Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Exploration Support for Performance Maps

Marcel Galuschka

Course of Study:	Informatik
Examiner:	Prof. Daniel Weiskopf
Supervisor:	Dr. Kuno Kurzhals, Moataz Abdelaal, Prof. Thomas Wortmann, Max Zorn
Commenced:	August 3, 2022
Completed:	Feburary 3, 2023

Abstract

In architectural design optimization multi-dimensional parameter space analysis is an important operation to find new solutions. In this work, a glyph-based approach is developed to combine the information of the parameter space and the continuous solution space. The glyphs, displayed on a uniform grid, additionally encode the uncertainty of the underlying continuous space. Interaction is added to obtain higher resolution information. A feedback loop is introduced to indicate areas that have already been explored. The application of the tool is demonstrated and evaluated in an expert study. The study results are classified and processed for technical improvements and future research.

Kurzfassung

In der architektonischen Entwurfsoptimierung ist die Analyse des mehrdimensionalen Parameterraums ein wichtiger Vorgang, um neue Lösungen zu finden. In dieser Arbeit wird ein glyphenbasierter Ansatz entwickelt, um die Informationen des Parameterraums und des kontinuierlichen Lösungsraums zu kombinieren. Die auf einem einheitlichen Gitter dargestellten Glyphen kodieren zusätzlich die Unsicherheit des zugrunde liegenden kontinuierlichen Raumes. Interaktion wird hinzugefügt, um Informationen mit höherer Auflösung zu erhalten. Eine Feedback-Schleife wird eingeführt, um Bereiche anzuzeigen, die bereits erforscht wurden. Die Anwendung des Tools wird demonstriert und in einer Expertenstudie bewertet. Die Ergebnisse der Studie werden klassifiziert und für technische Verbesserungen und zukünftige Forschung aufbereitet.

Contents

List of Figures

List of Tables

List of Listings

Acronyms

CAD Computer Aided Design. 15

ICD Institute for Computational Design and Construction of the University of Stuttgart. 59

MDS MultiDimensional Scaling. 15

PCP Parallel Coordinate Plot. 66

RBFOpt Radial Basis Function Optimization. 9

Rhino 7 Rhinoceros 7. 11

SUS System Usability Scale. 9

t-SNE t-distributed Stochastic Neighbor Embedding. 70

UMAP Uniform Manifold Approximation and Projection for Dimension Reduction. 15

VIS Institute for Visualization and Interactive Systems of the University of Stuttgart. 60

WPF Windows Presentation Foundation. 48

1 Introduction

Our world population is rising steadily and with it the demand for new housing. To meet this demand, it is necessary to adapt new buildings to these requirements. The field of architectural design optimization tries to help architects to find new solutions.

In architectural design optimization, problems often have a multi-dimensional parameter space, which can easily become obscure. The challenge is to visualize the problem in a way that it can be understood by humans, i.e. to reduce the number of dimensions represented without having to make assumptions that may not be true in general. This work focuses on problems where parameters in a multi-dimensional parameter space influence a single objective value, such as material consumption. In general, we are not able to obtain an easily understandable representation of the function projecting from the parameter space to the actual objective value, which can therefore only be seen as a black box function. Evaluating such a black box function can be computationally intensive, so we are forced to work with only a few evaluations. We are also interested in learning how different parameter configurations affect our objective value, so that we can explore the space of possible designs with a particular objective in mind.

Current visualization tools such as the *Glyphboard* [KKG+20] focus on displaying multivariate features without an objective value as a result. They focus on comparing different dimensionality reduction techniques such as Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) and MultiDimensional Scaling (MDS) to find clusters and outliers within the data set. Our focus is on a comparable objective value that is influenced by the parameters, and we want to find correlations between them.

In this work, we extend the Performance Explorer plug-in called *Opossum* created by Wortmann [Wor17b]. The tool is designed to work with models created in Grasshopper 3D[1], a graphical algorithm editor included in the Computer Aided Design (CAD) software Rhino 7[2]. The models are then analyzed either by a physics engine such as Karamba3D[3] or Kangaroo[4] or any other metric, resulting in a numerical objective. With these inputs, the plug-in can be used to explore the behavior of the objective value for different input parameters. One example model which can be optimized is shown in Figure 1.1 where the heights of the arcs are parameterized.

In our work we will use a surrogate model to estimate data in places where we are not able to have evaluated results for the investigated model. We will use and know the basics of the optimizer used to generate the dataset with a good representation even with only a few function evaluations. The

[1]https://www.grasshopper3d.com/

[2]https://www.rhino3d.com/

[3]https://www.food4rhino.com/en/app/karamba3d

[4]https://simplyrhino.co.uk/3d-modelling-software/kangaroo

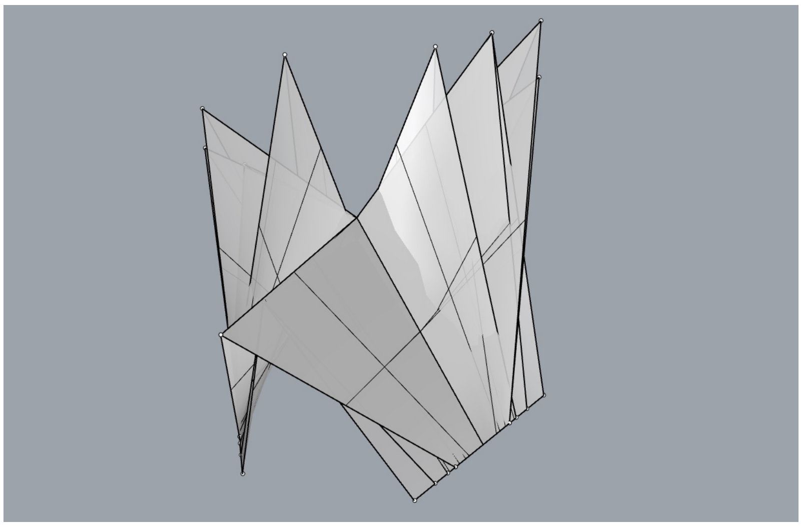


Figure 1.1: Example model with the heights of the arcs being parameterized

surrogate model is then projected into a two-dimensional plane using *StarCoordinates* [Kan00]. For interpolation in parameter space we use the *Barycentric Coordinates* [Wei23]. We use this interpolation in the two-dimensional representation of the data set.

We present our visualization tool which extends the Performance Explorer by adding a tighter connection between the parameter space and the objective value. We use a new glyph design based on star plots on a uniform grid to include the parameter space into the overview of the data set. In addition, we add several options to get more detailed information in needed areas. We also track user interaction, which gives us even more insight into how the performance map is perceived subconsciously.

We have a case study conducted by experts in the application area of our tool and discuss the results from it. From the experience of the study we describe improvements that would lead to a more consistent tool. We also propose future work regarding different dimensionality reductions and multi-objective values.

2 Background

In this Chapter we describe the basics we need for this work. This also includes works we use in the background, which are not related to the visualization part of our work as well as mathematical concepts. Starting with how the data we later show is being generated in Section 2.1. In Section 2.2 we will discuss how we distribute the data on a two-dimensional map. We will also specify how we are able to estimate data in locations where no data is simulated in advance in Section 2.3.

All of these ideas are fundamental to understand the overall concept which is described in Chapter 4. They are already used in the original visualization plug-in created by Wortmann [Wor17b]. We are extending the plug-in named *Opossum* to achieve easier exploration in this work.

2.1 Simulation based optimization

The objective value is given by a physics engine, that means we are in general only able to generate the objective value for some arbitrary parameters. It is therefore not possible to calculate the objective value continuously for all parameters. This is why we need a way to have a set of parameters for which we have the objective value calculated. This set having to optimize the objective value with a low number of physics evaluations and without being able to calculate the gradient efficiently. The set of parameter configurations and the corresponding objective value is called surrogate model and will be later used for the visualization.

Gutmann [Gut01] use the *RBFOpt*[Gut01] algorithm to create the surrogate model. The optimizer helps us to have very few function evaluations, since the physics engine calculating the objective value will dominate the computation time which can take several seconds or minutes to evaluate one parameter configuration. The *RBFOpt* algorithm tries to calculate a reasonable global minimum, converging to the correct minimum eventually. As the name suggests they use radial basis functions to interpolate the surface of the function hidden in the physics engine combined with a metric using the roughness of this interpolation. With this roughness metric they can avoid finding a local minimum. Since the minimization given the interpolated surface is much faster than evaluating the physics engine again, the interpolated function is used to estimate the parameters which lead to the global minimum. Wortmann [Wor17a] use these parameters to evaluate the black box function at this new estimated location. In the Conference paper [Wor17a] the Model-based Optimization part in *Opossum* is further described.

In Table 2.1 we can see the output of the optimizer storing the input parameter values along with the simulated objective value. The goal of this optimization is to maximize the outcome which can be easily achieved with an intermediate step which inverts the objective value after each simulation. This negative objective value can then be minimized with the *RBFOpt* algorithm. We can see that in this case the objective value can be highly increased with only eight evaluations of the physics engine.

index	parameters										objective value
0	99.0	0.6	96.6	2.7	2.0	98.0	0.9	99.8	99.6	0.7	15450
1	24.5	68.0	18.9	17.4	25.1	99.7	18.2	11.3	32.1	52.3	82058
2	78.8	25.3	10.5	87.5	62.2	16.1	58.8	40.2	57.5	87.6	245378
3	7.34	96.07	99.6	99.8	98.2	99.9	99.5	5.2	99.9	99.5	1103088
4	97.3	98.9	16.5	99.2	3.6	99.2	98.8	99.4	99.5	59.0	1173323
5	99.8	98.2	98.6	99.4	98.3	99.3	1.2	98.8	98.2	95.5	1951573
6	99.7	99.5	99.9	99.8	99.7	99.9	99.9	99.7	99.5	77.2	2979642
7	99.9	99.9	99.9	99.8	99.6	99.9	99.8	99.8	99.8	75.3	2987714

Table 2.1: Sample of the data from a surrogate model created by the *RBFOpt* algorithm described in Section 2.1. Ten input parameters influence the objective value which we want to be maximized.

2.2 Star Coordinates

The output of the surrogate model defines the data which will be shown in the visualization. This multi-dimensional data is mapped on a two-dimensional plane by using the *Star Coordinates* from Kandogan [Kan00]. In the *Star Coordinates* the coordinate axes are distributed on a circle, where each axis meets the previous and the next dimension with the same angle. Every single coordinate axis is scaled to have the same length. The projection of the n-dimensional data D, with $|D|$ entries, to a point (P_j) in the Cartesian Coordinates:

$$P_j(x, y) = \left(\sum_{i=0}^{n-1} \sin\left(\frac{2\pi \cdot i}{n}\right) \cdot o_i \cdot (d_{ji} - \min{}_i), \sum_{i=0}^{n-1} \cos\left(\frac{2\pi \cdot i}{n}\right) \cdot o_i \cdot (d_{ji} - \min{}_i) \right)$$

With $o_i := (\max{}_i - \min{}_i)$, $\min{}_i$ being the minimal valid value in dimension i and $\max{}_i$ being the highest valid value. Here j denotes the data item to be projected, i is the dimension in the n-dimensional dataset starting with zero, d_{ji} is the parameter value for data item j and dimension i in the dataset D.

In Figure 2.1 the data item, being the ten input parameters and the objective value, with index 0 from Table 2.1 is plotted based on the idea of the Star Coordinates. The blue line presents how the point is shifted on the plane in each direction. The trajectory shows that for this data item five dimensions C_0, C_2, C_5, C_7 and C_8 influence the position of the resulting point the most. The red cross in the Figure shows the resulting position in the point P_0.

There will be points with different coordinates in the n-dimensional space being mapped to the same location in two-dimensions. The projection from an n-dimensional space to the two-dimensional Cartesian Coordinates generates ambiguities. We can see that there are many constellations of parameters resulting in the same position after the projection. The original article from Kandogan [Kan00] resolves these ambiguities with an interactive part, where the user is able to modify the scale for each coordinate axis. This part of the Star Coordinates will not be used in this work, we will focus on different visualizations and interactions to investigate the dataset.

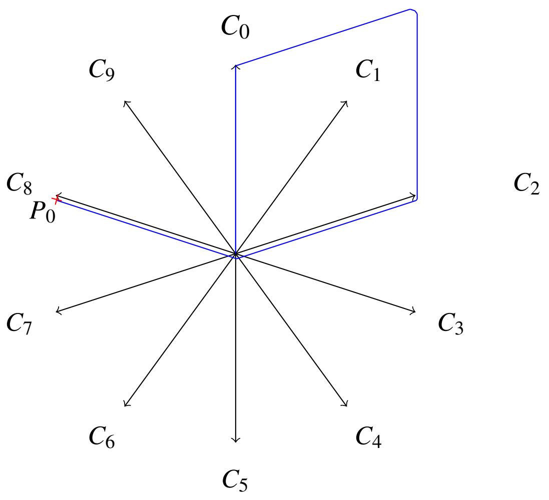


Figure 2.1: *Star Coordinates* [Kan00], 10-dimensional data item mapped on Cartesian Coordinates. The blue trajectory shows how each dimension shifts the point on the two-dimensional plane.

2.3 Barycentric Coordinates

We have only a small dataset D, the data items are not enough to gain insights. This is the reason why we use interpolation between the points we evaluated before. We use *Barycentric Coordinates*[Wei23] to calculate an arbitrary point in the Cartesian coordinate system. To be able to use the *Barycentric Coordinates* we need a structure of triangles, which is generated with a Delaunay Triangulation, giving us triangles with small edges.

Inside these triangles, as well as on the edge of them, we are able to estimate the value for any point P_x based on the three corner points. The points are weighted with the area stretched from the two opposite points and the point P_x we want to interpolate. This leads to a homogeneous interpolation having the highest weight for the nearest corner point to out new point P_x. In Figure 2.2 the area of each triangle in $T := \{t_a, t_b, t_c\}$ correspond to the corner point denoted in the subscript. The actual weight for the corner point w_i can then be calculated with: $w_i := \frac{\text{area}(t_i)}{\sum\limits_{t \in T} \text{area}(t)}$.
For the values $V := \{v_a, v_b, v_c\}$ of the corner points we can calculate the value of P_x as following: $P_x := \sum\limits_{i \in \{a,b,c\}} w_i \cdot v_i$.

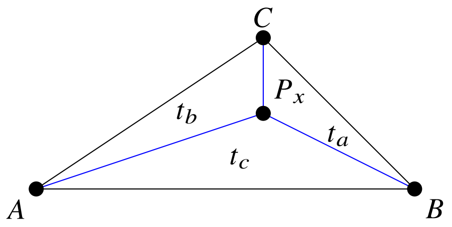


Figure 2.2: The area of the triangles t_a, t_b and t_c corelates with the influence of point in the subscript.

3 Related Work

The visualization of multi-dimensional data is not a new topic, there are already several approaches. In this Chapter we will introduce some of them and discuss the advantages of each and which aspects we have adopted in our own work. In Section 3.1 we give a brief overview of *Glyphboard* [KKG+20], they combine dimensionality reduction and displaying the parameters in a glyph. Furthermore, in Section 3.2 we show *Opossum* from Wortmann [Wor17b], which we will extend in our work. In Section 3.3 we present a study by Yihan Hou, et al. [Yih22] comparing different star glyphs along with recommendations for them.

3.1 Glyphboard

The core idea of *Glyphboard* [KKG+20] is the creation of a general visual analysis tool for high-dimensional datasets. In contrast to many others, this visualization tool is designed for experts in a certain domain with potentially less technical expertise. Therefore, the tool offers an easy to understand dashboard.

The whole dashboard contains a main view and a settings panel. The setting panel can be hidden to maximize the size of the main view. Each data item is mapped on the two-dimensional plane with the help of dimensionality reduction which can be chosen and adapted by the user. Since dimensionality reduction leads to clusters, and due to the anyway limited space on the display, it is generally not possible to display glyphs for each data item. Kammer et al. [KKG+20] use three different levels of details.

The first level shows only the data point as a dot, where the color is mapped to either the selected feature from the original dataset or a cluster variable which is calculated beforehand. By zooming, the user can get into the second level. In the second level, glyphs become visible instead of the dots. There are two types of glyphs that the user can choose. One of them being the star plot, where the feature value of each feature is encoded by the length of a line in each direction. These lines are connected to be a convex polygon or star, making it easier to recognize different shapes. The other being the flower glyphs, here value is encoded in the length of the petals, making it easier to recognize individual values. The most detailed level adds more details like coordinate axis for each glyphs.

In each level of detail hovering over a data item a tooltip shows its exact feature values and information. In the lowest level of detail only showing dots, a magnifying lens can be activated. Inside the lens the glyphs from the user selected level are shown. It is also possible to avoid overlapping of glyphs in a force-directed graph or in a uniform grid. On top of that the user can activate filters on the data and compare data items in different contexts.

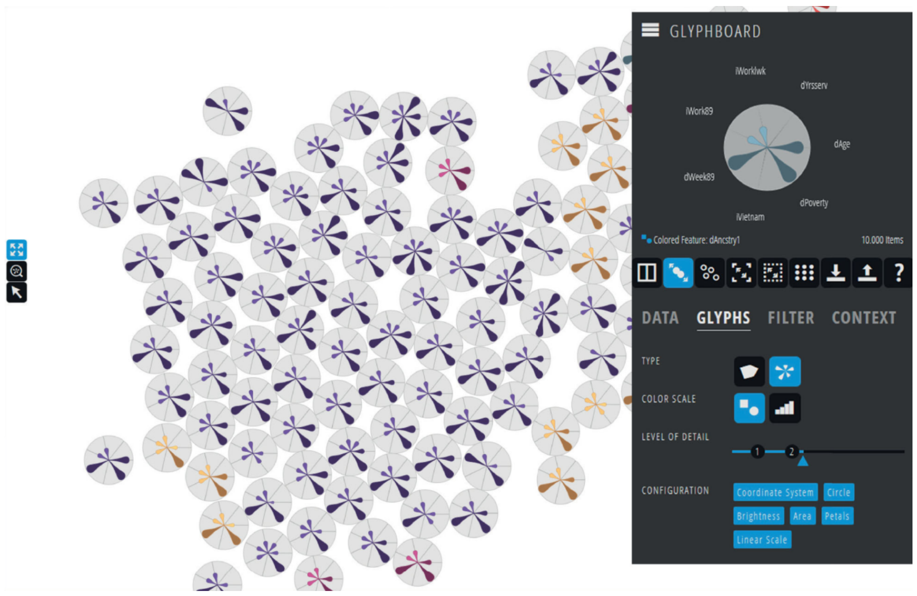


Figure 3.1: *Glyphboard* [KKG+20] on most detailed level. Showing a cluster with flower glyphs, where the overlap of glyphs is avoided.

In Figure 3.1 the dashboard from *Glyphboard* [KKG+20] shows the most detailed level. In the configuration panel on the right we can see that the force-directed is activated, because the level of detail slider passed the number two. This leads to a misplacement from the location calculated from the dimensional reduction in favor of having non-overlapping glyphs.

They evaluate their dashboard they conduct a case study. They invited data scientists and domain experts to use the tool on four different exemplary domains with which they are familiar. The general result was that the tool had a positive influence in stating new hypotheses and identifying clusters which was the goal of the tool.

In our work we also used similar star plots with a visual differentiation between plots. As well as a configuration panel where the user is able to see a selected glyph in detail. We tried to avoid dimensionality reduction algorithms that are difficult to understand, so that it will be even easier for users to create a connection between the location of a glyphs and the feature value for each dimension. Since we want to keep the number of evaluations of the physics engine small our dataset is much smaller. We use interpolation between the data items to have enough data to visualize, resulting in a continuous map. We can later sample the interpolated map at arbitrary points, we can therefore implicitly avoid overlapping of glyphs. We encode the uncertainty of the glyph value, due to the fact that there are more points underneath, in the stroke width around our glyphs. Another aspect which differs to our approach is, that we have an implicit mapping from our features to a performance value which we are able to encode with color on an arbitrary color scale palette. They on the other hand use one of the features or a cluster value from the dimensionality reduction which is used.

3.2 Performance Map

In our work we extend the approach from Wortmann [Wor17b]. They created a visualization tool for analyzing architectural models. Each architectural model can be evaluated with a certain metric defined in a physics engine. Since the evaluation of induvial models is often computational intense they have only a small dataset which can be used for the visualization. The dataset is generated with a machine learning algorithm using radial basis functions, mainly using *RBFOpt* [CN18] as described in Section 2.1.

The visualization tool shows the performance map encoding the objective value, also named the performance, of models. The high-dimensional dataset is projected on the two-dimensional plane using the *Star Coordinates* [Kan00]. As described in Section 2.2 this can lead to multiple data items which are projected on the same location in the two-dimensional space. If this is the case they compare data items at the same location and only keep items with a better performance. Having only few data items mostly in regions where the optimizer was in exportation phase, calculating new evaluations with parameters similar to the current optimum. Since they aim for a visualization that helps to understand the full black box function which maps from parameter space in performance space, these local data points are not sufficient.

The only feasible method to generate a continuous map leads them to interpolate with *Barycentric Coordinates* [Wei23] explained in Section 2.3. They create new data items by interpolating in the parameter space at an arbitrary location and use a surrogate model. The surrogate model approximates the performance by fitting curved surfaces through the calculated data items. With enough interpolated points being estimated they are able to show the continuous map where the data is color encoded using the rainbow color palette.

The full performance map consists of two parts which are plotted on top of each other. In the background they show the continuous map and on top of it the evaluated data items are shown as small circles separated from the background with a black stroke and filled with the performance value encoded with the same color palette. Figure 3.2 shows the performance map. In the top left corner a legend shows the value which is associated to a performance value, in this example the optimizer was searching for a low objective value. On the colored map it is possible to distinguish between two areas with low objective values. In the top left there is an orange colored area indicating values below 233. The right side of the map is dominated with objective values evaluated by the optimizer. All of them highlighted with the circle inside the yellow area having values below 261.

In our work we use the backend of the *Performance Map* [Wor17b]. This is why we get the same constraint having only a few data items. We also use the same approach to interpolate between evaluated design models giving us a continuous map of performance values. We were focused on helping the user to learn the black box function. This means the user is in need of a tight connection between performance value and the parameter space. We discarded the idea of a continuous bitmap in favor of aggregating nearby performance values and presenting them along with the parameter values in a star plot. The overview of parameter values and corresponding performance should reduce the interactions needed to strengthen hypothesizes.

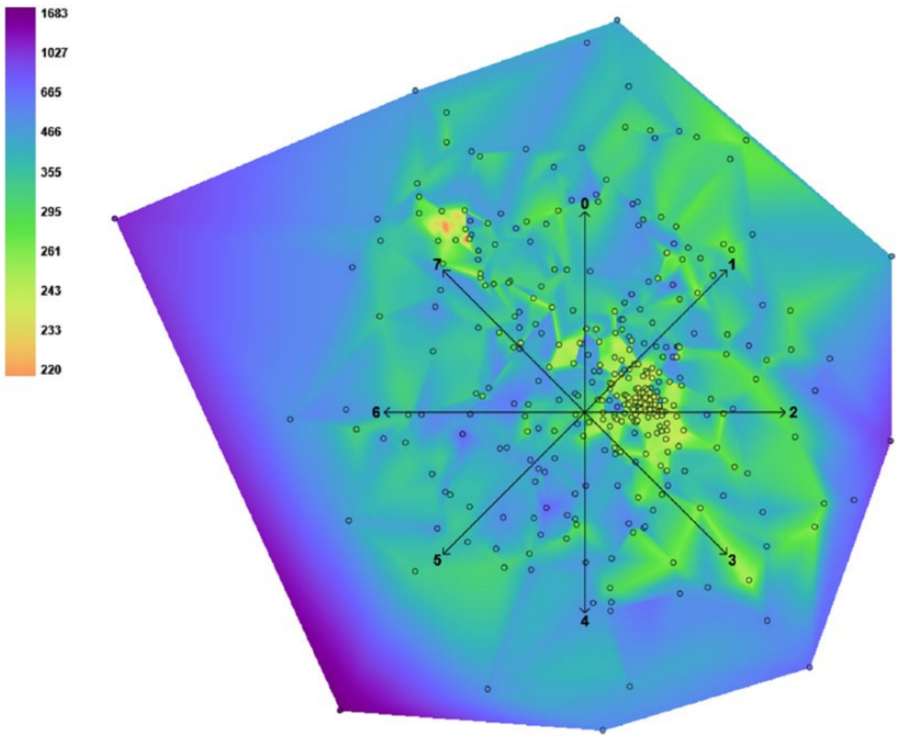


Figure 3.2: *Performance Map*[Wor17b] showing two areas with high performance, one just to the right of the center and the other in the upper left.

3.3 Effect of Star Glyphs

In the paper *A study of the effect of star glyph parameters on value estimation and comparison* [Yih22] they conduct a user study comparing different versions of the star glyph. They have five designs for the star glyph four of them involve the common known contour which encodes the values in the corner distance from the center. The plain contour is analyzed against the addition of points highlighting the corner points and the length shown with another line drawn from the center into the corner points. The forth contains both additions in the glyph and the fifth encodes the value in the area of the sector of a circle. They perform the study on a low-dimensional glyph showing four variables and a higher-dimensional glyphs with ten variables.

Figure 3.3 shows the ten different glyph types. The rows show the low and high dimensional cases, and the columns show the different versions. The first column shows the simple star glyph, which only encodes the value of each dimension in the outline. For the second column, they add the extended position to the contour by highlighting the corner point with a dot. In the third, they add the length of the distance with an additional line from the center of the glyph to the corner point. The fourth version is a combination of the second and third, with the augmented position and length being displayed at the same time. In the last case, the value is encoded in the area of the circle sector. For all star glyphs, a dashed circle has been added around each of the vertices to facilitate comparison of non-adjacent attributes.

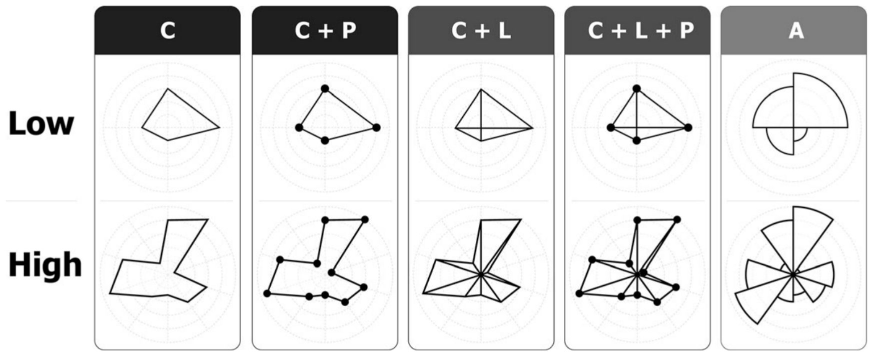


Figure 3.3: Five different star glyphs from the user study. Encoding the value in C (contour), P (position), L (length) and A (area) in both a low-dimensional and an high-dimensional case.

These glyphs are compared by Yihan Hou, et al. [Yih22] in terms of efficiency, accuracy, and usability. The tasks are finding extreme values, estimating the values of a given data attribute, comparing the value of two adjacent attributes, and comparing the values of two non-adjacent attributes. The evaluation of efficiency is based on the time needed for each task, while accuracy is calculated by the correctness rate. The user experience is provided by the user feedback, which is obtained through a questionnaire asking for confidence and preference.

The result of the study is the general preference of the participants. The reason for this is the simple sense of extreme values in the glyph. Another preference is that the extended dot helps in solving tasks, and the length encoding is the least liked. However, the measurable data tell a different story. For the given tasks, the area glyph is still the most efficient, but the length encoding also helps to significantly reduce the completion time.

With the results in mind, we use the simple star glyphs for our work, since the glyphs will be small for finer grids, we aim to have basic glyphs. The task in our work is to find similarities in a large number of glyphs, shifting the focus from the actual values to the shape of the glyphs. A similar approach to additional length encoding can be added on demand within the ParamView. The idea of adding circles around the contour to compare different attributes may be helpful and added in the future.

4 Concept

In the following the concept of our extensions to the existing performance map is described. We will start by outlining the requirements by presenting the core glyph we will use throughout this work and describing how our application is structured.

In this Chapter we use the following naming convention:

data item The data item refers to an actual item, these points are evaluated by the black box function.

data point A data point is estimated based on the surrogate model and lies in the two-dimensional plane. In the GridView a data point is a glyph shown in the main view.

4.1 Requirements

There are several aspects we want to extend to the performance map. We want to strengthen the connection between the parameter space and the performance space in an overview of the whole dataset. This will eventually lead to the need to reduce the level of detail we are able to show, this issue will be addressed as well. The data drill down relies on interaction and may be a personal preference, so we add several interaction configurations so that we can later evaluate which one works best. An important aspect is that the user should be able to see the product they are investigating in three-dimensional view. A rendered view of the parameter set should be available on demand. As the user learns more and more of a black box function to optimize his design, he might get distracted and miss important information, so we need a visual tool to track where the user has dug into the information. By showing him where he has looked for deeper insights, he will be able to see areas he has not explored where good information could be hidden. In addition to all of the above, we want the user to have control over the visualization, and we want them to be able to use our add-ons in many different combinations to meet their needs.

4.2 Grid Levels

In the Section 4.5.2 we will use a grid with different levels of fineness. We generate five grids with 256×256, 128×128, 64×64, 32×32 and 16×16 entries. The finest grid with 256×256 entries is generated first and calculated using the surrogate model. The other grids are calculated by merging four entries into one by computing the mean value of the objective values and the parameters as well as the standard deviation of the objective values.

4.3 Glyph

We use the star plot since it was used in a newer version than the one described in *Performance Map* [Wor17b]. One reason using it again is, that the user is already familiar to it, since they have already seen it in the *Performance Map* before. Another reason is that Opach et al. [OPDR18] found out that they perform well by using the star plot if the goal is to find similarities between glyphs, while sacrificing the ability to read parameter values.

In our star plot we normalize the parameters to scale from 0 to 1 in the range defined by the boundaries in Grasshopper. The star is built by an n-sided polygon where n is the number of dimensions in parameter space. The parameters are distributed in the same way as in the *StarCoordinates* [Kan00]. In each direction ($i \in \{0, \dots, n-1\}$) one corner of the polygon is defined by the parameter p_i scaled between 0 and 1. For a glyph with size r the corner points are defined by:

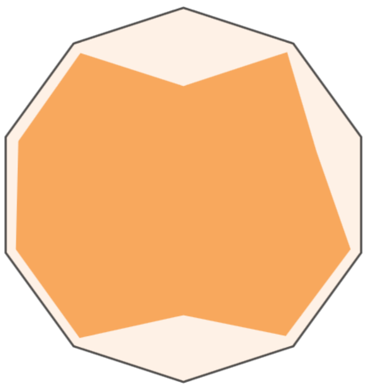
$$(x_i, y_i) := \left(\sin\left(\frac{2\pi * p_i}{n} \cdot r\right), \cos\left(\frac{2\pi * p_i}{n} \cdot r\right), \right) \forall i \in 0, \dots, n-1$$

These glyphs can not exceed the size of a circle with radius r. The polygon is filled with the color defined by the objective value and the color palette which is used. By default we use the rainbow color palette to be consistent with the original *Performance Map*.

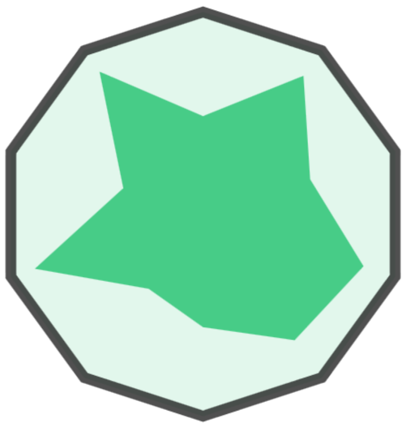
For glyphs presenting parameters with high values, the polygon area will be large and the color encoded objective value can be read easily. However, the size of the area of the polygon will change the subjective feeling of color intensity, small polygons lead to a wrong interpretation of the color encoding the objective value. We try to reduce the misinterpretation by adding a background color to the full polygon, having all parameters showing their maximum value with a lower color hue. This helps to detect the color encoding of the glyph even for glyphs with low parameter values, still making it possible to detect similar forms created by the polygon.

Since the glyph has to be larger than a singular pixel, we have to merge several points in a single glyph. Therefore, we show the fluctuation of the points under a glyph by calculating the standard deviation of all points and project it in the thickness of the black border around the colored background. The thickness is scaled by the maximum and the minimum of the standard deviation on the complete dataset. The thickness is at least one pixel and can be a maximum to be fraction of the glyph size. The user can select if a high standard deviation is encoded with thick borders or low deviation is associated with thick borders or he can even disable the border at all.

Figure 4.1 shows two different glyphs from an example dataset with ten input parameters. The glyph on the left, which represents a higher target value, is color-coded with an orange color from the rainbow color palette. It can be seen that on the left side all the parameter values are high because the polygon almost reaches the edge. The two parameters mapped in the two vertical directions are relatively low in comparison. The standard deviation is also low compared to the glyph on the right. However, the glyph on the right has a medium objective value because it is encoded with a green color. The standard deviation, which tells us that there are glyphs underneath that are changing in their performance. This could be due to different parameters being mapped to a similar location with the star coordinates, giving a different objective value, or that the target value varies with similar parameters performing a local extreme in hyperspace. The parameters in this data point are smaller than on the left, so we chose to set the background of the glyph to encode the same color but with higher transparency, keeping the shape of the glyph in focus.



(a) Glyph with high objective value in the rainbow color palette with high parameters for seven of the ten dimensions.



(b) Glyph with lower objective value in the rainbow color palette, the parameters are lower in average. Underneath the area of the glyph there is a higher deviation of objective values.

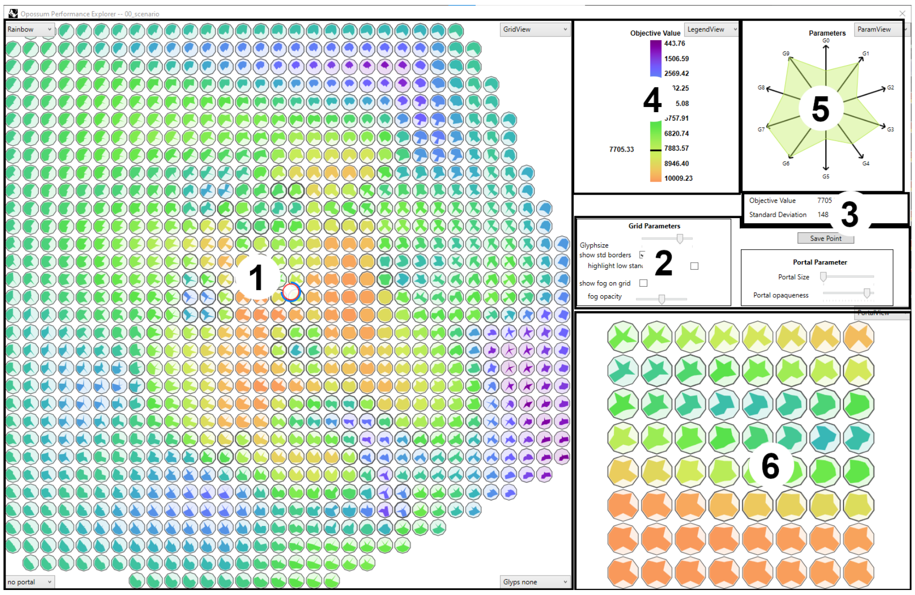
Figure 4.1: Our star glyphs showing the parameter space in the form and the objective value in the encoded color. The background showing the same color with higher transparency and the border presenting the standard deviation underneath the glyph.

In addition the user can display if there are evaluated data points underneath a glyph by changing the background color to be gray instead of the encoded color from the color palette or by reducing the opaqueness of the glyph.

4.4 Dashboard Organization

Our dashboard is designed to be used in a day-to-day use and therefore the focus lies in a steep learning curve without sacrificing features that could be helpful when working with it for a longer period of time.

The dashboard is divided in six parts which you can see in Figure 4.2. The main view is designated to show the overview of the complete dataset. The user can choose between the ContinuousView as well as the GridView described in Section 4.5.2 and add a visual help of investigated regions can be activated as explained in Section 4.5.3. The configuration panel is one of the views that is always visible and includes user defined parameters for the GridView and the portal shown on top of the grid. With the grid parameters the user can select size of the glyphs in the grid, change settings of the standard deviation shown in the border of the glyphs and toggle the intensity of the FogView inside the grid. With the portal parameters the portal size and the background opacity can be changed. The save point button can be used to store the currently selected point to disk in a csv-file. The ValueView which is as well always visible, shows the estimated value and standard deviation of the location which is currently hovered with the mouse, but can be locked with the right mouse button. This enables the user to change settings or leave the application without loosing the information he was just investigating.



Objective Value 7705
Standard Deviation 148

Figure 4.2: Our visualization tool contains six different views. (1) main view, (2) configuration panel, (3) ValueView, (4) - (6) additional views assigned individually.

Since it is not possible, being limited in screen space, or preferred by the user to use all additions we made, the user is able to choose which of them he will use. Our dashboard has therefore 3 additional views where the user is able to define what it shows by selecting the content with a dropdown menu which is available in each of the views. In addition the user can have interactive tools displayed in front of the main view. These can be selected with a dropdown menu in the bottom right corner of the main view.

In Figure 4.2 the overall organization of views is shown.

1. The main view contains GridView by default but can be changed to the ContinuousView. It shows an overview of the complete span of the surrogate model created by the optimizer.
2. The configuration panel is always visible and is used to change parameters of the visualization.
3. The ValueView shows the objective value and the standard deviation of the currently selected point being visible at all times.
4. The first additional view shows the LegendView by default but can be changed with the dropdown menu in the top right corner.
5. The second additional view shows the ParamView when the tool is started and can be changed with the dropdown menu.
6. The third additional view is the largest one and displays the PortalView, but can be changed as well with the dropdown menu.

4.5 Main View

The main view is made to show an overview of the complete dataset. The user can choose between the ContinuousView as shown in Subsection 4.5.1 and our new approach by selecting the GridView as described in Subsection 4.5.2. In both configurations the user is able to use the mouse to get more information for the objective value and the parameters. Therefore, each time the location of the mouse pointer is changed the additional views get triggered to update with the point currently under the mouse pointer in the ContinuousView or with the glyph under the mouse pointer in the GridView. It might be helpful to remember the currently visible information even when the application is left or settings should be changed. This is why it is possible to interrupt these updates by clicking with the right mouse button once, locking the information visible. By clicking the right mouse button a second time the information will update again. It is even possible to store the parameter set and values to disk by hitting the space bar or by clicking on the save point button in the configuration panel, this will store the current parameters as well as the objective value to disk. The continuous map and the discretized map showing glyphs having their own set of benefits and drawbacks. The main view has four dropdown menus in all of the four corners. We placed these configuration menus inside the main view because the configurations are directly coupled with the main view leaving the least confusion on where the configuration can be found. We chose to place the dropdown menus in the corners because the convex hull which is spanned by the dataset is most likely not to fill the rectangle perfectly, leaving free space in the corners with the highest probability. In addition to these two versions of the map it is possible to show the FogView as described in Subsection 4.5.3 on top of the selected map, here the user can select the amount of fog which is visible.

4.5.1 ContinuousView

The ContinuousView contains the information from the *Performance Map* from Wortmann [Wor17b]. The interpolated map is drawn as a bitmap and stretched to fill the size of the main view. On top of the bitmap we show the axes indicating which parameter is associated to a specific direction. In front of the axes we display for each data item, which is evaluated with the black box function, a circle with a black stroke. This gives us nearly the same visuals as in the original *Performance Map*, only the legend is missing. We excluded the legend from the main view because it is likely that it would either lead to a map which is either not maximized or overlap with the map.

In addition to the original *Performance Map* the user is able to hover with the mouse over locations to get more information. Therefore, the main view triggers the additional views to update on each movement of the mouse.

The continuous map helps the user to find the estimated global optimum as it is visible directly in the bitmap. Additionally the circles indicate the location of points with a good evaluated objective value. The continuity in the map might as well help the user to spot anomalies or structures resulting from the *StarCoordinates* [Kan00], which has to be proven by comparing parameters considering the ambiguities from the dimension reduction.

The process of excluding these ambiguities to form noticeable structures and proving that the actual parameter leads to a certain behavior involves interaction with the map. Since there are more ambiguities with higher dimensional parameter spaces the actual position of a data point becomes less relevant. A visual representation of the parameters, instead of only showing the objective value, could be helpful to reduce the number of interactions needed to spot hypotheses of correlations between parameters and the objective value.

4.5.2 GridView

In the GridView, we switch from a continuous to a discrete representation. Therefore, we add the values of the parameters to the map by showing the glyphs as described in Section 4.3. The GridView is the default visualization for the main view and can be changed with the drop down menu in the top right corner of the view. We have a set of grids that we calculate once when the application is started. These grids are discretized versions of the continuous map of different sample rates The grid which is shown can be selected in the configuration pane.

By displaying one grid, each data item in the grid is printed regularly to fill the complete view. Here the location of each glyph contains the data which would be overdrawn by the glyph and the rectangular augmented area in the continuous case, as shown in Figure 4.3. Furthermore, the mean of the objective value as well as the mean value for each parameter set which is within the associated area are shown

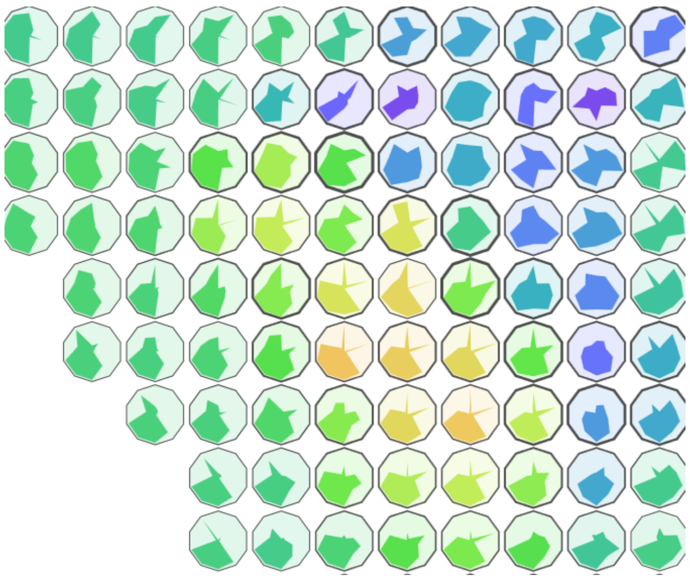


Figure 4.3: Extract of the GridView showing distinct shapes for a high objective value. Every glyph with with yellow or orange color has two spikes in the top right and high parameter values in the bottom left.

A grid of glyphs is shown in Figure 4.3. Here one can see that G9 and G3 has to be close to zero to achieve a high objective value. Whereas the glyphs forms a ’belly’, having the genes G4 through G7 to be maximized in the most of the cases. The two spikes for dimension G0 and G2 is visually appealing as well in this dataset. In this grid size we could therefore assume that the genes G1 and G8 have the least influence on the objective value leaving the most freedom to those parameters.

This hypothesis should be checked by comparing more glyphs showing the same or a similar behavior. This could be achieved by either increasing the resolution of glyphs which implicitly will make each glyph smaller. Another method is to use the mouse interactively to drill down single glyphs in a higher resolution. The user is able to use different techniques which we will discuss later in Section 4.6.

Compared to the ContinuousView, the GridView has the advantage of displaying the parameters and the objective value in one overview. Therefore, it is easier to post hypotheses about the dataset without interacting with the map and having to investigate single points.

In comparison to the ContinuousView the GridView lacks accuracy because fewer points are plotted in the map. We try to compensate this drawback with the help of interaction as described in Section 4.6. We also loose the information of data item which are evaluated by the black box function, because they would overlap with he glyphs making it harder to identify the form. This is why we the user is able to highlight glyphs where real data items are covered by them. It is possible to highlight them with a gray background under the colored polygon or by reducing the opacity of the complete glyph. Since there are in general more glyphs with no evaluated points, the once with reduced opacity are still appealing to the eye.

4.5.3 FogView

The FogView can be activated and will be visible in front of the correctly selected view being either the ContinuousView or the GridView. The fog shows the user where he was looking for results in the map. This can help the user to understand where areas with interesting information was already found and investigated by him. It could also lead the user to see where he was not paying attention or he did not spend time looking for more detail. Underneath the unexplored area in the map there might be interesting points hidden. This fog is designed to help the user exploring a model configuration for a long time and getting stuck in the same result over and over. However, it could also help to identify clusters of interesting areas if he is satisfied with the results.

The location which is identified by the fog as visited is based on the position of the mouse cursor. The gauss normal distribution is used, with the portal size for the standard deviation, to update not only the singular point but also the area around the mouse pointers exact location.

With the FogView it is possible to detect areas visited by the user. In Figure 4.4 one can see that there are several parallel thin lines from the right to the lower left of the heatmap. The user may have looked for changes in the dataset. The most prominent is a circular motion which can be seen in the thin lines in lower left. These patterns occurred while having the overview in focus, because the portal was not visible. The larger spots came from the portal being enlarged, here the user was drilling down the data. With this heatmap we can see that there has to be an interesting region in the bottom left, it might be that the border of the region towards the bottom right is not trivial as the user was looking for information there with the portal.

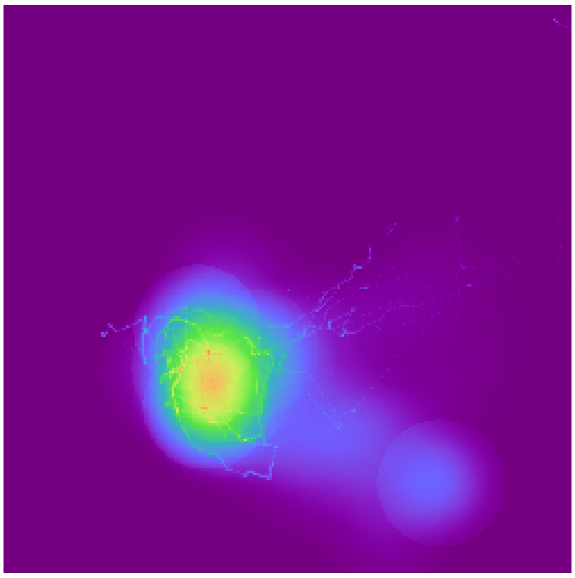


Figure 4.4: The FogView showing a heatmap of locations visited by the user. The width is depending on the portal size selected.

4.6 Interactive Views

As we already stated above the user is able to interact with the tool to get more insights and additional information about the location he is currently investigating. In this Section we will describe the different additional interaction techniques and views helping the user to learn the correlation between parameters and the objective value.

4.6.1 ParamView

The location of a data point on the map can not conclude to the actual parameters associated to it. These ambiguities result from the *StarCoordinates* [Kan00] used to Figure out at which location a certain parameter set is plotted. The ParamView is, if the ContinuousView is used, the only source to verify the parameters which result in the objective value on the map.

In our application the ParamView can be selected to be shown in any of the additional views. There we show a star plot of the hovered data point with the color hue encoded with the color palette. However, the surface of the polygon has a higher transparency and only the edge of the polygon is fully opaque. We reduce the opaqueness of of the polygon because we add an axis for each parameter showing the maximum value. This allows the user to estimate the parameter values of the currently hovered data point. This not only helps the user comparing individual points by the parameter configuration, but also makes it possible to name the parameters and estimate the parameter value easier since there are axes showing the maximum value for each of the parameters.

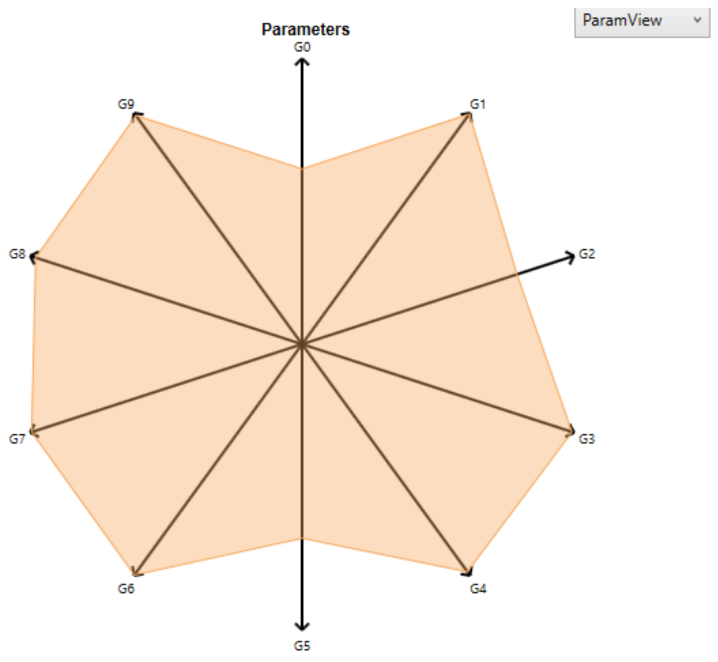


Figure 4.5: The ParamView showing the star glyph with the axes indicating the names of the dimensions. All parameters being maximized apart from G0, G2 and G5.

In the example data point in the ParamView in Figure 4.5 a similar point to the one in Figure 4.1a is shown. With the arrow in each axis it is easier to estimate the real value of the parameter, since the part of the arrow covered with the polygon is easier to estimate. Another benefit of the ParamView is that for the single glyph it is easy to read the name of the parameters. In this case the parameters G0, G2 and G5 are not maximized but the other seven are.

4.6.2 LegendView

The legend is the primary source of information when it comes to relating the color encoding in the main view to the objective value. It shows the objective value for ten colors in the scale, evenly distributed across the entire color palette. In addition, the objective value of the currently hovered point is printed next to the legend, along with a black line indicating its location on the color scale. This is important when the user is viewing the dataset for the first time or when moving between datasets. As the user explores the dataset and drills down for more detail, the legend becomes increasingly obsolete as the user is confronted with the actual objective value for each individual point. The legend takes up a lot of space in the main view and constantly overlaps with the map. To solve this problem, we would have to reduce the size of the map, which would take up a lot of space in the main view. This led us to remove the legend from the main view and add it as an option to be displayed in one of the additional views. Now, the user can interactively select the legend to see at first glance Once he knows the basic correlation between color and objective value, he may prefer another view to get deeper insights.

4.6.3 LensView

In the GridView, we combine multiple data points from the continuous map into a single glyph. With the GridView only, it is not possible to get a deeper insight into the data hidden in each glyph. Our first idea was to create a lens similar to a fisheye lens, but with the addition that the glyph in the center of the lens expands to show its children, providing even more insight.

Since we want to keep the focus on the glyph, the four glyphs in the center, coming from the one where the mouse pointer is, are not merged into its four children. To keep the lens content consistent, we wanted to show all the glyphs under the lens, which had to be much smaller than the original glyph, since the four children of the center glyphs should be comparable by shape. This led us to unmerge only the one glyph in the center to keep the others a bit larger. The size of each glyph is calculated with a similar approach to inverse distance weighting. Each of the four directions up, down, left and right from the center contains a number of glyphs. For each direction we calculate the weight for all glyphs in this direction with $w := \frac{1}{\text{dist}+1}$ with dist being the Euclidean distance to the mouse pointer. These weights are normalized to sum up to one and the weight will describe how much space the glyph will take. The glyphs expanded and nearest to the mouse pointer will have add the weights for the up and down and the left and right direction and use the minimum of these two values as their own weight. This ensures that the center glyph will be the largest. The result is a lens that displays glyphs with a continuous size as one move the mouse.

As we discussed earlier, this leads us to display the glyphs in the outer regions of the lens in a much smaller size than before in the grid. This is why we made it possible to display the contents of the lens next to the original grid. The only indication of the size of the lens is a circle in the main view that indicates the location and size of the lens, but it is otherwise completely transparent. This helps the user keep track of the glyphs displayed in the lens and understand where the unmerged glyph came from.

The LensView can be seen in Figure 4.6. Here the glyph in the lens with the largest diameter along with the three largest ones being the one to the right, the one directly below the largest one and the green one to the lower right of it, being from a finer grid. The other ones are from the same grid but changed in size. Glyphs further away from the center are smaller than the ones in in the middle. It is possible to gain information of the neighborhood namely to the right of the center it is still possible to see the orange glyphs and to the left the glyphs are more blueish. However, it is impossible to detect shapes inside the glyphs which are not directly next to the center one. The information about the neighborhood coloring is hard to see and it would be easier to move the lens than trying to compare the colors. Making this approach nice to look at but more or less impossible to use. One can also see that some of the glyphs are highlighted with a gray background. These glyphs have actual evaluated points underneath.

We rejected this approach because it led to misunderstandings about which glyph in the lens was associated to which level. In addition, the jump from one unmerged glyph to another is still not continuous, resulting in a wobble that changes the size of the glyphs in the lens, but changes immediately when another glyph is unmerged. Even if these problems had been solved, it would only be possible to see four glyphs for each unmerged glyph in the main view, and any deeper digging into the source of the glyph merged by many data points would not be possible or would require more complex techniques.

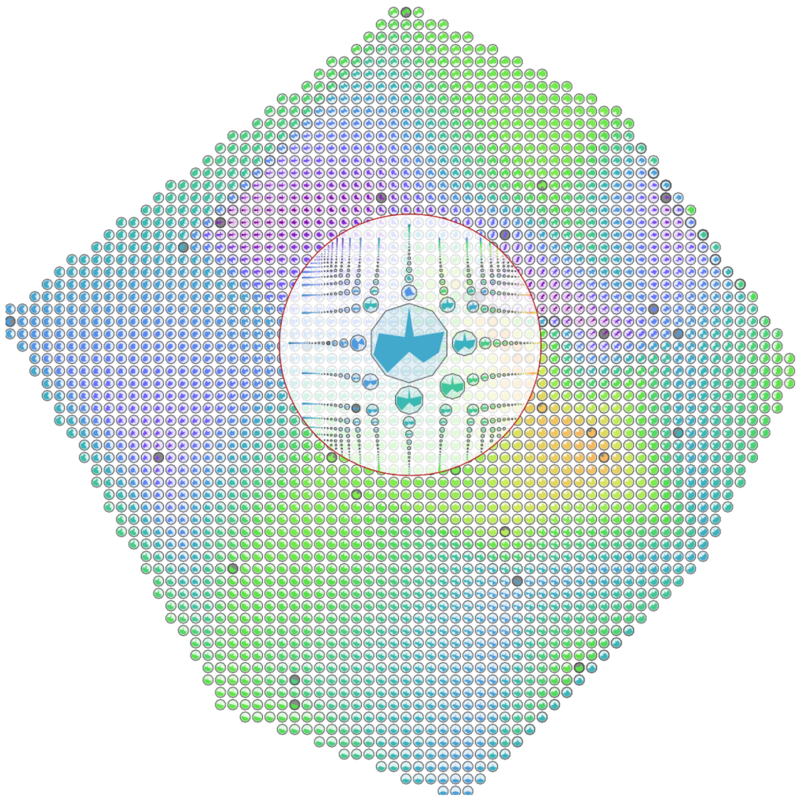


Figure 4.6: The LensView was the first approach to dig deeper into the grid. The fisheye lens shows four glyphs larger than the rest and all of the glyphs inside the lens smaller and shifted to fill the circular lens.

4.6.4 PortalView

Our new approach uses a mechanic to drill down a single glyph in its children. We call it the portal since we are able to see deeper levels of the shown grid and is shown on top of the GridView. Within the portal only the data point which has the least distance to the mouse pointer will be unmerged. This reduces the amount of information highlighted inside the portal in contrast to the lens described in Section 4.6.3. Since we remove any information about the neighborhood from the portal the content inside the portal is consistent in this approach. The location of the portal follows the mouse pointer showing the center of the portal exactly under the mouse pointer. The content inside the portal changes every time the user is hovering the mouse pointer over a different data point in the grid.

By default the portal shows the same glyph as in the grid shown anyway. The user can change to a finer grid by scrolling upwards with the mouse and decrease the fineness by scrolling downwards. For each finer grid the glyph is unmerged in its 4 children, therefore the new glyphs are having each only half the size in each direction. A ring around the lens shows how much the user can semantically zoom. If the ring with a blue color is not visible the grid level and the level inside the portal are identically. For each level another segment of the ring will be visible until the finest grid is shown and the ring is completed.

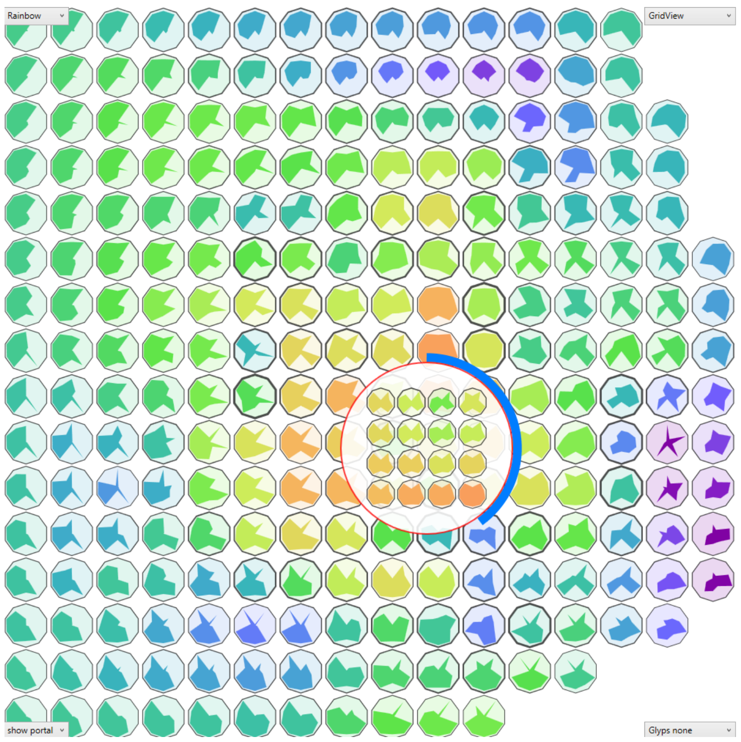


Figure 4.7: The GridView showing a coarse grid in the background. The portal in the front adds the possibility to dig deeper into single glyphs getting the information from a finer grid.

Since the portal moves with the mouse, it is by default not possible to reach any glyph inside the portal other than in the coarsest level where only one glyph is visible. This is why we added the functionality to lock the portal with a right click with the mouse pointer the black circle around the portal content get red while the portal can not be moved. While the portal is locked the ParamView, the LegendView and the ValueView updates with the glyphs inside the portal, this is how the user can get more insights in the finer levels. With finer grid levels the number of glyphs in the portal increases, the user can use a slider to interactively change the size of the portal. This enables the user to select in how much of the details he is interested, or how much he prefers to see the overview of the dataset in the GridView.

In Figure 4.7 the PortalView inside the grid is shown. The blue ring around the portal shows, that the user can still zoom in finer grids. In this case there would be three levels deeper than the current one in the portal. The user would likely have to increase the size of the lens since for finer levels the glyphs are going to be smaller. In the visible state the portal will not change being locked which is indicated by the red circle. It is possible to see that the yellow glyph contains different parameter configurations. In the lower two rows the parameters are similar, they differ only slightly in the vertical directions. The green glyphs have very different parameters in contrast, minimizing the vertical directions and having lower values for directions being 20° rotated from the vertical ones, namely G9 and G4. The user can inspect the names of the parameters in this state by hovering over the glyphs inside the portal. This would update the ParamView if it is visible in the current configuration. The ValueView showing the actual objective value and the LegendView as well if visible.

While the portal can show information of finer grid levels the glyphs are either small and not distinguishable or the portal has to be huge overlapping the GridView. This might lead to a possible confusion in which data point is actually unmerged when the portal overlaps several glyphs from the grid. We added the feature to have the portal in an additional view and displaying only the level indicator with the blue circle and the lens locking indicator with the red circle in the grid. The portal itself acts the same as before, the user locks the lens and is able to hover over the glyphs inside the portal and the ParamView, the LegendView and the ValueView triggered to update. This let's us increase the individual glyphs without intersecting with the GridView. However, this adds the additional effort to change the focus more often between the GridView and the addition view showing the portal content. Moving the lens in the GridView will potentially leads the user to pay attention to the portal content and therefore eliminates potentially the connection between the overview and the actual seen data.

4.6.5 RenderView

The tool is build to develop model configurations with a certain goal. The task of an architect is to validate the configurations presented by our tool presents. This includes other measurable goals which have to be fulfilled as well as the overall aesthetics and practicality.

To discard model configurations often the visualization of the model in the real world is sufficient. We added the feature that the user can include the rendered view from the Rhino 7 application directly in our tool. The user can also use bad performing models to see how they would look like in the real world and discard architectural ideas if it is not reasonable to build them. Therefore, the RenderView can act as filtering of ideas and source of inspiration at the same time. However, creation of the model inside Rhino 7 is in general not fast and may take some time. This is why the new rendering will be generated only on demand.

4.7 Color Palette

In the original *Opossum* from Wortmann [Wor17b] the rainbow color palette was used. In fact they used color palette created in the essay *The rainbow is dead... long live the rainbow!* [Mat12]. The idea behind his essay is that the rainbow color palette being the most used palette has some major drawbacks even resulting in negative impact on solving tasks. In the paper from Michelle A. Borkin [Mic23] it is stated that the rainbow color palette even resulted in more misdiagnoses of heart diseases.

Matteo Niccoli [Mat12] tried to understand the problems with the rainbow color palette, being still the most commonly used palette at that time. The main problems with color palettes is a non-linear luminance change, which is especially a problem for the rainbow color palette. The luminance drops for high values being a good reason for misdiagnoses, since the luminance drop leads a user to identify high peaks as valleys. He tried to fix the rainbow color palette by adjusting the the three color curves in the RGB space until the luminance is monotonically increasing.

He came up with his *cubeYF* palette having a 100 % monotonically increasing perceptual luminance and nearly linear increase, however he discarded the red and yellow colors at the end of the spectrum. Wortmann [Wor17b] used the *cube1* palette including a red color at the end of the color spectrum.

This results in less perfect luminance change at the end of the spectrum, nevertheless the added color adds more contrast in the hue space. This helps the user to find and remember values using the legend, since there are more distinct colors we are familiar with and is less dependent on the intensity of the each color.

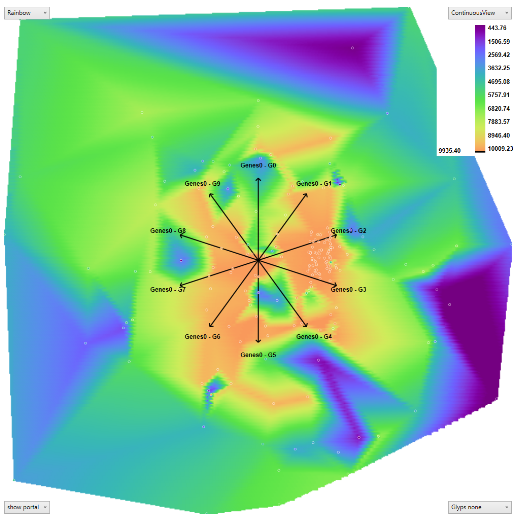
We added several other color palettes referring to the advice from Kenneth Moreland [Ken15] publicly available on his own website[1]. The Kindlmann being a similar palette to the *cube1* discussed earlier having the rainbow color map adjusted to change monotonically in brightness and ending in white instead of yellow. The black body color map ranging from black over red and orange to end in a complete white. In the inferno color map adding a purple color to to the black body color palette. Since all of these color maps use the complete spectrum of the luminance, we will have white glyphs in front of the white background. This forces us to add black outlines around the inner polygon which reduces the efficiency of the color encoding and the outer stroke indicating the standard deviation.

In Figure 4.8 the four color palettes are displayed. The similarities between the default color palette *cube1* and the Kindlmann are obvious. The main difference being that the Kindlmann in Figure 4.8b scales from a complete black to white which does not occur in the *cube1* as seen in Figure 4.8a. In the examples it is visible that it is hard to distinguish between similar values with a high objective value. While the other two color maps are similar as well where the black body color map having a similar but less significant problem with extreme values.

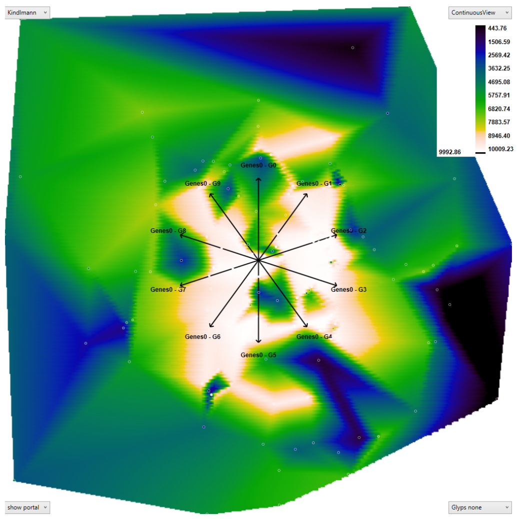
4.8 Customizability

Our tool is designed to help architects learn how parameters influence the model in a certain objective value. Since the resulting performance map and the weight of the used metric the tasks can differ a lot. This is why we built the tool to be customizable by the user, who might also have their own preferences. Our application has the described views and features which were described earlier. Each of them can be selected as described in Section 4.4 giving the user the freedom to choose the additions they need to solve their tasks as they prefer. In Chapter 5 we describe among others how this customizability is achieved technically.

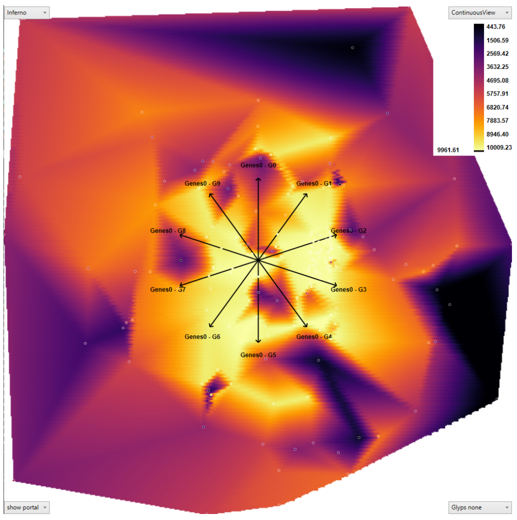
[1] https://www.kennethmoreland.com/color-advice/



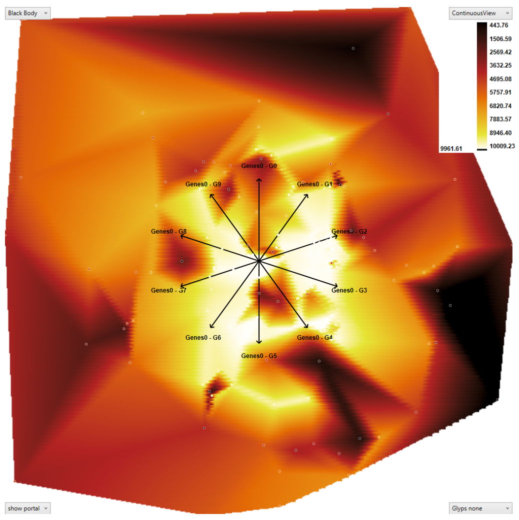
(a) Cube1 Color Palette



(b) Kindlmann Color Palette



(c) Inferno Color Palette



(d) Black Body Color Palette

Figure 4.8: The ContinuousView of one dataset in all four supported color palettes.

5 Implementation

The tool is based on *Opossum*[1] of version 2.2.4, which was the latest version when we started work. We added the functionality described in the previous Chapter 4. In this Chapter we will explain the technical details and limitations that result in the tool created in this thesis.

5.1 Grasshopper

Grasshopper is a graphical algorithm editor. Within Grasshopper it is possible to define models that are parameterized. These models can then be analyzed with the help of a physics engine, of which there are several available as plug-ins for Grasshopper and can be found on Food4Rhino[2]. For example a commonly used one is Karamba3D[3]. The bridge between the analyzing physics engine and the tool we created is kept very general, so the only requirement is that the engine generates a numerical value containing a score based on the chosen metric.

The optimizer tool needs three different inputs, which are connected to the wires in the editor. The first are the Variables, which are automatically changed by the optimizer in use. The second input is called Simulators, this input is primarily used to improve performance by disabling the model when it is not needed. Since Grasshopper ignores any component with a wire coming out of it, the physics engine will be idle, saving computing power. The third input is the numerical objective that the optimizer uses to find good solutions.

Each input can be connected to multiple components. All variables are taken into account to find a good objective. In the visualization, these Variables are aggregated and each is displayed equally. The Simulators are primarily built to disable the physics engine when its computation is not needed. However, since the Simulators are the actual three-dimensional model, we use those that define the space required for the largest rendering. Although it is possible to feed the optimizer with multiple targets, we only use the first one for visualization. The optimizer can be opened with a double click, where the settings for the optimizer can be set and the results table can be displayed.

Since the visualization tool depends on the results of the optimizer, the two are connected by a single wire. The visualization component depends on a correctly built model from the optimizer, otherwise it will exit with an error. The visualization tool can be opened with a double-click and uses the input from the optimization component.

[1] https://grasshopperdocs.com/addons/opossum.html

[2] https://www.food4rhino.com

[3] https://www.food4rhino.com/en/app/karamba3d

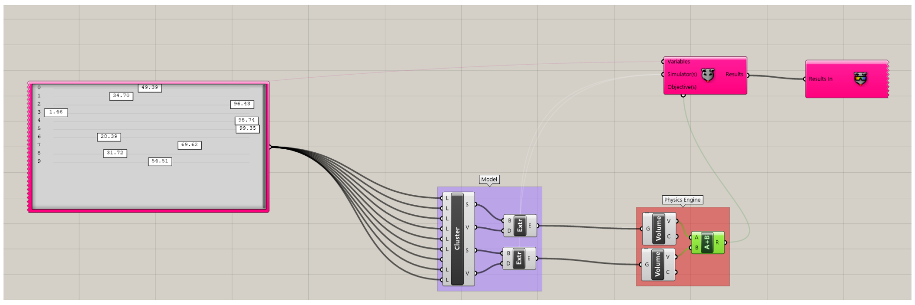


Figure 5.1: An example structure in Grasshopper showing the connections which are mandatory for our tool to work.

In Figure 5.1 one can see the mandatory connections. The variables in this example are combined in a gene pool shown on the left side with pink outline and is connected with the pink wire. The Simulators are created in the purple group and connected with the optimizer with white wires. In this example there are two models created both connected to the optimizer. The objective value is calculated in this example by computing the volume of the two models. They are grouped in red and labeled with physics engine. The objective value is connected with a green wire. The optimizer then is connected with a single wire to the visualization tool in the top right corner.

5.2 Program Structure

Figure 5.2 shows the overall structure of the interaction loops. In this Section we will explain how the user interaction is handled in the backend. The focus is on having a usable experience when using the tool even on mid-range hardware. This implicitly forced us to precompute parts for the visualization during tool startup.

When the tool is started we **generate** the **grid** levels as described in Section 5.3. Along with the grid we calculate the bounding box of each model configuration in the dataset coming from the optimizer and store the largest one. The perspective view bounding box is used for the RenderView ensures that all models are comparable in size when the bitmap is shown in the RenderView. As soon as the grid levels are built, the **continuous map** is **generated**, therefore a bitmap is calculated based on the selected color palette. The bitmap has the same resolution as the main view to minimize computational effort since it is currently not possible to enlarge the bitmap.

By preparing the ContinuousView, the precalculation is completed. Now the **background layer** in the main view can be **updated**. By default the GridView is opened showing the highest level showing the uniform grid with 8×8 glyphs. Each entry in the grid point array which has points inside the convex hull will be displayed, given the grid point entry of the current level. When the ContinuousView is shown, the previously calculated bitmap will be drawn inside the background canvas.

The following part of the program is executed for the center point of the main view once and repeated for the mouse position afterwards. We **update** the **fog array** at first. For this, we map the location in the canvas to the corresponding entry in the fog array and update the values around this entry as described in Section 5.5. Afterwards we **update** the **portal inside the grid** by clearing

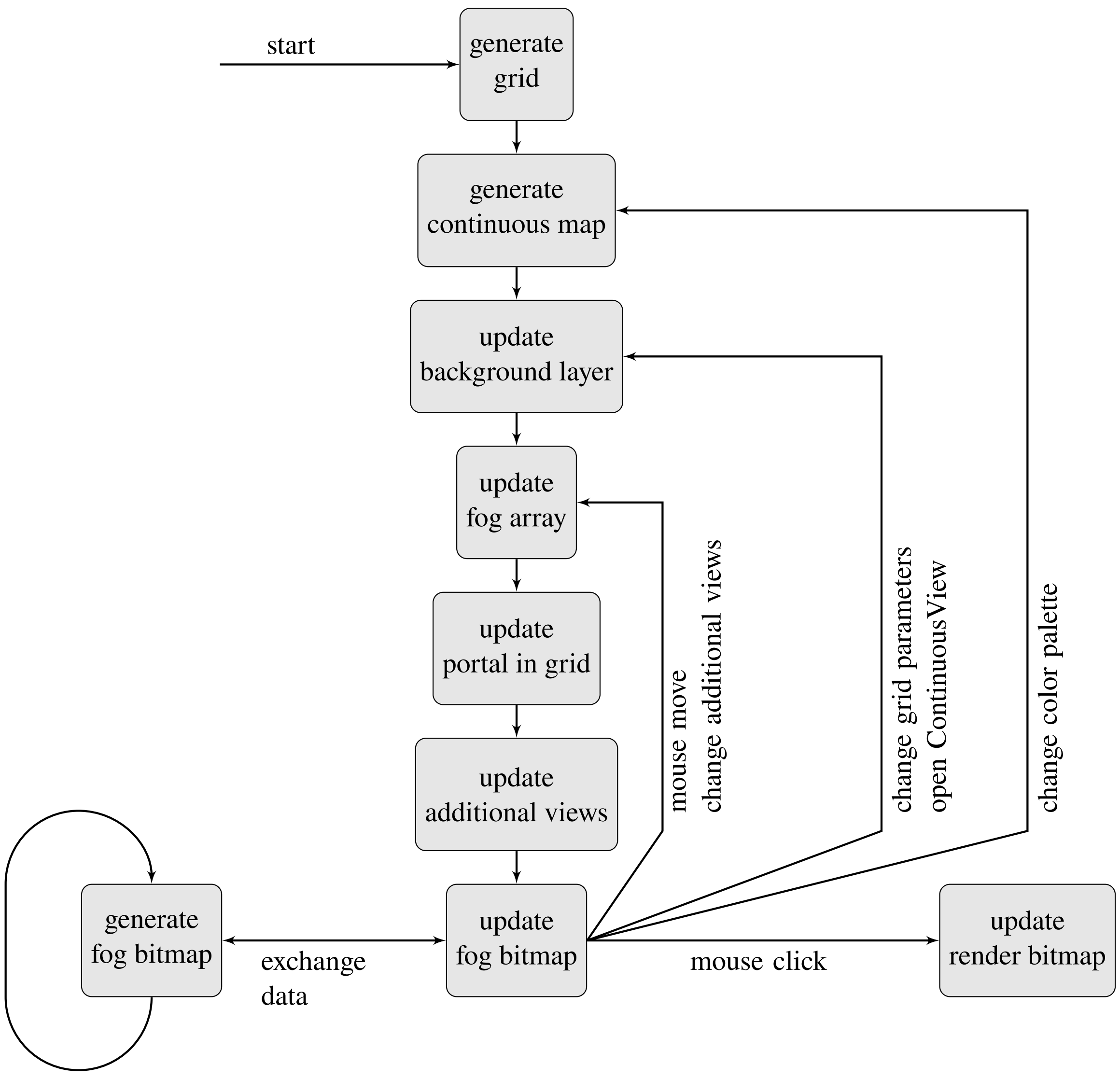


Figure 5.2: The program structure shows how the visualization tool handles interactions to keep latency low.

the corresponding layer and plotting the new glyphs around the mouse location. For each of the **additional views** we search for canvases tagged respectively. If there is a canvas with a specific tag, we compute the content which should be displayed. If the ContinuousView is activated we still use the grid to update the additional views. Therefore, the lowest grid is used, which decreases the latency in the pipeline and because each grid point is shown for only four pixels, it is therefore correct for this location. Since it is nearly impossible to point at a singular pixel, the drawback of having a slightly non-continuous interaction rather than a grid which has to be interpolated at an additional level, which doubles the precomputation step, should be acceptable. We scale the content to best fill the canvas and display it there. To conclude one interaction loop, the FogView will be displayed in front of the GridView or ContinuousView if it is selected to be visible. Therefore, the bitmap is updated from an external thread is scaled to the size of the canvas in the main view and the opacity of the displayed bitmap is set based on the grid settings.

When there is no interaction, our visualization tool will idle and wait for a setting to be changed, the mouse pointer to be moved or a mouse button to be clicked. The pipeline shown in Figure 5.2 will be used for any interaction. It is only possible to start the interaction pipeline earlier for more significant changes. This approach helps us to have consistent updates and prevent us from calculating the same things twice or more in one iteration.

For light interactions like simple mouse movement only necessary parts of the visualization are redrawn. When the grid level is changed or the continuous is selected, the background layer has to be redrawn and therefore we have to start the pipeline in an earlier state. Since the continuous map is rendered as a bitmap depending on the color palette, we have to recalculate the map as well if the palette is changed and we have to start the pipeline with the generation of the continuous map. The grid levels store the actual values rather than the color information This means we do not have to generate the grid again as long as the tool is alive. The user is also able to lock the lens which means the portal does not have to be updated. This is why it is possible to skip the **update portal in grid** step as long as the lens is locked. This is however the only part of the pipeline which can be skipped.

The only interaction which is different from the others is the RenderView. Whenever the RenderView is visible and the left mouse button is clicked, the visualization will start to **update** the **render bitmap** and locks until the bitmap can be shown in the corresponding additional view.

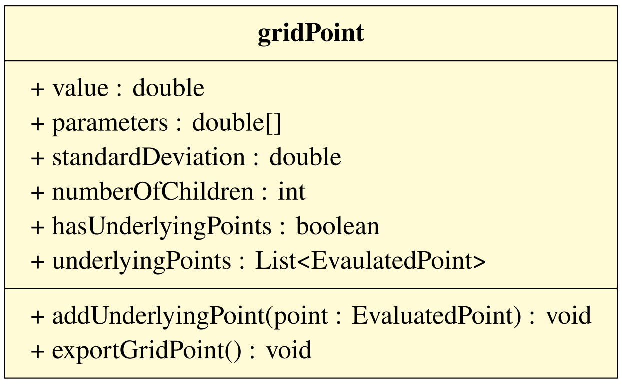
With the start of the tool another thread is started at the same time. This additional thread frequently copies the fog array to a local array in a transactional manner. It will **generate** a new **fog bitmap** given the current color map and write it back to the bitmap owned by the main thread again as a transaction. When one of the two transactions can not be performed because the main thread reads or writes the object, the iteration will be skipped. This behavior is very unlikely and leads to an old fog map shown in the main thread. However, it is still valid with a correctly scaled color palette and eventually the FogView will update to the new version.

5.3 Grid Levels

Since the interpolation in the Barycentric Coordinates is not fast enough to calculate on demand along with a responsive application, we precompute all the data which can be visualized in the GridView. In this Section we will describe how the precomputation is performed when the tool is started.

We have calculated six levels of grids, starting with the finest one containing 512 entries in each of the two dimensions. The highest level is merged for 5 times having 8 entries per dimension. Our visualization tool has a fixed height of 1000 pixels, displaying the lowest grid in the main view each glyph would therefore have a size of less than four pixels. The different levels bring the continuous performance map in a multi-scale representation. Each higher level contains half of the entries in both dimensions, therefore each level contains in total a quarter of the entries of the lower one.

The architecture of one grid point can be seen in Figure 5.3. Each grid point contains the objective value and the list of parameters both being calculated by the mean of the underlying points in the surrogate model. One grid point is responsible for the area underneath the glyph which will result from it being displayed, this are increases for higher levels. The standard deviation is calculated from the objective values the grid point is responsible for. The number of children can vary from



```
gridPoint
+ value : double
+ parameters : double[]
+ standardDeviation : double
+ numberOfChildren : int
+ hasUnderlyingPoints : boolean
+ underlyingPoints : List<EvaulatedPoint>
+ addUnderlyingPoint(point : EvaluatedPoint) : void
+ exportGridPoint() : void
```

Figure 5.3: The class diagram modelling one grid point.

zero up to four in our levels. A grid point can contain zero children only when it is in the lowest level and four children will be the default of all of the other levels. However, when there are not enough children to merge there will be less. This occurs in regions where the area of a grid point is not completely inside the convex hull span by the dataset mapped with *Star Coordinates*. We want to have the opportunity to show in which areas real objective values were evaluated. We keep track of the evaluated points with a list of them. In addition we store whether there are points at all facilitating access of the information needed for the visualization. Each grid point can be exported in a file, storing the information stored in the grid point for later inspection.

The initial grid representing the lowest level is generated by interpolating the surrogate model. Therefore, for each grid point the values of the four underlying points are calculated. The lowest grid having a resolution of 512×512 pixels we interpolate the continuous map with a resolution of 1024×1024 pixels. For the four objective values and the parameter values the mean is calculated and stored. Since we calculate with a higher density, we can calculate the standard deviation of the four values. Each grid point votes for a rectangular area underneath the glyph which will be shown at its location. If there is an evaluated point we add this point to the list of underlying points.

The higher grids are generated by merging grid points from the level below. We merge the grid points in general with a similar approach than the gaussian pyramids [E H84]. The objective value and the parameters are calculated with the mean of the underlying grid points. For the underlying points all grid points are merged by adding the lists to one and store them in the grid point.

5.4 Layering

Our visualization tool is started within Rhino 7 and shares resources with the main program allowing us to have access to the inputs of the component. We encountered serious performance issues. We had especial problems displaying and computing glyphs in a grid with 128 glyphs in each directions which has a maximum of 16,384 glyphs to be computed with a reasonable refresh rate. However, since the grid does not have to change in each frame, this overhead can be avoided by displaying the grid only once and remaining constant until the grid level is changed. Only updating glyphs moving with the mouse location drastically reduces computational effort. Therefore, we used a layer based approach, this gives us an efficient and easy way to redraw only certain parts of the visualization. Each layer is a transparent canvas in the Windows Presentation Foundation (WPF), since it is easier to redraw everything from a single canvas than moving only some of the objects of a canvas.

The layering approach comes with additional benefits. In one canvas the objects are drawn on top of each other which makes it impossible to show objects behind others without having to draw them over and over again. We avoid this with our approach since the same behavior is true for complete canvases. We add multiple canvases in advance based on the number of layers we need for each view, wherein the order of these layers will not change. Based on the configuration chosen by the user the layers are tagged with the content it should contain in the next iteration.

For the tagging of the canvases we use a list of references to the canvases. We add a list for each role a canvas take. For example there is a list for the background of the GridView and one for the foreground of the GridView. Respectively for the additional views there is one list containing all canvases showing the portal content and so on. This enables us to calculate the content of each different view only once. For each canvas, the content just has to be scaled according to the size of the canvas.

5.4.1 Main View

In the main view we have four different layers each one designated to fulfill a different job. The background layer showing data which does not change frequently, the layer above containing the interactive objects moving constantly together with the mouse pointer. On top of that, the next layer contains the FogView if it is enabled and in the foreground the dropdown menus are located to be never overdrawn by other objects.

The layer in the background shows either the performance map if the ContinuousView is enabled or the grid if the GridView is chosen respectively. This layer has no direct interaction and can be only triggered with the dropdown menus changing the color palette or between continuous and grid representation. In the GridView the indication type of evaluated points can be selected and through the configuration panel changing the grid size or the uncertainty indicators where the borders can be deactivated or inverted to highlight areas with smooth objective values. Since these are events that do not happen all the time, it is possible to have changes in this canvas to take up to 400 ms for finer grids, but most of the time they are static.

On top of the background layer the interaction layer is located showing the indications of portal size, the portal level and if it is locked or not. If the portal is drawn on top of the grid, the content of the finer grid is shown inside the circular portal expanding only the glyph which is next to the mouse pointer. In this canvas we can force the lens indicators and the semi-opaque background to be drawn behind the portal content itself, since we are able to draw the background before the content. This canvas is responsible to track the mouse movement, since we do not have to redraw this canvas if the mouse location changes in the outside. In the ContinuousView this layer is empty but tracks the mouse movement as well, since the LegendView and others have to be updated anyway. In addition this layer tracks clicked mouse buttons as well and changes the program flow as described in Section 5.2. This reduces again the computational effort while the mouse is out of the main view.

The FogView in the main view is drawn in front of the first interactive layer. Showing a bitmap which is generated constantly in the background and updated with each mouse movement. The opacity of the fog is defined by changing the transparency of the bitmap which is placed in this canvas. Therefore, the fog can be created as described in Section 5.5 without taking the definitions in the configuration panel into account.

Above all of them we show the four dropdown menus in each of the corners. Since they have their own events triggered by hovering over them the interactive canvas underneath will not be triggered.

5.4.2 Additional Views

All of the three configurable additional views are build in the same manner and differ only in their size and location. One additional view contains three layers where the foreground layer only includes the dropdown menu to switch the content of the view. The other two layers can be used individually and can be updated individually to reduce the number of objects which had to be drawn in each iteration.

In the LegendView we use the background layer to display the color scale as a set of rectangles having a height of one pixel. Together, these rectangles form a legend fading correctly as the color palette defines. In total, we display at ten locations equally distributed over the height of the complete legend the value assigned to the color. In the foreground we only have to update the objective value of the currently hovered value in the main view or the PortalView in another additional view.

The FogView uses two layers the upper one showing two buttons to export the heatmap and to reset the generated heatmap. In the layer underneath the generated bitmap is stretched in both axes by the same factor to fill the available space without giving a wrong impression, changing the aspect ratio. The bitmap is created on demand and may take some time depending on the model. It is created as described in Section 5.5.

The PortalView uses only one layer. The content is defined in the same way as in the main view. However, the rectangular canvas can be used completely to show the glyphs in a size that fills the view.

The RenderView also uses only one layer and shows the generated bitmap in the addition view. The bitmap is created on demand and based on the model could take some time to show it is created as described in Section 5.6

5.5 FogView

For the FogView each movement of the mouse is added to a two dimensional array. For performance reasons we use a double array with 200×200 entries. We calculate a gaussian normal distribution around the mouse location with a standard deviation being the site of the portal. This value is multiplied with the time the user has spend at this location. For each array entry within the standard deviation around the nearest array point the value is added.

Listing 5.1 Code showing the update of the fog bitmap running inside a while loop.

```
// gridarray to bitmap
for (int i = 0; i < fogArray.GetLength(0); i++)
{
    for (int j = 0; j < fogArray.GetLength(1); j++)
    {
        Color color = ColorPalettes.GetScaledColor(-fogArray[i, j], colMap);
        fogBitmap.SetPixel(
            i, j, System.Drawing.Color.FromArgb(color.A, color.R, color.G, color.B));
    }
}

System.Drawing.Rectangle drawRect = new System.Drawing.Rectangle(
                  0, 0, fogArray.GetLength(0), fogArray.GetLength(1));

try
{
    fogBitmapDraw = fogBitmap.Clone(drawRect, fogBitmap.PixelFormat);
}
catch (Exception e)
{
    System.Diagnostics.Debug.WriteLine(e.ToString());
}
Thread.Sleep(100);
```

We show this array as a heatmap with the same color palette chosen for the performance map scaled linearly from zero to the maximum value in the array. The heatmap has to be displayed efficiently. This is why we compute a bitmap with the described color palette. We can not create this bitmap with a high frame rate because it is not mandatory to have the last interactions in the heatmap. We compute the bitmap without blocking the update loop by generating a bitmap on a new thread with a refresh rate of approximately five frames per second. This thread runs in the background without stopping until the visualization tool is closed.

Since it should not be possible to have the main update loop interact with the array at the same time as the background thread and the bitmap should not be changed while it is used in the main loop to be displayed, we copy the array atomically and only create the bitmap using the copy. We use the same approach for the bitmap, this atomically created copy can only contain correct arrays and valid bitmaps.

In listing 5.1 the code generating the bitmap for the FogView is shown. Here the color creation and pixel changing for 40.000 pixels in the array with 200×200 entries take up the majority of the time. When another bitmap is generated the program will try to overwrite the bitmap. This procedure will fail if the bitmap is currently drawn, which is extremely rare and did not happen once while testing. However, if the procedure fails, a new bitmap will be generated in the next iteration. This eventually update the bitmap in the main thread.

In the FogView this bitmap is stretched to fill the most space in the canvas as possible without manipulating the aspect ratio. In addition to the shown bitmap in the additional view, the heatmap can be saved to disk using the export button. The bitmap is stored next to the open grasshopper file.

Next to the heatmap we add a sequence of the visited data points as a csv file which can be used for further evaluation. The reset button sets all values inside the array back to zero and a new heatmap will be generated.

5.6 RenderView

The RenderView is the only view which is included in our tool and is not able to update immediately with the user looking trough the shown performance map. The model has to be created inside Rhino 7 for the viewport to update and show the correct model. This view i the only view relying on the actual Grasshopper file to be open in Rhino 7 as well. This means that the user cannot open multiple instances of Grasshopper at the same time to compare different configurations.

In our implementation the user should know that he actively asks for a new model being build. The user has to click with the left mouse button to confirm that he wants to see the model in our RenderView. As soon as the user clicks, the parameters will be set in Grasshopper to be the one shown in the data point. The model will then be created which can take some time. When the model is built, the viewport is set to fit even the largest possible model which can be achieved in the surrogate model. In Rhino 7 a screenshot of the viewport is saved as a bitmap which is then shown in the RenderView. The RenderView however can be seen in the three dimensional space inside the actual Rhino 7 application for further investigation. For each displayed image of the model, the viewing direction from the perspective view in Rhino 7 will be used.

The code in listing 5.2 will be run every time the user triggers the RenderView to update the image. The majority of the runtime is used by updating the sliders with the active model. Afterwards a screenshot of the perspective view in Rhino 7 is taken. The loop in the end of the function shows the procedure of showing the content on every canvas in the list. This list contains all canvases which are tagged to be RenderView and are in the correct layer being responsible to show the content. The update is done in this way in all additional views according to the list they are part of.

Listing 5.2 Code showing update RenderView. Interacting with the Rhino 7 application and display content in every associated canvas.

```
// update model
UpdateSlidersSimulated(dap.getParams());

// set bounding box and take screenshot
myActiveView.ActiveViewport.ZoomBoundingBox(perspectiveViewBoundingBox);
System.Drawing.Bitmap bitmap = myActiveView.CaptureToBitmap();

foreach (Canvas c in canvasListBitmapMid)
{
 img = new Image();
 Int32Rect int32Rect = new Int32Rect(0, 0, bitmap.Width, bitmap.Height);

 img.Source = System.Windows.Interop.Imaging.CreateBitmapSourceFromHBitmap(
    bitmap.GetHbitmap(), IntPtr.Zero, int32Rect,
    BitmapSizeOptions.FromEmptyOptions());

  c.Children.Clear();
  img.Width = c.Width;
  img.Height = c.Height;

  c.Children.Add(img);
}
```

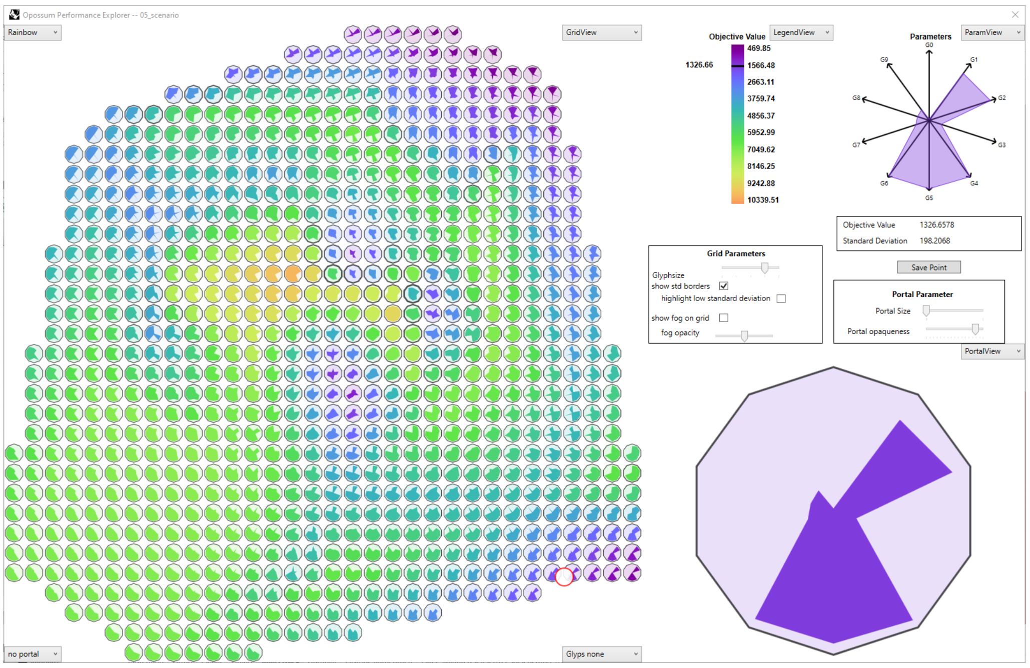
6 Use Case Study

In this Chapter we will give an example of how to use our visualization tool. The underlying model created is artificial and generates three different rectangular boxes, where our goal is to have a large volume.

6.1 Getting an Overview

When we start the tool, we can see the GridView that gives us an overview of the entire dataset shown in Figure 6.1. At first glance we focus on the main view, here we can see that there are three circular areas with a purple color in the center of the map. If we look at the LegendView, we learn that we should avoid the purple areas to complete our task.

We should look at the more orange areas that give us higher objective values. We can see that there is a region with several yellow and orange glyphs to the left of the center. Slightly to the right of the center there is another glyph with a light orange color.



Objective Value 1326.6578
Standard Deviation 198.2068

show std borders ☑
highlight low standard deviation ☐
show fog on grid ☐
fog opacity

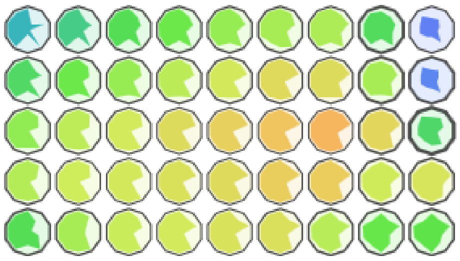
Figure 6.1: Overview of the dataset, three purple regions in the center and one larger region in orange can be seen.

6.2 Find Similar Glyphs

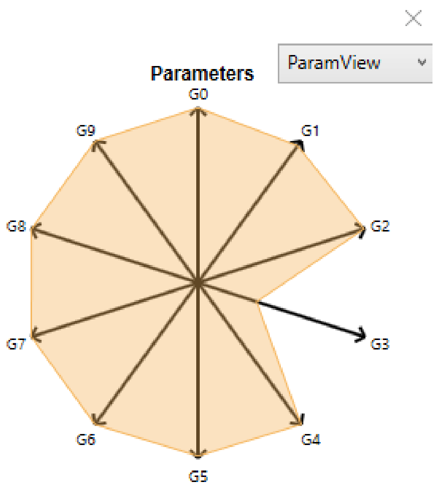
In the orange region, it is possible to see similarities in the glyphs. Here, all orange glyphs have a similar shape with only one dent. In Figure 6.2a the previous overview is enlarged. This means that all parameters are maximized, expect one. If we move the mouse over it, we can see in Figure 6.2b that the parameter that is not maximized is called G3.

We can also see that the glyphs that change color to yellow have at least one parameter that is smaller than in the orange glyphs. In this case they are G1 and G5, depending on the direction of the orange glyph. With this, we can already hypothesize that G3 does not have to be large to have a high target value. However, we do not know if it has to be small to achieve this.

Therefore, we should dig deeper and find representative glyphs where G3 is maximized together with a large target value.



(a) Orange region from the previous dataset. All glyphs in orange share a similar shape.



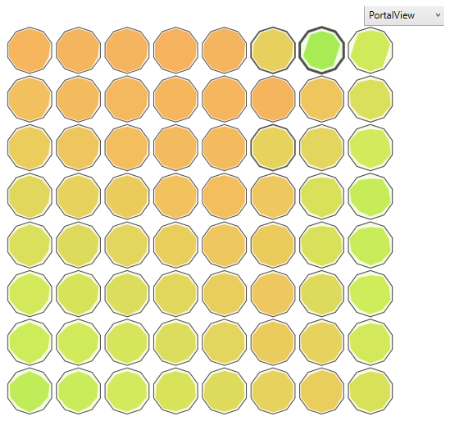
(b) The parameter being low can be identified as G3.

Figure 6.2: More detailed information about the region can be detected.

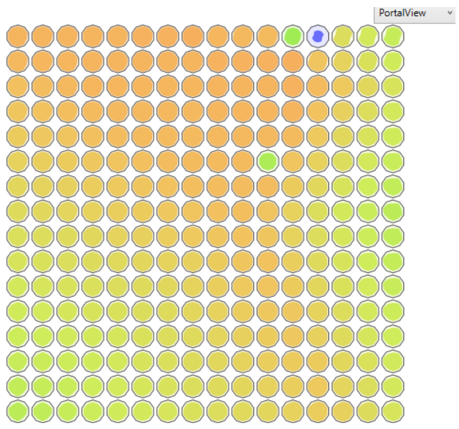
6.3 Find Representative Glyph

As mentioned earlier, the light orange glyphs were visible in the overview in Figure 6.1. We can use the PortalView to see the glyphs hidden in a finer grid. In Figure 6.3a we can see that there are glyphs with a similar orange color that were not visible before. Using the right mouse button to lock the lens, we can examine that the actual objective value is higher than 9800 and in the same range as in the other region, with objective values between 9600 and 9900. This leads us to conclude that the G3 parameter does not affect the lens value.

Within the PortalView shown in Figure 6.3a we can see an obvious outlier with a high standard deviation and a green value. We can investigate this further by visiting a finer grid, where we can see that it hides a purple glyph with much lower parameter values.



(a) Second to lowest level shown in PortalView.

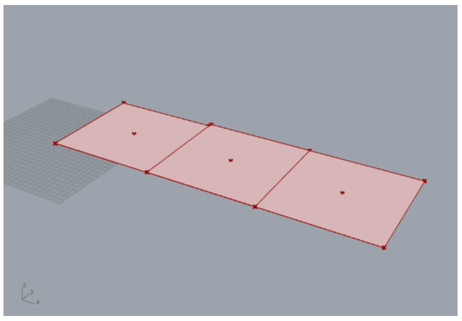


(b) Lowest level shown in ParamView.

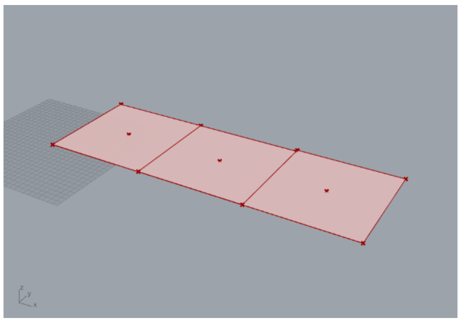
Figure 6.3: Light orange glyph on the right, expanded in the PortalView.

6.4 Confirm in RenderView

In the RenderView we can also confirm that the two regions result in similar models. In this case they are three flat cuboids stacked together, in real world scenarios there would be more complex structures visible in the RenderView. Therefore, we can ask the tool to generate a model in three-dimensional space and visualize it in the tool.



(a) RenderView from the darkest orange glyph in the left region.



(b) RenderView from the light orange glyph in the right region.

Figure 6.4: RenderView from two data points.

6.5 Low Objective Values

To complete the inspection, we can take a closer look at the regions with low values. In the Figure 6.1 we can already see that the glyphs with a purple color have very small polygons. We could assume that the other parameters all need to be maximized to get good results. This is hard to prove, however, because the optimizer only expanded the search space in a direction where an optimum can be assumed. Since there was no reason for it to evaluate regions with bad values, the assumptions for values in regions far from the optimization goal are very vague.

7 Expert Evaluation

We conducted an expert study with participants with expertise in architectural design. The purpose of the study was the following. First, the participants shoud be familiarized with the visualization tool. In addition, the study was to find out how our additions are accepted and used. In this Chapter we will state how the study is designed, what our expected outcome is, how the expert study was performed, report the results of it and finally discuss our hypotheses.

7.1 Study Design

Our study is designed to focus on usability and correctness. We want to answer the following questions:

Q1 (Usability) What configurations are preferred? Which configurations help people to solve the task subjectively?

Q2 (Correctness) What configurations have an effect on the correct solution of the problems?

Our goal is to compare the continuous map with the portal used in the grid and the portal shown in an additional view. There are three configurations locked to ensure the user to use the desired configurations. The three configurations used during the study can be seen in Figure 7.1 and are explained in more detail below.

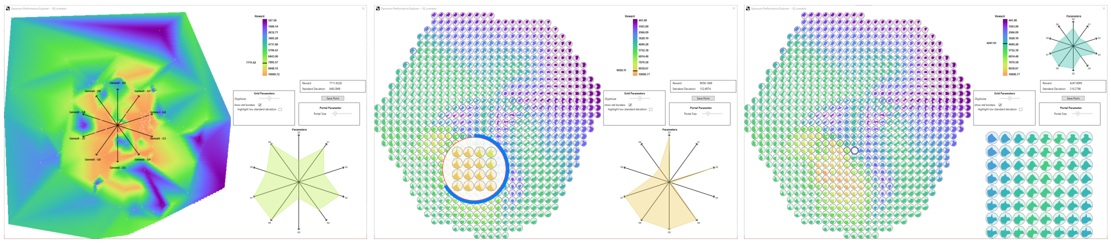
C1 (ContinuousView) The student can use the ContinuousView, LegendView, and ParamView in the largest additional view.

C2 (GridView Portal in Grid) The attendee can use the GridView with a fixed grid size, the portal is displayed within the grid itself. He can adjust the size of the portal and the level of detail. In addition, they can see the LegendView and the ParamView in the largest additional view.

C3 (GridView portal in external View) The attendee can use the GridView with a fixed grid size, the portal content is displayed in the largest additional view. The participant can change the level of detail shown in the portal. They can also see the LegendView and the ParamView, in this configuration the ParamView is moved to the top right corner.

For each configuration, we use a different black box function to avoid reusing solutions from previously seen configurations. Each black box function has ten input parameters and a target value between 0 and 10000. The three functions have the following characteristics:

F1 There are eight parameters that have a positive influence on the target value. The remaining two parameters are zero weight and have no effect on the objective.



(a) ContinuousView **(b)** GridView Portal in grid **(c)** GridView and PortalView

Figure 7.1: The three configurations used during the study.

F2 The individual weighting of the parameters is changed slightly. There are two different parameters that do not affect the result.

F3 In this scenario, there are two parameters that have a negative impact on the objective and need to be minimized. Again, there are two different parameters that have no impact. The six remaining parameters lead to a high objective value.

These black box functions are evaluated by the *RBFOpt* optimizer, which optimizes for the maximum value, before the actual study. This makes the study independent of the non-deterministic optimization, because it is run only once and the same dataset is shown to each participant.

The participants are asked to complete the following five tasks for each configuration – with slight modifications in the ContinuousView described in Section 7.2:

T1 Find two points in the displayed overview with an objective value above a given threshold.

T2 Find two points in the displayed overview with a standard deviation above a given threshold.

T3 Find a point in the portal with an objective value above a given threshold. The threshold could not be met only in the overview grid being shown.

T4 How many regions with a high objective value (given threshold) can be distinguished?

T5 Name two parameters that do not affect the objective value – Which parameters can be changed in their value without changing the objective value?

We choose the tasks **T1**, **T2** and **T3** to familiarize the user with the tool. If he performs these tasks correctly, we can assume that he is able to use the tool. Since the scores and standard deviations are displayed, it is unlikely that a student will fail this task. The tasks **T2** and **T3** are added because the functionality of finding maximum values was implicitly given in the ContinuousView and is missing in our approach. We want to see how our additions work to compensate for this inconvenience.

With the task **T4** we want to get a deeper insight into the user's understanding of the visualization technique. While it is possible to detect exactly one region in each configuration and function, it is still possible to distinguish some regions with outstanding parameter values.

The task **T5** is the main task in the study, the participant has to face the parameter values to find the solution. In a real application, the knowledge of parameters with less impact on the objective value can lead to more freedom in aesthetic adjustments.

7.2 Study Procedure

In this Section we give insights about the participants and the procedure of the study itself.

For our expert study we invited students and staff of the Institute for Computational Design and Construction of the University of Stuttgart (ICD)[1]. We had 9 participants in total of which 4 were female and 5 male. Of these 5 had a bachelor's degree and 4 a master's degree. None of the participants had received any compensation for their participation.

The participants had different knowledge of the original tool *Opossum* from Wortmann [Wor17b]. All participants were familiar with the surrogate model generated by *RBFOpt* and had seen the original *Opossum* at least once. Therefore, all of the participants knew the basics. However, 4 participants used the tool more often but not as part of their daily work, one participant considered himself an expert.

Given our small amount of participants the study used a within-groups design. Each participant had the same tasks and was given the same configurations only varying in the order the configurations where given.

Before starting, each participant completed a demographic questionnaire about their gender, education, knowledge of multivariate data visualization, and knowledge of the *Opossum* tool. Each participant then performed a color vision test using the "standard Ishihara color plate test"[2]. None of our participants were aware of having a color vision deficit, and we were unable to detect one with our test. At the beginning of our study, we introduced the artificial problem and explained to the participants that the first configuration was the ContinuousView. After that they had time to familiarize themselves with the tool, we did not have a time limit for this, but it usually took less than two minutes. When they were ready, they were given the tasks to solve, namely **T1**, **T4** and **T5**. We skipped the other two tasks in this configuration because there is no need to expand the glyphs to find better solutions, since they are already visible, and high standard deviations are almost impossible to find due to the high resolution of the map. After solving the tasks, they filled out a questionnaire about the difficulty of each task and had the opportunity to write down which strategy they used to solve these tasks.

For the second configuration, we randomly chose whether they would solve the tasks using the external portal or the portal inside the GridView. Again, we gave a brief presentation of the tool and the configuration, after which they had time to interact with the tool. Now the participant solved all five tasks and filled out the small questionnaire. For the third and final configuration we had the same procedure as for the second.

After all three configurations, the participant was asked about their preferred configuration for solving such tasks again. To summarize the study, they filled out a SUS [Bro95] questionnaire for each of the configurations.

For the participants we recorded the screen and the room audio, this helped us to analyze the behavior shown later. Throughout the study, participants were asked to voice their thoughts and questions. We also took notes of any outstanding behaviors or basic questions.

[1]`https://www.icd.uni-stuttgart.de`

[2]`https://colormax.org/color-blind-test/`

The entire process for each participant took an average of 50 minutes and never took longer than an hour.

The final design of the study resulted of a pilot study. Three participants from the Institute for Visualization and Interactive Systems of the University of Stuttgart (VIS) took part in the pilot study. We planned to compare even more configurations and more tasks. However, our pilot study showed that for new users, not working with the original tool on a daily basis, the tasks took way longer than initially expected.

7.3 Hypotheses

We expected that the configurations shown to the user would change the behavior of solving the tasks. This would lead to a difference in user experience and the correctness of the tasks which were solved.

For the question **Q1**, regarding the influence of configuration on the user experience, we expected different results based on the solved tasks.

In the task **T1**, where the user has to find a point on the map with a certain value, we expected the ContinuousView to perform best. Within the continuous map, the participant can use a kind of gradient descent to find a value that matches our constraint. Unlike the GridView, there are no distracting shapes and gaps between data points, making this task particularly easy. In addition, we expected the portal's location within the grid to be advantageous for this task, since it is possible to have the glyph currently under the mouse pointer visible within the overview grid. In contrast, when the portal content is displayed next to the browse, the glyphs hide the browse to some extent.

The task **T2** can only be compared between the two GridView alternatives. For the same reason as for task **T1**, we expected the portal in the grid to perform better in the case where a high standard deviation has to be found.

In task **T3**, the user has to dig deeper into the grid and find a target value that is higher than the one accessible once in the overview grid. Here we expect the external portal to be beneficial. The external portal helps to solve the task by having generally larger glyphs in the separate view. In addition, the portal does not block the overview of the grid, making it easier to find new glyphs if there is no solution below the currently selected and enlarged glyph.

For task **T4** we assumed that the participants prefer the ContinuousView the most because it is easier to find isolines in the continuous case. This would help to detect the outlines of regions. In this task we see an advantage in the internal portal having the finer grid shown on top of the overview can help to identify borders in higher definition if the portal size is set correctly.

In task **T5** the goal is to identify parameters not changing the objective value. We expect the ContinuousView to perform the worst in this task since the parameter values are not visible in the overview, each parameter set has to be checked with the ParamView. Here we assume the GridView to have major benefits, glyphs can be compared in the overview giving a first hint of glyphs with same objective values having changing parameter values. When the participant will try to confirm the potential solution they will look into finer grid levels using the portal. Since the parameter values are important for this task, we assume that the external portal with the larger glyphs will have benefits.

We state the following hypotheses for **Q1**:

H1.1 People would like to use the ContinuousView the most for the task **T1**, and the portal inside the grid has a slight advantage in the task to find a high objective value.

H1.2 Both alternatives would lead to similar results again, with a slight advantage for the portal in the GridView when looking for high standard deviations.

H1.3 The external view would provide a better experience for finding a higher objective value that is not shown in the overview grid.

H1.4 People would prefer the ContinuousView because more isoline detection would make them more confident in their decision when finding regions. There is a use case where the internal portal helps more than the external one.

H1.5 The larger glyphs with the external portal will have an advantage over the internal portal. However, both will perform better than the ContinuousView when searching for parameters that do not change the objective value.

With the question **Q2** we want to find out if there is a correlation between the chosen configuration and the correctness of the answers.

The tasks **T1**, **T2** and **T3** can be evaluated by the participant. There are no incorrect answers in the first three items, so we cannot answer the question based on these items.

In the **T4** task, the respondent looks for distinguishable regions on the map. The task itself is difficult to evaluate as strictly correct or incorrect, since our functions are rather smooth. However, we assumed that the participant chooses the regions based on a different strategy. For the ContinuousView, we expected the regions to be detected based on isolines, so we assumed that regions are interpreted as locations of connected colors. For the GridView, we expected that the participants separate different regions based on the shape of the glyph rather than the location on the map. We assumed more accurate results for the external portal because the glyphs are larger and therefore easier to compare.

For the **T5** task, we expected the GridView to have a significant advantage over the ContinuousView because there are glyphs with similar target values with different parameter values. In the ContinuousView the participant would have to manually ask for the parameter values by hovering over the map with the mouse. The external portal again has the advantage that the larger glyphs in the portal provide a clearer distinction even at lower levels.

We state the following hypotheses for question **Q2** and the tasks **T4** and **T5**:

H2.1 The found regions would be based on isolines in the ContinuousView and based on the glyph shape in the GridView, with the external portal providing a more accurate answer.

H2.2 The external portal would give the most accurate results, followed by the internal portal. In the ContinuousView it will be harder to compare glyphs giving the worst results in comparison.

These hypotheses are discussed in Section 7.5 based on the following study results.

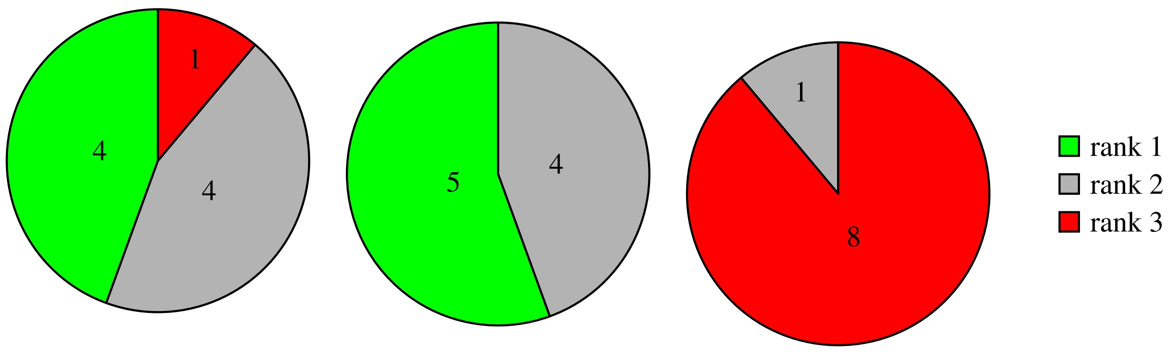
7.4 Results

We had four different sources of results. These were the participant's preferred configuration, the ranking of the difficulty of each task in each configuration, the SUS for each of the configurations and the correctness of the task **T5** in each configuration. Furthermore, we show some comments and feedback of the participants.

7.4.1 Preferred Configuration

In Figure 7.2 the user ranking for each configuration is shown. Participants had a strict preference for the GridView, either the portal inside the grid or the external portal are liked the most. There is only one participant which would prefer the ContinuousView over the GridView with the portal next to the grid. Here the participant argued that it was hard to remember to lock the lens before changing from the grid to the portal. Since the user scrolls trough the grid when the portal is not locked, he will loose track of the position.

Another interesting fact is that for the four participants having the original tool used more than once before there is a strict preference of using the external portal. All of them had the external portal ranked to be the best and the ContinuousView ranked the least.



(a) GridView with external Portal

(b) GridView with internal Portal

(c) ContinuousView

Figure 7.2: Ranking of Configurations preferred to solve the tasks.

7.4.2 Task Difficulty

In this Section we compare how the participants classified the tasks in their difficulty. Here we have five categories encoded from red on the left having a very hard task via a light red color indicating a hard task. In gray we encode votes for the task to be classified as doable. The light green indicates a task to be seen as easy and in the darker green that the task was classified to be very easy.

For the task **T1** there was no difference in the configurations we had each time one vote that the task was easy and the rest concerned the task to be very easy.

In the second task **T2** there is again a majority in both configurations of the GridView that voted this task to be very easy. However, in this case there are outliers as shown in Figure 7.3. Here one can see that for two participants it was difficult to see the standard deviation in the external portal.

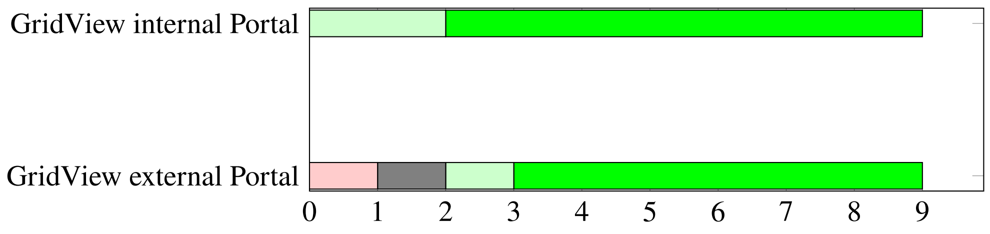


Figure 7.3: Task Difficulty for task **T2**

The third task **T3** shows that finding a glyph hidden in lower levels was much harder as shown in Figure 7.4. In the data one can see that it there are in total eight participants judging this task to be doable or even hard. This behavior results from one scenario where a glyph with a high objective value was constantly overseen.

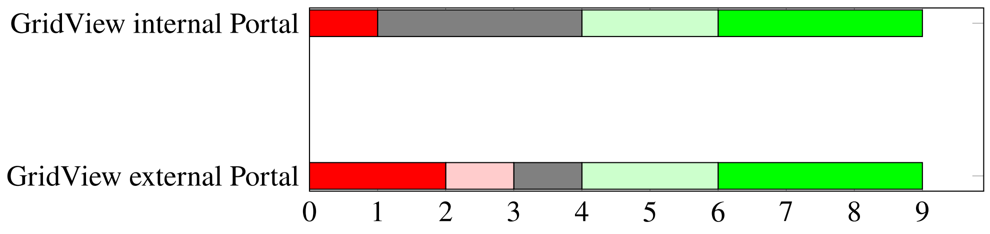


Figure 7.4: Task Difficulty for task **T3**

The results for task **T4** are shown in Figure 7.5. Here a clear preference for the portal inside the lens can be seen with the majority of participants see the task as easy or very easy. Only two said that it was not easy but doable to find distinguishable regions in the map.

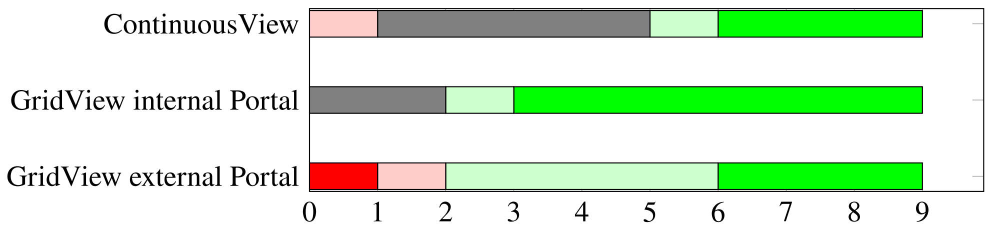


Figure 7.5: Task Difficulty for task **T4**

The task **T5** shows a similar image as the task before, which can be seen in Figure 7.6. Here the portal inside the lens was classified to be the easiest. Again the results for the ContinuousView are mixed with a minority of the participants thinking that the task was easy.

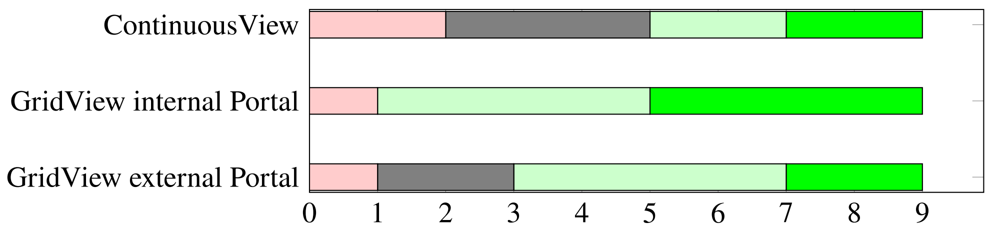


Figure 7.6: Task Difficulty for task **T5**

7.4.3 System Usability Scale

The users filled three SUS questionnaires for each of the configurations. With these questionnaires we were able to calculate the SUS scale which is a value between 0 and 100. Where 0 would mean that the software has considerable usability issues and is the worst imaginable software. A score of 100 would mean that is the best conceivable system with no usability issues at all. All of the three configuration achieved a mean value between 76 and 78 and a median between 72 and 80. We can therefore assume that we got reasonable good results.

In Table 7.1 the results are listed for the three configurations. We can read from the Table that the standard deviation is least for the external portal and both of the GridViews share the same mean. Looking at the minimum value we can see that the ContinuousView was the only one which achieved values below 50 and the external portal could not achieve values below 60.

Metric	ContinuousView	GridView internal portal	GridView external portal
Mean	76	77.5	77.5
Median	80	77.5	72.5
Standard Deviation	13.6	12.7	10.8
Minimum Value	41.5	54	64
Maximum Value	89	91.5	96.5

Table 7.1: Comparison of the SUS scale values for the three configurations

7.4.4 Correctness

For the correctness of the task **T5** the results don't show any anomaly. The participant was asked to find two insignificant parameters, so the maximum number of correct parameters is two. The results were quite similar for all three configurations tested. In Table 7.2, it is only noticeable that the standard deviation is lower for the external portal and similar in the other two cases

Metric	ContinuousView	GridView internal portal	GridView external portal
Mean	1.22	1.33	1.33
Median	1	2	1
Standard Deviation	0.79	0.82	0.67
Minimum Value	0	0	0
Maximum Value	2	2	2

Table 7.2: Comparison of the correctly found insignificant parameters for the three configurations.

7.4.5 Strategies / Feedback

In this Section, we present the strategies used by the participants to solve tasks **T4** and **T5**.

In the task **T4**, the participant looked for distinguishable regions with a high target value.

In the ContinuousView we saw that the regions were mostly distinguished by color. One stated that he used ”color temperature to gather initial search regions followed by a gradient descent”, another wrote that he: ”explor[ed] orange regions randomly”. Most participants stated that they ”trace the color of minimal threshold to identify shapes [regions]”. In summary, all participants used color and location on the map to distinguish different regions.

In the GridView, most participants showed the same behavior to identify regions. Since the grid shows a smaller number of values, most of them were more confident in their answer. Only one participant actually used the shape of the glyphs to identify different regions and was therefore able to classify them by their parameter values. However, the different portals did not show a significant change in behavior. No participant used the portal inside the grid to get a more detailed outline of individual regions.

In the task **T5**, the participant had to find two parameters that did not affect the target value.

In the ContinuousView three of the nine participants stated that they used the ”arrows as guidelines” reading the direction with the most points, with high objective values, as insignificant. Another two participants started with the same approach and followed the axes in the map. One participant was not looking in the main view and kept the focus on the ParamView hovering randomly over the dataset. They confirmed their solution by using the ParamView to identify that the parameters they expect change when moving the mouse pointer. The remaining four participants were using the regions they identified earlier and searched within them to identify if there are parameters which change obviously.

In the GridView, all participants began using the shape of the glyph to inspect parameters. All of the participants looked for glyphs that had a similar objective value and looked for prominent differences in shape. Three of the participants looked for glyphs with low target values in the outer regions of the map. In our optimization runs, these regions were not evaluated enough to get a clear insight, which was the main reason for the bad results.

The only difference we could see between the portal in the grid was that the user was looking for a good compromise between detail and size of the portal. The portal displayed in the additional view was used at the lowest level in all cases. On the other hand, when the portal was in the grid, the level of detail was reduced to have the glyphs in a reasonable size.

The participants were asked to provide feedback on our tool for each configuration.

In the ContinuousView, participants asked for the ability to zoom into the continuous map and to lock multiple axes to restrict movement on the map. For the region detection task, they were looking for an isoline visualization that highlights different heights of the target value. One participant suggested using a Parallel Coordinate Plot (PCP).

In the GridView they stated that for the region detection task was a bit difficult because the glyphs ”blur” the outlines of the regions. Two participants noted that it was much easier to make point-to-point comparisons with fewer data points displayed at once. One participant had difficulty seeing narrow spikes in the star plot, another one stated that it would help if the glyphs could be filtered based on their objective value or standard deviation. For glyphs inside the portal the colors of the glyphs can be very similar, this made it hard for some to find glyphs with high objective value.

In the GridView with the portal inside the grid, the main concern was that it is not possible to change the portal size intuitively. There were some suggestions about how to interact with the portal, which we will discuss later in Chapter 8.

7.5 Discussion of our Hypotheses

In Section 7.3 we stated five hypotheses about the user experience and two hypotheses about the correctness of the task completion. In this Section we will describe how the study supports or discard these. With our expert study having only a small number of participants we have not enough evidence to generalize our results.

For hypothesis **H1.1** we are not able to reject or accept this hypothesis, because we could not find any tendency in our results having the same answers for all three configuration.

In our study hypothesis **H1.2** turned out to be true in our case. The participants had more problems with external portal than with the portal in the grid. However, there was no significant difference.

For hypothesis **H1.3** the results show something different. There were again more participants classified that the task was easier with internal portal.

For hypothesis **H1.4** we can say that identifying regions was easier with the GridView, which rejects our hypothesis.

Hypothesis **H1.5** is not completely true, we can see that the internal portal makes the task **T5**, finding insignificant parameters, significantly easier than the ContinuousView. However, with the external portal, the task is harder to solve than with the internal portal.

For the hypothesis **H2.1** we studied the behavior of the participants. We were right with our hypothesis that most of them used the color coding to identify regions, not the parameters to identify them. The portal was not used for task **T4** and therefore the external portal did not help to find more accurate outlines.

In hypothesis **H2.2** we stated that the results would be more likely to be correct with the GridView in contrast to the ContinuousView. For this hypothesis our data is not sufficient since we have very similar results for all three configurations.

8 Discussion

In this Chapter we will reflect the study results.

We did not expect that the *StarCoordinates* [Kan00], which were introduced because of their easy to understand construction, would lead to a wrong assumption. Some participants assumed that two points on an axis can only have differences along that axis, but there is a possibility that two parameters lead to this behavior. This behavior was only shown in the ContinuousView and no one showed this behavior in the GridView. The ambiguity could have been resolved with the ParamView, but not everyone used this option.

The GridView worked well when it came to finding correlations between the parameter space and the objective value. We were able to reduce the number of interactions required. In our expert study, participants only used the portal to see the content of a few glyphs. In the ContinuousView, they had to hover over many data points to get the information they needed.

When using the GridView, participants focused on the GridView or portal and used the ParamView only to name the parameters. This meant that they did not have to hover over individual glyphs to estimate the parameter values in the ParamView. This shows that in most cases the star plots worked even without an outline and a background of a similar color.

However, there are some cases where our star plots were not easily recognizable by the participant. The feedback we received shows that spikes in a high luminance color, in this case yellow, were difficult to see.

With the additional feedback from our expert study, we are able to identify other technical improvements that could be added to facilitate interaction with the tool.

For example, the visualization tool lacks geometric zooming in both the ContinuousView and the GridView, and the convex hull that is currently displayed is often only relevant in a small area. In addition, it was requested and useful to include a filter for the target value and standard deviation. Another feedback was that a user would be interested in the real parameter values and not just an estimate from the star plots, so a view showing the parameter values could be easily added.

We received the most feedback and suggestions for the portal to be within the grid. In our current implementation, the only way to change the size of the portal is to use a slider next to the main view. This feature was introduced because we use the mouse wheel to change levels within the portal. One participant stated that we could increase the size of the portal as the user changes levels. With this approach, the glyphs inside the portal would be the same size regardless of the level. Another approach would be to press a key while scrolling to change the level with the scroll wheel, and the scroll wheel would act as expected in the other case and change the size of the portal. Additionally it would be helpful to have the mouse pointer centered in the location of the portal when the portal is not locked anymore this would stop the portal to jump around whenever it is unlocked.

To complete the technical improvements, the visualization tool could be made resizable. In its current state, the tool does not work with resolutions lower than 1600×1000 pixels. Additionally, if the tool is resizable, users with a higher resolution would benefit from the larger screen size.

9 Future Work

With the GridView and the several other additions we created a visualization tool which comes with significant advantages when the correlation of parameters and objective value is searched for. However, the study results and additional needs of experts show that there are multiple ways for improvement and future research.

9.1 Dynamic Resolution

The performance maps we display in our tool will have flat areas for large parts of the map. In these areas a high number of glyphs is probably not too helpful. It could be more important to see draw larger glyphs in these areas where not too much is happening.

We would think of a dynamic resolution where glyphs are shown on a lower level, showing more glyphs, only when the standard deviation in the parents is high. Therefore, a cutoff estimation or a selection would be needed to work properly.

9.2 Multi-dimensional objective values

In the architectural design it is rare that there is only one objective value which has to be fulfilled. With the version presented in our work a user has to search for a model configuration which satisfies one target and check if this is possible for the other targets which had to be achieved as well.

Since the underlying optimization component is already able to run the optimization even for multiple objectives, it would be possible to add features for multi-dimensional objective values. We could think of a hybrid approach using the GridView to show the objective value in the star glyph and the ParamView showing the parameter space or the PortalView to be used the see the different objective values for one glyph in the GridView. The multi-objectives encoded in the GridView and the color encoding the objectives weighted interactively by the user might lead to a good overview.

However, this will introduce the same interactions for the parameter evaluation we tried to eliminate with the GridView. This topic is a potential research project and would reduce the effort of the users drastically.

9.3 Projection Techniques

During the expert study, we found that the easier to understand *StarCoordinates* [Kan00] did not provide the benefits we had hoped for. The relationship between the placement and the resulting parameter values was misleading in some cases. It may be that other projection techniques and clustering algorithms can help us understand the unknown black box function even better. Plotting the evaluated data points along with some interpolated values may lead to a sufficient dataset that can be visualized using e.g. t-distributed Stochastic Neighbor Embedding (t-SNE) or UMAP to project the data onto the two-dimensional plane. This approach is very different from our work, but could give good results which has to be evaluated.

9.4 Further Evaluation of Additions

Due to the complexity of the task, our expert study had to be designed to compare only the core functionality of the tool including two versions of the portal in combination with the GridView and the ContinuousView.

There are many different permutations for each view displayed in a given location, there are many different configurations that can be compared. In our expert study, we did not include the FogView or RenderView, which may be essential parts to get deeper knowledge of the data. It would be interesting to know if there are noticeable behaviors when the user is free to use and customize the tool for his own needs. Therefore, it might be possible to add features to track configurations within the tool with the possibility to send back these user statistics manually or automatically. With this long-term study it might be possible to learn which of the views are actually used when the user is not forced to do so.

In addition, we did not record task completion time because we wanted the participants to use our tool as they would use it in their daily work. Nevertheless, it is an interesting question whether there are significant and measurable differences in completion time based on the configuration used.

9.5 More Glyph Designs

The star glyphs we were using are known to work well for shape recognition, however we encountered some major drawbacks in our usage of them without any outline of the shape. The problem being narrow spikes in the glyph with a color similar to the white background. The usage of the flower glyphs also described in *Glyphboard* [KKG+20] could help to reduce this problem.

However, there are many different glyphs which could be used for multivariate data visualization, not even being forced to radial glyphs. Showing the parameters in a profile glyph, showing a small chart, or in a PCP could be possible glyphs as well.

In addition to the multivariate data being visualized, we added a basic visualization for the uncertainty underneath a glyph. In the future there could be added more complex structures to encode the uncertainty similar to Gortler and Schulz et al. [Gor18] in *Bubble Treemaps*.

9.6 Separate Color Palettes

A rather special addition would be to use a different color palette inside the PortalView. In some regions the area underneath a glyph is rather flat resulting in similar colors inside the portal. It would be possible to scale the color palette for the portal view again, this would make differences in the portal more obvious. Because the opportunity of comparing children from different glyphs is missing in this approach, this addition would need more research.

10 Conclusion

In our work we extended the tool from *Performance Map* [Wor17b]. The original tool displays the multi-dimensional parameter space in a continuous map showing the objective value encoded in color. Our goal was to facilitate the exploration by adding a new connection between the parameter space and the resulting objective value.

We made the connection using a glyph-based approach. These glyphs represent the multi-dimensional parameter space in the form of a star plot. The glyphs are placed using the same strategy as in the original tool. The target value is encoded with the color in the same way as in the continuous performance map.

These glyphs take up much more space than a single dot. Therefore, we added a background to our glyphs that has the same color as the color palette, but is more transparent to show the shape of the glyph. We displayed the standard deviation of the underlying area, as this information would otherwise be lost. We added an interaction to access data points below the glyph with a portal. We also added a view that shows the areas of the map that the user was investigating and a feature to request a perspective view of the three-dimensional model, which is then displayed in the tool. All of these enhancements can be selected and displayed at different locations within the tool. This results in a customizable visualization that can be configured to the user's needs.

We conducted an expert study to evaluate a number of our extensions. Our participants preferred the extensions over the original tool and achieved similar results in the tasks they solved. The parameter space influencing the objective value was understood with less interaction than before. However, there are technical improvements to facilitate the interactions that had to be performed even more.

For the future there are several fields to expand this work even further, for example, by adding support for multi-dimensional objective values different projection techniques or different glyph designs.

Bibliography

[Bro95] J. Brooke. “SUS: A quick and dirty usability scale”. In: *Usability Eval. Ind.* 189 (Nov. 1995) (cit. on p. 59).

[CN18] A. Costa, G. Nannicini. “RBFOpt: an open-source library for black-box optimization with costly function evaluations – Optimization Online”. In: (Aug. 27, 2018). URL: https://doi.org/10.1007/s12532-018-0144-7 (visited on 12/29/2022) (cit. on p. 23).

[E H84] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, J. M. Ogden. *Pyramid methods in image processing*. RCA Engineer, 1984. URL: https://www.researchgate.net/profile/Joan-Ogden/publication/246727904_Pyramid_Methods_in_Image_Processing/links/544006600cf21227a11ba140/Pyramid-Methods-in-Image-Processing.pdf (cit. on p. 47).

[Gor18] Gortler and Schulz et al. “Bubble Treemaps for Uncertainty Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018), pp. 719–728. DOI: 10.1109/tvcg.2017.2743959 (cit. on p. 70).

[Gut01] H.-M. Gutmann. “A Radial Basis Function Method for Global Optimization”. In: *Journal of Global Optimization* 19.3 (Mar. 1, 2001), pp. 201–227. ISSN: 1573-2916. DOI: 10.1023/A:1011255519438. URL: https://doi.org/10.1023/A:1011255519438 (visited on 12/29/2022) (cit. on p. 17).

[Kan00] E. Kandogan. “Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions”. In: *Proceedings of the IEEE information visualization symposium*. Vol. 650. Citeseer. 2000, p. 22 (cit. on pp. 16, 18, 19, 23, 28, 31, 34, 67, 70).

[Ken15] Kenneth Moreland. *Why We Use Bad Color Maps and What You Can Do About It*. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2015. URL: https://www.kennethmoreland.com/color-advice/BadColorMaps.pdf (cit. on p. 40).

[KKG+20] D. Kammer, M. Keck, T. Gründer, A. Maasch, T. Thom, M. Kleinsteuber, R. Groh. “Glyphboard: Visual Exploration of High-Dimensional Data Combining Glyphs with Dimensionality Reduction”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.4 (Apr. 2020), pp. 1661–1671. ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.2969060 (cit. on pp. 15, 21, 22, 70).

[Mat12] Matteo Niccoli. *The rainbow is dead... long live the rainbow! – series outline*. May 29, 2012. URL: https://mycartablog.com/2012/05/29/the-rainbow-is-dead-long-live-the-rainbow-series-outline/ (visited on 01/28/2023) (cit. on p. 39).

[Mic23] A. P. e. a. Michelle A. Borkin Krzysztof Gajos. *Evaluation of Artery Visualizations for Heart Disease Diagnosis*. Jan. 28, 2023. URL: https://vcglab.org/gvi-files/borkin-InfoVis2011_camera-ready.pdf (cit. on p. 39).

[OPDR18] T. Opach, S. Popelka, J. Dolezalova, J. K. Rød. "Star and polyline glyphs in a grid plot and on a map display: which perform better?" In: *Cartography and Geographic Information Science* 45.5 (Sept. 3, 2018), pp. 400–419. ISSN: 1523-0406. DOI: 10.1080/15230406.2017.1364169. URL: https://doi.org/10.1080/15230406.2017.1364169 (visited on 08/23/2022) (cit. on p. 28).

[Wei23] E. W. Weisstein. *Barycentric Coordinates*. MathWorld–A Wolfram Web Resource. Jan. 5, 2023. URL: https://mathworld.wolfram.com/BarycentricCoordinates.html (cit. on pp. 16, 19, 23).

[Wor17a] T. Wortmann. "Opossum: Introducing and Evaluating a Model-based Optimization Tool for Grasshopper". In: (Apr. 11, 2017) (cit. on p. 17).

[Wor17b] T. Wortmann. "Surveying design spaces with performance maps: A multivariate visualization method for parametric design and architectural design optimization". In: *International Journal of Architectural Computing* 15.1 (Mar. 1, 2017), pp. 38–53. ISSN: 1478-0771. DOI: 10.1177/1478077117691600. URL: https://doi.org/10.1177/1478077117691600 (visited on 08/21/2022) (cit. on pp. 15, 17, 21, 23, 24, 28, 31, 39, 59, 73).

[Yih22] Yihan Hou, et al. *A study of the effect of star glyph parameters on value estimation and comparison*. Aug. 14, 2022. URL: https://link.springer.com/content/pdf/10.1007/s12650-022-00888-x.pdf?pdf=button (cit. on pp. 21, 24, 25).

All links were last followed on January 31 , 2022.

Declaration