Computers & Graphics: X 2 (2019) 100010

Contents lists available at ScienceDirect

Computers & Graphics: X

journal homepage: www.elsevier.com/locate/cagx

Technical Section

Cellpackexplorer: Interactive model building for volumetric data of complex cells☆

Magdalena Schwarzl[a,d,*], Ludovic Autin[b], Graham Johnson[c], Thomas Torsney-Weir[e], Torsten Möller[a]

[a] University of Vienna, Währingerstraße 29, 1090 Wien, Austria
[b] The Scripps Research Institute, California, United States
[c] University of California, San Francisco (UCSF), California, United States
[d] Visualization Research Center, University of Stuttgart, Allmandring 19, 70569 Stuttgart, Germany
[e] Department of Computer Science, Swansea University, Fabian Way, Swansea, SA1 8EN, UK

ARTICLE INFO

Article history:
Received 20 December 2018
Revised 1 July 2019
Accepted 5 August 2019
Available online 4 October 2019

Keywords:
Interactive visual analysis
Probabilistic 3D data
Ensemble visualization
Biological data

ABSTRACT

In this paper, we describe cellPACKexplorer, a system designed to help developers of cellPACK find errors in and improve their algorithm. cellPACKexplorer focuses on visualizing the effects of cellPACK recipe parameters on the final packing output. We found that the developers have two different methods for understanding the output, numerical and visual, depending on their background. We designed cellPACKexplorer with a flexible interface to support both types of users. We evaluated our tool through case studies and questionnaires. Novice users were able to create cell models with cellPACK and explore the behavior of different parameters. Further, expert users discovered an error in the code and were able to locate the problem quickly with our new analysis tool. We conclude with a discussion of the implications of our findings in the wider visualization community.

1. Motivation

A typical modeling process consists of a model setup, an optimization process and a validation of that model. This is true whether dealing with agent-based modeling [1], statistical modeling (such as regression, classification, or clustering) [2] or computational modeling [3]. However, if the parameter space is vast, or if the optimization function is qualitative (e.g. through visual inspection) then this model building process can be quite tedious. This is one of the first papers that focuses on the visual support of model building in the biological domain. The main motivation is to observe a very specific model building process and to show that visual support can tremendously speed up and help in model building.

In this work we focus on cellPACK [4], an open-source framework designed to generate and refine geometric structures of whole cells for researchers. The current setup of cellPACK requires the user to specify a number of input parameters to build virtual cells. These parameters influence the complex interactions among the various molecular "ingredients" (e.g proteins) for a particular packing to produce a final molecular cell. As these interactions between different parameters are very complex it is hard to predict the output related to a specific input setting. For more complex cases it is even impossible. The computation time for cellPACK outputs can be up to several days which further increases the difficulty of heuristically finding a fitting input parameter set. Further, a proper validation of the model often happens visually by comparing the result to textbook images (e.g. [5]) or prior experience. The aforementioned problems are a major bottleneck for the development of cellPACK and to the community participation required for consensus shaping on the scale of whole cells. New approaches are needed to make cellPACK more robust, easier to develop, and easier to test.

We have developed cellPACKExplorer to make developing cellPACK easier and assist the developers of cellPACK in the ongoing development. One of the complexities of understanding the results of cellPACK is that the packing algorithm is stochastic. In other words, for a particular parameter configuration cellPACK produces

☆ This article was recommended for publication by S. Bruckner.

* Corresponding author at: University of Vienna, Währingerstraße 29, 1090 Wien, Austria.
E-mail addresses: magda.schwarzl@gmail.com (M. Schwarzl), t.d.torsney-weir@swansea.ac.uk (T. Torsney-Weir).

https://doi.org/10.1016/j.cagx.2019.100010

an ensemble of volumes. cellPACKexplorer supports a new workflow to simplify the development of cellPACK, setting up cellPACK experiments and for analyzing and sharing cellPACK outputs and experiments.

Our contributions include a detailed user, data, and task analysis comparing the model building tasks of the cellPACK developers to model usage tasks which have been explored more thoroughly in the visualization community [6]. We focused on aiding the developers in improving the core packing capabilities of their model to better help them select the crucial features (input parameters), hide less important ones from future users, and to find proper defaults for some others so future cellPACK users (e.g. biologists, illustrators) are able to quickly create cellPACK outputs themselves. The developers of cellPACK were able to speed up the setup to large experiments from 30 min to 1 min and were able to analyze ensembles of hundreds of cellPACK outputs which they could not do before. It also revealed unknown behavior of their tool to them and helped them to validate the influence of input parameters on the generated outputs. Another advantage to cellPACK developers is that cellPACKExplorer makes collaboration and sharing of experiments easier.

2. Related work

The goal of cellPACKExplorer is to help the developers understand the effects and the range of possible packing parameters as they add and modify input parameters to the packing operations of cellPACK. Our approach with cellPACKexplorer is to combine parameter space analysis [6] with ensemble analysis of the set of 3D outputs.

We will first discuss how our work relates to existing taxonomies to characterize the vital user, data, and task characteristics of a design study. We focus on parameter space analysis [6], data types [7], and the computational pipeline [8].

In the language of von Landesberger et al. [8], cellPACKExplorer has the assumption, number of combinations, ease of comprehension, and subjectivity of output requirements. Based on their survey, no tool addresses all these tasks. In terms of data, for a given parameter combination, the cellPACK algorithm returns a set of locations for geometrically modelled proteins (ingredients). Therefore, our data falls squarely in the category of multirun simulation data in the framework of Kehrer and Hauser [7]. A crucial aspect of developing effective models is understanding the expressiveness of a model. Hence, cellPACKExplorer assists in grouping the outputs of the model for various parameter combinations based on similarity (of the output). An examination of the parameter sets that have created these groups helps to reason about the importance of specific parameters. Sedlmair et al. [6] identify these tasks as *partitioning* and *sensitivity* tasks. We compare our own work to other methods designed for these tasks in the following sections.

2.1. Parameter space analysis on fixed models

As cellPACK is under constant development, the available parameters are constantly being extended. One of the core questions is whether these parameters properly capture the range of realistic cells or whether they might be redundant with little influence to the final output generated. Our approach to help the developers answer such questions is to let them visually inspect the influence of new parameters on the range of possible outputs. This could be considered as a hybrid approach between code-level debugging and visual model building.

Many tools assist the user with the parametrization of a fixed model (i.e. a fixed algorithm). These methods are usually tied to a specific model. For example, in the context of segmented regression [9] and treed regression [10], Guo et al. [11] focused on the development and evaluation of linear models on subsets of the data. This approach was extended by Mühlbacher and Piringer [12] to include non-linear trend discovery. Likewise, McGregor et al. [13] present a system for Markov decision processes. CVVisual [14] provides code snippets that can be introduced into image processing source code to provide debugging-type visualizations.

2.2. Ensemble analysis

In cellPACKExplorer we want to analyze ensembles at two different levels. We want to group a number of outputs into distinct sets of outputs based on large-scale differences (see the previous section) and analyze the smaller-scale variations within these sets. While there are approaches to ensemble analysis and approaches that treat ensembles as distributions (see Kehrer and Hauser for an overview [7]), to the best of our knowledge there is no work in a similar setup, that requires a flexible interface due to a constantly changing underlying algorithm and changing analysis requirements.

2.2.1. Distributional approaches

For stochastic simulations, one usually examines the distribution of output possibilities resulting from a single parameter configuration. To help show these distributions, the notion of a boxplot was extended for curves with the introduction of contour boxplots and curve boxplots [15,16]. These were used, for example, for visualizing the range of possibilities of storm tracks. While these work well for showing distributions of 1D functions cellPACK produces two- or three-dimensional outputs which require a different solution. VAICo [17] considers a set of 2D images and computes the regions of difference of the set, clusters them, and then gives the user controls to browse these differences. While the aim of VAICo is to identify pixel level differences in images we are looking at understanding structural volumetric differences in the set of (probabilistic) volumes.

cellPACK is primarily intended for 3D output. Three-dimensional objects are often represented as either voxels or parametric objects. For voxel-based data, there are visualization methods such as probabilistic marching cubes [18] or MObjects [19]. One can also animate between all 3D objects in an ensemble as in Ehlschlaeger et al. [20] or Lundström et al. [21]. While animation techniques will work on general 3D output, animation can contribute to a higher cognitive load for users especially if the time axis in the animation does not correspond to time in the data [22]. In cellPACKexplorer, we combined the intuitive notion of a 1D distribution with the detail of 3D. We show 1D distributions of derived metrics with a user-selectable view of a single projected 3D output.

2.2.2. Clustering approaches

Partitioning is often done using a clustering approach. For example, Design Galleries [23] uses a distance metric to present a set of visually distinct possible renderings of a scene. Likewise, Fluid explorer [24] clusters a set of fluid simulation animations into animation segments that are then inspected. While convenient, both approaches to a clustered presentation of the model outputs showed deficiencies. Hence, other researchers adopted a more manual adjustment. For example, Paramorama [25] is focused on finding which parameter settings produce good segmentations based on manual inspection of the resulting images. They group outputs in a hierarchical fashion based on input parameter settings. Paraglide [26] enables a manual partitioning of the output space in order to draw conclusions on the input parameter values. An a priori partitioning scheme is not clear in the case of cellPACKexplorer. In addition, the partitioning scheme might be refined as the algorithmic description (and therefore the underlying model)

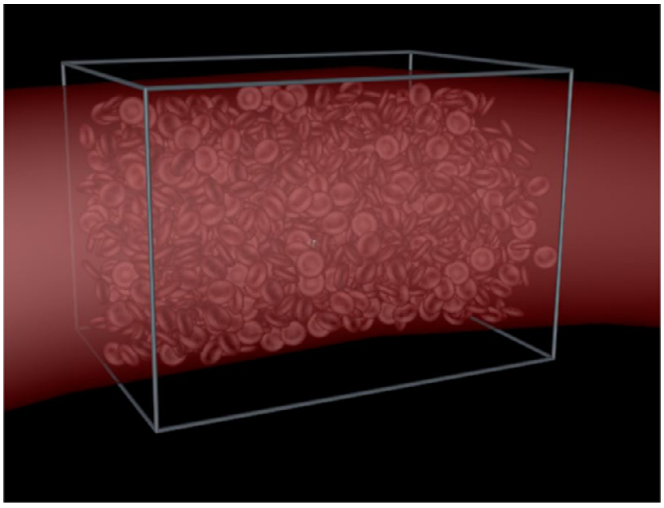

Fig. 1. A simulation of the red blood cell distribution (1413 cells) in a blood capillaries of radius 30μm and length 100μm built with cellPACK. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

changes. An evolving model description often brings new parameters with unknown effects. Therefore, we support a manual partitioning of the data by letting the user filter on input parameters and metrics computed on the output.

3. cellPACK

Since cellPACKExplorer is designed to work with cellPACK we provide an introduction to cellPACK itself here. cellPACK is an open-source biological software framework designed to assemble large-scale cells and cellular substructures from small-scale molecular building-blocks. cellPACK was designed to combine data from all branches of biology spread over different small-scale studies into comprehensive cells. As parameters reflect real biological properties of proteins and their interaction, cellPACK can be used for hypothesis generation and experimentation (imitating localizations and interactions), validation, communication, education and to view the mesoscale ($10^{-7}, \ldots, 10^{-8}$m) with atomic-resolution detail. The ultimate goal is that cellPACK will be accessible to audiences without a technical background and serve as a structural and informatics foundation for broader projects[1] at the Allen Institute for Cell Biology, which aim to generate dynamic virtual representations of whole cells for predictive experimentation. cellPACK can be used, for example, to fill an architectural engineering shape with concrete aggregate in preparation for earthquake simulations, or it can fill an artery with blood cells at appropriate densities to generate a histological representation for a medical illustration (see Fig. 1).

A major research focus of the developers of cellPACK is to create structural cells of the Human Immunodeficiency Virus (HIV). They use a packing approach to place smaller building blocks (modelling proteins and smaller cells) into larger volumes, both described geometrically. A cellPACK input file (called a recipe) contains a list of molecular building-block components (called ingredients) with behaviors (input parameters) that mimic biological constraints (e.g. attraction and repulsion). Each ingredient has its own set of input parameters that govern how it will pack with the other components of the cell. Fig. 2 demonstrates an example of the packing problem for HIV. Since HIV illustrates a rather complex scenario, the developers use simpler models for development. One of their approaches is to pack spheres with different radii into a box or on a plane. They explore how the generated outputs change under different input configurations (e.g. different parameter values for attraction between two sphere types).

cellPACK's input parameters can be split into two groups, general parameters and ingredient parameters. *General parameters* influence properties of the packing algorithm affecting the whole cell. Two examples include the resolution of cellPACK's spatial tracking grid, and a variety of options for how the next point on the grid to be packed (assigned to an ingredient) is selected. This point is then proposed to an ingredient as position in the output. In the literature this set of parameters is described as *model parameters* [27] which do not provide biological information.

Ingredient parameters on the other hand mimic the behavior of real-world biological proteins, they have to be set for each ingredient type independently and specify which protein an ingredient mimics. An example for this parameter set is known binding partners for an ingredient (i.e. attraction to a specific other ingredient type). They decide whether an ingredient accepts or declines the position proposed by the algorithm. In the literature they are referred to as *control parameters* [27], as they are meaningful in the biological domain.

4. Methodology

cellPACKexplorer was developed over ten months using the design study methodology [28]. This period consisted of two major development circles each ending with the evaluation of a prototype tool by the cellPACK developers. During the first cycle we had weekly meetings with the cellPACK developers. During this time we developed our initial user, data, and task characterization. We also presented the developers with successively refined prototypes. After we evaluated our first prototype with the developers, we discovered that their analysis interests varied as the code was updated. Further we found that they preferred to keep control of their analysis and therefore did not like the automatic clustering. Additionally we observed that they hardly went back to the input part of the interface. We updated our user, data, and task characterization and designed a new prototype tool. We present the results of this second development cycle in the following sections.

5. Problem description

We analyzed the workflow of the cellPACK developers in weekly meetings, asking them about their usual work and future goals for the cellPACK development. After collecting this data we characterized their data types and tasks following the taxonomy of Munzner [29]. We then synthesized what we consider the main model building tasks conducted by the developers of cellPACK in order to help them improve the understanding of their model and ultimately improve cellPACK. In what follows we summarize the result of our analysis.

5.1. User characterization

The cellPACK algorithm is still under development and we focused on the core developers of cellPACK as users for the time being. cellPACKExplorer assists them in their work of improving the cellPACK algorithm and simplify the parameter configuration for future users. The ultimate goal of the developers is to automatically generate realistic cells of biological structures (e.g. HIV, Blood-Plasma) from small components (called ingredients). They extract information from reports of various small-scale studies on chemical and biological properties of cells and convert it into input parameters that mimic these biological constraints. In their current workflow the developers iteratively refine cellPACK output models (e.g. HIV or Blood-Plasma cells) by changing the input parameter

[1] http://www.scripps.edu/newsandviews/e%5F20150921/vmcc.html.

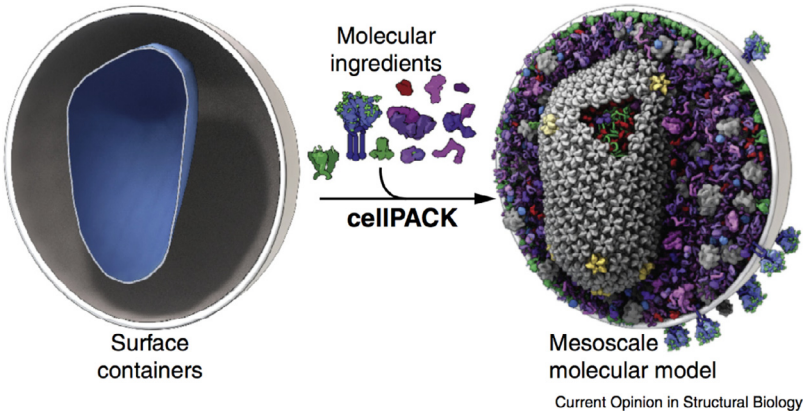

Fig. 2. Example of a cellPACK output that results from packing an HIV recipe. Left: the empty packing volume. Middle: different ingredients (proteins) to be packed. Right: one stochastic packing result (of hundreds). Note the emergent complexity of the protruding green/blue ingredients, which packed with a bias towards one side of the spherical surface, as the result of several simple molecular building blocks interacting in a variety of localized manners. Figure used with permission [4]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

settings (model-usage) or adapting a parameter's functionality and adding new parameters (model-building) through updates to the core algorithm. This loop continues until the produced outputs correspond to the anticipated results. To validate that their algorithm behaves correctly the developers build a number of different cells.

We found that different developers have different approaches in validating the code. The cellPACK developers come from a diverse set of backgrounds. In our case, one of them has a background in scientific illustration and analyses the outputs visually for correctness. Another developer has a technical background and focuses on derived statistical metrics to explore and validate the outputs. In our design process we aimed to support all types of cellPACK developers. In order to build a system that supports both we designed an interface that supports a wide range of skills and preferences. In addition, the developers do not just fine-tune cellPACKs's parameters to create the output of interest but rather adapt the underlying model (i.e. the cellPACK algorithm) directly to work as expected. We argue that an interface of high flexibility is specifically appropriate for model building in general as the underlying algorithm changes and thereby the used analysis tools and metrics change as well. The developers continuously update the cellPACK algorithm, develop new recipes to create different cells and include information from more and more studies from the literature. We found this to be one of the distinguishing aspects of model building as opposed to model usage.

5.2. Data characterization

Since we are focusing on model building rather than model usage, the data analyzed in cellPACKExplorer is the cellPACK algorithm itself. cellPACKExplorer takes input parameter configurations for cellPACK to initiate the computation of multiple cellPACK output models. The cellPACK input parameters have different data types. Some of them are categorical, for example, specifying which algorithm should be used to handle intersections of ingredients. Others are numerical, e.g. influencing the binding probabilities between ingredient types of cellPACK have changed many parts and often add new parameters and functions. New parameters can supersede or otherwise affect other parameters and the downstream results that are produced. For each input configuration, cellPACK's output consists of a 3D position for each copy of every ingredient type resulting in a spatial cell. Each single parametrization of the cellPACK model produces a number of outputs by stochastically varying the naturally occurring variations in the biological cells (i.e. initializing the algorithm with different random seeds). These outputs differ through the used random number.

We define a run as the creation of R different simulation outputs by re-running the simulation R times with the same input parameter configuration but a different random seed input setting. In the interface all filters work on the level of runs as atomic units. We define an experiment to be a subset of parameters that are varied over a range of parameter values. An experiment consists of N different runs created by N different input parameter configurations. This results in N sets of R results giving a total number of $N \times R$ volumes.

At the moment the developers work with smaller datasets varying about 3 parameters and generating roughly a hundred output models. In the simple test cases the ingredients are simplified to spheres of different radii and limited to about 5 ingredients with about ten copies per ingredient type. For the more complex models such as HIV the number of ingredients packed scales up to millions of ingredients.

5.3. Tasks

We first describe the current workflow of the cellPACK developers, present how we abstracted their tasks (T1-Setup through T6-Improve) and designed cellPACKexplorer to work more efficiently. A conceptualization of our resulting approach can be found in Fig. 3.

5.3.1. Current workflow

Currently, the developers use a simple trial and error [6] strategy, exploring one run at a time. They create a hypothesis what the cellPACK output for specific parameters will look like and then run the model to verify their hypothesis. This works well for experiments with a single ingredient type and a short computation time. As recipes get more complex and more ingredient types are packed, interaction effects between parameters make it impossible to predict the output.

To analyze these recipes they compute a few outputs one by one to rough out parameters and ranges to sample. These simpler experiments consist of only two to three parameters and about ten different seeds. They also run these small experiments to confirm that the system is still working after changing the code. The next step in their analysis is to write custom scripts in Python to sample the parameter ranges by varying less than twenty specified random seeds and less than five parameters. They analyze the outputs visually and statistically in MS Excel. For the different analysis aspects the developers use a suite of different 2D or 3D viewers to look at the raw outputs, create density maps of multiple ensemble members and illustrate renderings of single output cells. Finally,

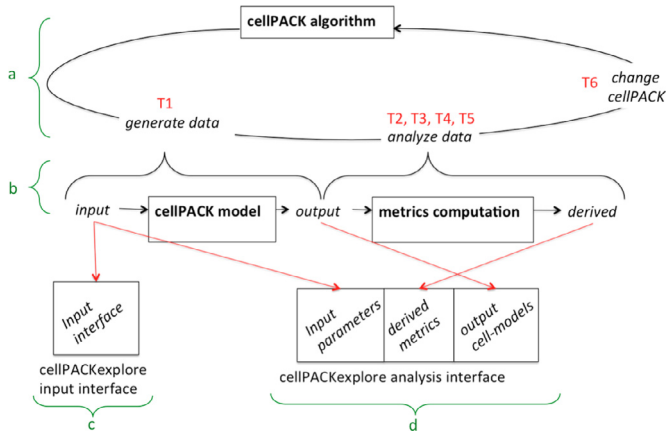

Fig. 3. The conceptual workflow of the developers of cellPACK using cellPACKExplorer. (a) The developers' mental representation of the workflow. They start with a version of cellPACK code, create a number of outputs to test it (T1-Setup), and, based on their data analysis (T2-Validate, T3-Identify, T4-Default), discuss their results (T5-Share) and create an improved version of the cellPACK code (T6-Improve). Prior to cellPACKExplorer, generating and analyzing the data was done as a tedious manual process. (b) cellPACKExplorer's pipeline guides the developers through the analysis process. (c) The setup of an experiment (see Section 7.1) as well as (d) the analysis of the experiment (see Section 7.2) are supported through visual interactive interfaces and improves their workflow.

they adapt the cellPACK algorithm and start a new round of the described workflow.

One bottleneck is the computation time. For a single output it can range from one 100th of a second for simple recipes to several days for more complex recipes packing millions of base pairs or proteins. Another barrier is the communication overhead. One developer has a background in scientific illustration and usually asks the other developer to write scripts to set up experiments sampling input parameters as these have to be written in python, where he has only limited experience. For statistical analysis of the outputs he again asks the other developer to implement scripts and measures. This approach requires a lot of communication between the two developers and the necessity for them to share data.

5.3.2. Tasks

Based on the cellPACK developer's current workflow and goals we abstracted the following tasks:

T1-Setup: Experiment setup: This requires selecting a subset of input parameters, a range for sampling, and a decision on the number of samples to be generated (compare *N*, *R* in Section 5.2). Usually the technically-trained developer was responsible of creating runs and outputs as well as statistical summaries, while the other developer engaged in the validation with respect to biological (or other) ground truth.

T2-Validate: Model validation through output comparison: When parts of the code are changed, it is important to make sure that the cellPACK model is still valid and produces correct results by comparing with data from the literature or textbooks (e.g. [5]) and current domain knowledge. This requires the analysis of the probabilistic volume ensemble set related to an experiment. cellPACK outputs are checked to ensure they satisfy several statistical constraints like concentration of ingredients and distribution over the volume.

T3-Identify: Identify parameters to be exposed: Parameters that greatly impact the range of 3D outputs should be exposed to the future users of cellPACK. However, too many parameters could overwhelm a new user with unnecessary complexity and hurt the adoption of cellPACK. In addition, packing parameters might not be intuitive to non-technical users. Therefore, the developers of cellPACK have to make a careful selection and need to understand the behavior of different parameters and their interactions.

T4-Default: Identify reasonable default values for other (hidden) parameters: After identifying which parameters to expose, the developers of cellPACK need to decide what are reasonable defaults for the remaining parameters. These default values should produce accurate results without additional configuration.

T5-Share: Share results: As the developers of cellPACK work on the code and analysis together they need to be able to share data to show findings to each other. This should be as automatic as possible to speed up the collaboration.

T6-Improve: improve cellPACK: The developers constantly improve the quality and speed of cellPACK. In addition, the insight gained on the impact of particular parameters leads to removing some and adding others.

5.3.3. Proposed changes

With our new tool we aim to reduce communication overhead, combine all analysis into one application, enable the developers to work independently and support a more systematic analysis. cellPACKexplorer enables the setup of an experiment (T1-Setup) without programming knowledge through a visual interface (see Section 7.1). Afterwards the ensemble of all outputs can be analyzed in a visual interface (see Section 7.2). This reduces communication overhead as both developers can set up, run and analyze experiments independently. In the old setup outputs were analyzed one by one, transferred to multiple tools and required a lot of communication overhead. cellPACKExplorer shows several statistical metrics through barcharts allowing both developers to inspect them. The interface of cellPACKexplorer is adjustable. Users can add and change the metrics as the cellPACK code changes. Fig. 3 shows the workflow. While all the tasks had to be done manually before, with cellPACKExplorer we support T1-Setup, T2-Validate, T3-Identify and T4-Default visually. We make cellPACKExplorer accessible through a common web browser which makes the data easily accessible for both developers to discuss findings (T5-Share). In the new cellPACKExplorer interface, the cellPACK developers first configure an experiment (T1-Setup) visually and start the computation of the output ensemble followed by the derived statistical metrics. The computation of all this data does not require any user interaction, therefore the developers are free to work on other tasks. When all the data is available, the cellPACK developer can explore the generated ensemble (T2-Validate, T3-Identify, T4-Default) using cellPACKExplorer's analysis interface. They can discuss and share the results (T5-Share) or modify the cellPACK code (T6-Improve). This is the only task not integrated in cellPACKExplorer and still done manually as code changes require a programmers expertise.

6. Design iterations

During our collaboration with the cellPACK developers they were constantly updating and changing the code. Therefore we had to account for these changes and provide a highly flexible interface for them that adapts to these changes. In this section we will talk about changes that we applied to the interface throughout the design process.

In the first prototype we had both the setup of an experiment and the analysis in the same window. After a revision of the first prototypes in collaboration with the cellPACK developers we decided to separate the five tasks into two sequential interfaces (each making use of the full screen), one for the setup of an experiment (T1-Setup) and one for the analysis of the experiment results (Task 2 through 5). This decision is based on the observation that the experiment setup was done carefully by the cellPACK developers and once an experiment was set up they switched to different duties until the computation of the ensemble of outputs was complete. During the analysis of outputs it was not necessary to see

the input configuration. Most time is spent on the analysis of the experiment, rarely reversing back to change the setup.

Another major change of the current design compared to the previous prototype is how we grouped the outputs. We first used an algorithmic clustering approach to group similar cellPACK outputs together. However, we observed that the developers, especially the scientific illustrator, preferred to compare outputs and draw conclusions themselves. Although an automatic clustering would require less work, it also limited the analysis to only the aspects considered by the algorithm. The users wanted as much flexibility as possible to analyze the data from various different aspects. Therefore, we decided to implement a manual filtering approach. This helps the users of our tool to build clusters (or groups) based on different aspects of the data.

In the first design of the interface we used a single barchart that showed the number of ingredients in different areas of the packing volume. For example, one bar represented how many ingredients were packed in the upper left subarea. Users could filter out results that had more than a specific number of ingredients packed in any subarea. Although a user could clean out runs that produced a very uneven distribution, they could not filter and analyze the outputs on different aspects. With the new spreadsheet layout, different metrics for analysis can be considered and easily added in the future if the requirements changing.

We decided to use histograms to represent frequency of the values sampled for the parameters and derived metrics. All input parameters are either numeric or categorical and can be represented through histograms. The same applies to the derived metrics shown as histograms. Although there are many other visual encoding to show this data type we found histograms to be best suited. They avoid the addition of further dimensions and reduce data to ink ratio. Additionally they work well for representing the input parameter frequencies as well as some of the output metrics reducing learning requirements for the user. Further we argue that histograms are a common and simple visual encoding for this type of data, commonly used, well known and easy to interpret by many potential users as well as the two developers with different backgrounds. Other chart types we considered in early paper prototypes. In discussion with the developers we decided for histograms already for the first high-fidelity prototype.

7. cellPACKexplorer

cellPACKExplorer is built as a client-server design with a web front-end. We chose this design as it simplifies installation and helped the developers of cellPACK to share (T5-Share) their results and discuss findings. Plots are implemented using D3 [30] and crossfilter [31] which supports interactive filtering of large datasets. We used the approach of Talbot et al. [32] for axis labeling.

We separated the six tasks into two sequential interfaces (each making use of the full screen), one for the setup of an experiment (T1-Setup) and one for the analysis of the experiment results (tasks 2 through 6). This decision is based on the observation that the experiment setup is done carefully and once an experiment is set up the developers switch to different duties until they wait for the computation of the ensemble of outputs to be complete. Later, they spend most of their time on the analysis of the experiment, rarely reversing back to change the setup. For a better understanding of the interactive nature of our tool, we provide a video demonstrating cellPACKExplorer in the supplemental material.

7.1. Input screen

The input screen (Fig. 4) supports task T1-Setup of the developers. The cellPACK developers use this screen to select a cellPACK recipe file and specify parameters and ranges they want to explore in the experiment.

To address the complexity of an experiment setup and make it accessible for both developers (the traditional workflow was one developer providing custom python scripts on request of the other developer) we organized the setup of the experiment into five steps, also reflected in the interface shown in Fig. 4. Since individual parameters influence different parts of the cellPACK algorithm and another set of input parameter is specific for different ingredient types they require different ways of specification. In addition the second set, working on a per ingredient basis has to be set carefully to avoid the computation of too many outputs.

(1) Set recipe: At first the developer selects a cellPACK recipe they want to analyze. This recipe is a *.json* file that specifies the packing volume, which ingredient types to pack, and default values for cellPACK input parameters. The recipe also declares the biological cell structure.

(2) Number of runs and output location: The second step configures the experiment and determines how many cells should be computed (N and R, see Section 5.2).

(3) Global packing parameters: In step 3, one sets general parameters of the simulation (see Section 3). The developer is only required to set the parameters they want to vary. Other parameters remain at default values as specified by the selected recipe file. For each sampled parameter the cellPACK developer can also choose a sampling method. They can either select (deterministic) grid sampling or stochastically uniform sampling. All parameters that are grid sampled are determined by a multidimensional Cartesian lattice. This is ok for a small number of parameters but can lead to a combinatorial explosion quite quickly. Hence, it should be used with caution and only for smaller test cases.

(4) Ingredient-specific parameters: Ingredient parameters require a more complex setup as each ingredient type can have its own set of parameter values (compare ingredient parameters Section 3). When using cellPACK itself, setting parameters is a very tedious process since each parameter value must be set by hand in a configuration file. To overcome this, we provide a searchable list of all available parameters. When the cellPACK developer starts typing, only parameters whose name matches are shown in the list. As we focused on the developers of cellPACK, they know the names of parameters they want to sample and therefore can save a lot of time using this feature. Because we observed the cellPACK developers changing the same parameter across multiple ingredients, we added the ability to modify parameter values for groups of ingredients. The tree representation of ingredients in the interface mimics the structure of ingredients in a cellPACK recipe. They can select ingredients in the tree and then select parameters and ranges to be sampled for the selected ingredients.

(5) Execute or export experiment: Once the cellPACK developer has finished setting up an experiment, they can either run it directly on the server or download the configuration. The download option is helpful if they want to run the experiment on another machine (i.e. with more computational power), send the setup to someone else (i.e. developers working together), or do the computation later. After the setup of the experiment, the computation of the outputs is done offline and does not require any interaction so the cellPACK developers can concentrate on other work while waiting for the results to be computed.

7.2. Analysis screen

Once all the outputs are computed and derived statistical metrics are ready, the developers use the analysis screen (Fig. 5) to inspect the generated ensemble. The layout of our interface is comparable to a visual spreadsheet structured as rows and columns. There are many reasons for this decision. First, one of the

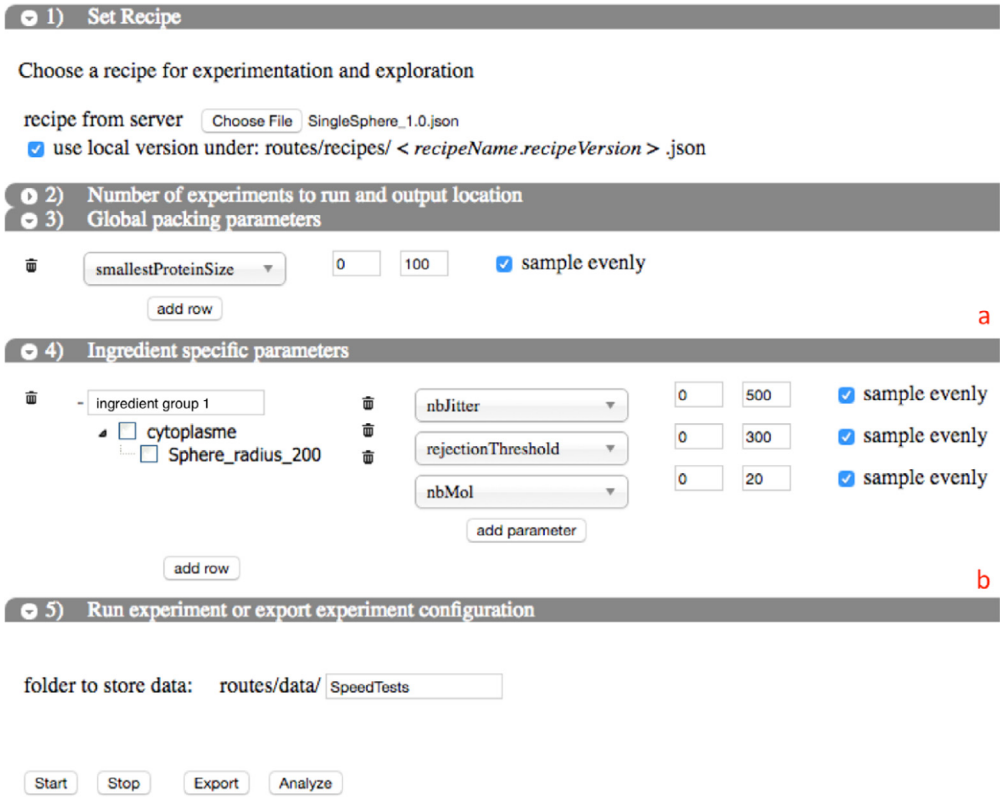

Fig. 4. The input interface. Each vertical panel is one step in the setup of an experiment. From top to bottom: (1) recipe specification (2) cellPACKExplorer settings (collapsed) (3) cellPACK general packing parameters (marked with a red a) influence general settings for the algorithm (4) cellPACK ingredient parameters (marked with a red b) are set per ingredient type 5) start/export configuration. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

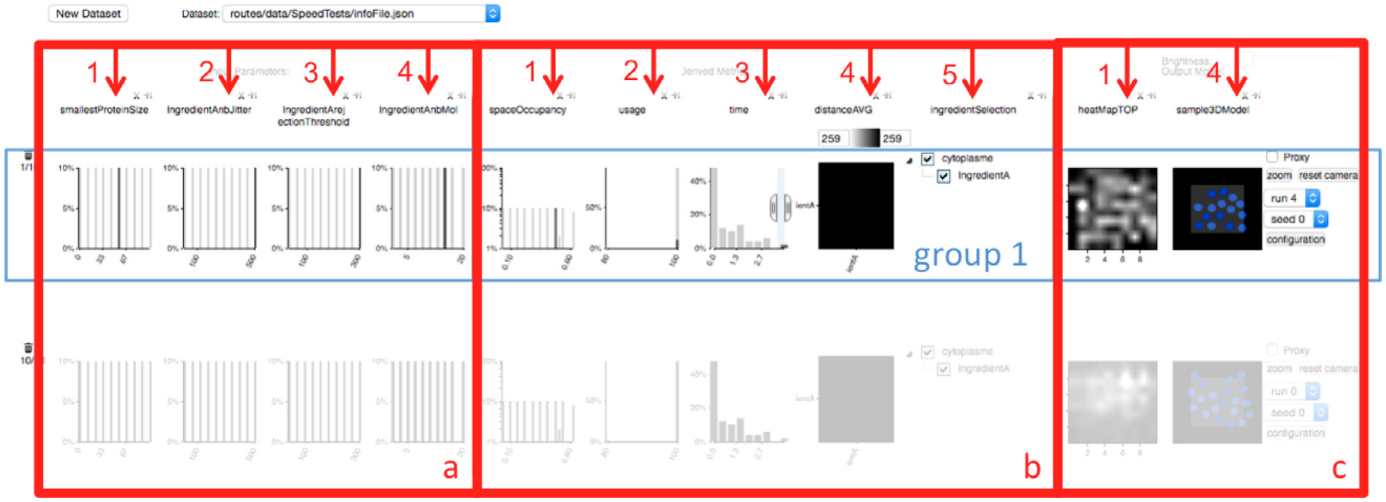

Fig. 5. The analysis interface. Three main columns for (a) input parameters (b) derived statistical metrics (c) spatial output presentation.

developers already worked with spreadsheets in the context of the analysis. Second, we wanted a flexible interface that adapts to algorithmic and analysis changes. With the spreadsheet layout, different analysis aspects (presented by columns) can be swapped in and out or added.

The plots for each row can be divided into three logical groups: distribution of input parameters (Fig. 5a), distribution of derived metrics (Fig. 5b), and renderings of the outputs (Fig. 5c). The total number of runs summarized in that row is shown on the left side of the interface. Each row in the interface represents a filtered subset of all runs of the selected experiment. This allows the cellPACK developer to compare (T2-Validate) different subsets of the output ensemble. A new row initially shows all runs of the experiment (i.e. no filters are applied and the whole output set is visible). The cellPACK developer can interactively adapt which runs are part of a horizontal group by creating a filter on any or several of inputs or derived measures represented by columns in the interface. The filters are combined with an AND operation such that groups are formed where each single output of a run has to fulfill all the filter constraints to be part of a horizontal row. Each filter only influences its own horizontal row in the interface. Filters in cellPACKExplorer can be adjusted in the bar charts directly. A click

on any location within the chart followed by a dragging operation limits the according parameter or output metric to the specified range. For more details please watch the video.

The column layout provides the flexibility to add other features (metrics) in the future or delete existing ones without changing the interface. This is something we found to be important for the model building workflow of the developers as the underlying algorithm (cellPACK in our case) changes regularly and requires different analysis metrics which can be easily added and removed without requiring the interface to change. We explain each of the different metrics the developers currently use in turn.

7.2.1. Input parameters

The leftmost columns (Fig. 5a) show one histogram for each sampled input parameter. It supports the developer in understanding an input parameter's influence on the generated outputs (T3-Identify) and identify good default values for parameters (T4-Default). Only sampled input parameters which have been selected by the developers in the input interface (Fig. 4) are shown. All others remain at default values for the whole ensemble. The horizontal axes of each graph represent the sampled parameter value (numerical or categorical). The vertical axes indicates the frequency, i.e. how often a specific value has been used to generate outputs in that row. In case of filtering, the full histogram for all runs remains transparent in the background to provide the context of the full ensemble dataset. The updated opaque part of the histogram shows the distribution of the currently selected subset of a row fulfilling all the filters.

7.2.2. Derived metrics

The center set of columns (Fig. 5b) show histograms of various derived outputs. We created these columns to help the cellPACK developer to quickly identify subsets of interest without scrolling through output images one by one. E.g. we watched the developers looking for outputs where some ingredients failed to pack or outputs that took a very long computation time. All the metrics have been developed in collaboration with the cellPACK developers to focus on their analysis goals. To provide a consistent structure of all columns the y axis on all histograms shows the frequency of each value on the x axis in the whole or currently selected dataset. This is consistent with the input parameter column.

As some graphs are computed on an ingredient basis, the cellPACK developer can focus on a subset of ingredients by individually selecting or deselecting them in the tree (Fig. 5b, column 5) the same layout for ingredient selection is also used on the input screen to support the user in the understanding of the interface. Selecting and deselecting ingredients enables the analysis of single ingredients, for example checking how much of the available space it covers within the packing volume.

Finding good derived metrics is difficult and hence, they are constantly revised. During the development of cellPACKExplorer and as our understanding of the packing algorithm improved, we suggested a number of new metrics. The developers of cellPACK also requested a number of different metrics and explored different parameters during the design process constantly updating used metrics. As we progressed through the development of cellPACK-Explorer, we refined the list of output metrics. The column layout in the interface gave us the possibility to easily swap in and out metrics and add new ones. In the current version of cellPACKExplorer we show the following metrics: *spaceOccupancy*, *usage*, and *distanceAVG*. These metrics show derived geometrical properties of the cellPACK outputs:

spaceOccupancy: The developers are interested in the concentration of different ingredients as this is crucial to assure biologically valid outputs. The *SpaceOccupancy* (Fig. 5b, column 1) histogram shows the distribution of the percentage (horizontal axis) of the total packing volume covered by an ingredient type. For example, if ingredient A takes up 50% of the whole cube in which we are packing then the *SpaceOccupancy* value is 50%. Within each run the occupancy for each ingredient is averaged over all cells computed with different seeds. This measure gives an idea of the concentration of ingredients compared to the total volume, the denser the volume is packed, the higher this measure will be.

U*sage:* An important aspect to assure that cellPACK produces correct outputs is to have the full usage of ingredients, i.e. the number of copies of an ingredient type the cellPACK developer wants to pack should be equal to the number of copies that is actually packed in the generated cell. In some cases these two numbers might differ. E.g., the ingredient might not find a place as there is not enough space left or its parameters do not allow certain positions. This would result in a usage below 100%. The developers want to identify these cells, and investigate them further. Ideally this histogram would only show one peak at a usage of 100% (as is the case in the example of Fig. 5b, column 2). This would represent an ensemble where cellPACK could place all ingredients specified in the recipe into the model. If this is not the case it means that cellPACK had to skip some ingredients as they could not be placed inside the cell due to their input parameters.

distanceAVG: Within a cell proteins rarely act alone. Molecular processes are carried out by the interactions occurring between specific proteins. Moreover, the interior of cells is a crowded environment. This crowding effect can make molecules in cells behave in radically different ways than in test-tube assays. It is thus important to have a metric that can represent the crowding property of a given cell generated by cellPACK. To analyze this in cellPACKExplorer, we developed the *distanceAVG* (Fig. 5b, column 4) measure. It measures the distribution of pairwise distances between each ingredient instance to every other ingredient instance for each cellPACK output (also known as the radial distribution function in physics) averaged over the subset of a run. As each ingredient type is repeated multiple times in a typical packing we compute a distance matrix using the average distance between ingredients. It is displayed as a heatmap (Fig. 5b, column 4), mapping low distance to white and high distances to black. In this figure, we have only one ingredient hence there is only a single distance in the matrix.

The *run-time* (Fig. 5b, column 3) of outputs generated in an experiment provides crucial information to enable the developers to improve cellPACK's efficiency (T6-Improve). It shows which parameters have the greatest impact on the computation time. Within an experiment it often happens that all outputs require approximately the same time except for one that takes much longer. Being able to filter on these outputs, the interface shows what input configuration caused the long computation time. This metric can also be used to find a proper trade off between a high density of packed ingredients and a reasonable computation time. The developer can quickly assess the computation time and compare it to the achieved accuracy of the outputs. If the developer is interested in accuracy represented by how dense ingredients are packed in a cell they can use the *spaceOccupancy* metric, showing how much of the space is occupied by an ingredient. If they want to check overlaps and intersections between ingredients they can make use of the *distanceAVG* graph and compare pairwise distances of ingredient types. Setting filters on these charts, cellPACKExplorer can be used to quickly focus on a subset of outputs that satisfy special criteria. Subsequently, these outputs can be inspected in more detail on the right side of the row (Section 7.2.3).

7.2.3. Packing columns

The right part of the interface (Fig. 5c) gives the cellPACK developers access to the direct output of cellPACK, which is a stochastic 3D volume. The first image of (Fig. 5c) shows the

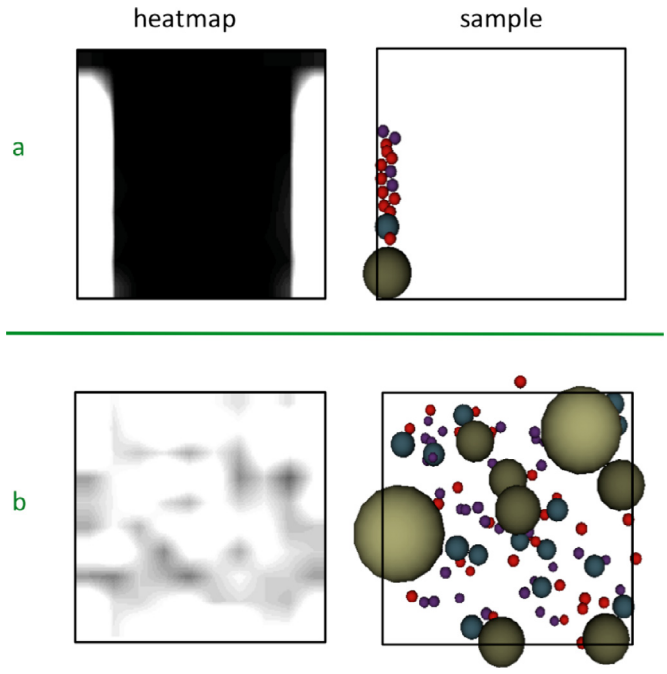

Fig. 6. Left column: heatmaps of all ensemble members projected along the *y*-axis. Space occupied by ingredients is colored in transparent white while empty space is colored black for all ingredients. Areas that are covered by ingredients (spheres in the example shown) in more ensemble members appear brighter, regions that are often empty appear gray, regions that are never occupied by an ingredient are black (e.g. center part in the top row). Right column: one cellPACK ouput part of the ensemble set that resulted in the heatmap shown on the left. Note the bias in the upper row (a) towards the left edge while the rest of the volume is empty (black). In the heatmap ingredients reaching out of the packing volume (black rectangle) periodically come back in on the opposite site (periodic boundary condition) which explains the white stripe on the right side in the heatmap in the upper row. The lower row (b) shows a random uniform distribution with almost all areas having the same brightness.

density of ingredients within the probabilistic volume for different orthographic projections (top, right, front). In Fig. 5c, columns 2 and 3 have been closed by the developer as a 2D packing is analyzed. To compute these heatmaps the packing volume is discretized into a user-defined number of subvolumes. For each of these subvolumes we compute the volume covered by an ingredient type divided by the total volume of the voxel. The resulting values are mapped to varying gray levels (black means an empty subvolume containing no ingredients). Fig. 6 shows a comparison of a biased cellPACK output with a lot of empty (black) space (upper row) and an output with uniform distribution (whole image is grey or white meaning that there are ingredients in all voxels) (lower row). We can see that there is a bias towards the boundaries of the packing volume in the top row because the border is brighter. Heatmaps have been used by the cellPACK developers before to analyze their cells. We chose to incorporate them in cellPACKExplorer to provide access to their initially used analysis methods. In our experience these (direct) visual depictions of the cells are easier to understand and were preferred by the less technically trained of our two users (cellPACK developers).

The last column (Fig. 5c, column 4) shows an interactive 3D view of one cellPACK output of a run in that row and gives the developer the option to inspect details for specific cells. The developer can interactively change which cell (run and seed) to present by selecting a different option in the dropdown menus. Outputs not part of the horizontal group are disabled. To interact with the 3D cell, the developer can use the mouse wheel to zoom and mouse dragging to rotate and translate the cell. We used billboard imposters for spheres to speed up the rendering. To further improve performance, in case of highly crowded cells (e.g. HIV), the developer can turn on the "proxy" option: each ingredient will be replaced by a single sphere encapsulating the original ingredient's shape. This representation shows the spherical proxy used by cellPACK to resolve the intersections between different ingredients while packing them to form the final cell. To better analyze one specific cell, the whole view can be enlarged by a click on the "zoom" button. After testing the tool, the developers of cellPACK were interested in the exact parameter configuration of the cell presented in the viewer. Hence, selecting "configuration" opens a tooltip with detailed information about the sampled parameters that yield that specific output.

The interface can easily be adapted by closing or opening columns of information (e.g. when exploring a 2D recipe, only one of the projected heatmaps is needed, features can be shown/hidden depending on the analyzed parameter). After the exploration of an experiment, the underlying model (cellPACK) might be improved (T6-Improve) and a new experiment can be started.

8. Evaluation

We used two different setups for the evaluation of the tool. In addition to regular feedback from the cellPACK developers we asked them to use the tool on their own and provide qualitative feedback for us guided by a questionnaire (Sections 8.1, 8.2, 8.3). Second we asked some users without a biological background or previous knowledge about cellPACK to fulfill some tasks and also pr ovide some informal feedback (Section 8.4).

8.1. Biological case study

In this case study, the cellPACK developers are interested in the smallestProteinSize parameter specifying the density of the packing grid. Additionally they investigate the influence of the rejectionThreshold of an ingredient deciding how often an instance of an ingredient tries to find a place until it gives up and does not pack itself. Further they also inspected the molarity of different ingredients specifying the number of copies of each ingredient type. Fig. 7 shows this setup. They start the experiment and work on some other tasks while the server generates the cellPACK results.

When all the outputs are ready our users evaluated which parameter setups caused a long packing time as they want to avoid these and get results faster. They set a filter on the time graph and focus on the runs that required a long packing time. With the filter set, they found out, which values for input parameters relate to long computation times. In the test case described a high value for the rejectionThreshold and a low smallestProteinSize in combination with a relatively high molarity causes a long packing time. The upper row of Fig. 8 shows this scenario. This makes sense as a high threshold implies that the ingredient tries to pack itself very often and a high molarity creates many instances that try to pack themselves. Since the packing grid is small there are many points to be proposed to the ingredient and different locations are close to each other.

Next, they create a new row and look at the usage graph. There was one run where not all the ingredients have been packed. Since this behavior is erroneous, they want to further see which input setup caused this and filter out all the valid runs (see Fig. 8 lower row). This let them discover that a small value for the smallestProteinSize, low rejectionThreshold and high molarity caused the faulty result. If the rejectionThreshold is low, the ingredient gives up in trying to find a location. Additionally with a high molarity a lot of copies try to find a location and there will not be enough space for all the instances to find a place.

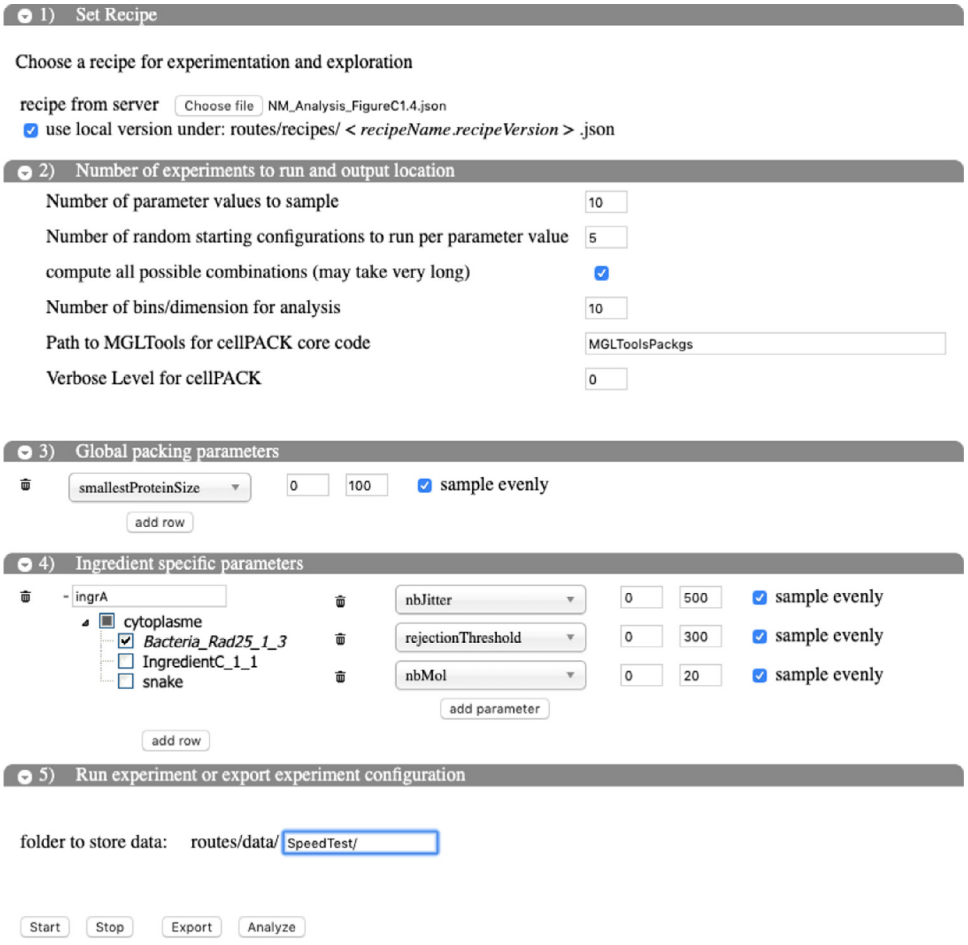

Fig. 7. Example experiment setup with cellPACKExplorer. The user is interested in the global parameter smallestProteinSize, modelling the density of the packing grid. Further they want to evaluate *nbJitter*, *rejectionThreshold*, and *nbMol* for an ingredient called Bacteria_Rad25_1_3.

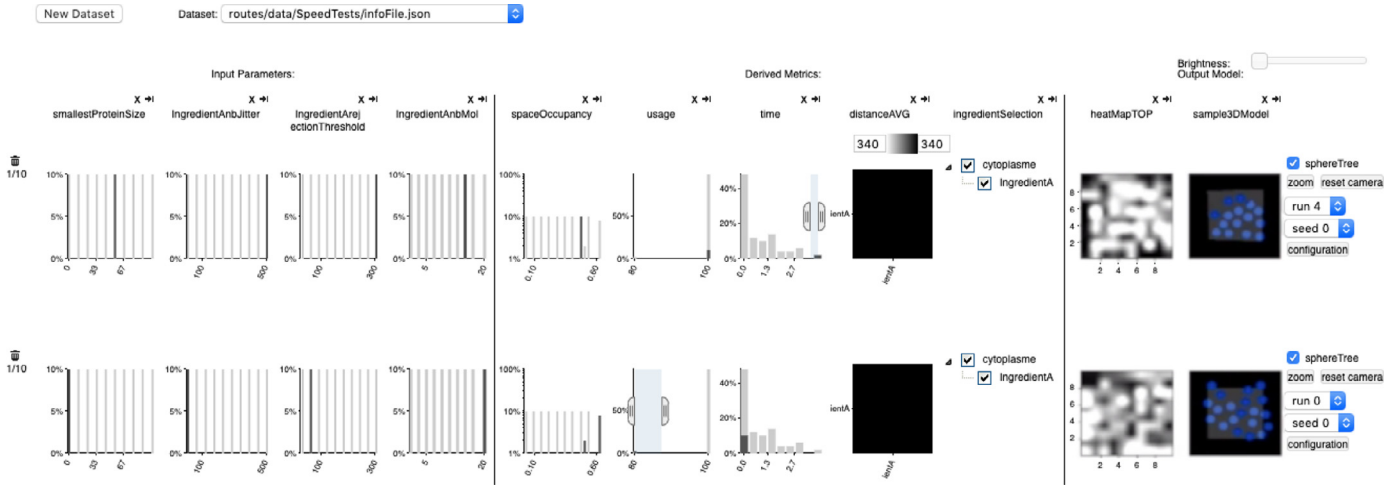

Fig. 8. The output results from the setup shown in Fig. 7. In the upper row the user wants to see which parameter setup required a long time to pack. The filter sets this horizontal group to focus on runs with the longest computation time. In the second row the user selected all the runs where not all ingredient copies have been placed (the filter is set on low usage).

cellPACKexplorer allowed the developers to quickly understand the influence of a number of parameters in a short, interactive session, which required many painful and tedious iterations before.

8.2. Debugging case study

The interface can be used to test new code for stability and functionality. Specifically, the developers wanted to test the behavior of a new ingredient parameter, *weight*. This parameter influences an ingredient's decision to pack close to a binding partner (another ingredient that has already been packed). Without looking at the code and using only cellPACKexplore, the other developer was able to quickly validate this new feature. They added the new *weight* parameter to a known recipe, and sampled it in a range from 0–100% of its range to confirm that at 0% the results computed by cellPACK were the same as without the parameter

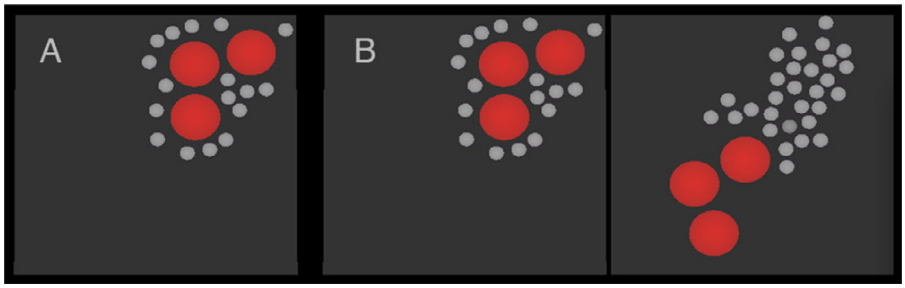

Fig. 9. A. The original recipe prior to adding the *weight* parameter code shows how IngredientB (small gray spheres) always packed close to IngredientA (large red spheres) B. The new version of the original recipe shown in A has the *weight* parameter added and cellPACKExplorer has been used to sample the *weight* probability from 0% on the left (always bind to IngredientA) to 100% (always bind to IngredientB) on the right. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(Fig. 9A and Fig. 9B-left). As the probability was increased up to 100% cellPACK produced results that matched the developer's hypothesis of how the weighting would decreasingly influence IngredientB (see Fig. 9B-right).

The developers were able to use cellPACKExplorer for large-scale debugging tests and to isolate more subtle issues with newly added code. Instead of debugging heuristically by adjusting parameters and viewing one result at a time, cellPACKExplorer enabled them to setup and analyze thousands of models at a time, which could reveal statistical subtleties more readily. In a typical scenario the developer would first run small experiments and sample only two sets of parameters at a time with typically just two seeds (T1-Setup) to ensure the interface and the program are running correctly (T2-Validate, T3-Identify, T4-Default). If the code failed, then they would debug it and adapt the cellPACK algorithm (T5-Share). Secondly, they would rerun with all the parameters they want to test with a small number of seeds (T1-Setup) to confirm that the pairwise tested parameters all worked together (T2-Validate, T3-Identify, T4-Default). Finally, they would greatly increase the number of seeds for a deep analysis to explore the behavior of the cellPACK algorithm (T2-Validate, T3-Identify, T4-Default). If the cellPACK output was incorrect they would adapt the cellPACK algorithm again (T5-Share) and start a second round.

Using this exhaustive approach and a wide sampling range, the developers of cellPACK observed some problems that could not have been noticed with the smaller experiments they were doing before. Manually scrutinizing hundreds of 3D models using their old approaches was time consuming and prone to error. cellPACKExplorer's filtering options helped them to quickly discover issues such as repetitive/identical models or incorrect distributions that resulted, for example, from errors in the core code or input parameter configurations that caused ingredient constraints (e.g. using a gradient or specifying that two ingredient types should pack close to each other) to be ignored or incorrectly applied.

Fig. 10 demonstrates one example of a difficult bug to spot manually that was relatively easy to find with cellPACKexplore. The developers wanted to implement a parameter that created hotspots in some areas of the packing volume for an ingredient type. At first (row a) the parameter did not create the anticipated results, as the distribution of the ingredient type remained uniform. After some updates to the code, outputs showed the anticipated distribution (row b).

8.3. Qualitative questionnaire results

After testing cellPACKExplorer, we interviewed the developers using a semi-structured interview. The interview was structured around 4 topics: the usability of the interface, new types of analysis enabled by the interface, what specific observations were made, and any suggestions for improvements. In summary, they described cellPACKExplorer as being extremely helpful. It allowed simpler access to the modelling tool making modelling more efficient. It also helped the developers debugging the cellPACK software, optimize recipes and generate hypothesis in biological research. In the following we describe their analysis approach and discuss feedback in more detail.

A major benefit of the new system is the speed-up in their workflow. One developer described a time savings of up to 100 times because they were able to setup experiments much faster than with custom Python scripts. The fact that they could run cellPACK and analyze cellPACK results on a web server eliminated their constant hassle of maintaining and running cellPACK across the diverse collection of computers at home and at work (multiple operating systems, incompatible Python versions, etc). In addition, many of the manual analysis tasks done via spreadsheets and custom scripts is now integrated into cellPACKExplorer. The developers also were able to spot and fix an error in their code, as described in Section 8.2. This would have been very difficult without our tool. Through the direct visualization of inputs and outputs, they are able to optimize parameter values for output generation and generate new hypothesis about biological behavior and interaction of molecules and how these could be mapped to parameters.

8.4. General usability evaluation

In order to measure the transferability of cellPACKExplorer to non-developers, we also conducted a usability evaluation with users who had not previously used cellPACKExplorer. We started with a brief introduction to cellPACKExplorer and a walk through of the interface. Then, each participant was asked to analyze the packing parameters for two recipes previously generated by the cellPACK developers through the online accessible tool. Both recipes described a packing problem of spheres on a 2D plane. Spheres occupy the intersecting circle on the plane. The developers often use this simplified setup to quickly test code changes. It is helpful for quick intersection tests and increased rendering performance of outputs. We chose these data sets for our study as they are simple and fairly easy to analyze.

In order to give context to our usability evaluation, we gave each participant a number of tasks, framed as a set of questions. The participants were tasked with answering:

1. how many ingredients had been packed in each of the data sets and their shape,
2. the input configurations where most or all ingredients found space in the packing volume, and
3. to specify a row which contains the best runs and how they defined the optimal outputs.

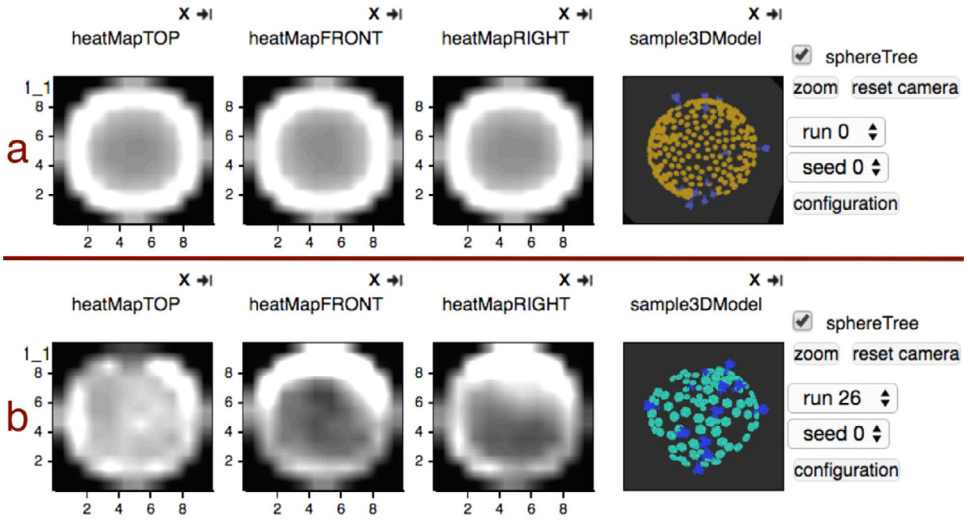

Fig. 10. An example of how a subtle bug was found in the cellPACK code that was written to pack objects close together on the surface of a sphere 100% of the time. The heatmaps quickly revealed a uniform random distribution on the spherical surface (a). The core code was adapted and a second experiment revealed the anticipated hotspots at some locations on the surface (b).

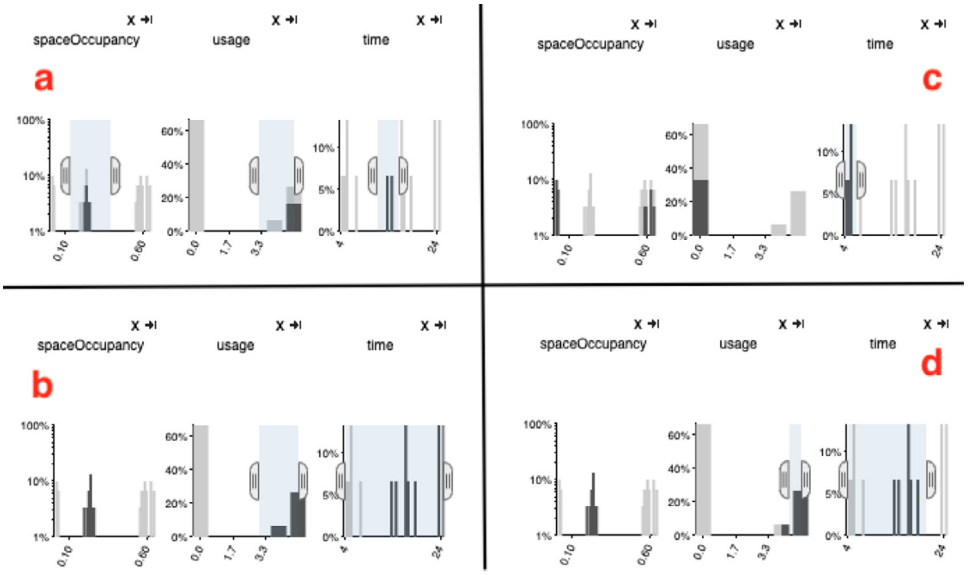

Fig. 11. The results of user's selection of the best runs on the first dataset. Most of the study participants with biological background focused on a high usage. This corresponds to outputs where all ingredients have been packed. In addition they set time to be low. Only the visualization researcher (c) set the filter on the lowest time only.

Our four participants consisted of a UXdesign expert, a bioinformatics specialist, a visualization researcher, and a specialist in molecular simulation and animation. All but two participants correctly answered all questions. The others correctly answered after explaining one of the graphs again. Fig. 11 and Fig. 12 show the final decisions of our users (a): UXdesign expert, (b): bioinformatics specialist, (c): visualization researcher, (d): specialist in molecular simulation and animation) for the two datasets.

Some of our participants had troubles interpreting the derived metrics charts. After a brief clarification they were also able to answer questions related to this chart. In addition, some users preferred that the *x*- and *y*-axes in the graph showing the usage would be the other way around. The biological and UXDesigner participants preferred the output renderings on the right side of the interface over the numerical metrics. Two of our participants were curious how to set up their own experiment and use cellPACKExplorer. We showed them how they can create their own datsets using our input interface.

9. Discussion

As we consider the close collaboration with the cellPACK developers including weekly meetings as one of our contributions we want to describe insights we gained during that process here.

We found that the users of cellPACKExplorer could be differentiated into two types of approaches. One was working more visually and the other is working with numerical performance measures. Users trained in visual confirmation (e.g. designers and biologists) prefer the former and more technical users prefer the latter. We also saw this during our general usability evaluation with users that were not involved in the development of cellPACKExplorer. To support this requirement we designed a flexible interface that supports both visual and numeric analysis.

Further, we observed that our users need, at times, close control over their analysis process. Specifically, while automatic clustering provides smaller groups of the ensemble dataset it also limits the analysis a user can do with a specific dataset. The clustering works

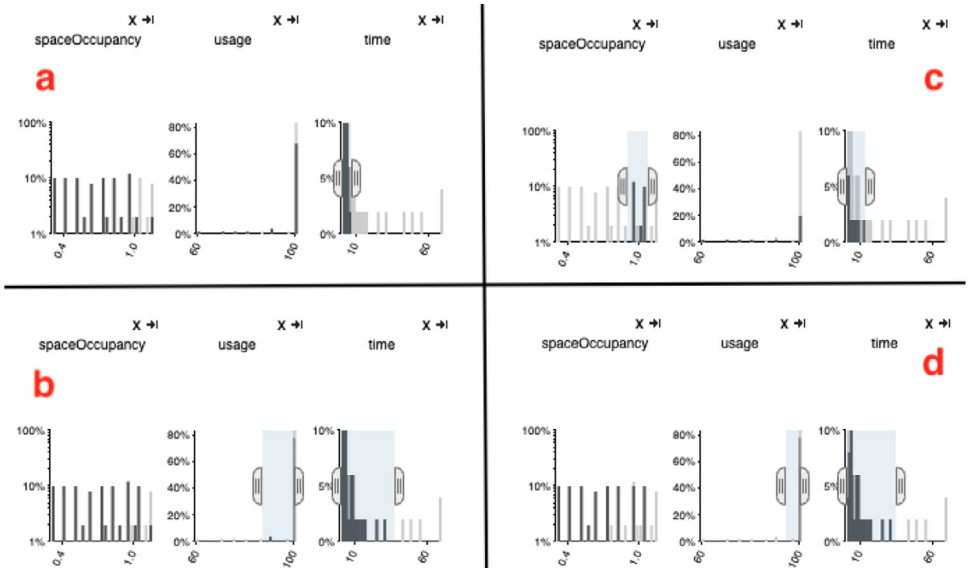

Fig. 12. The results of user's selection of the best on dataset 2. This time all study participants set a filter constraint on low time. Some of them also set a high usage. The visualization researcher (c) also filtered on a relatively high space occupancy.

on a pre-defined set of features and is usually applied in the offline stage of the workflow (i.e. after data generation but before the user starts his analysis). With our approach of manual filtering on inputs and outputs we keep the user in the loop and provide freedom to investigate the same dataset from different aspects.

To provide recommendation for a faster design process for future design studies we emphasize the use of a web based application. This eases installation for the users and allows us to deploy updates easily. This makes it easy to access the tool for a variety of users with different levels of technical skill.

10. Future work

While cellPACKExplorer is a great support in the analysis of simple packings the scalability to realistic biological cells remains future work. A cellPACK recipe can consist of thousands of ingredients. For example, *E. Coli* has about 1 million proteins made up of about 4000 unique protein types, Mycoplasma Mycoides is formed of about 50,000 proteins made up of about 800 unique protein types. Each of these unique proteins is a separate ingredient. This will require a change in our interface to account for such a large number of ingredients.

The developers of cellPACK have requested the ability to select other derived metrics in the center column (Fig. 5b) and the possibly to upload their own derived measures. We see a lot of potential future work in developing new metrics that reveal other features of the ensemble set and can be interpreted visually as easy as the outputs. We further realized that as the cellPACK outputs inspected changed some of our proposed metrics required an adaption. The *spaceOccupancy* and the heatmaps will not support analysis in case of a recipe that packs ingredients on the surface of a sphere.

The flexibility of cellPACKexplorer in terms of what metrics are shown opens it up to other application areas. For example, it could help to explain to meteorologists which parameters of weather simulations influence temperature, air pressure, or precipitation. Another possible domain is bioengineering where scientists generate simulations of human organs and inspect the influence diseases can have on potential fields of the human heart [33]. As the visual columns in the right most side of the interface can be closed if a simulation does not provide visual output, our tool also is applicable to simulations in areas such as software engineering or mathematics where simulations do not always result in visual outputs.

Finally we want to experiment with different sampling strategies. The cellPACK developers currently prefer the 'full combinatorial' method, that creates all possible combinations for the sampled input parameters (after discretization of the continuous parameters). This approach does not scale well if the sampling range or number of parameters sampled increases. We also anticipate to provide some guidance in choosing a proper sampling strategy and number of samples required to get a statistically meaningful output.

11. Conclusion

To the best of our knowledge, this work, for the first time in the visualization literature, is presenting a task analysis of a model building process in the biological domain. We specifically focus on the development of cellPACK for generating complex virtual cells that are difficult to parameterize and validate. We compared and contrasted the challenges and tasks performed by the cellPACK developers, related to model building and model usage. Specifically, we identified the need to add, remove, and find proper defaults for parameters guiding the modeling process as a novel task to be performed. Further, the ability to incorporate and validate new derived measures proved crucial and difficult for the success of the modeling pipeline. Based on this breakdown we created cellPACKExplorer supporting the developers in analyzing input parameter effects on outputs as well as the distribution of objects in a volume for a probabilistic volume ensemble dataset.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cagx.2019.100010.

References

[1] Miksch F, Pichler P, Espinosa KJ, Popper N. Agent-based methods for simulation of epidemics with a low number of infected persons. In: Hutchison D, Kanade T, Steffen B, Terzopoulos D, Tygar D, Weikum G, et al., editors. Proceedings of the second information and communication technology - EurAsia conference (ICT-EurAsia). Information and Communication Technology, LNCS-8407. Bali, Indonesia: Springer; 2014. p. 21–8. doi:10.1007/978-3-642-55032-4_2. Part 1: Information & Communication Technology-EurAsia Conference 2014, ICT-EurAsia 2014 https://hal.inria.fr/hal-01397140.

[2] Bishop CM. Pattern recognition and machine learning (Information science and statistics). Secaucus, NJ, USA: Springer-Verlag New York, Inc.; 2006. ISBN0387310738.

[3] Committee on Mathematical Foundations of VerificationValidation, and Uncertainty Quantification; Board on Mathematical Sciences and Their Applications, Division on Engineering and Physical Sciences, National Research Council. Assessing the reliability of complex models: mathematical and statistical foundations of verification, validation, and uncertainty quantification. The National Academies Press; 2012. doi:10.17226/13395. ISBN 9780309256346.

[4] Johnson GT, Autin L, Al-Alusi M, Goodsell DS, Sanner MF, Olson AJ. Cellpack: a virtual mesoscope to model and visualize structural systems biology. Nat Meth 2015;12(1):85–91. doi:10.1038/nmeth.3204.

[5] Goodsell DS. The machinery of life. Copernicus; 2009. doi:10.1007/978-0-387-84925-6.

[6] Sedlmair M, Heinzl C, Bruckner S, Piringer H, Möller T. Visual parameter space analysis: a conceptual framework. IEEE Trans Vis Comput Graph 2014;20(12):2161–70. doi:10.1109/tvcg.2014.2346321.

[7] Kehrer J, Hauser H. Visualization and visual analysis of multifaceted scientific data: a survey. IEEE Trans Vis Comput Graph 2013;19(3):495–513. doi:10.1109/TVCG.2012.110.

[8] von Landesberger T, Fellner DW, Ruddle RA. Visualization system requirements for data processing pipeline design and optimization. IEEE Trans Vis Comput Graph 2017;23(8):2028–41. doi:10.1109/TVCG.2016.2603178.

[9] Muggeo VM. Segmented: an r package to fit regression models with broken–line relationships. R News 2008;8(1):20–5.

[10] Alexander WP, Grimshaw SD. Treed regression. J Comput Graph Stat 1996;5(2):156–75. doi:10.2307/1390778. http://www.jstor.org/stable/1390778.

[11] Guo Z, Ward MO, Rundensteiner EA, Ruiz C. Pointwise local pattern exploration for sensitivity analysis. In: Proceedings of the 2011 IEEE conference on visual analytics science and technology (VAST). IEEE. Providence, RI: IEEE; 2011. p. 131–40. doi:10.1109/VAST.2011.6102450.

[12] Mühlbacher T, Piringer H. A partition-based framework for building and validating regression models. IEEE Trans Vis Comput Graph 2013;19(12):1962–71. doi:10.1109/TVCG.2013.125.

[13] McGregor S, Buckingham H, Dietterich TG, Houtman R, Montgomery C, Metoyer R. Facilitating testing and debugging of Markov decision processes with interactive visualization. In: Proceedings of the IEEE symposium on visual languages and human-centric computing (VL/HCC). IEEE; 2015. p. 53–61. doi:10.1109/VLHCC.2015.7357198.

[14] Bihlmaier A, Worn H. CVVisual: interactive visual debugging of computer vision programs. In: Proceedings of the 20th conference on emerging technologies & factory automation (ETFA). IEEE; 2015. p. 1–6. doi:10.1109/etfa.2015.7301408.

[15] Whitaker RT, Mirzargar M, Kirby RM. Contour boxplots: a method for characterizing uncertainty in feature sets from simulation ensembles. IEEE Trans Vis Comput Graph 2013;19(12):2713–22. doi:10.1109/TVCG.2013.143.

[16] Mirzargar M, Whitaker RT, Kirby RM. Curve boxplot: generalization of boxplot for ensembles of curves. IEEE Trans Vis Comput Graph 2014;20(12):2654–63. doi:10.1109/tvcg.2014.2346455.

[17] Schmidt J, Gröller E, Bruckner S. VAICo: visual analysis for image comparison. IEEE Trans Vis Comput Graph 2013;19(12):2090–9. doi:10.1109/tvcg.2013.213.

[18] Pöthkow K, Weber B, Hege H-C. Probabilistic marching cubes. Comput Graph Forum 2011;30(3):931–40. doi:10.1111/j.1467-8659.2011.01942.x.

[19] Reh A, Gusenbauer C, Kastner J, Gröller E, Heinzl C. MObjects–A novel method for the visualization and interactive exploration of defects in industrial XCT data. IEEE Trans Vis Comput Graph 2013;19(12):2906–15. doi:10.1109/tvcg.2013.177.

[20] Ehlschlaeger CR, Shortridge AM, Goodchild MF. Visualizing spatial data uncertainty using animation. Comput Geosci 1997;23(4):387–95. doi:10.1016/S0098-3004(97)00005-8. Exploratory Cartograpic Visualisation

[21] Lundström C, Ljung P, Persson A, Ynnerman A. Uncertainty visualization in medical volume rendering using probabilistic animation. IEEE Trans Vis Comput Graph 2007;13(6):1648–55. doi:10.1109/TVCG.2007.70518.

[22] Tversky B, Morrison JB, Betrancourt M. Animation: can it facilitate? Int J Hum Comput Stud 2002;57(4):247–62. doi:10.1006/ijhc.2002.1017.

[23] Marks J, Andalman B, Beardsley PA, Freeman W, Gibson S, Hodgins J, et al. Design Galleries: a general approach to setting parameters for computer graphics and animation. In: Proceedings of SIGGRAPH 97. In: Annual Conference Series. ACM; 1997. p. 389–400. doi:10.1145/258734.258887.

[24] Bruckner S, Möller T. Result-driven exploration of simulation parameter spaces for visual effects design. IEEE Trans Vis Comput Graph 2010;16(6):1467–75. doi:10.1109/TVCG.2010.190.

[25] Pretorius AJ, Bray M-A P, Carpenter AE, Ruddle RA. Visualization of parameter space for image analysis. IEEE Trans Vis Comput Graph 2011;17(12):2402–11. doi:10.1109/TVCG.2011.253.

[26] Bergner S, Sedlmair M, Möller T, Abdolyousefi SN, Saad A. Paraglide: interactive parameter space partitioning for computer simulations. IEEE Trans Vis Comput Graph 2013;19(9):1499–512. doi:10.1109/tvcg.2013.61.

[27] Santner TJ, Williams B, Notz W. The design and analysis of computer experiments. Springer-Verlag; 2003. doi:10.1007/978-1-4757-3799-8.

[28] Sedlmair M, Meyer M, Munzner T. Design study methodology: reflections from the trenches and the stacks. IEEE Trans Vis Comput Graph 2012;18(12):2431–40. doi:10.1109/TVCG.2012.213.

[29] Munzner T. Visualization analysis and design. AK Peters visualization series. Boca Raton, FL: CRC Press; 2015.

[30] Bostock M, Ogievetsky V, Heer J. D3: Data-driven documents. IEEE Trans Vis Comput Graph 2011;17(6):2301–9. doi:10.1109/TVCG.2011.185.

[31] Square I. crossfilter. 2017. http://square.github.io/crossfilter/ [Last Accessed 18 July 2017].

[32] Talbot J, Lin S, Hanrahan P. An extension of Wilkinson's algorithm for positioning tick labels on axes. IEEE Trans Vis Comput Graph 2010;16(6):1036–43. doi:10.1109/TVCG.2010.130.

[33] Rosen P, Burton B, Potter K, R Johnson C. muView: a visual analysis system for exploring uncertainty in myocardial ischemia simulations; 2016. p. 49–69. doi:10.1007/978-3-319-24523-2_3. ISBN 978-3-319-24521-8.