. This change in level of detail complies with requirements **R5, R6** and **R7** (see Section 3).

The last part of the web application, the `Parameter Table` (see Figure 2 **d**), is dedicated to the parameters used in the simulation, as domain experts want to examine the same model with different parameters set. It shows the used parameters and corresponding values. When analysing ensembles of simulations the user will be able to see a summary here, as well as have the ability to brush the data here, i.e. select subsets of the simulations.

The user is able to animate the changes of values throughout the simulation in a linear manner with variable playback speed and pause whenever an anomaly is hit. Still, watching the whole simulation, even at increased speed, can take a long time, depending on the length of the simulated drive. To avoid this, we provide a play-

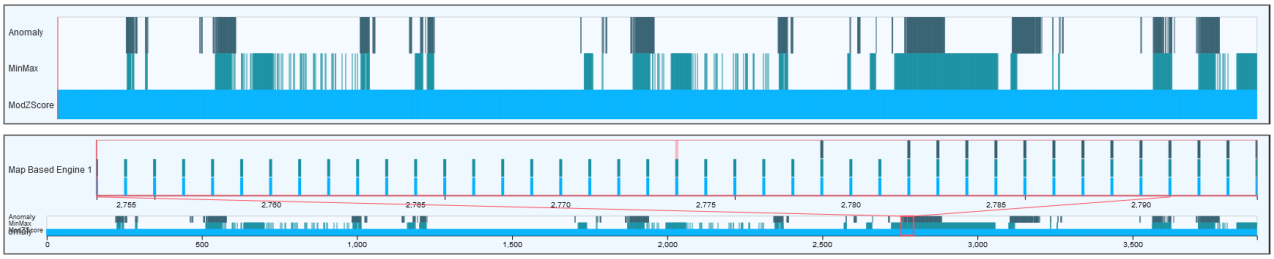

Figure 3: The *Anomaly Timeline* displays the anomalies, subdivided by anomaly type; one row each. A line indicates the occurrence of an anomaly somewhere in the system at the given point in time. The top figure shows the summarized *Anomaly Timeline* showing all anomalies of the system, subdivided by anomaly types Iforest Anomaly, MinMax Threshold and Modified Z Score. The second figure shows on the bottom the summarized timeline, and on top the timeline of a specific, user selected, component. Here the timeline is zoomed according to the zoom window (indicated by the orange rectangle in the summarized timeline) and shows all three anomaly types that arose in this component.

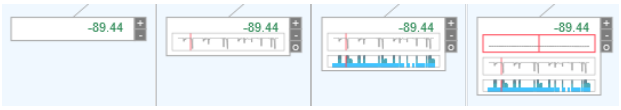

Figure 4: A *Data Label* showing its four levels of detail (LOD). First, on the left, only the current energy flow value is shown (LOD 1). Next to it, the label is extended to LOD 2, showing the whole, unzoomed sparkline. For LOD 3, the third *Data Label*, the anomaly timeline corresponding to this component is added. The right-most *Data Label* is extended by a zoomed version of the sparkline, centered around the current time stamp (LOD 4).

back option to accelerate the animation until the current time frame is near an anomaly. Before hitting the anomalous point, the animation slows down, so that the user can carefully watch the following time frames and examine the changes. The user can choose that, when hitting an anomaly, the `Data Label` associated with the component where the anomaly arose automatically expands, creating a visual link to the anomaly occurrence in the system.

6 Conclusions & Future Work

A vehicle undergoes many cycles of evaluation and adaptation before it can be cleared for production. The evaluation process is based on several simulations of a drive through various terrains and under divers conditions. The analysis process is a tedious task and current systems leave engineers wanting. We propose an interactive web application, which alleviates this task by aggregating the simulation results as well as performing anomaly detection techniques on it in a python-based server. The processed data is displayed in an interactive web visualization with four main views: the *Simulation Player* showing the system layout, giving playback control and encoding additional information in the flows (lines connecting two system components). The *Anomaly Timeline* turns the users

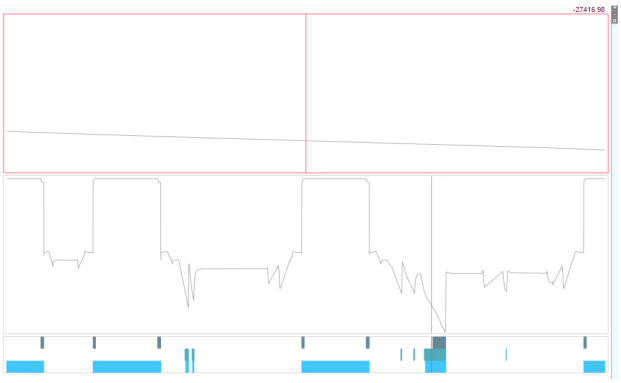

Figure 5: The **Energy Flow Explorer** with a selected Data Label enlarged, giving more room and revealing more details of how the component was affected throughout the simulation.

attention to points of interest in the simulation. The *Data Labels* give the user additional information on a flow and component at multiple levels of detail, if desired. The parameters used for the simulation are depicted in the *Parameter View*. The whole application is highly interactive and gives the user an overview as well as detailed information on demand and non-linear animation of the simulation.

The system is currently being extended to cater to the analysis of multiple simulations concurrently, where the user will also be able to further investigate subsets of selected simulations brushing and linking. We also plan to include a visualization showing the balance of energy flowing into a component versus leaving it, revealing loss or gain in energy. We plan on doing a thorough evaluation together with domain experts, but first feedback has been very positive.

7 Acknowledgements

VRVis is funded by BMK, BMDW, Styria, SFG, Tyrol and Vienna Business Agency in the scope of COMET -

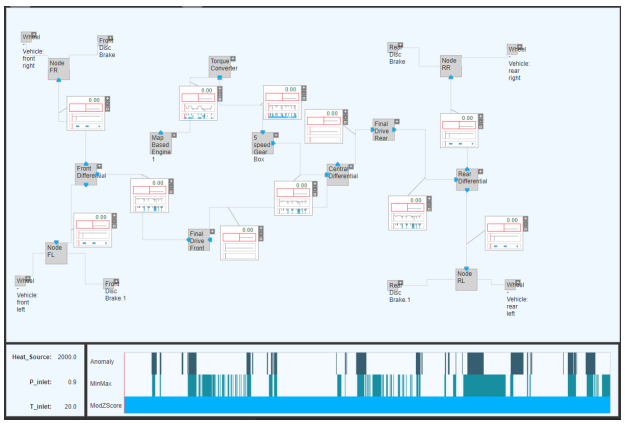

Figure 6: The **Energy Flow Explorer** showing the system components and associated *Data Labels* at the highest level of detail. Below the *Parameter View* and summarized *Anomaly Timeline*

Competence Centers for Excellent Technologies (879730) which is managed by FFG.

References

[1] Moez Ali. *PyCaret: An open source, low-code machine learning library in Python*, April 2020. PyCaret version 1.0.0.

[2] Jerry Banks, John Carson, Barry L. Nelson, and David Nicol. *Discrete-Event System Simulation (4th Edition)*. Prentice Hall, 4 edition, December 2004.

[3] J. Bertin and W. Berg. *Semiology of Graphics*. University of Wisconsin Press, 1983.

[4] Mike Bostock. Data-driven documents. `https://web.archive.org/web/20201031193629/https://d3js.org/`. [Online; accessed 14.02.2023].

[5] Cambridge Intelligence. KronoGraph.

[6] Brian Fisher. *Illuminating the Path: An R&D Agenda for Visual Analytics*, pages 69–104. National Visualization and Analytics Ctr (January 1, 2005), 01 2005.

[7] Wolfram Hasewend. AVL CRUISE. *ATZ Automobiltech. Z.*, 103(5):382–392, May 2001.

[8] B. Iglewicz and D.C. Hoaglin. *How to Detect and Handle Outliers*. ASQC basic references in quality control. ASQC Quality Press, 1993.

[9] Ingo Lütkebohle. AVL CRUISE™ M. `https://www.avl.com/cruise-m`, 2009. [Online; accessed 27.01.2023].

[10] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[11] Krešimir Matković, Helwig Hauser, Reinhard Sainitzer, and Eduard Gröller. Process visualization with levels of detail. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pages 67–70, 01 2002.

[12] Gabriel Moldovan, Alessandro Mariotti, Laurent Cordier, Guillaume Lehnasch, M. Salvetti, and Marcello Meldi. Multigrid sequential data assimilation for the large-eddy simulation of a massively separated bluff-body flow. *pre-print*, 12 2022.

[13] Xian Qu and Jun Xie. Simulation analysis for effect of rear structure of hatchback car on rear field characteristics. *Journal of Physics: Conference Series*, 1815(1):012001, feb 2021.

[14] Maria Riveiro. Evaluation of normal model visualization for anomaly detection in maritime traffic. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 4, 04 2014.

[15] Lukas Ruff, Jacob Kauffmann, Robert Vandermeulen, Gregoire Montavon, Wojciech Samek, Marius Kloft, Thomas Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, PP:1–40, 02 2021.

[16] K. Santhosh, Debi Dogra, and P. Roy. Anomaly detection in road traffic using visual surveillance: A survey. *ACM Computing Surveys*, 53:1–26, 12 2020.

[17] Young-Kyoon Suh and Lee Kiyong. A survey of simulation provenance systems: modeling, capturing, querying, visualization, and advanced utilization. *Human-centric Computing and Information Sciences*, 8, 12 2018.

[18] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, 2 edition, 2001.

[19] Johan Wagemans, Jacob Feldman, Sergei Gepshtein, Ruth Kimchi, James R. Pomerantz, Peter A van der Helm, and Cees van Leeuwen. A century of gestalt psychology in visual perception: Ii. conceptual and theoretical foundations. *Psychological bulletin*, 138 6:1218–52, 2012.

[20] Kamila Zdybał, Giuseppe D'Alessio, Gianmarco Aversano, Rafi Malik, Axel Coussement, James Sutherland, and Alessandro Parente. *Advancing Reacting Flow Simulations with Data-Driven Models*, pages 304–329. Cambridge University Press, 01 2023.