ParaGlide: Interactive Parameter Space Partitioning for Computer Simulations

Steven Bergner sbergner@cs.sfu.ca
Michael Sedlmair msedl@cs.ubc.ca
Torsten Möller torsten@sfu.ca
Sareh Nabi Abdolyousefi sna39@sfu.ca
Ahmed Saad aasaad@cs.sfu.ca

Abstract—In this paper we introduce ParaGlide, a visualization system designed for interactive exploration of parameter spaces of multi-dimensional simulation models. To get the right parameter configuration, model developers frequently have to go back and forth between setting input parameters and qualitatively judging the outcomes of their model. Current state-of-the-art tools and practices, however, fail to provide a systematic way of exploring these parameter spaces, making informed decisions about parameter configurations a tedious and workload-intensive task. ParaGlide endeavors to overcome this shortcoming by guiding data generation using a region-based user interface for parameter sampling and then dividing the model's input parameter space into partitions that represent distinct output behavior. In particular, we found that parameter space partitioning can help model developers to better understand qualitative differences among possibly high-dimensional model outputs. Further, it provides information on parameter sensitivity and facilitates comparison of models. We developed ParaGlide in close collaboration with experts from three different domains, who all were involved in developing new models for their domain. We first analyzed current practices of six domain experts and derived a set of tasks and design requirements, then engaged in a user-centered design process, and finally conducted three longitudinal in-depth case studies underlining the usefulness of our approach.

1 Introduction

Many researchers seek to study real-world phenomena by describing them using computational models. Such models usually take a set of input parameters and map them to a set of outputs that describe the behavior of the system. Consider, for instance, modeling the aggregation behavior of animals such as flocks of birds, swarms of insects, or schools of fish (see Section 3.2). The input parameters for a model in this domain might describe the social forces between individual animals, such as attraction, repulsion, and alignment. Based on a specific configuration, the model can compute an output image that shows the collective group behavior of animals.

A major modeling challenge is to guarantee a close correspondence between the real-world system and the model. Efficient human interfaces have the potential to greatly improve the fitness and applicability of computer simulations, because they enable model developers, domain experts, and decision makers to make adjustments and interpret model output according to their real world experience and knowledge. Selecting relevant training data, assessing model output, choosing among alternative model versions, adjusting preference or constraints, and trading-off multiple performance objectives are standard tasks for these types of users. Typical data mining, machine learning, or image processing research strives to ultimately automate all of these steps in order to improve model performance and consistency of decisions. However, no single computational method is capable of handling all varieties of possible modeling scenarios and especially early stages of model development involve potentially costly tasks that need to be carried out by human experts.

These challenges have motivated a number of recent contributions in visualization research [2], [23], [29], further discussed in Section 4.1, that seek to make interactive model analysis feasible and more efficient. These approaches focus on finding optimal input parameter configurations. By observing real model developers we confirmed that *optimization* certainly is a valid and very important task. However, we further found an additional, yet unsupported need for *input parameter space partitioning*. Specifically, model builders wanted to understand and discover regions in the input parameter space that exhibit qualitatively different model behavior in the output space. In the animal aggregation example, certain regions in input parameter space might, for instance, lead to the formation of groups that may rest or move together. Other parameter configurations may produce patterns that are not biologically plausible or may not converge to a solution at all.

In this paper, we study the problem of and provide a solution for input parameter space partitioning. Towards this goal, we make the following contributions:

- the problem characterization and abstraction of parameter space partitioning based on the in-depth analysis of three application domains (see Section 3);
- ParaGlide, a system that supports interactive parameter space exploration and partitioning (see Section 5).
- a field study of ParaGlide providing anecdotal evidence for the benefits of interactive parameter space exploration and partitioning (see Section 6).

ParaGlide's system design proposes several distinct features that support interactive exploration and partitioning of param-

- *S. Bergner, A. Saad, and T. Möller conducted this work at the Dept. of Computing Science, Simon Fraser University (SFU), Burnaby, BC, V5A 1S6, S. Bergner is currently at the Dept. of Statistics and Actuarial Science.*
- *M. Sedlmair is with Imager Lab, University of British Columbia.*
- *S. Nabi Abdolyousefi did this work at Dept. of Applied Mathematics, SFU. She currently is with the Dept. of Economics, SFU.*

eter spaces (see Section 5): (1) The cost of uniformly sampling parameter configurations inside a parameter region of interest can be very high. We reduce these costs by offering user control for iterative region refinement and provide alternative region types that cover the same value ranges, but have a smaller volume. (2) An important task in model analysis is sensitivity analysis: the determination of how "sensitive" model outputs are to changes in the input parameter space [8]. We show that visualizing the shape of the parameter partitions can support the task of sensitivity analysis. (3) ParaGlide allows for manually clustering the output space into groups of similar model results. Relating these output groups back to their corresponding input configurations provides the user with insight into the effect and importance of the model parameters. (4) Furthermore, ParaGlide allows for a close integration between visual model analysis and model development environments, such as R or MATLAB. Hence, a model developer can derive additional features of the outputs of the model and test novel hypotheses on-the-fly without the need to tediously switch back and forth between different tools.

2 METHODS

ParaGlide was developed in a user-centered design process with five users from three different target domains: biological modeling, image segmentation, and fuel cell engineering. We met with our users in person covering longitudinal time ranges of four years (fuel cell engineering), two years (biological modeling) with monthly meetings, and five months (image segmentation) with weekly meetings.

Our collaborations were inspired by Design Study Methodology [27] and were roughly organized in three phases: First, we engaged in a problem characterization phase in which we analyzed current practices of our collaborators to get a detailed understanding of needs and requirements. Second, we used the insights from the problem characterization to (further) inform the design of ParaGlide and discussed design mockups and prototypes with our collaborators. Third, we deployed ParaGlide and observed our collaborators working with it. During these observations, we (1) gathered formative feedback in terms of usability and feature requests that we used to improve ParaGlide's design, (2) refined our understanding of user practices and design requirements (see Section 3), as well as (3) gathered summative feedback and anecdotal evidence to validate ParaGlide's design. These field investigations can be seen as a form of multi-dimensional in-depth long-term case studies (MILCs) [28].

3 PROBLEM CHARACTERIZATION

We start this section with the data and problem abstractions that we derived from our field investigations in the three application domains. The abstraction introduces the terminology that we will use throughout the paper. We then discuss our three application domains as examples that illustrate our problem characterization and abstraction. The section concludes with a list of abstract tasks and requirements that we derived from our field investigations and that we supported/addressed with ParaGlide's design. We provide this abstract problem description, including problem, data and tasks, as one of our major contributions that can be used by others as a starting point for further technical contributions.

3.1 Data Abstraction

In contrast to many other cases where data is given as a static source that can be loaded into a tool for inspection, in our case the data is dynamically sampled from running a model. Data production and data analysis are therefore tightly coupled together.

For our purpose, we define a ***model*** as the description of possible states of a real world system via mathematical relations among variables that at least partly represent observable quantities. We call all variables that go into the model ***input parameters*** and collectively refer to all model input parameters as the ***(input) parameter space*** of the model. A specific choice of values for the model parameters constitutes the input to a piece of software, called the ***simulator*** that can produce complex objects as ***outputs***, such as time curves, plots, or images. Collectively, these outputs characterize the model ***(output) behavior***. From the output of the model, one can further extract particular features or compare one output with others to measure their similarity using a distance metric. We refer to these features as ***derived variables*** that form the ***feature space*** of the model. A specific combination of parameter values and their simulation output together is referred to as an ***experimental point*** that due to deterministic computation has a unique location in input and feature space of the model.[1] Figure 1 provides a conceptual sketch of this computational pipeline. In Section 5, we discuss how ParaGlide allows the user to interact along the pipeline.

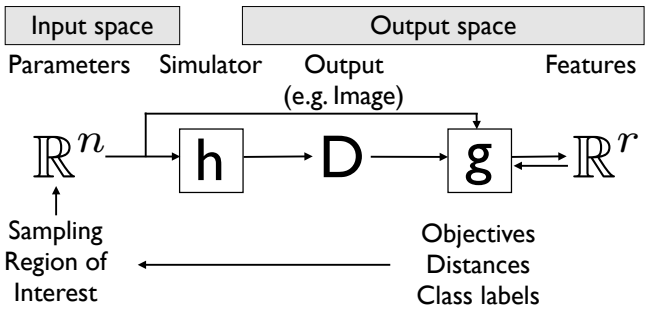


Fig. 1. Flow of processing steps when working with a simulation model. The input parameter space is mapped to the output behavior via the simulator, which in turn can be further mapped to a feature space that is comprised of derived variables. Mapping functions are marked with a rectangle. Optimization objectives, class labels, and also embedding coordinates as described in Section 5.2 are represented as derived variables. Experimental points that define input parameters for the computation of simulator outputs are generated within a user specified region of interest as discussed in Section 5.1.

1. Vocabulary note: In other domains different terminologies are used to describe models. For instance, input variables is used instead of input parameters, or inputs are referred to as *independent* and outputs as *dependent* variables [31]. We opted for a coherent terminology that reflects the vocabulary used in the application domains that we analyzed.

We now provide a more formal description of the outlined problem: The overall computation of model behavior for given input parameters can be summarized as a function $f : \mathbb{R}^n \to \mathbb{R}^r$, where $f(\mathbf{x}) = \mathbf{y}$. It is formed by composing a potentially costly output computation $h : \mathbb{R}^n \to D$ and derived variable computation $g : D \to \mathbb{R}^r$ to give $f = g \circ h$. A particular output captures the result of the computation for a particular parameter configuration $\mathbf{x} \in \mathbb{R}^n$. It is possible to cache outputs of h for later processing. Code to compute the output of h is given as a black box, the simulator. While the function abstraction imposes that the code is deterministic, uncertainties of the system can still be modelled by specifying probability distributions for some of the input parameters.

In simple computation pipelines the derived output variables are computed directly and f and h are identical with $D = \mathbb{R}^r$ and g being the identity. In other scenarios, g could also depend directly on the input parameters $\mathbf{x}$. The meaning of g can be interpreted as a feature, but also as embedding coordinate, cluster membership label, likelihood of the model configuration for given training data, distance from a template point, or objective performance measure—to give a few practical examples.

In order to obtain a data set of experimental points for further analysis, the model computation can be invoked for a set of parameter configurations (experimental points) $X = \{\mathbf{x}_k\} \subset \mathbb{R}^n$ of finite size $m = |X|$. This set X is referred to as a ***design*** [25, p. 15]. With a prescribed ordering of input parameter configurations, choosing X amounts to the construction of a design matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$. A sampling method to produce a design X takes a region description $\mathcal{M} \subset \mathbb{R}^n$ as input from the user and, if adaptive, may also be informed from certain outputs. The mapping f gives a set of outputs $\mathbf{Y} = \{f(\mathbf{x}_k)\}$. By concatenating these values as $[\mathbf{X}\ \mathbf{Y}] \in \mathbb{R}^{m \times (n+r)}$ the ***data table*** is obtained that can be used in further processing or visualization. Its $n + r$ columns represent the values of the n input parameters and the r derived variables. We collectively refer to these columns as the ***dimensions*** of the data table. Each of the m rows represents a single experimental point, namely a specific instance of input parameter settings and the derived variables that have been computed for them.

3.2 Example: Biological Modeling

Background: Our first target users were two developers of models that describe biological aggregations, including movement and migration of flocks of birds, swarms of insects, schools of fish, herds of quadrupeds, bacterial swarms, etc. These aggregation models evolve a spatial distribution of individuals over time and, therefore, form spatio-temporal patterns. People seek to understand these patterns to better predict animal migration behavior, for instance, to contribute to more efficient fishing strategies [21] by determining when and where fish aggregations form.

Model: The specific model employed by our users [12], [16] consisted of partial differential equations (PDEs) which represent the entire animal population as a continuous density function in one spatial dimension and describe how this spatial distribution changes over time. Movement of all individuals is expressed for separate travelling densities of individuals that move and may turn to the other direction. The basic idea is to govern the movement via three kinds of social forces — namely attraction, repulsion, and alignment — that act globally among the densities of individuals. Attraction is the tendency between distant individuals to get closer to each other, repulsion is the social force that causes individuals in close proximity to repel from each other, and alignment represents the tendency to sync the direction of motion with neighbors (see details in Appendix A.3). Altogether, there are 14 different *input parameters* to this model that include a set of weights to balance the different forces. Solving the model for different input parameter settings produces images showing many complex spatial and spatio-temporal patterns (the *output*). Ideally, these patterns then reflect patterns that can be observed in nature, such as stationary aggregations formed by resting animals, zigzagging flocks of birds, milling schools of fish, and rippling behavior observed in Myxobacteria.

Practices and problems: To better understand the space of possible outputs, our users manually explored the parameter space of their model. By visually inspecting the resulting output images, they were able to demonstrate the model's capability to reproduce a variety of complex patterns. Here, our users were particularly interested in comparing two versions of the model [12]. In the first one the velocity of movement is constant. In the second one the individuals speed up or slow down as a response to their social interactions with neighbors. Comparing these models required to solve them numerically for several different parameter configurations. Each one of them corresponds to one specific choice of the 14 input parameters. Yet, the computation of individual output images and their visual inspection, which our users did by running the model's MATLAB implementation, was a time consuming tasks. Iterating on individual parameter adjustments, computation of outputs, and their inspection was ineffective and tedious. By choosing the spatial and temporal resolution of the simulation its computational complexity can be adjusted to lie between 2 minutes and one hour per run of the model. With 5 minutes each, it is possible to compute close to 300 model outputs in the duration of a single day.

To produce parameter configurations, our user's current practices were simple random sampling and nested `for`-loops that produce all combinations of values for certain input parameters (so-called factorial designs). While computed outputs can be loaded and inspected individually, an overview of the data space was desired but not available. In order to sufficiently explore interesting outputs of the model, a method is required to systematically construct sets of parameter configurations.

3.3 Example: Image Segmentation

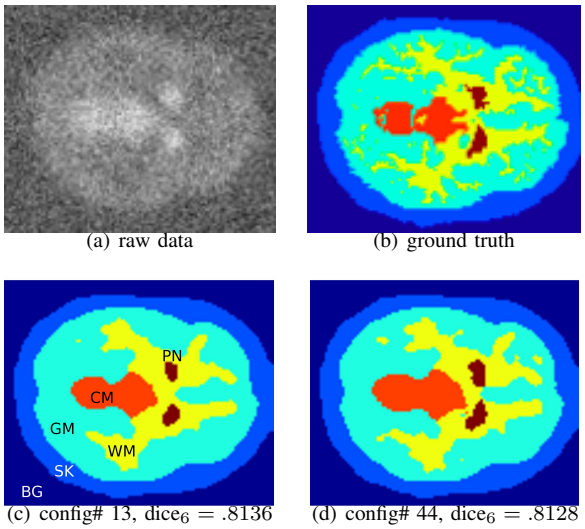
Background: Medical image segmentation is concerned with the identification of different anatomical shapes in a medical image scan of a patient. A reliable method has great potential to improve a physician's diagnostic means to provide better health care.

Model: Researchers in this domain build models of how different anatomical shapes are demarcated. The computational model here is therefore a segmentation algorithm that, for a particular adjustment of input parameters, outputs a segmentation of a medical image. The goal is to find input parameter configurations that lead to "good" segmentation outputs. Our user, for instance, used a kinetic model to devise a segmentation algorithm for molecular image quantification of dynamic-positron emission tomography (d-PET) data [24]. Such data is difficult to segment due to severe noise and partial volume effects. Our user therefore developed an extension to the random walk segmentation method [14] by adding input parameters that account for desirable criteria, such as data fidelity, shape prior, intensity prior, and regularization.

In order to attain a superior segmentation quality, a proper choice of input parameters is crucial. To facilitate this choice of input parameters, our user specified additional numerical performance measures (derived variables) that assess the quality of each segmentation (the output). As one such measure he picked the Dice coefficient [10], which gives a ratio of overlap with labeled training data. Another measure expresses an error of the quality of the kinetic modeling. Overall, the algorithm is influenced by 8 input parameters, and overall 10 derived variables were used to provide quality measures for all anatomic region. Figure 2 shows an example of output images; the different anatomic regions are marked with different colors.

Practices and problems: Our user initially calibrated his model by numerically optimizing the performance with respect to the input parameters. However, a closer look at the Dice coefficient revealed that this approach is not necessarily exact and reliable. For instance, using the two configurations of Figure 2(c) and (d), both have a Dice value above the 90^{th} percentile of all computed segmentations and are less than 0.003 standard deviations apart. Numerically, this situation means that both segmentations are of the same, near optimal quality. Yet by visual inspection, it becomes apparent that the putamen (PN, dark red in the figure) shape in (d) is closer to the ground truth in (b) than the one obtained in (c). Hence, guidance of a human domain expert is desirable to visually sort among several candidate solutions in order to find an improved segmentation.

The decisions of whether the algorithm attains a good or a bad solution in this example depends on taking multiple derived features as well as the output images into account. However, automatic optimization alone, as practiced by our user, is challenging due to two reasons: First, it is hard to properly account for multiple competing, sometimes contradicting objectives. Second, for the model to work robustly, it is important that the segmentation quality does not decay too quickly for slight changes to the chosen parameter configurations. A proper understanding of the sensitivity of certain parameter settings is necessary. Hence, extracting the space of solutions that lead to 'good' segmentation results enables the user to pick a robust solution within that space. These challenges were not supported by our user's current practices though.



(a) raw data (b) ground truth

(c) config# 13, $\text{dice}_6 = .8136$ (d) config# 44, $\text{dice}_6 = .8128$

Fig. 2. Raw image data from a brain scan (a), ground truth segmentation provided by a physician (b), and two output segmentations from our user's model from two different parameter configurations (c/d). Anatomic regions are shown with different colors: background (BG), skull (SK), grey matter (GM), white matter (WM), cerebellum (CM), and putamen (PN). Labels can be found in (c).

3.4 Example: Fuel Cell Prototyping

Background: A fuel cell takes hydrogen and oxygen as gaseous input and converts them into water and heat, while generating an electric current. Affordable, high-performance fuel cells have the potential to enable more environmentally friendly means of transport by greatly reducing CO_2 emissions. A reliable synthetic model that can simulate different configurations, would allow for much broader exploration of design options than real prototyping.

Model: Our users' model can be used to describe individual cells in a fuel stack [9]. The model takes 100 input parameters and produces outputs of 43 different plots of various physical quantities (see for instance Figure 8 below) characterizing the behavior of the cell stack, such as current density, temperature, or relative humidity variance across the geometry of the cell stack.

Practices and problems: Our users' main task was finding an optimal setting for the input parameters. They found suitable values for most of the input parameters from automatically fitting the model to available derived variables. However, they were also particularly interested in getting a deeper understanding of potential groups of parameters that represent the geometry of the assembly (size and number of cells in a stack), material properties (permeability), and running conditions (temperature, pressure, concentration). Such an understanding would allow for better studying failure mechanisms and further optimize the model's performance. This analysis was however not supported by their current practices.

3.5 Tasks and Requirements

In the following, we describe abstractions for tasks and requirements that generalize across the three use cases we

Model analysis tasks (A)	
A-view	visually inspect data table in overview and detail views of individual outputs
A-opt	find parameter configurations for which a certain output feature is optimal
A-sa	analyse sensitivity of feature values to a change of input parameter configuration
A-cpr	compare different model versions
A-psp	find input regions that produce similar output
Data set generation (D)	
D-roi	specify region of interest (ROI) possibly spanning only a subset of dimensions
D-smp	sample and generate set of experimental points inside ROI
D-drv	compute or enter manually: features, objectives, embedding coordinates, cluster labels
D-dist	construct a distance metric
Workflow support (W)	
W-int	integrate with existing practices and code
W-rep	save/load state of the project for reproduction

TABLE 1
Summary of the task and requirements analysis.

observed. We used these tasks and requirements to inform the design of ParaGlide (see Section 5) and offer them as the second part of our problem abstraction. Table 1 summarizes our findings and is organized in three categories:

1. Model analysis tasks (**A**) represent high-level objectives that drive the investigation of model behavior. *Viewing* the data table in an overview, as well as inspecting specific outputs of the model in detail (**A-view**) are important tasks allowing for a basic form of interactive model analysis. We found the need for visual inspection in all three of our use cases: In the biological modeling example the users looked at plots of animal aggregation patterns; in the image segmentation example images like Figure 2 were inspected; and in the fuel cell example the users investigated model behavior via plots such as the ones in Figure 8. Yet, while all users iteratively analysed individual experimental points, none of their current practices supported a way to directly view the correlation between experimental points. ParaGlide successfully filled this gap as discussed in Section 6. Specific for model tasks is that the data has to be generated before it can be viewed. This gives rise to a separate group of tasks (**D**) discussed below.

A very common task in modeling is *finding optimal parameter configurations* (**A-opt**) such as in the fuel cell stack case and the image segmentation, also pursued by Torsney-Weir et al. [29]. Two major goals of optimization are: a) to calibrate the model to have a good fit[2] with given measurements, and b) to adjust further free parameters to optimize other objectives, e.g., electric current output of the calibrated cell stack model.

Sensitivity analysis (**A-sa**) has been studied, for instance, by Booshehrian et al. [8]. Our fuel cell users were also highly interested in analysing the sensitivity of their model output to changes in the input parameter space. While identified as a separate task, it can be understood as optimization of an additional derived variable representing a local measure of sensitivity.

Model comparison (**A-cpr**) is a task that can be understood on different levels, depending whether one considers different parameter configurations to represent different models. Indeed, a categorical parameter can choose among different model families. One goal of the animal aggregation study was to compare constant and non-constant velocity models, as detailed further in Section 6.1.

Parameter space partitioning (**A-psp**) is a form of model analysis that decomposes the input parameter space into a number of regions where each represents similar output behavior. Once such a partitioning is obtained and the shape of the partitions is inspected in the input space, it is possible to answer questions about model complexity, sensitivity, allow for qualitative comparison, and provide guidance for context-dependent optimization. All of our users were interested in understanding regions both in the (input) parameter space and the (output) feature space, as well as how these input and output regions relate to each other. We characterize a workflow of tasks that support this need and offer it as an extension to our knowledge of needs in model development. In contrast to the general tasks discussed above, we did not observe these tasks in our users' current practices but actively designed this workflow. We found it being adopted by our users as soon as they had ParaGlide available (for more details see Section 6).

2. Data generation (**D**) refers to a number of steps to choose parameter configurations (**D-roi**, **D-smp**) and compute outputs of a computational model. Further processing of the outputs leads to other derived variables (**D-drv**, **D-dist**) that represent features, objectives, distances, embedding coordinates, or labels for clusters of experimental points. A more detailed discussion of this sub-task decomposition is provided in Section 5 where we explain how data from a simulator is produced and how further processing, such as parameter space partitioning (**A-psp**) is enabled.

3. Workflow requirements (**W**) are meant to facilitate adaptation of novel practices by avoiding technical hurdles. In particular, we found a need for *close integration* (**W-int**) between novel tools and the given model code. In line with previous findings [26], close integration is particularly important in our situation where a seamless transition between data generation through the model and its exploration is imperative. Some collaborators reported difficulties to reproduce findings that were made during past trial-and-error investigations. Hence, *saving and loading (**W-rep**)* of system states for later reproduction and documentation of results became another important requirement. This requirement also includes organization of a possibly large number of output files that are generated in multiple sessions.

2. For instance, a derived variable could be added that represents likelihood (probability of a parameter configuration for given data). An optimization of that variable w.r.t. the input parameters could be applied or the shape of the likelihood surface be inspected.

4 Related Work

In the past, a number of systems have been created specifically to explore variations in rendering parameters. Some of these systems are concerned with the capture of trajectories through the multi-dimensional space and less with providing a global view of parameter space (e.g., VisTrails [1], Jankun-Kelly et al. [15], Bhagavatula et al. [6]). Design Galleries [18] on the other hand, do not provide the connection between input and output space required by our users.

In this section, we will focus on visualization systems that support the exploration of parameter spaces for a general simulation system. A particular emphasis is put on the problem of parameter space partitioning.

4.1 Interactive Environments for Parameter Adjustment in Computer Experiments

Computational steering adjusts parameters during execution of time-dependent simulations [20]. Since our users do not modify parameters during the run of a simulation, our problem setting is different from classical steering in that we do not need to handle live updates of parameters that are shared between simultaneously running modules. However, there is enough similarity to benefit from a comparison.

An evaluation of their computational steering environment (CSE) by van Wijk et al. [30] recognizes major uses for debugging, presentation, and assistance in technical discussions that progress faster when "What if?" questions can be answered immediately. A follow-up survey by Mulder et al. [20] identified further uses for model exploration, algorithm experimentation, and performance optimization. While these systems inspire numerous design decisions, they do not fulfill several of our design requirements. These include efficient specification and sampling of ROIs (**D-roi** and **D-smp**), an easy integration of end-user codes for simulation (**W-int**), and certain derived variable computation (**D-dist**).

Berger et al. [2] discuss a system to visualize engineering and design options in the neighborhood around an optimal point (**A-opt**) in parameter space of a computer simulation. Based on a continuous function abstraction they provide a local analysis method (**A-sa**) that benefits domain experts. However, in order to not get stuck in local extrema, optimization methods usually benefit from an additional global perspective on the problem domain. The purpose of parameter space partitioning is to enable such a global perspective by decomposing the input parameter space of the model into regions of distinct output (**A-psp**).

Tuner [29] is a system that supports a global view of the parameter space through a Pareto pane, but their exploration is focused on the local behavior (**A-sa**). Further, it is constrained to only two derived variables and is solely focused towards an optimization task (**A-opt**). Our discussion focuses on a global partitioning of the parameter space.

The challenge of devising a user interaction for sampling the parameter space has recently been taken on in the Paranorama system of Pretorius et al. [23]. Their users can specify different ranges of interest for individual parameters (**D-roi**) along with the number of requested distinct values per range (**D-smp**). Since their parameter configurations are constructed via a Cartesian product and inspected through slices of the space of outputs, it does not scale well. Section 5.1 will give some consideration to the number of involved parameters and the volume of the region of interest.

4.2 Parameter Space Partitioning

The computational model analysis cases of Section 3 benefit from an overview of regions of distinct system behavior marked out in their input parameter space (**A-psp**). While there is no prior work in academic visualization research that provides such a representation, there is further evidence in physics and psychology that such a system is needed.

Bhatt and Koechling [7] study the behavior during impact of two solid bodies with finite friction and restitution, which results in a tangential sliding velocity that continuously changes direction after impact. The problem is characterized by 9 parameters. The first step of their analysis determines a reduced set of three dimensionless parameters that completely define the tangential flow of sliding velocities. An important observation is that the qualitative behavior of the flow is characterized by 4 main cases with up to 3 sub-cases each (**A-psp**). An implicit expression of the boundary between the cases is derived that is quartic in terms of the 3 dimensionless parameters. By fixing one parameter and showing slices through this boundary, the enclosed regions can be visually distinguished and are labeled with the different cases they represent (**A-view**). These labeled slices provide a comprehensive overview of all possible sliding behaviors.

Pitt et al. [22] also promote parameter space partitioning (**A-psp**) and give an example analysis of a model that predicts, whether visual stimuli are recognized as words or non-words. In their overview of analysis techniques, they distinguish two axes that separate quantitative from qualitative and local from global techniques. In their view, partitioning is a global, qualitative method, and sensitivity analysis (**A-sa**) a case of more local, quantitative inspection. Their partitioning algorithm proceeds from a given classifier function that determines the type of a particular model output and a given set of valid parameter configurations (seed points). The regions around these configuration points are sampled using a Metropolis-Hastings algorithm with uniform target distribution (**D-smp**). Rejected points have fallen into non-equivalent regions and are explored subsequently.

The studies they present determine the number and volume of qualitatively distinct model behaviors and apply these measures to reason about suitable model fidelity. However, a discussion of user interactions and computational procedures beyond sampling to obtain these results are not part of their exposition. In a more interactive setting, our work emphasizes the benefits that arise from a visual evaluation of a partitioning.

5 Design of the ParaGlide System

Figure 3 shows a screenshot of ParaGlide's GUI. Like many other visualization systems, ParaGlide supports multiple coordinated views: The main window (Figure 3F) contains a menu bar on top and, from left to right, (B) dimension group

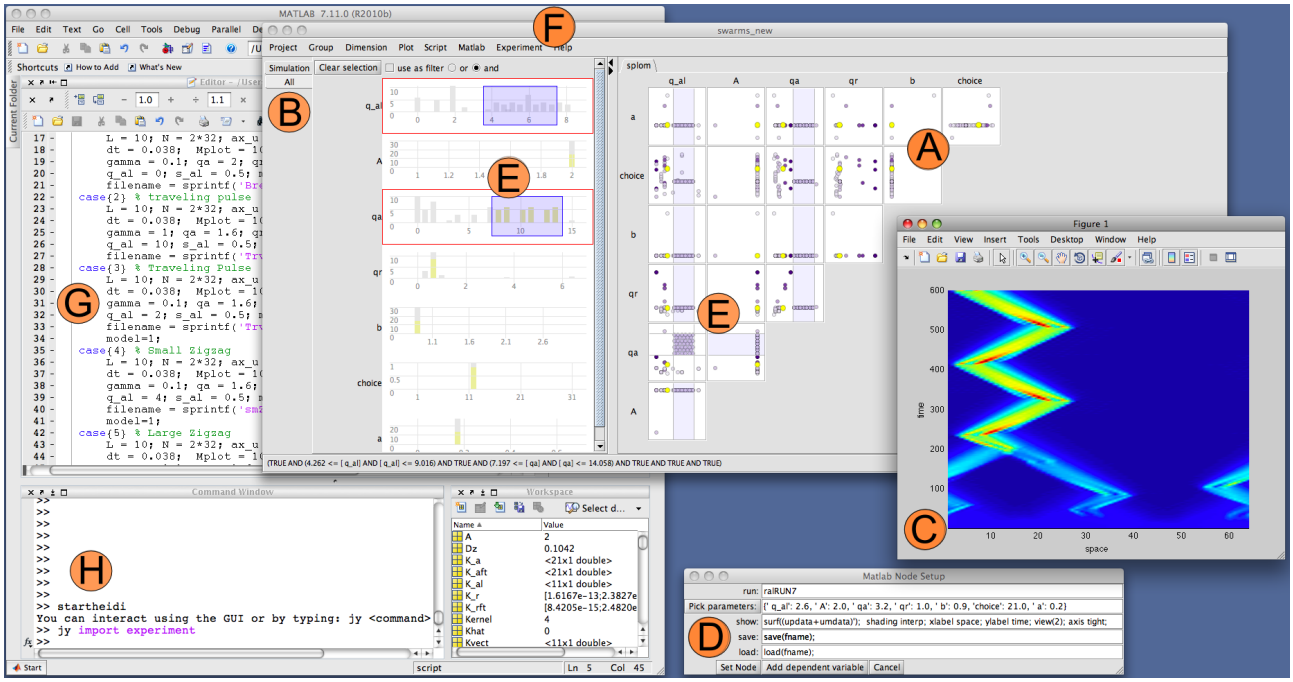


Fig. 3. ParaGlide GUI running inside a MATLAB session in the biological modeling use case of Section 3.2. An overview (**A-view**) of the data is given with histograms (A1) and a scatter plot matrix (SPloM, A2), both for a chosen group of dimensions (B). Clicking the experimental point highlighted in yellow in the SPloM, opens a detail view (C) and depicts a motion pattern of two groups that merge and then progress upward in time in a 'zigzag' movement. The view is produced with the `show` command as specified in the dialog (D) that sets up the MATLAB compute node (**W-int**). The data set was obtained by combining two methods: A parameter region of interest (**D-roi**) is selected within the light blue selection boxes (E) and a sampling method is chosen from the Experiment menu (F) to produce a set of lattice points (**D-smp**). Also, manually created parameter configurations are imported from a switch/case script (G) by sampling the case selection variable of that script and recording the parameters it sets. To add these sampling methods to ParaGlide, a Jython command was issued inside the MATLAB command window (H) to import the experiment module.

selection tabs (see below), (A1) views for dimensions of the chosen group in form of histograms and (A2) a scatterplot matrix (SPloM), and on demand, (C) a separate window with a detail view for individual outputs such as the 'zigzag' pattern image from the biological modeling example as shown in the figure.

Grouping of dimensions: A visualization challenge that is inherent due to the multi-dimensionality of our data (e.g., eight dimensions in the case of the image segmentation study) is the limited screen space and visual clutter. To address this challenge, we provide a basic method that allows the user to split the entirety of dimensions into smaller groups for more focused inspection. We call them ***dimension groups***. Grouping decisions can be based on user preferences or previous analysis and group membership can be dynamically changed by the user. The dimension groups in Figure 3B are called "All", showing all dimensions—input parameters and derived output variables—and "Simulation", where only input parameters are included. The multi-dimensional overviews per tab only show parameters of the currently selected dimension group, which reduces the required screen space. This approach of manual selection proved sufficient in the first two use cases. In the fuel cell case, the simulator already provides a division of parameters into several meaningful groups that ParaGlide can import and represent. Multi-dimensional data viewing (**A-view**) is considered in more detail in Appendix A.

Simulation code integration: Since ParaGlide serves as interface to the simulation model, no additional programming effort is needed to enable the interactive generation and exploration of the parameter space. This design simplifies interfacing with the simulation and allows us to do so with a user dialog (Figure 3D), called ***compute node*** that specifies the following aspects: input parameter names (parameters) and their default values; a command to compute outputs (run), a command to produce a plot of the output (show), and commands to derive variables from existing outputs. This simple interface directly supports our requirement **W-int**. A more detailed example of ParaGlide's way of model integration that also explains portions (G) and (H) of Figure 3 is given in Appendix A.2.

We designed ParaGlide to have a flexible analysis approach that is able to adapt to a large variety of possible practical settings. A design decision at the heart of our tool design was to enable a human in the loop to take care of decisions that are difficult to make algorithmically and leave other aspects to automatic processing. In such a setting, ParaGlide allows to address complex tasks through a number of basic interaction steps that interoperate with the computational problem pipeline shown above in Figure 1. These interactions tasks include

- Specifying and sampling a region of interest (**D-roi** and **D-smp**—see Section 5.1)
- Clustering of high-dimensional outputs (**D-drv**, **D-dist** and **A-psp**—see Section 5.2)

5.1 Specifying and Sampling a Region of Interest (D-roi, and D-smp)

ParaGlide is not initialized with experimental points and needs to produce them first. Hence, a region of interest $\mathcal{M} \subset \mathbb{R}^n$ needs to be specified first, followed by parameter configurations inside $\mathcal{M}$ [11]. In this way, the choice of region of interest (ROI) and the construction of a set of parameter configurations inside it are closely related (**D-roi** and **D-smp**).

It is common, that the user specifies a multi-dimensional region by its one-dimensional ranges for each parameter separately. Combining these ranges via AND operations results in a hyperbox-shaped region. It is comparably simple to display and manipulate. Specifically, in the SPloM and histogram interface of Figure 3E, a 2-dimensional selection box is shown in light purple.

Further, the number of samples within the region could be governed by concerns such as sampling density, but more often is constrained by the available running time of the simulator. Hence, our sampling strategy is to query the user for the total number of parameter configurations to be created as well as selecting from one of four sampling strategies [25, Ch. 5]: simple random sampling, lattice sampling, Latin Hypercube, and low-discrepancy sampling (Halton). All of these strategies lead to a distribution that fill the region of interest with uniform density. Lattice sampling gives a very regular structure in the visual arrangement of sample points and allows users to study model behavior changes along certain lines or regularly spaced points. In cases where this sort of regularity would be misleading, less structured sampling patterns may be preferred.

Managing sampling costs: To sufficiently sample an increasing number of parameters poses numerical challenges (e.g., see effects of dimensionality in Bergner [3, §2.1]) and can be very costly. ParaGlide's design enables to address this issue in several ways. Firstly, the separation of region specification (**D-roi**) from sample construction (**D-smp**) in an interactive loop illustrated in Figure 1 enables the user to adaptively increase the sampling density only after the ROI has been narrowed down to promising areas. In this phase, it is also possible to use a sampling method that interprets the ROI box as a range around a center point of focus and only places experimental points in the part of the box that extend up to a certain distance from the center. This allows to span the same parameter value ranges with a much reduced volume and at lower computational cost.

The computation of outputs can be adjusted between more or less computationally expensive levels of output fidelity. At the extreme end this completely bypasses running the simulator and directly invokes a cheaper feature extractor that can help with the initial region refinement. An application of this approach in practice is documented in Section 6.1.

In terms of usability of region-based sampling, we found that—in addition to the rough box selection—our users were also interested to see and set exact numerical values of region bounds. ParaGlide supports this requirement, which found frequent use. This specific need for an intuitive manipulation that is at times quantifiable should be kept in mind when designing user interactions for multi-dimensional navigation.

5.2 Clustering of High-dimensional Outputs (D-drv, D-dist and A-psp)

In this section we describe a workflow to obtain a clustering of experimental points based on similarity of their corresponding outputs. A parameter space partitioning (**A-psp**) is then obtained by viewing the color coded clusters in the input parameter space of the model.

To address the task of producing a meaningful decomposition of the parameter space through a clustering of the outputs into regions of distinct model behavior, Pitt et al. [22] assume that a discriminant function that labels different types of outputs is given. We do not make this assumption and instead assist the user in the construction of a clustering, which proceeds in three steps: First, to compare among experimental points the user chooses or constructs derived variables (**D-drv**) that pick up on meaningful features, possibly combining multiple features into a single vector. To assess whether the similarity or distance measure derived from these feature vectors is meaningful, a distance preserving embedding of points representing the feature vectors is computed and shown (**D-dist**/**A-view**). In our case, we have found spectral embedding [17] to work well using a similarity matrix that represents cosine similarity (i.e. Pearson's correlation coefficient) among pairs of feature vectors. It is also possible to decide to only include a subset of the output or derived variables in this computation (see Appendix A.3 for more details). Examples of the resulting embeddings are shown later in Figures 5 and 7. A particular advantage of these embedding methods is that, once a similarity matrix is determined, the complexity of computing an embedding only depends on the number of experimental points m. Hence, distance or similarity based data analysis is very suitable in high-dimensional settings involving complex outputs for many parameters.

If the detail inspection of different outputs that cluster together in the spectral embedding agrees with the user's expectations, individual clusters can be labeled via interactive brushing. Region selection comes again to aid for this purpose.

ParaGlide supports linking clusters of similar outputs back into the input space by simply linking and brushing scatterplots in the SPloM (**A-psp**). In the input space, these clusters can have arbitrary, possibly disconnected shapes and manually clustering the output space gives the user a rich opportunity to explore potential relations between outputs and inputs. While it might be possible to use one of various automatic clustering methods [5] to replace manual clustering of outputs, it is (1) impossible to select one of them and assume it to work for diverse problems, (2) to provide a diverse suite of clustering methods and hope that the user will find an appropriate one, and (3) cover all—previously unknown—relations that might be interesting for the user. Hence, we constrained ourselves to a manual method to assign cluster labels.

For manual clustering, the user determines a plot in the SPloM, e.g., the low-dimensional embedding, where the clusters of interest are sufficiently separated (or not). After enclosing the outputs (represented by points in the plot) by drawing rectangular regions in the plot, it is possible to manually assign cluster labels or to directly work with the multi-dimensional

region description. Non-rectangular regions can iteratively be constructed from smaller rectangular ones. If visible cluster separation is not sufficient, the distance and embedding construction described above can be adjusted to include more features in the comparison that improve separation of clusters. In order to be able to tell clusters apart visually, the derived variable containing the cluster labels is mapped to different colors. This is done by associating it with one out of several prepared palettes of distinguishable colors.

Viewing clusters of similar outputs in parameter space provides a point based view of the partitioning (**A-psp**). Direct display of the underlying continuous relationships between parameter and feature space is also a possibility [3, §2.4.2], but is not pursued further in this initial system design.

6 VALIDATION

We present the summative findings in form of case studies in which our users (a) were able to do something that they were not able to do without ParaGlide, (b) could gather new insights and improve their model by using ParaGlide, or (c) felt to be able to conduct tasks more efficiently with ParaGlide than with traditional methods.

6.1 Case Study: Biological Modeling

This case study is based on several sessions with one of our users from the biological modeling example of Section 3.2. Figure 3 further illustrates this case study.

Overview of previously computed experimental points: At the beginning of the investigation our user had an implementation of the model and a set of example parameter configurations that were given as a MATLAB script with annotated switch/case statements (see Figure 3G). Using ParaGlide, our user imported all their previously defined parameter configurations and presented them in a scatter plot matrix overview, as shown in Figure 3A2. Each point in these linked views represents one parameter configuration. Viewing the SPloM, our user was able to get a quick overview of how they had sampled the parameter space so far. This overview alone was already an interesting insight for them, revealing dense and sparse areas of sampling.

After triggering computation of outputs for all parameter configurations (**D-smp**), our user then started to inspect pattern images (**A-view**) by simply clicking on an experimental point in the SPloM representing a particular parameter configuration. Figure 3C shows an example of a zig-zag movement which corresponds to the parameter configuration highlighted in yellow in the SPloM. Seamlessly transitioning between different output images while being able to see them in context of their location in parameter space, facilitated our users' discussion of hypotheses about the model. Prior practice was to manually load simulator outputs and to iteratively inspect them together with the parameter configurations for each output as numbers in a script. Our way of visually supporting this reasoning, made it less tedious as it did not force the users to recall previously inspected outputs and parameter configurations. Instead, ParaGlide offered them a way to mentally navigate through *"a map of parameter configurations"*.

Finding interesting patterns using linear stability analysis: To further facilitate parameter space exploration, our user employed ParaGlide to set up an automated screening for relevant patterns in the output. Performing a linear stability analysis of steady states of the PDE system for a given parameter configuration (further details in Appendix B) provided an indicator for possible pattern formation—without requiring computation of the corresponding simulator output. This screening allowed to sample more input configurations without necessitating much of our user's time.

She set up the screening as follows: After doing the analysis and implementing a feature extractor (**D-drv**) that indicates the stability type arising from a given parameter configuration, she added *stability type* as a derived variable via the `ComputeNode` interface (see Figure A.1 in the Appendix). Color coding experimental points according to this derived variable, as shown in Figure 4, helped her to further focus more expensive pattern computation in the interesting unstable parameter region.

Our user computed the more detailed, but also more costly, output images only after carefully inspecting the relation between parameter configurations and the stability type derived variable. In doing so, she roughly reduced the computation time (full output images vs. stability feature only) by a factor of 60. This procedure successfully demonstrated the use of feature computation (**D-drv**) to address costs related to detail inspection (**A-view**).

Iterative sampling of the parameter space: ParaGlide allows to generate parameter configurations simply by specifying the containing region, as proposed in Section 5.1. Our user considered this approach a convenient way to generate input configurations and manage a repository of computed outputs. Aside from saving time, the interaction also put the user's focus on core questions of choosing and combining parameter value ranges. Coarse sampling to provide an overview within the selected range, followed by finer sampling in a more focused region to acquire details proved to be a good strategy—the structure in Figure 4, for instance, was found in this way and inspired our user to further analytic investigation. A particular finding was that an increase in repulsion (horizontal axis) leads to increased stability (dark dotted region) which corresponded well with out user's biological experience.

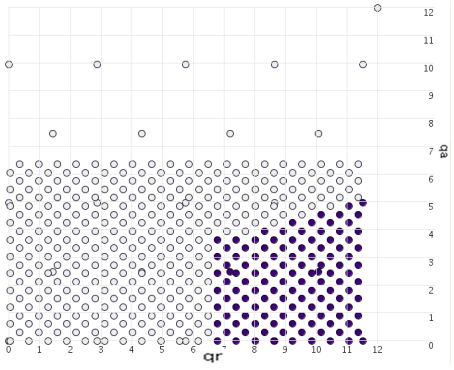


Fig. 4. Illustrating the parameter configuration creation in a sub-region of the parameter space iterating from coarse to finer sampling. (Un-)filled circles indicate parameter configurations that lead to an (un-)stable steady state.

Comparison of different model versions (A-cpr): Our user successfully employed ParaGlide to compare two different versions of the model with non-constant and constant velocities respectively. Since both models have the same input parameters, our user could use one simulator creating two different derived variables (**D-drv**), one for each model, and compare the different parameter configurations by color coding them (**A-view**, **A-psp**). ParaGlide facilitated a main insight in our user's work: Visualizing the stability type parameter for both model versions showed that the instability of most steady states tends to increase in the presence of non-constant velocities. The comparison task was not possible with prior practices.

Overall, the user in this case found ParaGlide to be *"a user-friendly tool that makes creating the parameter configurations and comparing the computations much easier [...] This tool gives me a better intuition about different outputs that correspond to various parameter configurations."*

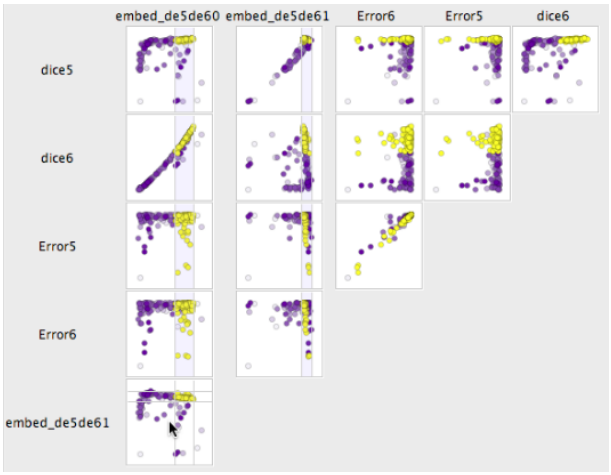


Fig. 5. Scatter plot matrix view that compares the point embedding (lower left) with the objective measures that went into computing its underlying similarity measure. The numbering of the responses corresponds to the class labels of Figure 2.

6.2 Case Study: Image Segmentation

In the following, we evaluate the use of ParaGlide in the context of the image segmentation use case described in Section 3.3. During three recorded meetings of overall 6 hours, our user employed ParaGlide to develop and implement a workflow that finds a good parameter configuration. The user's overall goal was to find a robust setting for eight parameters of a segmentation algorithm that produces *good* results for different patients and noise levels, assessed by ten derived variables (error measures). The term *good*, in the sense of this discussion, refers to all segmentations on a plateau of the (negative error) optimization landscape that have target values close to the global optimum. His goal was therefore not simply to find "the" optimal parameter configuration as in a standard optimization task (**A-opt**). When chosen from an initial, explorative sample of parameter configurations good configurations are also referred to as candidates, or representatives of a good cluster. Since *the* optimum is an ambiguous term in the context of multi-objective optimization, our method proceeds by first clustering all outputs that are similar to each other. This clustering leaves the task of finding out which cluster of outputs is a good one. The user found the good cluster by manual inspection of detail views for different outputs. The shape of the plateau of good outputs, viewed as color coded points in the dimension group of input parameters, informed the user about which parameters to keep and which ones to drop. Ultimately, it also informed a choice of parameter calibration for the algorithm.

Find good outputs by visual inspection: For easier inspection, our user initially focused on just the input parameters and two performance criteria. A SPloM view of this custom parameter group verified that the sampling method (**D-smp**) indeed distributed samples evenly over the 2D scatterplot projections. The user inspected the outputs by clicking on points in the scatter plot. However, it was not possible to improve all 10 performance criteria at the same time using this manual exploration process. The user considered optimization with this approach a "*very difficult to infeasible task that needed to be simplified*". The interaction steps with ParaGlide that allowed to address this task within a few meetings are described in the following.

Construct the good neighborhood: For most configurations of parameters, a small change in parameter values leads to a small change in the output behavior of the segmentation algorithm. This behavior means that for each good parameter configuration, it is worthwhile to explore the neighborhood around it to find additional good or even better configurations [19]. With ParaGlide's distinction of inputs and outputs it is possible to construct and combine different notions of neighborhood around a point. In order to obtain a notion of distance (**D-dist**) for this purpose, our user created a vector-valued derived variable that combined all performance criteria and normalized their dynamic ranges. Dot products between vectors in this feature space allowed to compute a similarity measure that was used to produce the spectral embeddings described in Section 5.2.

Our user then inspected the arrangement of outputs in the embedding which enabled him to decide whether the similarity measure was meaningful to sort out among "good" and "bad" segmentation results. Figure 5 shows the similarity embedding in the lower left plot, where the "good" cluster is highlighted in yellow. Judging from the strong diagonal distribution in two plots in the matrix, the horizontal embedding dimension is dominated by `dice6` and the vertical one by `dice5`. Since both variables should be maximized by good results, it is not surprising that visual inspection quickly identified the good cluster in the upper right of the embedding. The user "*found it convenient to make the cluster selection in the embedding*", which underlines the point that a manual, interactive clustering method may improve the user's confidence in the resulting cluster decomposition (see Section 5.2). The embeddings helped to simplify the selection of intervals for good parameter configurations and proved as an aid in a number of tasks:

- find good candidates around which to refine the sampling of configurations,

- group adjacent good outputs into cluster(s), and
- check the embedding by inspecting it in a SPloM view together with the feature variables as in Figure 5.

Assessment of multiple parameter effects: To determine the relevance of each parameter for the overall performance of the segmentation algorithm, our user viewed the distribution of the good cluster in input parameter space. To do so, he simply used the linking and brushing feature in the SPloM view. Being able to see the shape of the cluster with respect to parameter axes gave him a notion of sensitivity, where a large enough size of the good region indicated stability of this behavior with respect to parameter changes. This was particularly helpful towards his goal of finding "good" parameter regions and not just a single optimum:

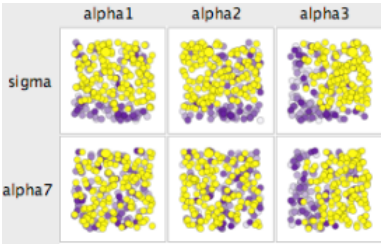


Fig. 6. Scatter plot matrix view of the good cluster (yellow) identified in Figure 5 viewed in the subspace of input parameters.

When projecting the cluster onto each variable individually, its extend was either spread out or localized in one or multiple density concentrations, as for instance shown in Figure 6. If the yellow good configurations in Figure 6 are spread out along a dimension, the corresponding input parameter is not impacting the quality of the output. For instance, parameters `alpha{1,2,7}` are considered as not impacting the quality of the output. While `sigma` and `alpha3` indicate clear thresholds beyond which the good parameter configurations are found. This observation informed our user of parameters to drop and, hence, *directly influenced model development*. Input parameters showing more localized good outputs or a clear transition were kept as part of the segmentation model and simply set to some robust value inside the good region.

Robustness and refinement of good regions: Our user identified multiple potentially good regions. For a final decision, he was particularly interested in how robust these regions were when applied to different patients or noise levels. ParaGlide directly supported his needs by allowing him to abstract region specifications (**D-roi**) and export/import them via XML (**W-rep**). He then applied and adjusted these region specifications for data sets of experimental points where segmentations were generated under different conditions, namely 2 different noise levels and 2 patients. Inspecting good regions across these multiple conditions not only allowed him to select among good regions, but also to fine-tune the one he selected. This resulted in a *"good and robust parameter choice"*.

Verify generalization to different patients: While the optimum has been made robust by constructing it over different experimental conditions, its performance has to generalize well beyond the condition used during adjustment. To verify this, the user performed a validation of the good configuration he found by testing it for 10 previously unseen patients.

Compared to the best configuration, the 10 validation segmentations showed very good Dice coefficients (mean=0.781, stdev=0.06) and excellent kinetic modeling errors (mean=0.062, stdev=10^{-4}) throughout. These results indicate that this configuration overall delivers high shape detection accuracy as well as low kinetic error. Two of the 10 data sets yielded just above average, yet acceptable, Dice coefficients, which *"inspired [me in] further investigation"*.

Overall, this user appreciated that the interaction steps suggested by ParaGlide greatly accelerated his daily practice.

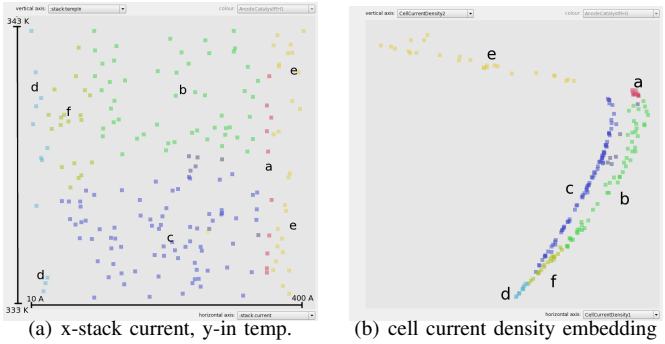
6.3 Case Study: Fuel Cell Prototyping

The fuel cell case introduced in Section 3.4 was concerned with a simulation model of a fuel cell stack that outputs 43 plots of different physical quantities characterizing the behavior of a cell stack. Figure 8 shows one such plot for 6 different configurations. Here, we report on a case study, in which the first author of this paper collaboratively with the two analysts from this use case used ParaGlide to investigate their model with parameter space partitioning. We use "we" to refer to this group of analysts and particularly focus on the use of low-dimensional embeddings that have only briefly been discussed in the case studies above.

In a prototypical experiment, we chose a region of interest over two input parameters: stack current ($10A..400A$) and stack inflow temperature ($333K..343K$). In this region 204 parameter configurations were sampled with a uniform random distribution shown in Figure 7a (x-axis: stack current; y-axis: temperature). To interact with the simulation code, we used ParaGlide's network connection feature. Connecting the model over the network allowed us to run multiple instances of the simulator to compute outputs and, in doing so, to leverage the computational power of parallel machines.

In Figure 7b the resulting 204 experimental points were arranged according to their output similarity in cell current density using spectral embedding as described in Section 5.2. In this embedding, simulation outputs could be inspected by clicking on points and we manually clustered, labeled and color-coded this output space using ParaGlide's rectangle selection. Note, that the clustering shown in Figure 7b, which was selected by our users, would have been hard to select with automatic clustering algorithms.

The view of the input parameter space (Figure 7a) was then encoded with the same color coding as the output space embedding (Figure 7b). That is, when going back to the input parameter space (Figure 7a), the color coding now reflected the parameter ranges of distinct behavior in Figure 7b. Our users explored this relation by inspecting cluster representatives as the ones shown in Figure 8 (Clicking on different points in either of these views). This exploration confirmed that *within* each cluster of parameter configurations the plotted curves are very similar. By comparing the representative behaviors of different clusters with each other, however, it became apparent that cell stack behavior *between* different parameter clusters can change significantly.



(a) x-stack current, y-in temp. (b) cell current density embedding

Fig. 7. Scatterplots showing 204 sampled experimental points in the fuel cell example: a) input parameter space showing two sampled parameters: current and inflow temperature, b) 2-dimensional embedding of current density outputs where spatial proximity reflects output plot similarity; the output space has been manually clustered into six regions; the concrete output plots are shown in Figure 8; These screenshots are from the 2007 C++ version of ParaGlide, and are also attainable in the currently discussed Java implementation.

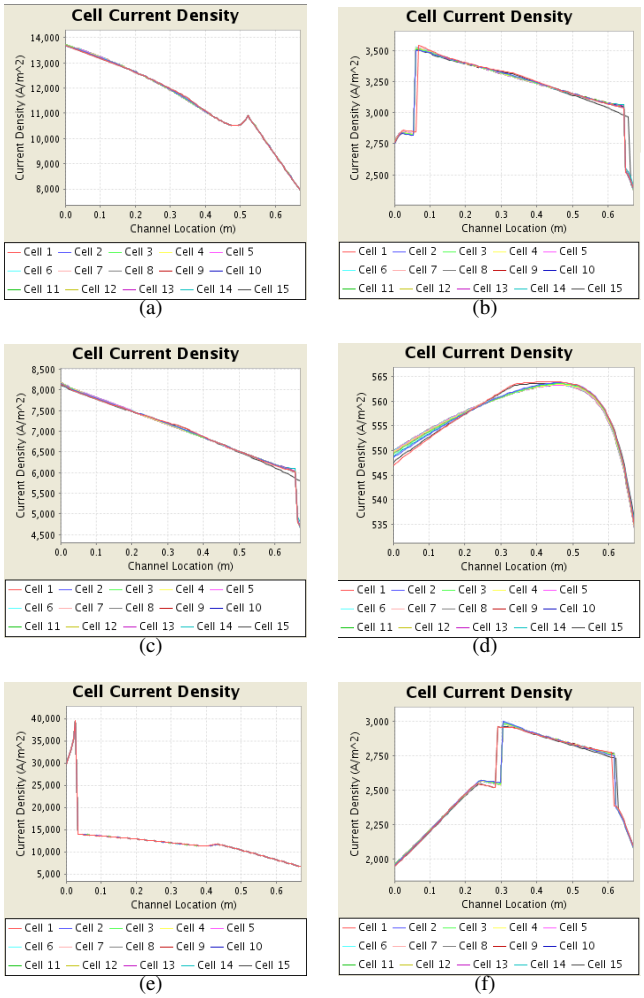


Fig. 8. Cell current density output plots for the parameter configuration, where (a-f) correspond to color-coded clusters in Figure 7b. The clearly differing cluster behaviors shown in these plots indicates that cluster boundaries in Figure 7 represent significant changes in output behavior.

Our users found the parameter space partitioning of Figure 7 intriguing, giving them a new method to study their model. For instance, they pointed out that while parameter cluster representative Figure 8e may be physically unreasonable, the corresponding yellow (e) region in Figure 7a was interpreted as a "bad" region, where adverse reactions occurred. Another useful observation was that response types (b, green) and (c, dark blue) are much less sensitive to changes in current than, for instance, (a, red), while all three of these clusters are robust against changes in temperature. This can be seen from the different size that these clusters have when viewed in the input parameter space of Figure 7a, where large vertical extend corresponds to robust behavior w.r.t. temperature and narrow horizontal extend to high sensitivity w.r.t. changes in current.

We also created other embeddings of output behaviors, for instance, showing the similarity of the water content of the membrane-electrode assembly. Due to restricted space, however, we do not report on these numerous other examples.

7 Discussion and Summary

We now discuss insights and lessons learned from applying ParaGlide in various settings of computational science.

7.1 Interaction

Human-in-the-loop: A key aspect of ParaGlide is the combination of user interactions for sampling the parameter space of a computational model together with methods for interactive analysis of the acquired data. We found that providing an interactive and intuitive way to explore a model a parametrization can have large impact on researchers' model development. In the example on movement of biological aggregations, the exploration possibilities offered by ParaGlide lead to a shift in the initial research questions. The initial goal of surveying types of patterns formed by different configurations (addressed by **D-roi**, **D-smp**) was achieved by the construction of a feature extractor (**D-drv**, **A-view**) that would help to accelerate this task by guiding the search for outputs with desired characteristics.

From a more general perspective, naturally not all settings in computational modeling require user interaction with the model as we propose it. To better understand when interactive modeling might be a fruitful path to go, we offer the following guidelines that we observed in our own field work. A need for interactive parameter space exploration might exist

- when user experience or *knowledge has not been fully integrated into the model yet* (development setting);
- in cases where model based optimization requires *visual assessment of graphical output*, e.g., if the model produces output images and these images are intended to be ranked by a domain expert; or
- if the model, rather than providing one optimal solution, is intended to *inform domain experts* about effects of mechanisms represented in the model (e.g., biologists reasoning about movement patterns, or a physician considering the shape of different possible segmented brain regions for diagnosis).

In short, all these modeling scenarios require human interpretation to gain understanding of the real system. A fully calibrated

model on the other hand that is only used in controlled, automated settings, often may not require human interaction at all. An exception might be for monitoring and verification of the assumptions it relies on.

Region-based sampling: As discussed in Section 3.5, parameter space analysis comes with the characteristic that data set generation is tightly interwoven with its exploration. In ParaGlide, the data set creation is split into two steps of region construction (**D-roi**) and the generation of experimental points (**D-smp**). We found that this workflow enabled our users to more consciously think about combining value ranges of different parameters. Analyzing previous practices revealed that most of our users simply used nested for-loops to generate uniform samplings across all dimensions. The important benefit of region based sampling is the support for actively steering the potentially expensive process of data sampling. Standard sampling methods can then be simply invoked in smaller regions, or iteratively to generate different granularities of sampling density.

Based on our parameter space partitioning method, we suggest that research efforts to further improve sampling efficiency can now approach the topic from two somewhat independent angles:

- by finding types of multi-dimensional ROIs that are understandable and intuitive to specify yet small in volume (to minimize computation), and
- by devising methods to adaptively sample a partially identified region by generating experimental points that explore the cluster transition boundary and descending into neighboring clusters. Currently, the arrow of Figure 1 that provides information of derived variables to guide the sampling method takes effect only through user interaction by relocating and refining the ROI based on observations made from prior computation or other context information.

Manual, high-dimensional clustering: Our partitioning workflow approaches labeling via manual clustering and involves the user in all stages of its construction. A by-product of the procedure given in Section 5.2 is a domain specific distance measure that can be used to partition computed outputs into different groups. Being aware that it is possible to automate some interaction steps, we were intrigued by the possibility of a manual procedure that produces meaningful clustering results for non-convex cluster shapes in a space with as many as 1502 dimensions[3], corresponding to the curves of Figures 8.

7.2 Impact of Parameter Space Partitioning

A major aspect of our work is to promote the idea of interactive, visual parameter space partitioning. As pointed out in Section 4, existing interactive systems to work with computer models mainly focus on interactive model optimization. While not originally intended for this purpose, a meaningful segmentation of the model parameter space provided by our partitioning method, may also have potential use in this context. We saw the coexistence of optimization and partitioning tasks, for instance, in our image segmentation, and fuel cell examples. Pitt et al. [22] already identified one major benefit, stating that the diversity of behavior that a model can produce provides a measure of the power of the modeling mechanisms, indicating possible risks of overfitting or oversimplification.

Building on these findings, our work showed that visual exploration of parameter space partitions can provide users with confidence that the right level of complexity of modeling mechanisms has been chosen. An example is the parameter assessment of Section 6.2. In particular, we identified two general situations in which parameter space partitioning supported our users' needs:

Context-dependent optimization: Parameter space partitioning can become helpful in context-dependent optimization, where depending on additional conditions (context) different parameter configurations lead to the intended good results. For instance, in our fuel cell example the engineers were interested in fuel cell characteristics in two different situations, where either very low or high current is drawn from a cell. Knowing the shape of the parameter regions that perform well under the respective conditions allows to choose a set of parameter configurations that robustly lead to good results.

Visual sensitivity analysis: Another very promising property of the visual partitioning approach we suggest, is its application to global sensitivity analysis. Clustering the output space and showing these clusters respectively in the input parameter space, gives the user an intuitive way to learn about the sensitivity of certain output behaviors. Since cluster boundaries represent a significant change in output behavior, the narrowness of an output cluster in the direction of change of a particular input parameter can be interpreted as a measure of sensitivity of the cluster behavior with respect to that parameter. An example of such a sensitivity interpretation is given in our image segmentation case study (Section 6.2). Here, the user leveraged the cluster boundary distance as a measure of robustness. The interpretive power of viewing several such regions is apparent in fuel cell case study (Figure 7a).

While parameter spaces in our examples are continuous, our analysis operates on sampled configurations. Alternatively, topological decomposition [13] and simplification [4, §2] of multi-dimensional functions are possible continuous approaches.

7.3 Conclusions

Developing, interpreting, and improving computer models can be a time and resource intensive process. In this paper, we propose parameter space partitioning and exploration as a strategy to support this process. After studying, formalizing and abstracting the problem of parameter space partitioning, we engaged in developing ParaGlide, a system that enabled us to address practical modeling settings, that have not been properly supported before. ParaGlide was developed in a user-centered design process and closely connects data generation with its visual exploration. Studying ParaGlide in the field revealed various examples in which our users could either

3. Also, due to the functional dependency on the two varied parameters all data lies in a non-linear, two-dimensional subspace of $\mathbb{R}^{1502}$.

speed-up their work or get new insights which informed model development. We hope that others will find our work inspiring and extend it with further innovations to support model developers in their data-intensive and complex endeavors.

ACKNOWLEDGEMENTS

We would like to acknowledge Paul Chang, Brian Wetton, and Razvan Fetecau for helpful feedback on our use cases. We are grateful for contributions to the implementation by Gary Lo (SPloM), Veronika Irvine (XML), and Stephen Ingram (glimmer). Tamara Munzner and Melanie K. Tory provided insightful perspectives during design discussions.

REFERENCES

[1] L. Bavoil, S. Callahan, P. Crossno, J. Freire, C. Scheidegger, C. Silva, and H. Vo. VisTrails: enabling interactive multiple-view visualizations. In *IEEE Conference on Visualization*, pages 135–142, Oct. 2005.

[2] W. Berger, H. Piringer, P. Filzmoser, and E. Gröller. Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum*, 30(3):911–920, 2011.

[3] S. Bergner. *Making choices in multi-dimensional parameter spaces*. PhD thesis, Simon Fraser Univsersity, 2011.

[4] S. Bergner, R. Pohle, S. Al-Zubi, K. D. Tönnies, A. Eitner, and T. R. Neu. Segmenting microorganisms in multi-modal volumetric datasets using a modified watershed transform. In L. V. Gool, editor, *Proc. Patt. Rec., 24th DAGM Symp., volume 2449 of LNCS*, pages 429–437, Zurich, Switzerland, September 2002. Springer Verlag.

[5] P. Berkhin. *Grouping Multidimensional Data*, chapter A Survey of Clustering Data Mining Techniques, pages 25–71. Springer, 2006.

[6] S. Bhagavatula, P. Rheingans, and M. desJardins. Discovering high-level parameters for visualization design. In *Eurographics / IEEE VGTC Symposium on Visualization (EuroVis)*, pages 255—262, June 2005.

[7] V. Bhatt and J. Koechling. Partitioning the parameter space according to different behaviors during three-dimensional impacts. *Journal of Applied Mechanics*, 62:740, 1995.

[8] M. Booshehrian, T. Möller, R. M. Peterman, and T. Munzner. Vismon: Facilitating analysis of trade-offs, uncertainty, and sensitivity in fisheries management decision making. *Computer Graphics Forum (Proc. EuroVis 2012)*, 31(3):1235–1244, June 2012.

[9] P. Chang, G.-S. Kim, K. Promislow, and B. Wetton. Reduced dimensional computational models of polymer electrolyte membrane fuel cell stacks. *J. Comput. Phys.*, 223(2):797–821, 2007.

[10] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[11] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+ context visualization of complex simulation data. In *Proc. of the Symp. on Data Vis. 2003*, pages 239–248. Eurographics, 2003.

[12] R. C. Fetecau and R. Eftimie. An investigation of a nonlocal hyperbolic model for self-organization of biological groups. *J. Math. Biol.*, 61(4):545–579, 2010.

[13] S. Gerber, P. Bremer, V. Pascucci, and R. Whitaker. Visual exploration of high dimensional scalar functions. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1271–1280, 2010.

[14] L. Grady. Random walks for image segmentation. *IEEE Trans. on Pattern Analysis and Mach. Intelligence*, 28(11):1768–1783, Nov. 2006.

[15] T. Jankun-Kelly, K.-L. Ma, and M. Gertz. A model and framework for visualization exploration. *Visualization and Computer Graphics, IEEE Transactions on*, 13(2):357—369, March 2007.

[16] F. Lutscher and A. Stevens. Emerging patterns in a hyperbolic model for locally interacting cell systems. *J. Nonlinear Sci.*, 12:619–640, 2002.

[17] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[18] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *SIGGRAPH '97*, pages 389–400. ACM Press, 1997.

[19] C. Mcintosh and G. Hamarneh. Is a single energy functional sufficient? Adaptive energy functionals and automatic initialization. *Proc. MICCAI, Part II*, 4792:503–510, 2007.

[20] J. Mulder, J. van Wijk, and R. van Liere. A survey of computational steering environments. *Future Generation Computer Systems*, 15(1):119 – 129, 1999.

[21] J. K. Parrish. Using behavior and ecology to exploit schooling fishes. *Environ. Biol. Fish.*, 55:157–181, 1999.

[22] M. Pitt, W. Kim, D. Navarro, and J. Myung. Global model analysis by parameter space partitioning. *Psychological Review*, 113(1):57, 2006.

[23] A. Pretorius, M. Bray, A. Carpenter, and R. Ruddle. Visualization of parameter space for image analysis. *IEEE Trans. on Vis. and Comp. Graph.*, pages 2402–2411, 2011.

[24] A. Saad, G. Hamarneh, T. Möller, and B. Smith. Kinetic modeling based probabilistic segmentation for molecular images. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008*, pages 244–252, 2008.

[25] T. Santner, B. Williams, and W. Notz. *The Design and Analysis of Computer Experiments*. Springer, 2003.

[26] M. Sedlmair, P. Isenberg, D. Baur, and A. Butz. Information Visualization Evaluation in Large Companies: Challenges, Experiences and Recommendations. *Information Visualization*, 10(3):248–266, July 2011.

[27] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: reflections from the trenches and the stacks. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 18(12):2431–2440, 2012.

[28] B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Proc. AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV)*. ACM, 2006.

[29] T. Torsney-Weir, A. Saad, T. Möller, B. Weber, H.-C. Hege, J.-M. Verbavatz, and S. Bergner. Tuner: Principled Parameter Finding for Image Segmentation Algorithms Using Visual Response Surface Exploration. *IEEE Trans. on Vis. and Comp. Graphics*, pages 1892–1901, 2011.

[30] J. van Wijk, R. Van Liere, and J. Mulder. Bringing computational steering to the user. In *Scientific Visualization Conference, 1997*, pages 304–304. IEEE, 1997.

[31] P. Wong and R. Bergeron. 30 Years of Multidimensional Multivariate Visualization. *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 3–33, 1994.

Steven Bergner is currently a PostDoc at the Dept. of Statistics and Actuarial Science at Simon Fraser University (SFU). Prior to that, Steven completed his PhD in Computing Science (SFU) in 2011 under the supervision of Prof. Torsten Möller. His pursuit of computational modeling in visualization arose from previous work in multi-shape recognition (Dipl.-Ing. Comp. Visualistics, Univ. of Magdeburg). Apart from data visualization and human-computer interaction, his research interests include algorithm development for signal processing, computer graphics, experimental design, and quantitative analysis. Studying various research workflows also woke an interest in reward systems for intellectual work that recognize sources as well as the hubs. Steven reviewed for IEEE Transactions and Conference on Visualization, ACM SIGGRAPH/Transaction on Graphics, and EuroVis holding memberships in IEEE, ACM, SIAM, and the German National Academic Foundation.

Michael Sedlmair is a PostDoc in the Imager Lab at the University of British Columbia, Canada, working with Tamnara Munzner. In 2010, he finished his PhD on the topic "Visual Analysis of In-car Communication Networks" under the supervision of Andreas Butz at the University of Munich. His PhD work was funded by and conducted at the BMW Group Research and Technology where he closely collaborated with automotive engineers to analyze and better support their data visualization needs. His research interests focus on studying visual data analysis techniques, tools and users in real-world settings.

Torsten Möller is a professor at the School of Computing Science at Simon Fraser University. He received his PhD in Computer and Information Science from Ohio State University in 1999 and a Vordiplom (BSc) in mathematical computer science from Humboldt University of Berlin, Germany. He is a senior member of IEEE and ACM, and a member of Eurographics. His research interests include the fields of Visualization and Computer Graphics, especially the mathematical foundations thereof.

He is co-director of the Graphics, Usability and Visualization Lab (GrUVi). He is the appointed Vice Chair for Publications of the IEEE Visualization and Graphics Technical Committee (VGTC). He has served on a number of program committees and has been papers co-chair for IEEE Visualization, EuroVis, Graphics Interface, and the Workshop on Volume Graphics as well as the Visualization track of the 2007 International Symposium on Visual Computing. He has also co-organized the 2004 Workshop on Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration as well as the 2010 Workshop on Sampling and Reconstruction: Applications and Advances at the Banff International Research Station, Canada. He is a co-founding chair of the Symposium on Biological Data Visualization (BioVis). In 2010, he was the recipient of the NSERC DAS award. He received best paper awards from IEEE Conference on Visualization (1997), Symposium on Geometry Processing (2008), and EuroVis (2010), as well as two second best paper awards from EuroVis (2009, 2012).

Sareh Nabi Abdolyousefi received a B.Sc. degree in Applied Mathematics at Sharif University of Technology, Iran, in 2007. She then joined the graduate program in the Mathematics Department at Simon Fraser University, Canada, in 2008. During her mathematics research, she focused on investigating and simulating mathematical models used to study the behavior of biological aggregations.

After finishing her M.Sc. in Mathematics, she joined the Department of Economics at Simon Fraser University in 2011. She continued to build upon her statistical, financial, economics, and programming backgrounds and received her M.A. degree in Economics in August 2012.

Her main research interests are in the areas of mathematical modeling, quantitative analysis, econometrics, statistical analysis, asset pricing models, and quantitative finance.

Ahmed Saad received the BSc and MSc degrees in Systems and Biomedical Engineering from Cairo University, Egypt. He is currently working toward the PhD degree at the School of Computing Science, Simon Fraser University. His research interests span the fields of medical image analysis and visualization. He is also interested in molecular imaging, computer graphics, visual analytics and medical informatics.