Paper:

Torsney-Weir, T., Saad, A., Moller, T., Hege, H., Weber, B., Verbavatz, J. & Bergner, S. (2011). Tuner: Principled Parameter Finding for Image Segmentation Algorithms Using Visual Response Surface Exploration. *IEEE Transactions on Visualization and Computer Graphics, 17*(12), 1892-1901.
http://dx.doi.org/10.1109/tvcg.2011.248

Tuner: Principled Parameter Finding for Image Segmentation Algorithms Using Visual Response Surface Exploration

Thomas Torsney-Weir, *Student Member, IEEE*, Ahmed Saad, Torsten Möller, *Senior Member, IEEE*, Britta Weber, Hans-Christian Hege, *Member, IEEE*, Jean-Marc Verbavatz, and Steven Bergner, *Student Member, IEEE*

Abstract—In this paper we address the difficult problem of parameter-finding in image segmentation. We replace a tedious manual process that is often based on guess-work and luck by a principled approach that systematically explores the parameter space. Our core idea is the following two-stage technique: We start with a sparse sampling of the parameter space and apply a statistical model to estimate the response of the segmentation algorithm. The statistical model incorporates a model of uncertainty of the estimation which we use in conjunction with the actual estimate in (visually) guiding the user towards areas that need refinement by placing additional sample points. In the second stage the user navigates through the parameter space in order to determine areas where the response value (goodness of segmentation) is high. In our exploration we rely on existing ground-truth images in order to evaluate the ”goodness” of an image segmentation technique. We evaluate its usefulness by demonstrating this technique on two image segmentation algorithms: a three parameter model to detect microtubules in electron tomograms and an eight parameter model to identify functional regions in dynamic Positron Emission Tomography scans.

Index Terms—Parameter exploration, Image segmentation, Gaussian Process Model.

1 Motivation

For visual analysis image data often need to be segmented. Segmentation refers to the process of partitioning the image into multiple segments, i.e. sets of pixels or voxels, that form contiguous and semantically meaningful regions. If each of these regions is marked by a unique identifier, image segmentation simply means labelling of pixels or voxels. In biomedical imaging, where images are acquired using some kind of tomography or microscopy, segmented regions might correspond to anatomical structures in the case of non-functional imaging, and to regions with specific physiological activity in the case of functional imaging.

In recent years a variety of semi- and fully automatic techniques have been developed to address the segmentation problem [32]. However, even the current state-of-the-art approaches fall short of providing a “silver bullet” for image segmentation. This has several reasons. One reason is that given some image, the segmentation problem is not well defined; in fact it depends on the application which regions are semantically meaningful. Another reason is that due to different image degradation factors such as low signal-to-noise ratio, imaging artifacts, partial volume effects and shape variability, different kinds of a priori knowledge need to be included. Additionally, the majority of the existing segmentation methods rely on and are sensitive to setting a number of parameters. For example, most of the algorithms contain weighting parameters between multiple competing image-driven or prior-driven cost terms in an attempt to mimic the cognitive capabilities of expert users (e.g. radiologists for medical images).

A good parameter setting is usually found by a manual trial and error procedure. The segmentation algorithm developer starts with a particular parameter configuration and then checks for a quality or response measure of the final segmentation measured against a ground truth image where the correct segmentation is available. If the segmentation quality is not satisfactory, another parameter configuration will be tested. This is a tedious, time-consuming, and error-prone task. Furthermore, once a good parameter setting is found the developer then goes on (using the set of found parameters) to apply the algorithm to images without a ground truth. Because the developer has no context for the space around these ideal parameters they have no real idea of the applicability to other datasets.

In this paper, we propose a visual analysis tool to systematically explore the multi-dimensional parameter space impacting the quality of image segmentation algorithms. We adopt a statistical model known as a Gaussian process model to interpolate the response values given a sampling of the parameter space. We then use an interactive visualization to enable the exploration and refinement of the parameter space. The proposed tool can be applied to any fully automatic segmentation algorithm controlled by a number of tunable parameters and a quality measure for the obtained results.

2 Problem Statement

Image segmentation algorithms are typically plagued by a plethora of different tuning parameters. Conceptually, we differentiate *model parameters* from *algorithmic parameters*.

2.1 Model Parameters

An important class of segmentation methods are variational methods, see e.g. [30]. They rely on the minimization of an objective (energy) functional whose minima correspond to “good” segmentations. For this class, model parameters are the weights of the different terms in the energy functional. Building an energy functional gives the algorithm designer the ability to allow multiple competing goals to be considered. The energy functional is generally formulated as follows

$$E(\phi, I) = \alpha_1 E_1(\phi, I) + \alpha_2 E_2(\phi, I) + ... + \alpha_k E_k(\phi, I), \qquad (1)$$

where I represents the input image to be segmented, ϕ represents a segmentation, and $E_1, E_2, \ldots, E_k$ represent k different energy terms. Therefore, the parameters $\alpha_1, \alpha_2, \ldots, \alpha_k$ represent a weighting of the importance of every energy term. The final segmentation $\widehat{\phi}$ is obtained by minimizing (Eq. 1) as follows:

$$\widehat{\phi} = \arg\min_{\phi} E(\phi, I). \qquad (2)$$

- *Thomas Torsney-Weir, Ahmed Saad, Torsten Möller, and Steven Bergner are with GrUVi (Graphics, Usability, and Visualization Lab) at Simon Fraser University, Burnaby, Canada, E-mail: {ttorsney,aasaad,torsten,sbergner}@sfu.ca.*
- *Britta Weber and Hans-Christian Hege are with Zuse Institute Berlin (ZIB), Takustr. 7, 14195 Berlin-Dahlem, Germany, E-mail: {britta.weber,hege}@zib.de.*
- *Jean-Marc Verbavatz is with the Max Planck Institute of Molecular Cell Biology and Genetics (MPI-CBG) in Dresden, Germany, E-mail: verbavat@mpi-cbg.de.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

For example, consider the popular Snakes algorithm [21]. Here an approximate boundary evolves to the desired boundary guided by minimizing two competing energy terms. A boundary energy term attracts the solution to pixels with high gradients. However, since boundaries optimized with only this condition tend to be jaggy due to noise in the image, an additional smoothness term is introduced to enforce the boundary of the segmented object to act like a membrane or thin plate that is trying the stretch out. Many other energy terms have been considered in the segmentation literature and we are not trying to provide a complete list here. For a good overview, see, e.g., Pham et al. [32].

2.2 Algorithmic Parameters

Algorithmic parameters fine-tune different parts of the algorithms. For variational segmentation, for instance, there exist approaches, like graph based approaches graph cuts [7] and random walker [16], where the energy term itself contains parameters to be tuned. We could describe these terms using $E(\phi, I, \sigma)$ where σ would impact how similar two nodes are that are connected through an edge.

Algorithms not based on energy minimization also have tuning parameters. For instance it is quite common to have thresholding parameters. One of the fundamental image processing algorithms is edge detection; the most popular algorithm, the Canny edge detector [10], uses three parameters, controlling the size of a Gaussian smoothing function and thresholding with hysteresis (using min/max thresholds).

In addition, almost any segmentation algorithm also includes parameters like number of iterations, accuracies for termination conditions, etc. Parameter tuning is an integral part of almost any image processing task.

2.3 Quality Measures

In order to produce an image segmentation, a particular parameter setting is determined and a segmentation algorithm is applied to the image. During algorithm development an expert-segmented image, or *ground truth*, is crucial to measure the quality of the segmentation. Often a visual comparison of the automatic segmentation to the expert-segmented images is desired, but fine subtleties or 3D images are hard to inspect properly. Therefore, a number of other quality metrics have been developed.

One of the most popular metrics is the Dice similarity coefficient [14]. It measures the overlap between a segmented region and ground truth, with a value of one corresponding to a perfect overlap. Precision and Recall are two other widely used quantities to assess the quality of a classifier. Precision measures the percentage of *true positives*, i.e., which of the segmented pixels have the right label relative to all the pixels labelled with this label by the segmentation algorithm. In contrast, Recall measures the number of correctly labelled pixels relative to all the pixels that should carry said label based on the ground truth. Ideally, both of these measures should come out to one, but often improved Precision comes at the cost of reduced Recall and vice versa. Therefore, it is useful to examine these different measures simultaneously to ensure better segmentation performance with respect to different criteria.

Other commonly used tradeoffs include likelihood versus prior in Bayesian methods [1], loss versus penalty in pattern recognition [47] and image thresholding algorithms [39].

Our proposed tool is able to work with any segmentation technique controlled by a set of parameters and associated with a set of numerical quality measures.

2.4 Finding the Right Parameters

Given a segmentation model, a ground-truth, and one or several quality measures that evaluate the segmentation output relative to the ground-truth, an algorithm developer typically enters a time-consuming, tedious, and error-prone process to find good parameter values. Experience often goes a long way to come up with an initial guess. Manual variation of the parameter settings give a hint to the user of whether an improved segmentation is possible and whether the segmentation result changes slowly (i.e. we have a stable parameter region) or quickly (i.e. the segmentation result is very sensitive to the exact parameter setting). Often a single segmentation could take minutes if not tens of minutes or hours and every new parameter combination that needs to be tested will add to the frustration of the experimenter. Furthermore, keeping track of all the previously tested parameter combinations amounts to a test of patience, good memory, and being well organized. At no time of the exploration process is one ever sure, whether all the relevant parameter regions have been found.

To facilitate the parameter exploration process, we identify a set of *tasks* that our tool needs to support:

Exploring the full parameter space: A comprehensive and systematic way is needed to explore the full parameter space efficiently. This requires a strategy and tools for getting a quick overview and overall understanding of the parameter space in order to identify interesting regions. Furthermore, it requires means for refining the search in interesting regions.

Finding optimal parameter settings: The tool should allow the user to quickly navigate to all local optima in the global parameter space or in a subregion of it.

Assessing the sensitivity of a parameter region: The tool should enable the user to quickly assess the sensitivity of segmentation results to parameter changes.

Simultaneous exploration of multiple quality measures: Trade-offs between competing quality measures should be made clear and easy to explore and comprehend.

2.5 Contributions

Given these design constraints, we introduce a two-stage process to find optimal parameter ranges for image segmentation algorithms. During the first stage we employ an approach that samples the complete parameter space as densely as the time budget allows and then (in a batch process) automatically acquires all the corresponding segmentations. While this process is running ”over night” the user can devote his or her attention to other matters. Our approach also employs an uncertainty measure based on statistical reasoning to automatically refine regions that have not been sampled well. In the second stage the researcher explores the results of the first stage in an interactive setting. We use multidimensional navigation tools to find areas of high interest and to investigate the stability of these regions.

The contributions of this paper can be summarized as follows: (i) We develop a systematic model to explore the full parameter space based on a Gaussian process model [20]. (ii) We allow the user to visually explore the full parameter space using sliced-based navigation (similar to HyperSlice [43]) of the response function for up-to tens of parameters. (iii) We allow the user to study the trade-off of up to two quality measures. (iv) We provide uncertainty visualization of the response surface as well as the expected gain in order to facilitate refined sampling of the parameter space.

3 Related Work

Understanding and analyzing high-dimensional spaces has always been a challenge in statistical graphics as well as visualization. Approaches such as scatterplot matrices [12], parallel coordinates [18], and star-glyphs [44] are now common for visualizing high-dimensional data. Their main purpose, is to understand “point-clouds” or discrete entities. However, if one needs to understand *continuous* high-dimensional spaces, these approaches fail mostly, since they do not properly convey the continuity of the underlying space. Recently, Bachthaler and Weiskopf [2] extended scatterplots in order to properly portray continuous functions. However, the essence of scatterplots and similar approaches is to separate the data values from its intrinsic embedding in some metric space. This embedding is crucial if we want to understand the (local) sensitivity of the response surface. Sensitivity analysis studies the variance of the function to its embedding.

In medical imaging it is common to create a mental model of a 3D image of a patient by studying three orthogonal axis-aligned slices. Creating a mental model of a higher-dimensional continuous function is next to impossible, but the local behavior of a function can be externalized leading to a cognitive relief of the user. Using a slice-plane matrix for the understanding of a high-dimensional function had been suggested by van Wijk and van Liere using a technique which they coined *Hyperslice* [43]. This idea was extended by Tweedie and Spence by what they called the Prosection Matrix [42]. Here, a thick $(n-2)$-dimensional slab is being summarized as opposed to a simple 2D slice of the n-dimensional space under study. Since we believe, that a slice will be a more accurate portrayal of the space, we use the Hyperslice approach in this paper.

While van Wijk and van Liere were inspired by the study of parameter combinations for computational steering in chemical reactions, a complete system for this study was not proposed. The idea of "seeing" into a high-dimensional parameter space in order to understand the distribution of optimal places and their sensitivity has recently found a lot of attention in the visualization community. Computational steering has been a known problem for a while, which is addressed by several researchers in the visualization community, most recently in World-Lines by Waser et al. [45]. This problem is fundamentally different from the problem we are trying to solve. In computational steering the user studies a simulation over time and actually wants to change the parameters while the simulation is running. In our case, we must set the parameters at the start of a new simulation with the intention of optimizing the final output according to some quality measure. This is closer to the work by Bruckner and Möller in FluidExplorer [9]. However, one of the major accomplishments of FluidExplorer was dealing with simulation outputs where temporal behavior is crucial. Furthermore, they did not address optimization of any objective function.

Approaches where an objective function is missing typically require the user to express their preference after comparing different simulation outputs (see e.g. the work by Brochu et al. [8]). This is an area also known as *active learning*. Very recently Pretorius et al. [34] have been developing a system for the exploration of parameter values for image segmentation. In their case they are not making use of any quality measures and don't assume the availability of any ground truth. Therefore, their system is quite different from ours.

Alternative approaches to facilitate parameter explorations have resulted in systems like Design Galleries [25], Image Graphs [23], spreadsheet-like exploration interfaces [19], and VisTrails [37]. None of these approaches is utilizing an optimization function nor is the user able to see a comprehensive overview of which places of the parameter space have been "looked at" and which have not.

The inspiring work by Piringer et al. [33, 4, 5] is perhaps closest to our work. Their system, *HyperMoVal*, was one of the first comprehensive environments for studying the impact of parameters on simulation experiments. HyperMoVal was also using the ideas of Hyperslice for navigating through a high-dimensional scalar function as well as facilitating a sensitivity analysis. However, HyperMoVal was geared toward industrial applications and was validated in the automobile industry. Our scope is slightly smaller and we are focused on finding good parameter combination for image segmentation. Therefore, in many ways, our problem is more constrained and requires a much less complex system. The major difference to HyperMoVal is that we use several different quality metrics in order to judge the goodness of the parameter settings. These quality measures are the basis of our exploration and simplifies the user interface immensely. In HyperMoVal, colored contour-plots, representing the model estimation, are overlaid over scatterplots representing the measured data. Furthermore, their sensitivity analysis is very different in that it focuses on dimensional graphs by varying exactly one parameter and one local neighborhood around a single high-dimensional parameter combination. This has been improved with the authors current work [5].

3.1 Statistics

Our technique was born out of the work by Box and Wilson [6] in 1951. Their method is to fit gradually more and more complex estimating models to a complex function. One well-established area of research in statistics employing this idea is known as DACE - the Design and Analysis of Computer Experiments. For a good introduction we refer the reader to the book by Santner et al. [36]. The particular model we are using has been well described by Jones et al. [20]. It has been successfully employed in a variety of computer experiments such as an ocean circulation model [15], a hazard-effect model for volcano eruption prediction [3], and an arctic sea ice simulation [11]. However, the typical approach by statisticians is to fit the Gaussian process model and then evaluate the results of a variance decomposition. Two approaches to this method are discussed by Schonlau et al. [38] and Oakley and O'Hagan [31]. Our approach is to provide insight by viewing and interactively exploring the full response space.

3.2 Automatic Parameter Tuning in Image Segmentation

Few papers tackled the automatic parameter estimation in image segmentation as the main focus are effective segmentation models and efficient algorithms. Kumar et al. [22] applied a pseudo-likelihood technique to estimate the parameters of a conditional random field algorithm. Szummer et al. [40] applied graph cuts to do maximum margin efficient learning of the segmentation parameters. Mcintosh and Hamarneh [26] optimized a non-convex energy function to find the optimal parameters. They later extended their technique using a constrained convex energy function to avoid sensitivity to the initial parameter settings [27]. The common goal in these algorithms is to learn the parameters so that the ground truth emerges as the optimal solution. This is achieved by optimizing yet another energy function. Therefore, these techniques optimize for one particular quality measure only. In contrast, our tool provides a way to examine multiple complex responses simultaneously. In addition, automatic parameter learning was constrained to a particular form of a segmentation technique (e.g. conditional random field or deformable model) and could not handle general algorithms as our method does.

4 Gaussian Process Model

The core idea of our approach is to take the known segmentation results for particular parameter combinations and evaluate a quality metric for these segmentations. The values from this quality metric are also known as the *response*. Knowing the response at discrete values, we build an emulating model that allows us to interpolate from the known values and to estimate the quality metric at all places of the parameter space, even though we have not yet computed the actual segmentation at these places. The continuous function is often referred to as the *response surface* and the prediction at new parameter combinations is known as *inference* in statistics.

In our work, we particularly employ a *Gaussian process model* for computing the response surface. This is born out of an area known as *Design and Analysis of Computer Experiments (DACE)*. We will only be able to briefly summarize the essential ideas here and refer the reader for details to the excellent treatment by Jones et al. [20] or Santner et al. [36].

The Gaussian process model is a well-known technique in statistics. It assumes that the response surface is governed by some unknown random function $Y(x)$. What "random" means in this case is that we are not making any assumptions about what the underlying function looks like. The Gaussian process model interpolates a response value at an arbitrary point from known sample points (often called *design points*). In our case, however, the response to a particular input is deterministic. However, we don't know it ahead of time before we actually compute the segmentation and quality measure at that point. In that sense, the model of a random function $Y(x)$ encapsulates the uncertainty we have about a particular predicted response value that we have not yet computed. Hence, this uncertainty will be zero at the design points themselves.

4.1 Building the Model

In its most general form the model assumes that the response value at a particular parameter combination x is governed by some sort of average response function plus a deviation which is a weighted average of the response from all known sample points:

$$Y(x) = \sum_{i=1}^{k} f_i(x)\beta_i + \varepsilon(x). \tag{3}$$

The first term, $\sum_{i=1}^{k} f_i(x)\beta_i$, is the regression term and the second term, $\varepsilon(x)$, is the error term. The various basis functions $f_i(x)$ can be any continuous function (often low-order polynomials) of the input variable x. While the choice of regression functions is often not further restricted, a common choice is to simply select the constant function only, which captures the mean behavior, μ, of the response surface. This may seem overly restrictive but it turns out that the error term is so effective that the restriction of the regression to the mean only does not inhibit the power of the method [36].

Assuming n design points, the error term relies on the fact, that our confidence in our estimation decreases as we move farther away from the design sites. The error at the design sites is assumed to be zero representing complete confidence in the output of a computer model. An alternative way to put this is that the error at an arbitrary location x_{new} is correlated with the design sites by some n-dimensional function $c(x_{\text{new}}, X)$ where X is an $n \times d$ matrix representing all the design sites at which we have taken samples. Again, there are a number of choices for this correlation function but a popular and effective one is the Gaussian correlation function which, for a d dimensional input, is

$$c(x^i, x^j) = \prod_{k=1}^{d} e^{\theta_k (x_k^i - x_k^j)^2}. \tag{4}$$

Each of the θ_k factors in the above equation are known either as *correlation parameters* or *hyperparameters*. These hyperparameters are modeled through an appropriately chosen likelihood function (which properly predicts the measured responses). The typical way to determine these hyperparameters is by maximizing this likelihood function. This can be done numerically through an optimization procedure such as Simulated Annealing or Newton's Method. In practice this optimization converges very quickly. In our test cases this process only took a few seconds for 8 factors and up to 250 design sites. Once these θ_k hyperparameters are calibrated with the design sites we generate an $n \times n$ correlation matrix R with entries $r_{i,j} = c(x^i, x^j)$, the correlation between design sites x_i and x_j.

4.2 Prediction

Once we have the correlation matrix R computed we can predict the response at an arbitrary point. It works out that the best linear unbiased predictor of $Y(x)$ is

$$\hat{Y}(x_{\text{new}}) = \hat{\mu} + c(x_{\text{new}}, X)R^{-1}(Y - \hat{\mu}1), \tag{5}$$

where Y is a vector of length n of the response values for each design site. We can find closed form solutions for $\hat{\mu}$ and $\hat{\sigma}^2$, the values for the mean and variance of the response surface given our correlation matrix. These are given by

$$\hat{\mu} = \frac{1'R^{-1}Y}{1'R^{-1}1} \tag{6}$$

and

$$\hat{\sigma}^2 = \frac{(Y - \hat{\mu}1)'R^{-1}(Y - \hat{\mu}1)}{n} \tag{7}$$

The Gaussian process model is a statistical extension of traditional interpolation schemes, which allows the assignment of an uncertainty to the predicted value. There is no restriction to the type of basis function used, hence, spline models work just as well. It works out that the squared error in prediction, $Z^2(x)$ is

$$Z^2(x_{\text{new}}) = \hat{\sigma}^2(1 - c(x_{\text{new}}, X)'R^{-1}c(x_{\text{new}}, X)). \tag{8}$$

This measure reveals the confidence of the model in making a prediction at a particular point.

4.3 Next Sample Point

The uncertainty measure by itself is helpful, but if we just sample in locations where the uncertainty is high we will sample in a number of spots that will never lead to a maximal value. Since we are interested in design points that optimize our quality measures, it would be far better to combine our current estimate at a location with the uncertainty and use that as a guide. We should sample in areas with high estimated response *and* high uncertainty.

There have been several attempts at picking the next sample point algorithmically. These methods all attempt to combine areas of high response and high uncertainty in order to find the best place at which to take more samples. We use the method outlined in Jones et al [20] due to its ease of implementation and explanation to the user. It is very important to make sure the user can understand what the meaning is behind what the application is displaying (and it can be difficult to explain the reasoning behind a complex statistical model).

Instead of just taking the predicted point, $\hat{Y}(x_{\text{new}})$, as a known scalar we assume that the prediction at that point is the mean of a normal distribution with standard deviation equal to the standard error of the predictor; $Z(x)$ of (Eq. 8). With this assumption the expected improvement at a particular point, $I(x)$, works out to be (note that E and I in this formula are not related to (Eq. 1) and (Eq. 2))

$$E[I(x_{\text{new}})] = (f_{\text{opt}} - \hat{Y})\Phi\left(\frac{f_{\text{opt}} - \hat{Y}}{Z(x_{\text{new}})}\right) + Z(x_{\text{new}})\phi\left(\frac{f_{\text{opt}} - \hat{Y}}{Z(x_{\text{new}})}\right), \tag{9}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative normal and normal probability density functions respectively and f_{opt} is the current best response value.

5 Walkthrough

Here we present our system, Tuner. Tuner is designed to guide the user through the full lifecycle of tuning the parameters of a segmentation algorithm. This takes them from selecting an initial sampling strategy, to finding regions of interest, to further examining these regions of interest by placing and evaluating additional samples in these regions.

The overall pipeline for the analysis is shown in Fig 1. The overarching idea is to start with a sparse initial sampling of the parameter space. Then, through an iterative procedure of identifying regions of interest and refining with additional samples. In this manner the user is able to identify regions that meet their criteria.

Tuner is responsible for generating the points at which to take samples. These are the *sample points* shown in Fig 1. These are passed to the segmentation algorithm which assigns one or more scalar values indicating the "goodness" of segmentation. Once these *design points* are passed back to Tuner we build an interpolation model in the form of a Gaussian process model and use that model to drive the interface.

The only requirement that we impose on the segmentation code (whose inputs are being sampled) is that it can be run in the background (i.e. non-interactively). We link to the segmentation code by means of a user-specified shell script. The contract for the shell script is that it must take a reference to a file containing sample points generated by our program and write out the classified points into a file specified by Tuner. This keeps Tuner independent of any particular algorithm or platform.

5.1 Initial Sampling

In order to facilitate a systematic initial sampling of the parameter space we provide a workflow to place these initial samples. When the user creates a new project they are presented with the initial sampling dialog, shown in Fig 2(a). Getting a dense initial sampling of the parameter space is critical for gaining a good overall estimation of the parameter space. Perhaps the most obvious way would be to lay out a large number of samples per dimension in a Cartesian grid. However, the combination of high dimensionality of the inputs and the expense of running one sample point make this strategy prohibitively expensive. We want to minimize the number of runs of the segmentation code while getting a good overall idea of the response manifold.

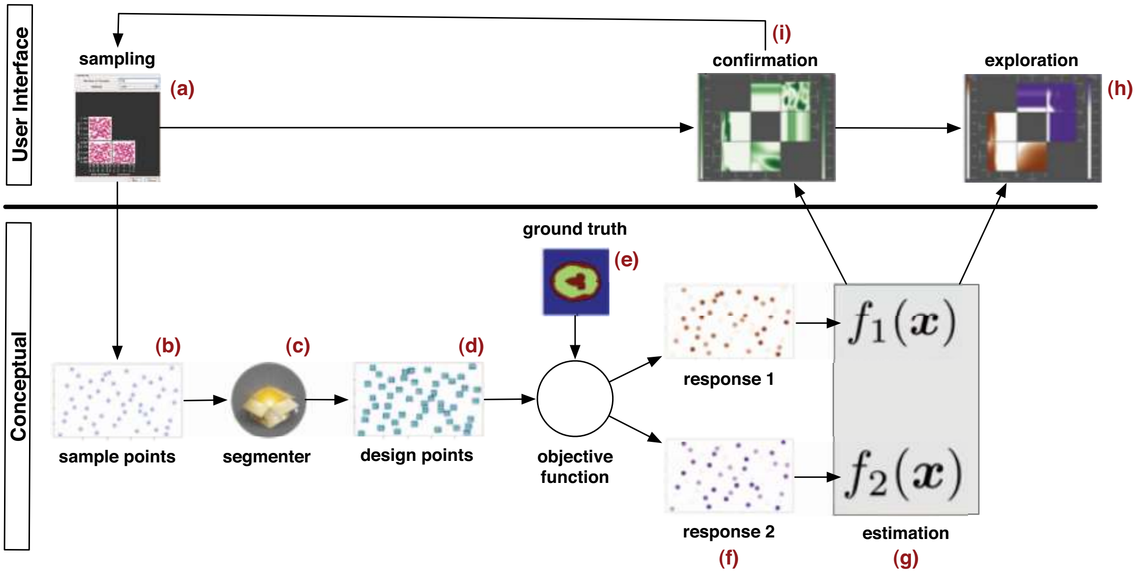

Fig. 1. An overview of the workflow in Tuner. The user starts (a) by taking an initial sampling of the space. This generates a set of sample points (b) at which we want to compute segmentations. These points are passed off to the segmenter (c) and design points — the segmented images — are generated (d). The segmented images are compared against a ground truth image (e) in order to generate scalar responses (f). We then estimate the full response space (g) and display it to the user such that they can explore it (h). At any time the user can generate additional sample points in order to build up a more accurate model (i).

As an alternative we provide the user two alternative sampling strategies: Latin Hypercube sampling [28] and random sampling. Both of these strategies place an exact number of samples in the parameter space. This allows the user to accurately interpret the running time. We show the generated sample points in a scatterplot matrix. This provides the user with a general idea of how well the number of sample points they have chosen fill the sampling space. Another advantage of these strategies is that both of these sampling strategies run in interactive time. When the user changes the number of sample points in the dialog they immediately see the updates in the SPLOM.

Clicking on the run button begins the sampling process. Tuner monitors the state of the sampling and provides a progress bar to show feedback. Once sampling is complete Tuner automatically builds the Gaussian process model for each non-input field found.

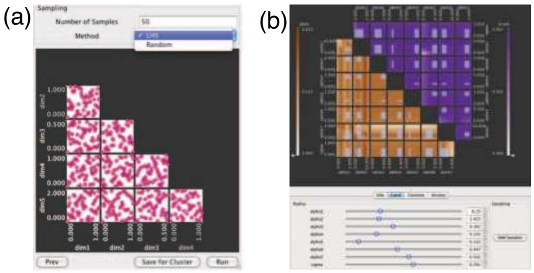

Fig. 2. a) Marking out an 8D hyperbox in order to place additional samples. The sliders below the box control the extent of the box centered at the current point. Adding samples brings up the sampling interface b) which uses a SPLOM preview of sample point locations. The user enters the desired number of sample points and the sampling strategy and the SPLOM automatically updates. The “Save for Cluster” button at the bottom allows the user to save the sample points to a file (e.g. for running on a cluster). The “Run” button runs the sample points directly through Tuner.

5.2 Project Viewer

Once the Gaussian process models are built the user is presented with the Project Viewer window, shown in Fig 3. This is the primary interface for interaction with the response surface. The main sections of this interface are the *Pareto Panel*, the *Response View*, the *Controls*, and the *Histograms*. These views are designed to support the tasks of analyzing the tradeoff of up to two response variables, finding optimum parameter settings, and brushing and refining regions of interest.

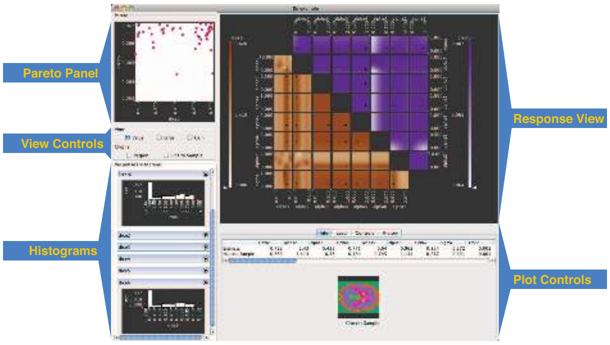

Fig. 3. The main interaction window in Tuner. The *Response View* is a slice-based view of the response surface. The *Pareto Panel* shows the tradeoffs between the two selected response values. The *View Controls* and *Plot Controls* adjust the type of plot shown and the current parameter settings. The user can also mark out a region for sampling here. The *Histograms* show estimated histograms for each of the response variables. We have linked these views in order to facilitate the evaluation of various tradeoff points.

A typical workflow with this view begins with the Pareto Panel. The user selects a favorable output combination. They then explore the area around that known tradeoff point and mark out a region in which to place samples. The user then runs these sample points through the external segmentation algorithm. The Gaussian process model automatically rebuilds and the updated view is presented to the user.

5.2.1 Pareto Panel

Pareto analysis is an established term in statistical data analysis. It refers to the fact, that when optimizing multiple objects, not all objectives can be maximized. The so-called *Pareto Front* are all points in parameter space where at least one objective cannot be further improved, given that all other objectives remain fixed.

The Pareto Panel, shown in Fig 3 gives the user an overview of the combination of known response values. This view shows a scatterplot of the sampled response values for the selected pair of response dimensions. The optimal tradeoff between pairs of response values is not clearly defined. Some tasks may place a heavy weight on one particular response dimension, effectively performing a 1D optimization while other tasks may weight all response dimensions equally. Finding these optimal tradeoff points by navigating through the input parameter space is akin to finding a needle in a haystack.

This is an interactive tool in the sense that clicking on a particular dot in the scatterplot will set the slices to the parameters backing that dot. Thus the user can start their exploration at a known tradeoff point and then explore the surrounding area. We found that the users were quickly able to reason about which response to prioritize over another using this view.

5.2.2 Response View

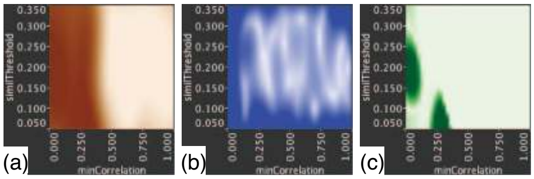

Fig. 4. The three types of plots in our system. a) Shows the response value itself. Here darker regions indicate higher response. b) Shows the uncertainty in estimation. Here darker regions indicate higher uncertainty. c) Shows the expected gain from additional sampling. Darker colors here are areas of higher expected gain.

The *Response View* is the main portal for interaction with the estimated response surfaces. The plots shown are analogous to a scatterplot matrix. However, instead of scatterplots for each pair of dimensions we are showing 2D slices as was suggested in HyperSlice [43]. This is a very familiar interface for our users as slice-based views are common in medical image visualization. There are two methods for interaction with these plots. One is by clicking and dragging in the plots themselves. This changes the slices accordingly. We also provide slider controls to change the slice and range filters to change the zoom level under the "Controls" tab in the "Plot Controls" section shown in Fig 3.

This view allows us to show and visually compare up to two response dimensions at a time. The first response variable is shown in the lower left matrix and the second response value is shown in the upper right matrix. We use different color maps for the different response values in order to visually distinguish them. Different response variables are distinguished through different hues using quantitative colormaps from Colorbrewer [17].

We provide three different plot types of the data to the user. An example image of each is shown in Fig 4. Each of these plot types support a different task that the users want to perform. Response shows the estimated response value at any given point. This view supports finding regions of high-quality segmentation. Error shows the standard deviation (Eq. 8) reported by the estimation model. Here, areas of high error are essentially gaps in our sampling. By placing sample points in these gap regions the analyst is able to build up a more accurate model across the entire input space. A model built up in this way is ideal for conducting a global sensitivity analysis. The third plot shows the expected gain in the maximum from sampling at that location. This expected gain measurement is a realization of (Eq. 5). By placing sample points in areas of high expected gain the user builds up a more accurate model in areas with a high likelihood of finding an optimum value. This supports the optimization task.

In practice we found that the error plots were not as useful as the gain plots. One possible explanation for this is that the error plots indicate areas of global uncertainty. This error value does not account for the fact that it may be guiding the user towards areas where the response is known to be low. However, the gain plots are taking into account the current best known sample point as well as the uncertainty. This is better suited to our optimization task since we will not spend as much time placing samples in areas where we do not expect to find any sort of optimal value.

The color maps are interactive. The user can filter out response values that are of no interest by compressing the color map. This allows the user to better distinguish interesting from uninteresting areas. Since we are searching for optimal values, the filtering of the colormap needs to constrain the range in only one direction. Filtered values are mapped to a neutral gray color in order to visually distinguish them from the unfiltered values.

5.2.3 Controls

In the *Plot Controls* section we placed the controls that affect the user's exploration and refinement of the response surface in a tab panel. The "Info" tab shows the user the details. This includes the numerical values of all input parameters and output values for the specified focus point. It can also include an image slice of the corresponding segmentation. The focus point is also indicated with a crosshair in each plot. The "Local" tab contains a slider for each dimension. These sliders specify the size of the region of interest which can be used as a bounding box for placing additional samples or simply as a reference. This tab also contains a table listing the number of sample points in the current region as well as the gradient with respect to each input parameter. The "Controls" tab contains controls that allow the user to adjust what slice they are viewing , the zoom level, and which response variables to view. The user is also allowed to save particular points they find interesting as a type of bookmark by pressing the "H" key at any time. The "History" tab contains a table of these saved points, one per row. The user can click on a row in that table and that will take them back to that point.

In the *View Controls* section we provide controls that only affect the view in the *Response View*. The plot types are changed with radio button controls. In addition, we provide controls to show and hide the currently selected region of interest and a control to show a line from the current slice to the nearest sample point. This shows users where the sample points are in the parameter space. However, neither of our users used this feature in their analysis.

5.2.4 Histograms

The response histograms are designed to give the user an idea about the distribution of response values. We show one histogram per response dimension. To generate these histograms we take a dense sampling of the *estimated* response values from the Gaussian process model.

We also use an arrow glyph in the x-axis of each histogram to show where the current slice lies in the range of outputs. This lets the user see at a glance how close the currently selected point lies in relation to all response dimensions, not only the ones shown in the Pareto Panel. The histograms can be individually hidden so that unimportant response dimensions will not clutter the interface.

5.3 Regions of Interest

Once the user has identified regions of interest via the response view or one of the two error views their next task is to place additional sample points in this region in so as to further refine the shape of these regions. Remember that we only took a few samples spread throughout the full high-dimensional parameter space. It is very likely that there are some regions that require a denser sampling.

The "gain" plots are well-suited to this task, an example of which is shown in Fig 5(a). In this particular plot areas where the expected gain is high are represented in dark green. An advantage of using this scalar gain value for the plots is that we can utilize the same navigational interface used for finding high response to find good areas in which to take additional samples. The expected gain measure is just treated as an additional albeit always available goal function. The user does

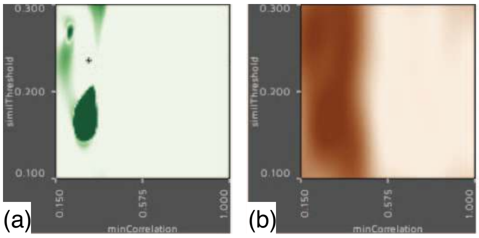

Fig. 5. a) An example expected gain plot from our system. Note the area of high gain around minCorrelation 0.2 and similThreshold 0.175. b) The corresponding response value for the gain plot. Note that the most gain is achieved around the location of highest response.

not have to learn two workflows and switching between sampling and response surface maximization is instantaneous.

In order to mark out a region of interest we provide the user with a set of radius sliders located under the “Local” tab in the controls section. An example of this is shown in Fig 2(b). These allow the user to mark out a hyperbox centered at the current slice over the current region of interest. The user then clicks on the “Add Samples” button on that control panel which opens up a dialog that presents an interface that is identical to the initial sampling dialog, Fig 2(a). The interaction here is also identical: the user selects the number of sample points to place in the region and a sampling strategy and the dialog provides a preview of the locations of the sample points in a SPLOM. When the user clicks on the run button in this case the project window is closed and the user is presented with a progress bar indicating the status of the sampling. Once the sampling has finished the Project Viewer window reopens and the user may examine their refined regions.

5.4 Task Solutions

We conclude the walkthrough of Tuner explaining how our solutions correspond to the tasks laid out in Sec. 2.4.

Exploring the full parameter space: We use slice-based navigation in order to explore the full parameter space. In addition, we allow the user to click on a plot in order to snap to that location. In order to allow the user to filter out uninteresting regions we provide zoom functionality in each dimension. We also allow the user to compress the colormap of either response value. Specific details about a particular point are displayed as a table to the user (details-on-demand).

Finding optimal parameter settings: Through the use of the Pareto Panel, identified in Fig 3, the user is able to understand and select a trade-off point to explore in further detail. Using the expected gain plots to find areas with high expected gain around this location, the user takes additional samples in order to find the maximum value.

Assessing the sensitivity of a parameter region: As the user changes a particular parameter value all plots dependent on it change interactively. At first this may seem to be confusing for the user since so many elements are changing simultaneously. However, what the user is able to very quickly grasp is how much each plot is changing as they move through a particular dimension. The change causes their attention to automatically move to the plot that is changing the most. In fact, we found that this interactive method was preferred over looking at static slices and deciphering the effect from those.

Simultaneous exploration of multiple quality measures: We show the two response surfaces in linked views thereby showing both quality measures simultaneously. The user is able to visualize different parameter settings and their effects on both quality measures without having to change views. We also provide a tradeoff plot for known sample points in the form of a scatterplot allowing the user to select a desired Pareto point as a starting location.

6 Case Studies

Our users were PhD students whose research involves the development of novel image segmentation techniques.

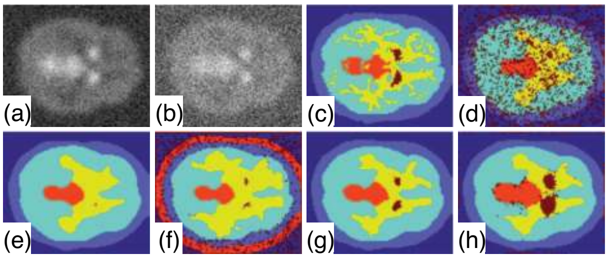

Fig. 6. Brain dynamic PET study images. a) The last time step slice (highest signal-to-noise ratio) of the synthetic dPET data blurred with a Gaussian kernel (i.e. introduces partial volume effect) with noise level 5λ, b) with noise level 10λ, c) the ground truth segmentation, d-h) represent multiple segmentations with different parameter configurations showing the need for carefully fine-tuning the algorithm parameters.

6.1 Brain Dynamic PET Study

In dynamic PET imaging, a series of 2D images are reconstructed from listmode data obtained by Gamma coincidence detectors. Kinetic modelling is the process of applying mathematical models to analyze the temporal tracer activity in order to extract clinically or experimentally relevant information. An extension of the algorithm by Saad et al. [35] is being developed by adding more energy terms to make the segmentation more robust. The goal is to segment 2D+Time PET images into six functional regions: background, skull, grey matter, white matter, cerebellum, and putamen. We used images from Saad et al. [35].

The proposed probabilistic algorithm is controlled by eight parameters: α_1 represents the weight of an image fidelity term, α_2 represents the weight of a random walker based spatial regularization term [16], α_3 represents the weight of a shape prior term, α_4 represents the weight of an intensity prior term, α_5 represents the weight of a non-negativity constraint over the segmentation probability field, α_6 represents the weight of a non-negativity constraint over the recovered regional TACs, α_7 represents the weight of a prior term over the recovered regional time activity curves (TACs) using a set of templated TACs, σ represents a parameter that impacts how similar two nodes are that are connected through edges which is common in graph-based approaches [16]. Two main quality measures have been calculated for the putamen structure, the dark brown object in Fig 6(c): the Dice metric [14] that measures the quality of the recovered shape and the Glucose metabolic rate recovery error that measures the error in recovering the physiological parameter under investigation.

Fig 6(a) and Fig 6(b) show the last dPET time step with noise levels 5λ and 10λ respectively. Here, λ is used to scale the unit variance of the random noise generator to the scale of the synthetic TAC intensity at each time step [35]. We show the last time step as it has the highest signal-to-noise ratio (SNR), as is typical in dPET (the preceding time frames are even noisier). Fig 6(c) shows the ground truth image that we hope to obtain. Fig 6(d)-Fig 6(h) show multiple segmentations with different parameter configurations demonstrating the need for fine-tuning the algorithm parameters to obtain suitable results.

Fig 7 shows the exploration stages for the dPET parameter space using Tuner. We first ran an experiment with 50 different parameter configurations sampled using a Latin Hypercube method. The first goal is to obtain a parameter configuration with low Glucose metabolic recovery error and high Dice metric for the putamen structure. In order to choose a starting point for our exploration, we examine the Pareto Panel showing the Dice metric versus the Glucose metabolic recovery error Fig 7(a). Examining the preview image at point *a*, shown in Fig 7(a), represents a low error value with moderate Dice value but the corresponding image shows a salt-and-pepper like noise in the segmented image. Point *b* represents moderate error with high Dice value and a better segmented image. Hence, we choose the parameter configuration corresponding to point *b* as a starting configuration. Fig 7(b) shows the slice plot matrix corresponding to the initial pa-

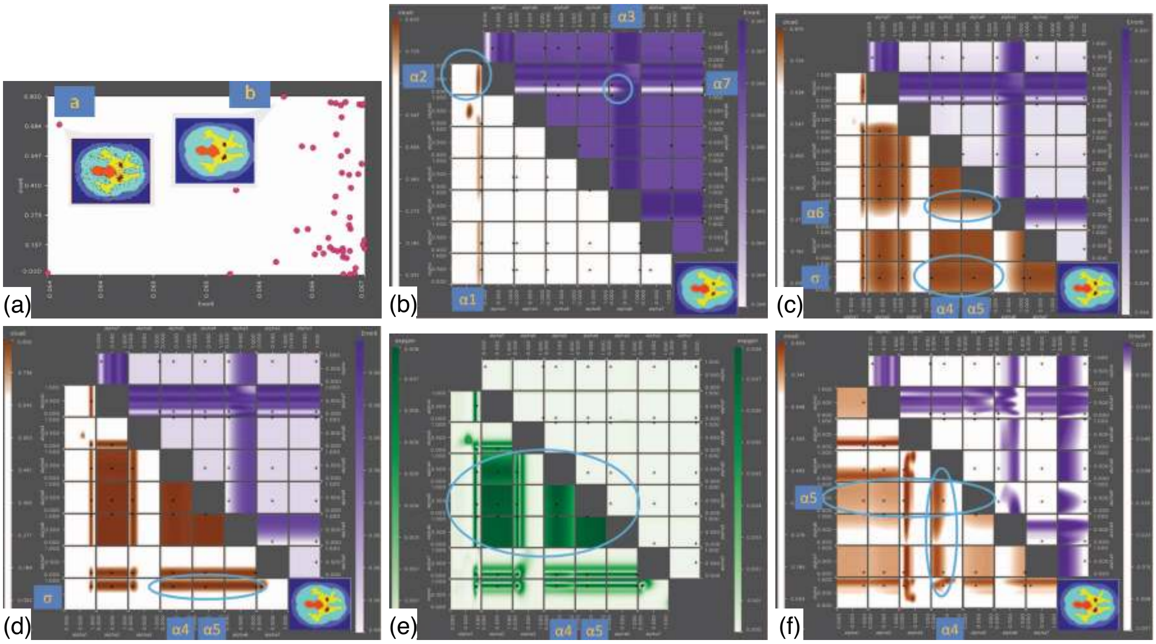

Fig. 7. Parameter exploration of the dPET parameter space. a) Dice metric versus the Glucose metabolic recovery error Pareto Panel that corresponds to 50 samples. b) slice plot matrix that corresponds to the initial parameter configuration of point b. The bottom-right corner shows the corresponding segmented image. c) Fine-tune α_3, α_7, α_1, α_2, and α_6. d) shows the slice plot matrix of higher noise level 10λ e) expected gain plot suggesting more needed samples especially in the regions of α_4 and α_5. f) 250 samples slice plot matrix.

rameter configuration of point b. It shows the possibility of obtaining lower recovery error in the α_7 vs. α_3 plot and better Dice in the α_2 vs. α_1 plot. The bottom-right corner shows the respective segmented image. Fig 7(c) shows the selection of the optimal α_7, α_3, α_2, and α_1. It also shows that $\alpha_6 \geq 0.7$ is a stable region in terms of the Dice coefficient. For α_4, α_5, and σ, the Dice plot shows larger stable regions.

To examine the performance with higher noise levels, Fig 7(d) examines the slice plot with additive noise of variance 10λ. It shows that there is shrinkage in the stable region in the Dice plot for σ suggesting a good range to be $0.6 \leq$ sigma ≤ 0.8 but still not a definitive answer yet for α_4 and α_5. Fig 7(e) shows the expected gain plot, which suggests adding more samples for this higher noise level especially in the α_4 and α_5 ranges. Fig 7(f) shows a 250 sample experiment for the 10λ noise level. It shows that lower values of α_4 give higher Dice values. However, it still shows wide stable region for α_5 suggesting that neither the Dice value nor the error metric are sensitive to the change of α_5. This has to be related to the fact that other spatial regularization parameters prevent the segmentation probability matrix to contain negative values which α_5 controls directly. The final parameter configuration yields a Dice value of 0.806 and error value of 0.064.

We have applied the same values to 11 datasets accounting for inter-subject anatomical variability. We find a mean Dice value of 0.754 with standard deviation of 0.084 and a mean error value of 0.0641 with standard deviation of 0.000078. This shows that the parameter settings picked from wider stable regions in the parameter space lead to reasonable segmentations for images from the same population [26].

6.2 Microtubule Extraction from Electron Microscopy Tomograms

We used Tuner to determine the best parameter settings for a biological segmentation algorithm designed to extract microtubule centerlines from Electron Microscopy Tomograms. The aim of this segmentation is to give reasonable measures for the density of microtubules in the specimen. Microtubules constitute a part of the cell cytoskeleton and are subject to extensive study in biological research due to their important role in scaffolding and cell division.

The segmentation algorithm proceeds by first enhancing the centerlines of the microtubules in the tomograms by computing normalized cross-correlation with an idealized cylinder [24], [46]. In the second step the geometry of the centerlines is reconstructed using a simple iterative greedy traversal in the enhanced volume. The tracing algorithm has three parameters: a threshold, *minCorrelation*, determining where to search for microtubules in the volume (*MC*), a model parameter, *orientation*, penalizing strong curvature of the traced lines (*OW*), and a second threshold, *similThreshold*, that determines when to stop the tracing of one line (*ST*). The outputs of this algorithm are polygonal lines representing microtubules in the volume.

Finding a good parameter set by visual inspection is nearly impossible since the microtubule network can be very dense, containing between 500 and 2000 lines. Many datasets $(30-100)$ have to be processed as well. Thus we aim to find common parameter settings yielding good results on *few* test data sets and apply the found parameters to the remaining data. The assumption is that a parameter set exists that yields reasonable results for all data. Hence, we need to assure that the selected test data sets represent quite well the variety of the complete set. Likewise, if no stable parameter range for this test set can be found, we can conclude that it is impossible to use the automatic segmentation algorithm with one parameter setting for all datasets.

To find suitable parameters, we created sets of microtubule centerlines manually (the *ground truth*) for the test data and compare the automatically computed centerlines to these. The measurement we use is similar to the one utilized by Cole et al. [13] and results in numerical measures Precision and Recall which measure the estimated error rate in the algorithmic segmentation. Note that using Precision and Recall as measures requires that the ground truth segmentation is complete, since missing lines would result in a low Precision even for a perfect automatic segmentation. Previous work [48], albeit from a different domain, leads us to believe that setting the Precision in our case to above 0.9 and Recall to above 0.8 will yield trustworthy results.

Tuning parameters according to these constraints is time consuming, since ST, OW, and MC depend on each other. Before Tuner, we investigated the effects of the various parameter settings by fixing two of the three parameters and then varying the third. We then plotted Recall over Precision [13]. In the resulting plot the best parameter choice can be found where the curve is closest to one for both Precision and Recall. A stable range of parameters can be found when plotting Pre-

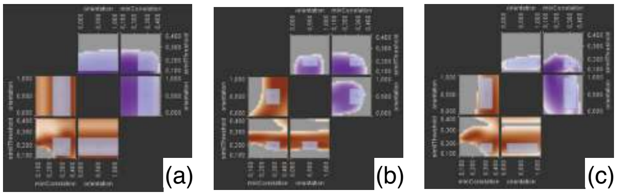

Fig. 8. Final parameter ranges for the three test datasets. Gray boxes enclose the areas chosen as final ranges for minCorrelation (*MC*), similThrehold (*ST*) and orientation (*OW*) fulfilling both Recall and Precision constraint for a) *DTHQ*, b) *DTMQ* and c) *STLQ*.

cision/Recall over the parameter, but only with the other parameters fixed. Because of the interdependence of the parameters, this tuning is an iterative process. Tuning by hand can take days to weeks. Adding another model parameter to the tracing increases the complexity even further, since checking the effect of the new parameter would require to compare its effect against all three other parameters.

Using Tuner, we estimated our parameters independently on three different data sets with varying acquisition quality (referred to as *DTHQ*, *DTMQ* and *STLQ*). For each data set we started with 50 samples on a Latin Hypercube within our three-dimensional parameter space. The number of samples was constrained by the fact that each segmentation took on average ten minutes. Hence, 50 samples took almost 9 hours to compute.

Using the Pareto Panel we first navigated to a sample point near or in the quality constraint. We filtered out all values for Precision and Recall below the constraint using the interactive colormap and drew a box marking ranges of parameters that lay within the unfiltered area, thus fulfilling the constraint. In order to mark out the full region, we iteratively moved the center point of the found box to points nearby to search for areas where a larger box could be drawn and resized the box. We refined with 20 new sample points. This procedure was repeated for the two other test data.

For each of the three data sets we again searched for valid ranges in the same manner using the refined samples, but this time precisely fulfilling the constraint. Fig 8 shows the resulting choices for each of the data sets. Grey boxes mark the finally chosen ranges. Table 1 lists the final ranges for each parameter and data set. A parameter set that fulfils the constraint for all the datasets must lie inside the intersection of these ranges. The intersection is given in the last line of Table 1.

The resulting parameter choices correspond to what we measured by using the above described Precision/Recall plots. However, it was an order of magnitude faster. Moving the slider of one parameter can be considered the same operation as creating a Precision/Recall plot as mentioned above. With Tuner, instead of having to recompute additional values, a simple move of the slider is sufficient. This reduced the work of days to a couple of hours.

Table 1. Resulting parameter ranges for the microtubule segmentation algorithm satisfying the quality constraint.

	min *MC*	max *MC*	min *ST*	max *ST*	min *OW*	max *OW*
DTHQ	0.256	0.37	0.168	0.28	0	1
DTMQ	0.26	0.348	0.197	0.247	0.307	0.702
STLQ	0.278	0.36	0.179	0.235	0.142	0.93
min/max	0.278	0.348	0.197	0.235	0.307	0.702

7 Implementation

Tuner is written in the Scala programming language using APIs provided by the Processing library and OpenGL. For the Gaussian process model and Latin Hypercube generator we are using packages for the R environment, `mlegp` and `lhs` respectively. The link between Scala and R is provided by the JRI Java library. The axes labels were determined using the algorithm and code described in Talbot et al. [41].

8 Lessons Learned and Future Work

Because of enthusiastic feedback by our collaborators and their colleagues, we plan to extend the capabilities of our tool and make it available to a wider audience.

Currently, we constrain our tool to focus on only two trade-offs at a time. An earlier version of Tuner allowed the user to simultaneously view the output of up to four objective functions. However, with the layout we used it proved too difficult to compare between objective functions to be effective. After discussing the case with our users we found that while they definitely needed to visualize two objective functions simultaneously there was not a strong case to be made for viewing more than two. An example use case is examining Dice coefficients for different structures simultaneously. There are approaches in the literature, that deal with the trade-off of multiple objective functions [42], [29]. We believe their work can help in improving Tuner.

Further, there is a need to compare the tuning of parameters for multiple images from different patients or multiple images from different noise levels simultaneously. However, this would add a categorical data type. Hence, we need to investigate how the Gaussian process model can be applied to categorical data types. Currently, the model assumes that all parameters are continuous.

As seen in our PET study, some objective measures are inexact, i.e. improving the objective measure (in our case Dice) doesn't necessarily improve the quality of segmentation. Still, the objective measure is vital to guide the algorithm (and the user) towards a good segmentation, but the fine tuning needs to happen by examining the segmentations directly. Therefore, it is desirable to not just predict the (scalar) objective measures, but also the actual segmentations. We investigated naïvely applying the Gaussian process model in order to estimate the voxels of the images between known segmentations but we found that the estimated images produced were not accurate enough to be reliable predictors of the actual segmentation algorithm. We need to find a proper, general methodology for image interpolation.

The Pareto Panel in Fig 3 was found to be incredibly useful in terms of selecting a starting location for one's analysis. Users are able to visually see the trade offs in maximizing one response variable against another. However, it only shows values for known segmentations. We intend to extend this widget to show the full Pareto Frontier for the response values by taking into account the estimated response values from the Gaussian process model. This would allow the user to fine-tune their trade-off analysis.

9 Conclusion

In this paper we have demonstrated Tuner, a tool designed to assist developers of segmentation algorithms with finding "good" parameter settings for a wide variety of algorithms. We began by introducing four tasks that are vital to segmentation algorithm developers: exploring the full parameter space, finding optimal parameter settings, assessing the sensitivity of a parameter region, and simultaneous exploration of multiple quality measures. We demonstrated how our technique of building a statistical model from a sparse sampling, iterative improvement, and high dimensional visualization allow the user to perform these tasks. By using this tool they are able to replace a tedious and manual process with a principled and systematic approach that allows them a much greater understanding of the effect of a parameter on their algorithm. Both our users only spent some hours on the exploration of the parameter space using Tuner before becoming confident on particular optimal parameter regions. Both users had used days and weeks on the same task before Tuner was made available to them.

Acknowledgments

We thank Derek Bingham for fruitful discussions. We acknowledge support from the Natural Sciences and Engineering Research Council of Canada. The microtubule project was funded by the Max Planck Institute of Molecular Cell Biology and Genetics, Dresden. We thank Tony Hyman for his support of this project.

REFERENCES

[1] S. P. Awate, T. Tasdizen, N. Foster, and R. T. Whitaker. Adaptive Markov modeling for mutual-information-based, unsupervised MRI brain-tissue classification. *Medical Image Analysis*, 10:726–739, 2006.

[2] S. Bachthaler and D. Weiskopf. Continuous scatterplots. *IEEE Trans. on Visualization and Computer Graphics*, 14(6):1428–35, 2008.

[3] M. J. Bayarri, J. O. Berger, E. S. Calder, K. Dalbey, S. Lunagomez, A. K. Patra, E. B. Pitman, E. T. Spiller, and R. L. Wolpert. Using statistical and computer models to quantify volcanic hazards. *Technometrics*, 51(4):402–413, Nov. 2009.

[4] W. Berger and H. Piringer. Interactive visual analysis of multiobjective optimizations. In *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*, pages 215–216, October 2010.

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

[6] G. E. P. Box and K. B. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(1):pp. 1–45, 1951.

[7] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[8] E. Brochu, T. Brochu, and N. de Freitas. A Bayesian interactive optimization approach to procedural animation design. In *Proc. of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 103–112. Eurographics Association, 2010.

[9] S. Bruckner and T. Möller. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Trans. on Visualization and Computer Graphics*, 16(6):1468–76, 2010.

[10] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, PAMI-8(6):679–698, Nov 1986.

[11] W. L. Chapman, W. J. Welch, K. P. Bowman, J. Sacks, and J. E. Walsh. Arctic sea ice variability: Model sensitivities and a multidecadal simulation. *Journal of Geophysical Research*, 99(93):919–935, 1994.

[12] W. S. Cleveland. *The elements of graphing data*. Wadsworth Publ. Co., Belmont, CA, USA, 1985.

[13] F. Cole, A. Golovinskiy, A. Limpaecher, H. S. Barros, A. Finkelstein, T. Funkhouser, and S. Rusinkiewicz. Where do people draw lines? *ACM Trans. on Graphics*, 27(3):1–11, 2008.

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

[15] W. Gough and W. Welch. Parameter space exploration of an ocean general circulation model using an isopycnal mixing parameterization. *Journal of Marine Research*, 52(5):773–796, 1994.

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

[17] M. A. Harrower and C. A. Brewer. Colorbrewer.org: An online tool for selecting color schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[18] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1:69–91, 1985. 10.1007/BF01898350.

[19] T. J. Jankun-Kelly and K.-L. Ma. A spreadsheet interface for visualization exploration. In *Proc. of the 11th IEEE Conference on Visualization (Vis '00)*, pages 69–76. IEEE Computer Society, October 2000.

[20] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[21] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[22] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proc. of ICCV*, pages 1150–1157, 2003.

[23] K.-L. Ma. Image graphs – a novel approach to visual data exploration. In *Proc. of IEEE Visualization 1999*, pages 81–513. IEEE, 1999.

[24] J. R. MacIntosh, editor. *Cellular Electron Microscopy*. Elsevier, 2007.

[25] 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 *Proc. 24th Ann. Conf. on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 389–400. ACM Press, 1997.

[26] C. Mcintosh and G. Hamarneh. Is a single energy functional sufficient? adaptive energy functionals and automatic initialization. In *Proc. of the 10th international conference on Medical Image Computing and Computer-Assisted Intervention*, MICCAI '07, pages 503–510, 2007.

[27] C. Mcintosh and G. Hamarneh. Optimal weights for convex functionals in medical image segmentation. In *Proc. of the 5th International Symposium on Advances in Visual Computing*, pages 1079–1088, 2009.

[28] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[29] K. Miettinen, P. Eskelinen, F. Ruiz, and M. Luque. NAUTILUS method: An interactive technique in multiobjective optimization based on the nadir point. *European Journal of Operational Research*, 206(2):426 – 434, October 2010.

[30] A. Mitiche and I. B. Ayed. *Variational and Level Set Methods in Image Segmentation*, volume 5 of *Springer Topics in Signal Processing*. Springer Verlag, 2011.

[31] J. E. Oakley and A. O'Hagan. Probabilistic sensitivity analysis of complex models: A Bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(3):751–769, Aug. 2004.

[32] D. L. Pham, C. Xu, , and J. L. Prince. Current methods in medical image segmentation. *Annual Review of Biomedical Engineering*, 2:315–337, 2000.

[33] H. Piringer, W. Berger, and J. Krasser. HyperMoVal: Interactive visual validation of regression models for real-time simulation. *Computer Graphics Forum*, 29(3):983–992, 2010.

[34] A. J. Pretorius, M. Bray, A. E. Carpenter, and R. A. Ruddle. Visualization of parameter space for image analysis. *IEEE Trans. on Visualization and Computer Graphics*, 2011. In Press.

[35] A. Saad, G. Hamarneh, T. Möller, and B. Smith. Kinetic modeling based probabilistic segmentation for molecular images. In *Proc. of the 11th international conference on Medical Image Computing and Computer-Assisted Intervention - Part I*, MICCAI '08, pages 244–252, 2008.

[36] T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer-Verlag, 2003.

[37] C. Scheidegger, H. Vo, D. Koop, J. Freire, and C. Silva. Querying and creating visualizations by analogy. *IEEE Trans. on Visualization and Computer Graphics*, 13:1560–1567, November 2007.

[38] M. Schonlau and W. J. Welch. Screening the input variables to a computer model via analysis of variance and visualization. In A. Dean and S. Lewis, editors, *Screening*, pages 308–327. Springer New York, 2006.

[39] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, 2004.

[40] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *Proc. ECCV*, pages 582–595, 2008.

[41] J. Talbot, S. Lin, and P. Hanrahan. An extension of Wilkinson's algorithm for positioning tick labels on axes. *IEEE Trans. on Visualization and Computer Graphics*, 16(6):1036–1043, 2010.

[42] L. Tweedie and R. Spence. The Prosection Matrix: A tool to support the interactive exploration of statistical models and data. *Computational Statistics*, 13(1):65–76, 1998.

[43] J. van Wijk and R. van Liere. Hyperslice: Visualization of scalar functions of many variables. In *Proc. of the 4th Conference on Visualization'93*, pages 119–125. IEEE Computer Society, 1993.

[44] M. O. Ward. XmdvTool: Integrating multiple methods for visualizing multivariate data. In *Proc. of the Conference on Visualization '94*, Vis '94, pages 326–333. IEEE Computer Society Press, 1994.

[45] J. Waser, R. Fuchs, H. Ribičić, B. Schindler, G. Blöschl, and E. Gröller. World lines. *IEEE Trans. on Visualization and Computer Graphics*, 16(6):1458–67, 2010.

[46] B. Weber, M. Möller, S. Prohaska, D. Günther, and H.-C. Hege. Fast tracing of microtubules in electron tomograms. Poster, 2011. Workshop on Visualizing Biological Data (VIZBI 2011), March 16-19, Cambridge, MA, USA.

[47] P. Zhao and B. Yu. Stagewise Lasso. *Journal of Machine Learning Research*, 8:2701–2726, 2007.

[48] Y. Zhu, B. Carragher, R. M. Glaeser, D. Fellmann, C. Bajaj, M. Bern, F. Mouche, F. de Haas, R. J. Hall, D. Kriegman, S. J. Ludtke, S. Mallick, P. A. Penczek, A. M. Roseman, F. J. Sigworth, N. Volkmann, and C. S. Potter. Automatic particle selection: Results of a comparative study. *Journal of Structural Biology*, 145(1-2):3–14, 01 2004.