Linking Advanced Visualization and MATLAB for the Analysis of 3D Gene Expression Data

Oliver Rübel[1,4,5], Soile V. E. Keränen[2], Mark Biggin[2], David W. Knowles[3], Gunther H. Weber[1,4], Hans Hagen[5], Bernd Hamann[1,4,5], and E. Wes Bethel[1]

[1] Computational Research Division, Lawrence Berkeley National Laboratory (LBNL), One Cyclotron Road, CA, 94720, USA. {oruebel, ghweber, ewbethel}@lbl.gov
[2] Genomics Division, LBNL, One Cyclotron Road, CA, 94720, USA. {mdbiggin, svekeranen}@lbl.gov
[3] Life Sciences Division, LBNL, One Cyclotron Road, CA, 94720, USA. {dwknowles}@lbl.gov
[4] Institute for Data Analysis and Visualization (IDAV), Department of Computer Science, University of California, Davis, One Shields Avenue, Davis, CA, 95616, USA. {bhamann}@ucdavis.edu
[5] International Research Training Group 1131, Technische Universität Kaiserslautern, Erwin Schrödinger-Straße, 67653, Kaiserslautern, Germany. {hagen}@informatik.uni-kl.de

Summary. Three-dimensional gene expression PointCloud data generated by the Berkeley *Drosophila* Transcription Network Project (BDTNP) provides quantitative information about the spatial and temporal expression of genes in early *Drosophila* embryos at cellular resolution. The BDTNP team visualizes and analyzes PointCloud data using the software application PointCloudXplore (PCX). To maximize the impact of novel, complex data sets, such as PointClouds, the data needs to be accessible to biologists and comprehensible to developers of analysis functions. We address this challenge by linking PCX and Matlab®[6] via a dedicated interface, thereby providing biologists seamless access to advanced data analysis functions and giving bioinformatics researchers the opportunity to integrate their analysis directly into the visualization application. To demonstrate the usefulness of this approach, we computationally model parts of the expression pattern of the gene *even skipped* using a genetic algorithm implemented in Matlab and integrated into PCX via our Matlab interface.

Key words: three-dimensional gene expression, integrating visualization and data analysis, Matlab integration, network modeling, genetic algorithm

[6] MATLAB is a registered trademark of The MathWorks Inc., 3 Apple Hill Drive Natick, MA 01760-2098, USA. Online at: http://www.mathworks.com/

1 Introduction

The BDTNP has developed a novel type of spatial and temporal gene expression data called *PointCloud*. Single PointClouds are obtained via segmentation of two-photon microscopy images of *Drosophila* embryos and provide a quantitative representation of spatial gene expression levels of the *Drosophila* blastoderm at cellular resolution [19]. Multiple PointClouds representing a variety of genes at multiple developmental time intervals are registered into a single *Atlas PointCloud* describing the expression of many genes at multiple points in time [7].

PointCloudXplore (PCX) is the standard tool of the BDTNP for visualization and analysis of PointCloud data [28]. PCX supports various 2D and 3D physical embryo views and abstract data visualizations and provides a suite of analysis tools, e.g., methods for cell selection and clustering [20]. Matlab —a commercial, high-level programming language and analysis environment— is widely used throughout the BDTNP, e.g., for PointCloud creation and embryo registration, and provides with its rich feature-set an ideal platform for development of custom analysis functionality. Despite its versatility, we anticipate that increasing numbers of researchers will benefit from being able to quickly develop and integrate custom analysis capabilities with PCX.

To address this challenge we developed an interface between PCX and Matlab, enabling researchers to easily integrate their analysis with the visualization and providing biologists faster and more convenient access to advanced analysis functions (Section 3). Via the PCX-Matlab interface, one can call functions implemented in Matlab directly from PCX, while PCX automatically handles all necessary communication, including, start/close of Matlab, data transfer to and from Matlab, and initiation of function calls. No Matlab knowledge is required to access Matlab functions via PCX. The interface also hides PCX's internal architecture from the Matlab developer, and very low effort is needed to make a Matlab function accessible to PCX. The interface supports fast prototyping and testing of new ideas and facilitates communication between bioinformatics researchers and experimental biologists.

We chose computational modeling of genetic regulatory networks as example to demonstrate the usefulness of the PCX-Matlab interface (Section 4). Expression regulatory models —e.g., by Janssens et al. [15] and Sanchez et al. [26]— often depend on extensive system-wide knowledge based on years of experimental work on mutants and transgenic constructs and specialized sets of equations and programs. With increasing number of components (i.e., genes), the number of potential interactions that need to be analyzed experimentally increases exponentially. Thus, computational methods are needed to identify probable candidate genes for experimental verification.

We describe a genetic algorithm for finding potential genetic regulatory interactions via optimization of a linear network model. We implemented this algorithm in Matlab and integrated it into PCX via our system interface. By integrating the modeling with the visualization, we can define the necessary

input to the analysis quicker and more accurately, and are able to effectively validate the input and output of the analysis. We discuss the modeling of the expression pattern of the gene *even skipped* (*eve*) to illustrate the advantages and disadvantages of the employed modeling approach.

2 Related Work

PCX uses the established concept of *linked multiple views* [2] for visualization of high-dimensional 3D gene expression data. In the WEAVE system, a combination of physical and information visualization views is used for exploration of cardiac simulation and measurement data [9]. Henze [12] developed a multiple-view-based system for exploration of time-varying computational fluid dynamics data. Advanced queries are supported by this system via selection of data subsets in each view (here termed portraits). The concept of using abstract views to define data queries was formalized by Doleisch et al. [6]. Interfaces to programming and analysis languages and systems are often used in the context of visualization to support scripting and to ease integration of the visualization with simulation [16] or data analysis, e.g., R and GGobi [18].

Someren et al. [27], De Jong [4], and D'haeseleer et al. [5] provide an overview of genetic network modeling. Janssens et al. [15] used simulated annealing to describe a predictive model of transcriptional control of the gene *even skipped* (*eve*) based on spatial gene expression, regulatory sequence data, and FlyEx 2D cellular resolution quantitative protein expression data of *Drosophila* embryos, which has also been used for modeling of *Drosophila* anterior-posterior gap gene system [14]. Fowlkes et al. [7] described first results on expression modeling based on the BDTNP quantitative 3D expression atlas using linear regression to identify for each target pattern six likeliest candidate regulators. In this work we describe the implementation of a genetic algorithm for finding genetic network models based on the BDTNP expression atlas. Genetic algorithms were surveyed by Whitley et al. [29] and Davis et al. [3]. The goal of this paper is not to develop a more sophisticated regulatory model but to provide a flexible environment to ease development of advanced analysis functions and to make them more accessible to the user.

3 The PointCloudXplore and Matlab Interface

The main goal of our research is to provide fast and easy access to new analysis functions for three-dimensional gene expression data, thus facilitating communication between biologists and bioinformatics researchers. To accomplish this goal, we added a Matlab interface (Figure 1) to PointCloudXplore (PCX), making PCX more easily extensible by bioinformatics researchers. From a biologist's perspective, this interface is transparent with respect to which functionality is implemented via Matlab functions. No Matlab knowledge is necessary to use these functions. From a bioinformatics researcher's

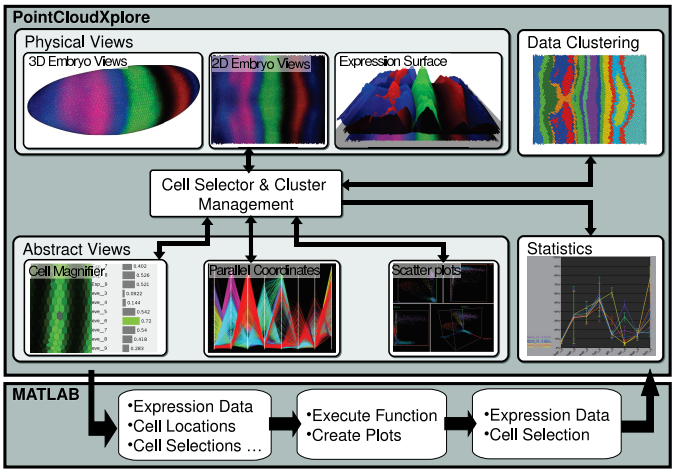


Fig. 1. Overview of PointCloudXplore and the interface to Matlab.

perspective, the interface hides the internal PCX architecture and requires minimal effort to make a Matlab function accessible to PCX.

The PCX-Matlab interface provides means to initiate Matlab function calls from PCX's graphical user interface (GUI), while automatically handling all necessary inter-system communication. Data classes exported from PCX to Matlab include multiple user-defined lists of gene expressions and/or cell selections and their names, 2D/3D cell locations, cell neighbors, and additional user-definable function parameters (scalar double, Boolean, integer, or string). Information imported from Matlab to PCX includes multiple derived gene expression channels and/or cell selections including optional names.

In the following we describe the PCX-Matlab interface (Section 3.1), discuss the interface from a biologist's and bioinformatics researcher's view (Section 3.2), and review various example applications (Section 3.3).

3.1 Interface Design

The definition of a Matlab function for PCX consists of an *M*-file with the function implementation and a *PCXM* header file, describing the function for PCX (Figure 3). This header file must specify:

- a function category (TYPE) used to group functions in the menu,
- a name for the function (NAME) to appear in the menu,
- the location of the *M*-file, if different from the *PCXM* file (DIR), and
- the Matlab command to invoke the function (CALL).

The TYPE and NAME parameter may define an arbitrary string, allowing the user to group and name functions according to personal preferences.

Using strict name conventions for function call input and output parameters, PCX can automatically identify what data needs be transferred to and

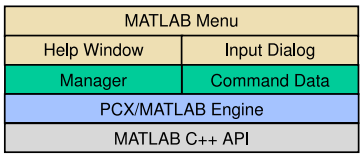


Fig. 2. Design of the interface between PointCloudXplore and Matlab.

from Matlab — e.g., gene expressions, cell locations and neighbors, cell selections, and user-definable double, integer, Boolean, or string parameters— based on the function call itself. All data transfers are implemented via a series of function calls to the Matlab C++ API, allowing us to directly transfer vector/matrix data of varying type between the two systems.

For some applications it makes sense to pass input genes and names in multiple groups. For each list of genes or cell selections, one may optionally specify the expected size of each list (IN_GENES, IN_BRUSHES). Furthermore, one can define an optional URL for a HTML help page (HELP_URL). Thus, a PCXM header file provides PCX with information about: i) what data needs to be exchanged between Matlab and PCX, ii) what parameters need to be exposed to the user, iii) how the Matlab function is invoked, and iv) how the function and its parameters should be represented within the GUI.

The PCX-Matlab interface consists of three main components: a communication layer (blue), a function management layer (green), and the GUI (tan) (see Figure 2). The communication layer is based on Matlab's C++ API (gray) and provides higher-level functions for inter-system communication.

The function management layer manages all Matlab functions available to PCX. The *Manager* class parses the PCXM files, stores and provides access to information about the Matlab functions, and initiates the execution of Matlab functions. For each Matlab function, the Manager creates a *CommandData* object containing all information about the corresponding function.

The GUI portion of PCX' Matlab interface consists of: the *Matlab Menu*, a parameter *Input Dialog*, and the *Help Window*. The Matlab Menu is part of PCX' main menu bar and lists all available Matlab functions (NAME) grouped by their logical category (TYPE). Furthermore, the Matlab Menu provides access to the Help Window and various management options, such as updating the function list. Using the Help Window, one can conveniently access the HTML help pages of all Matlab functions. When selecting a Matlab function in the Matlab Menu, PCX dynamically creates an Input Dialog at runtime listing all user-definable input parameters of the corresponding function.

The basic work flow of a use-case involving both PCX and Matlab is as follows (Figure 1): Upon start-up PCX parses all available PCXM files and initializes the Matlab Menu and Help Window. To initiate the execution of a Matlab function, one selects the desired function in the Matlab Menu. Subsequently, PCX creates an Input Dialog for the function and starts Matlab if it is not already running. After one has specified and confirmed the function

input in the Input Dialog, PCX transfers all the necessary data to Matlab and invokes the Matlab function. Finally, PCX imports any output expression channels or cell selections, once the Matlab function has terminated.

3.2 Perspective of the Biologist and Bioinformatics Researcher

When calling a Matlab function via PCX, the user interacts only directly with PCX. PCX automatically handles all communication with Matlab. In terms of access to the function it is irrelevant from a biologist's perspective whether a function is implemented in Matlab or directly in PCX. However, unlike Matlab, the interface of PCX is much more user-friendly and does not require any command-line input. To simplify repeated analysis and test-runs, PCX supports saving and restoring gene input lists, and the integration of Matlab and PCX enables one to refine input parameters quickly between individual Matlab command executions by making it possible to quickly investigate input and output expression patterns. The interface between Matlab and PCX provides the user faster and more convenient access to advanced analysis functions.

From a bioinformatics researcher's perspective, the effort needed to make an existing Matlab function (defined in an M file) accessible in PCX is very low and limited to creating a simple PCXM header file. No recompilation of source code or knowledge of PCX design or implementation details is required. Matlab functions can be updated at runtime of PCX, making development and debugging more convenient and less time-consuming. In addition, PCX provides a convenient GUI for the Matlab functions and parses PointCloud data, i.e., a developer does not need to know how to read PointCloud data. PCX's advanced data display and selection mechanisms in combination with the simplicity of the Matlab interface simplify the development of usable scripts for processing PointCloud data. The PCX-Matlab interface provides an easy way to make new analysis functions quickly available to biologists. The interface supports fast prototyping and testing of new ideas and facilitates communication between bioinformatics researchers and experimental biologists.

Figure 3 shows a simple example function that computes the cell-by-cell difference between two expression patterns, illustrating the effort required for deploying a Matlab function in PCX. The function takes two genes and their names as input and returns a new expression channel and its name as output. The PCXM header file specifies the TYPE, NAME, and function call to be issued (CALL). In addition, the developer defined a HTML help page (HELP_URL) and specified that both input gene lists should contain only a single gene (IN_GENES). Difference patterns are useful to analyze, e.g., the temporal variation of an expression pattern (Figure 3, right side) or compare the pattern of different genes. In our example, we consider a PointCloud atlas file containing late (76-100%) and early (0-3%) blastoderm *hunchback* (*hb*) patterns. By applying the difference function to these genes, we can explore the temporal expression pattern change. As this example illustrates, the Matlab interface allows us to easily and quickly extend PCX with additional features.

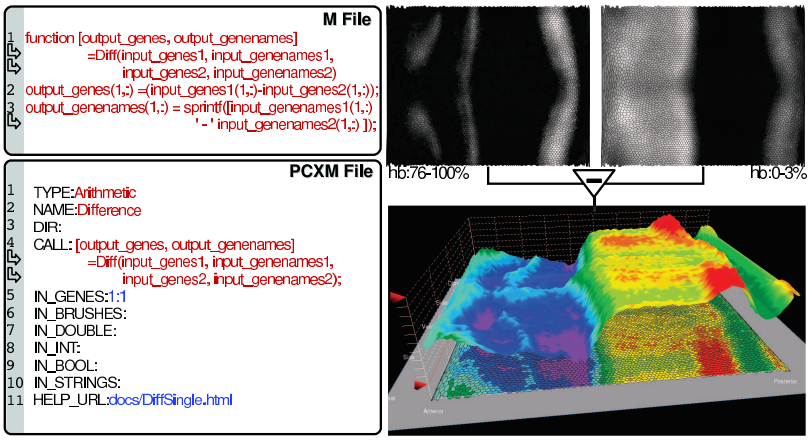


Fig. 3. M-file (top left) and PCXM header file (bottom left) of an example function for the computation of the cell-by-cell difference between two expression patterns. Color of text is used to illustrate which parts are mandatory (red), optional (blue), or provided in a template file (black). Right: Example showing the use of the function to compute the difference between late- and early-stage *hunchback* expression.

3.3 Applications

The combination of PCX and Matlab has a wide range of applications ranging from plotting and cell selection to expression filtering.

Often, one needs to quickly create a specific plot, e.g., for a presentation. Matlab provides a rich set of plots that can be deployed quickly via PCX' interface to Matlab. Plots created in Matlab are displayed in an external Matlab plot window, supporting direct interaction with the plot. We implemented, e.g., a line-out plot in Matlab for comparison of the expression profiles of multiple genes along a line defined on the embryo surface (not shown).

A second application of PCX' Matlab interface is cell selection. This category of applications includes, e.g., functions for importing/exporting cell selections, segmenting gene expression patterns [13], clustering cells [20], and detecting features.

Expression filtering functions select expression patterns or create new derived "expression" patterns. Simple filtering functions include functions for expression data import or export, arithmetic functions for computing the cell-by-cell difference between expression patterns ($c = a - b$) or the cooperative pattern of genes (e.g., $c = a * b$). Other applications of expression filtering include smoothing and post-processing of expression patterns, masking of expression patterns, or dimensions reduction (e.g., using principal component analysis). Advanced analysis functions, such as the gene expression modeling described in Section 4, often bridge various of the application categories mentioned above.

4 A Predictive Model of Expression Control

The BDTNP gene expression atlas provides us with information about the spatio-temporal expression pattern of genes, i.e., the input and output of the complex genetic networks responsible for the regulation of gene expression patterns. Based on this information one can attempt to infer potential regulatory interactions leading to the formation of a selected target pattern.

To explore and better understand gene pattern formation we first define a basic model for a genetic regulatory network. We use a linear model, i.e., the output of a model network is defined through a linear combination of its weighted inputs (Section 4.1). Given the input, output, and basic network model, we need to find a specific network model that defines the target well. To find a good network model we use a genetic algorithm [29, 3], which we implemented using Matlab (Section 4.2).

In the described network model, the output directly depends on the input. Validation and understanding of the input patterns is, therefore, crucial for the interpretation of a predicted model. Furthermore, we need to be able to compare the output of a network with the target pattern. PCX supports validation and comparison of expression patterns via a combination of physical and abstract visualizations and statistical analysis tools, making PCX ideal for these tasks. Using the PCX-Matlab interface we integrated the Matlab optimization function with PCX making the analysis easily accessible for the target user (Section 4.3).

In Section 4.4 we describe the example application of our system to model the expression pattern of the gene *eve*. It is beyond the scope of this paper to provide a conclusive model for the *eve* regulation. The goal of the research covered here is rather to investigate the advantages and disadvantages of the employed direct network modeling approach.

4.1 Expression Model

Next, we describe a continuous, linear expression model. Based on the expression values of the measured expression patterns of a set of input regulators $e_{regs} = \{e_1, e_2, ..., e_k\}$ we define the model expression of the cell c as:

$$v(w, e_{regs}, c) = \sum_{i=0}^{k} w_i e_i(c), \tag{1}$$

with $w = \{w_1, w_2, ..., w_k\} \in [-1, 1]^k$ being the weights of the different regulators e_{regs}. A weight of $+1$ indicates strong activation and a weight of -1 indicates strong inhibition of expression. In practice, a gene can only show positive expression, i.e., further inhibition of a non-expressed gene has no effect on the output. To account for this behavior, we define the model expression of a cell c as:

$$m(w, e_{regs}, c) = max(v(w, e_{regs}, c), 0). \quad (2)$$

The expression pattern defined by a network model w is defined as:

$$m(w, e_{regs}) = \{m(w, e, 1), m(w, e, 2), ..., m(w, e, n)\}. \quad (3)$$

A gene can also be ubiquitously expressed at some basic level even if no spatially regulated activator is present. To account for this behavior, a user can include a constant expression $e_{const} = 1$ as input factor with the corresponding weight $w_{e_{const}}$ defining the degree of basal expression. By including cooperative patterns — e.g., of the form $e_{ij} = e_i e_j$ — as input regulators, a user can also account for simple non-linear effects of cooperative regulation, i.e., two genes acting in cooperation to form a pattern.

Expression modeling — using binary as well as continuous models — has proven to be a powerful tool for the study of genetic regulatory networks [27, 4, 26]. The main limitation of linear expression models, such as the one described here, is that they are not able to account for non-linear regulatory effects — such as heterodimerization of two transcription factors for a specific regulatory effect [22] — and may not be able to predict all targets correctly.

4.2 Optimization Algorithm

The goal of the optimization is to find a model w for which $m(w, e)$ fits the target expression pattern e_{target} well. To define the similarity between two expression patterns e_i and e_j we use the correlation measure

$$corr(e_i, e_j) = \frac{\sum_{c=0}^{n}(e_i(c) - \overline{e_i})(e_j(c) - \overline{e_j})}{(n-1)s_{e_i}s_{e_j}} \quad (4)$$

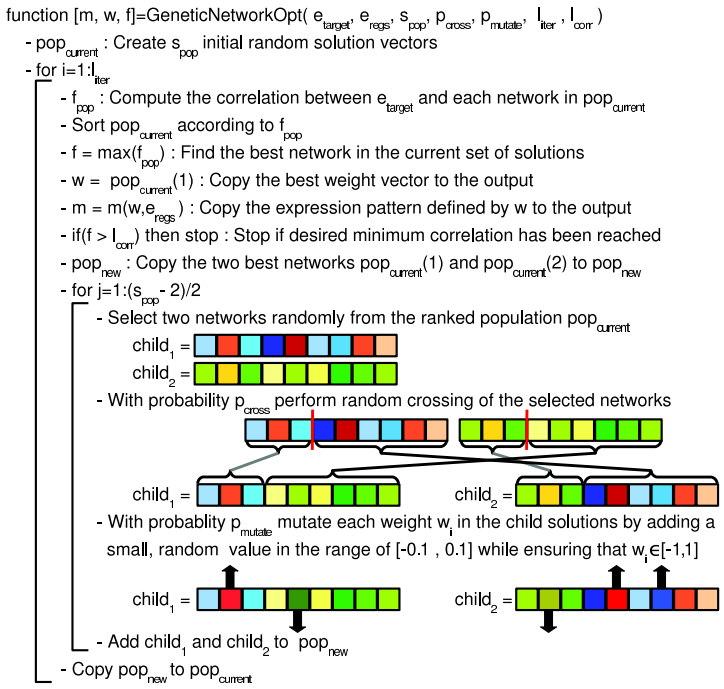
as distance function, with s_{e_i} and s_{e_j} being the sample standard deviation of e_i and e_j, respectively. The correlation is a scale-invariant measure — i.e., $corr(e_{target}, m(w, e_{regs})) = corr(e_{target}, t * m(w, e))\ \forall\ t > 0$ — allowing us to find models that capture the relative structure of the target pattern rather than its absolute expression. The correlation measure further has the advantage that it is easy to compute. The main disadvantage of the correlation measure is that it describes only the global fitness of a possible solution but does not consider the local structure of expression patterns in physical space.

The optimization problem we have to solve is:

$$\max_{w \in [-1,1]^k} (corr(e_{target}, m(w, e_{regs})). \quad (5)$$

The goal of the optimization is to find a weight vector w so that the correlation between e_{target} and $m(w, e_{regs})$ is maximal.

Finding the optimal weight vector w is an NP-hard problem. To find a possibly near-optimal solution we use the following genetic algorithm:



The algorithm is based on the canonical design of a genetic algorithm [29] and finds a single network w that defines the model expression pattern m with correlation f to the target pattern e_{target}. Besides the target pattern e_{target} and the input regulators e_{regs}, the algorithm has the following additional parameters: i) s_{pop}, size of the population of possible solutions, ii) p_{cross}, probability of crossing between two selected solutions and iii) p_{mutate}, probability of mutating a particular entry w_i of a solution vector by a small random value. The parameters l_{iter} and l_{corr} serve as termination conditions defining the maximum number of iterations and the desired correlation score, respectively. Optionally, we also allow the user to specify: i) whether the two best solutions should always be preserved, ii) whether mutations should only be accepted if they improve the solution, iii) whether the population should be initialized randomly or with all zero values, and iv) which types of plots should be shown during the optimization procedure.

Genetic regulatory networks are highly dynamic and often contain redundancies, i.e., in practice various good solutions may exists to the same problem. We repeat the optimization several times to investigate the diversity of good solutions and to increase the probability of finding a near-optimal solution.

4.3 PointCloudXplore and Matlab for Network Modeling

We implemented the described optimization algorithm in Matlab and integrated the function with PCX via our Matlab interface. Figure 4 illustrates

the design of a network modeling experiment using PCX and Matlab. Using the visualization and analysis capabilities of PCX, the user first identifies the target pattern of interest and a set of potential regulators, allowing the user to effectively validate the input expression patterns. A gene is often expressed in multiple distinct spatial domains — e.g., in seven stripes in the case of *eve* — which may be regulated by different *cis-regulatory elements* (CREs). Using PCX a user can select the different expression domains of a gene. These cell selections can then be used to mask the expression pattern of the target to simulate the corresponding CRE pattern. Alternatively, one could use the cell masks directly in the optimization to compute the fitness of the model only in the area of interest to find a local model. Finally, the user is asked to specify the input parameters of the optimization algorithm, i.e., s_{pop}, p_{cross}, p_{mutate}, l_{iter} or l_{corr}, in the Input Dialog created automatically by PCX.

After the user has confirmed the input, PCX automatically starts Matlab, transfers the input data to Matlab, and initiates the execution of the optimization function. During the optimization process the user is presented a set of plots visualizing the progress of the analysis. After completion of the optimization process, PCX imports the model expression pattern(s) $m(w, e_{regs})$, the relevant data is saved in a user-defined *.mat* Matlab state file, and a set of plots are shown to provide an overview of the results produced.

As this example illustrates, the close integration of PCX and Matlab allows advanced analysis functions to be deployed quickly to the biology user community while improving usability and accuracy of the analysis.

4.4 Modeling the *eve* Expression Pattern

In the following we describe the application of these methods to the modeling of the expression pattern of the gene *even skipped* (*eve*). The *eve* pattern is characterized by seven stripes, which are usually presumed to result from the action of five CREs. While some stripes appear to have their own CRE, other stripes may share a module, and stripe 7 (the posteriormost stripe) has been speculated to be under redundant regulation. We here focus on modeling stripe 2 and 7 of the *eve* pattern. The goal of this experiment is to demonstrate the strengths and weaknesses of expression-based network modeling.

Figure 4 illustrates the basic setup of the experiment. In order to identify how the pattern of *eve* is initially formed we use the *eve* pattern at stage 5:9-25% as target, while applying selective masks to isolate individual stripes of the pattern. For the potential regulators (see Figure 4) we used protein patterns at stage 5:4-8% because of the transcriptional delay between the regulator input and the output patterns. In cases where only mRNA expression data was available, we selected the patterns from stage 5:0-3% because of the translational delay from mRNA into a protein.

Figure 5 summarizes the results for the modeling of *eve* stripe 2. The algorithm consistently found similarly high-scoring models with $corr > 97\%$. The

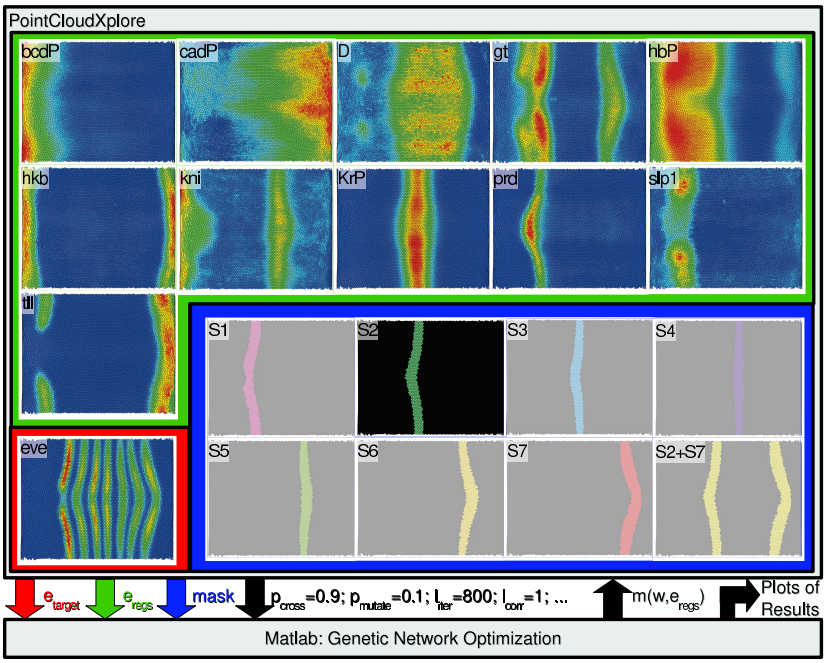


Fig. 4. Overview of the design of a network modeling experiment. Using PCX the user specifies the necessary input of the optimization function, e.g.: (i) the set of input regulators e_{regs} (green box), (ii) the target e_{target} (red box), (iii) an optional mask specifying the area of interest (blue box), and (iv) additional input parameters, such as p_{cross}. After the user has confirmed all settings, PCX sends all data to Matlab, calls the optimization function, and afterwards imports the model expression pattern $m(w, e_{regs})$. The optimization function also creates a set of plots with an overview of the analysis results.

expression pattern of the predicted model fits the target well and even resembles the D/V variation of the target pattern. Commonly, the input regulators used are only considered as regulators along the A/P body-axis. The shown results suggest that the input regulators are also able to function as regulators along the D/V body-axis (Figure 5d). These results are consistent with earlier modeling results based on data clustering [20]. The modeling suggests that *D*, hb, and possibly *prd* function as main activators of *eve* stripe 2 while *slp1*, Kr and gt function as inhibitors. The absolute weights of the remaining input regulators (i.e., *kni*, bcd, *hkb*, *tll*, and cad) are much lower, indicating that the function of these factors is not well characterized by the analysis.

Experimental data on *eve*-stripe regulation has shown that *eve* stripe 2 is activated by bcd [25] and hb [25] and potentially by D [21], and inhibited by gt [25], Kr [25] and weakly by slp1/2 [1]. prd is known to be able to activate late *eve* stripe expression via late enhancers [8], but is not essential for *eve* stripe 2 early upregulation. Notably, the modeling is able to find similarly

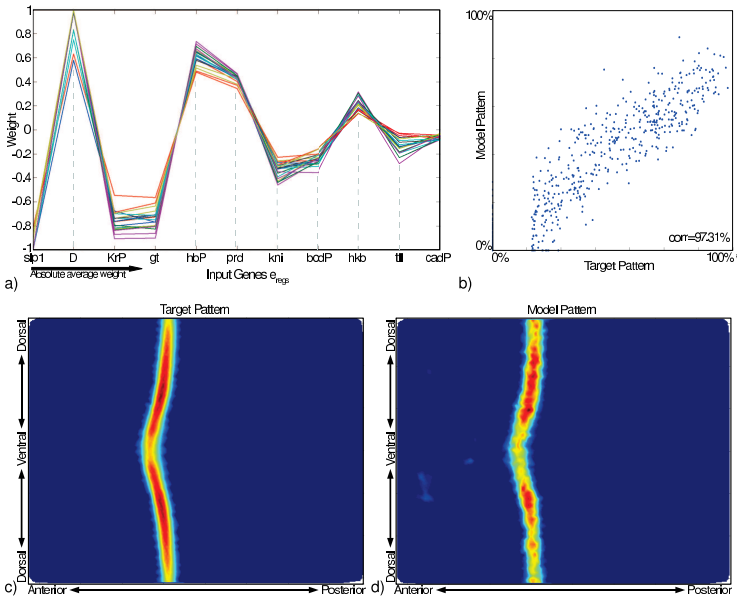


Fig. 5. Overview of the modeling results for *eve* stripe 2. We repeated the genetic algorithm 20 times while figures b-d show the model with the highest correlation. All experiments achieved a correlation of $> 97.x\%$. a) Plot of the 20 models with each curve representing one model. The genes are ranked according to their absolute average weight. b) Scatter plot of the target pattern and the best model pattern. c,d) Unrolled view of the target and the model pattern, respectively. Color shows relative expression with blue=low and red=high expression. We can see that the model fits the target well and even resembles the D/V variation of the target stripe.

high-scoring models even when removing *prd* from the input, indicating that *prd* may not be essential for defining *eve* stripe 2 (not shown). The reason why bcd was predicted as a weak inhibitor instead of an activator may be because (i) the highest bcd levels appear in a relatively wide gradient in the anterior of the embryo where *eve* stripe 2 is not located, and/or because (ii) bcd activates also inhibitors of *eve* stripe 2, such as slp [10] and gt [17].

We also modeled *eve* stripe 7 using the same set of input regulators (Figure 6a-c). The models we computed for stripe 7 score in general lower than the stripe 2 models ($corr \approx 88\%$) most likely due to a strong bias resulting in lower "expression" on one side of the embryo. The input regulator cad — which functions as a main activator in the stripe 7 model — shows a similar bias due to measurement or data normalization errors, explaining the observed behavior. The boundaries of *eve* stripe 7 are, however, well-defined in the generated model. The model suggests *D*, cad, and *prd* as main activators and gt, Kr, *kni*, *hkb*, and *tll* as main inhibitors of *eve* stripe 7.

It is commonly presumed that *eve* stripe 7 arises from the action of *eve* stripe 3+7 CRE. With the standard *eve* 3+7 CRE, inhibition by kni has been

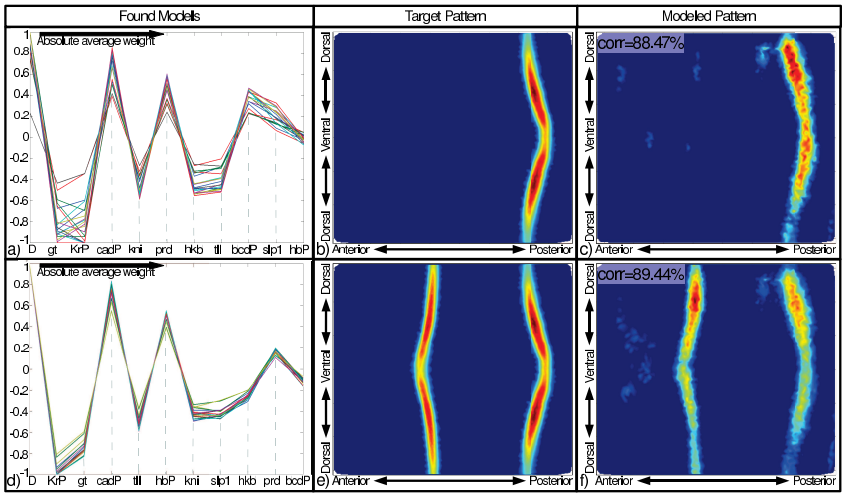


Fig. 6. Overview of the modeling results for *eve* stripes 7 (a-c) and 2+7 (d-f). We repeated the genetic algorithm 20 times while figure c and f show the result for the best model for stripe 7 and 2+7, respectively. All runs achieved a correlation of 87.x% for stripe 7 alone and $>$ 88.9% for stripe 2+7. a,d) Curve plot of the 20 models. b,c,e,f) Unrolled view of the target patterns and the respective best model pattern. Color shows relative expression with blue=low and red=high expression.

experimentally verified, but mutation in tll actually abolished stripe 7 [24], and the demonstrated clear negative *hb* regulation was not found. Moreover, mutations in gt, Kr, and hkb have been reported not to affect the output of the 3+7 stripe element [24], though gt and Kr have been here predicted to be strong inhibitors. Likewise, experimental data does not support D as an activator of stripe 7 [21], and prd is not essential for early *eve*-stripes 7 upregulation [8]. However, Yan et al. [30] predicted *eve* stripe 7 to be activated by *Stat92E*, which was not available in the input dataset. The differences between *eve* stripe 3+7 CRE and other experimental data and our results suggest that modeling may be able to closely mimic the target pattern via a different network, even when important regulators are missing in the input.

While *eve* stripe 7 has usually been described as part of an output of the *eve* stripes 3+7 CRE, Janssens et al. [15] have proposed that *eve* stripe 2 CRE might also participate in regulation of *eve* stripe 7. The *eve* stripe 2 element is known to often produce a weak *eve* stripe 7 expression and the same is observed in sepsid *eve* stripe 2 enhancers [23, 15, 11], while the stripe 7 produced by *eve* 3+7 CRE can be weaker than stripe 3 [24]. Therefore, we modeled *eve* stripe 2 and 7 at the same time (Figure 6d-f). Qualitatively the model for *eve* stripe 7 does not change significantly while *eve* stripe 2 shows the same D/V bias as stripe 7, probably due to the higher-weighted *cad* pattern. The boundaries of *eve* stripe 2 are, however, still well-defined. The Janssens et al. [15] model supports our prediction of cad as an activator and gt and tll

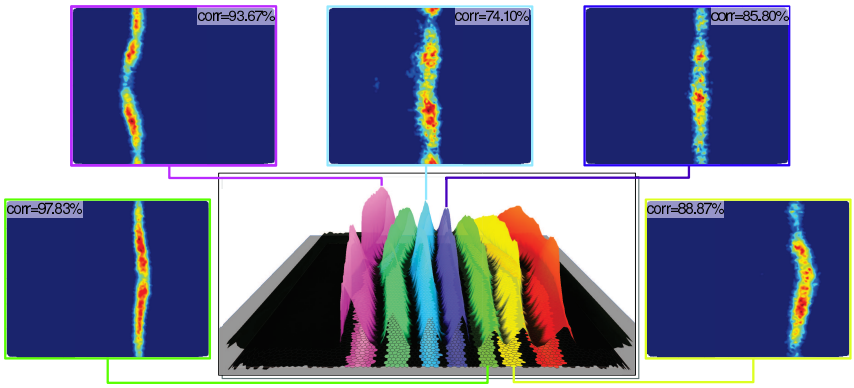


Fig. 7. Overview of modeling results for eve stripes 1, 3, 4, 5, and 6.

as repressors of *eve* stripe 7(+2) (Figure 6a,d). These results suggest that *eve* stripes 2 and 7 could, indeed, be formed by a common network model and may be co-regulated. However, our model does not account for potential regulation by multiple or diffuse CREs, which may cause distortion due to too high a contribution of stripe 7 to a particular CRE model. To eliminate potential problems due to interference between CREs one would ideally measure the actual output of individual CREs, e.g., from transgenic constructs.

Figure 7 shows preliminary results for the modeling of *eve* stripes 1, 3, 4, 5, and 6. For *eve* stripe 1 and 5 we found high-scoring models ($corr > 93\%$) which — similar to the *eve* stripe 2 model — also reproduce the main variations in expression along the D/V axis. While the models for *eve* stripes 3 and 4 show expression in the expected location, the stripe-borders are not well-defined. Interestingly the algorithm is able to model stripe 3 well when including the cooperative factor $e_{hbP,KrP} = e_{hbP}e_{KrP}$ as input regulator, achieving much higher scores of $corr \approx 90\%$. While cooperative regulation of stripe 3 by *Kr* and *hb* may not be real [24], this behavior nevertheless indicates that — besides the fact that important input regulators may be missing — non-linear regulatory effects not captured by the used linear model may be responsible for the correct formation of at least *eve* stripe 3.

The fact that the modeling was able to predict a large range of regulators correctly for *eve* stripe 2 shows that modeling can provide interesting insights into, or at least hints of, possible regulatory interactions. Missing regulators (see stripe 7), noise (see stripes 7 and 2+7), and limitations of the employed computational model (see stripe 3), however, directly affect the quality of the predicted model and may also lead to false negatives (missing regulators) and false positives (misidentified regulators) in predictions (see stripe 7). Modeling results should, therefore, always be validated experimentally.

5 Conclusions

To fully exploit the collaborative research potential of teams of biologists, computational biologists, and computer scientists, it is essential to overcome true and perceived obstacles for collaboration. Biologists rarely do computation and computer scientists rarely do biology. To maximize the impact of novel, complex, high-dimensional data sets acquired via modern imaging or computational methods — such as the BDTNP 3D gene expression atlas data — the data needs to be accessible to biologists and comprehensible to developers of analysis and visualization software.

We addressed these challenges by linking the visualization system PointCloudXplore (PCX) and Matlab via a dedicated interface, providing biologists seamless access to advanced data analysis functions and enabling bioinformatics researchers to integrate their analysis directly into the visualization. By being able to test new analysis functions during development, biologists are able to provide feedback early, facilitating communication between the developer and the user. By utilizing PCX and Matlab, a developer can develop new functions more efficiently without having to know anything about the PointCloud data format or the architecture of PCX.

One potential focus for future research are methods for linking other types of data, such as *in vitro* and *in vivo* binding data, with the gene expression data analysis. Being able to incorporate different types of data is essential, e.g., for the development of advanced predictive models of gene expression.

6 Acknowledgments

This work was supported by the National Institutes of Health through grant GM70444 and by Lawrence Berkeley National Laboratory (LBNL) through the Laboratory Directed Research Development (LDRD) program. Research performed at LBNL was also supported by the Department of Energy under contract DE-AC02-05CH11231. In addition, we gratefully acknowledge the support of an International Research Training Group (IRTG 1131) grant provided by the German Research Foundation (DFG), awarded to the University of Kaiserslautern, Germany. We thank the members of IDAV at UC Davis, the BDTNP at LBNL, the IRTG, and LBNL's Visualization Group.

References

1. L. P. Andrioli, V. Vasisht, E. Theodosopoulou, A. Oberstein, and S. Small. Anterior repression of a *Drosophila* stripe enhancer requires three position-specific mechanisms. *Development*, 129(21):4931–4940, November 2002.
2. M. Q. W. Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 110–119, New York, NY, USA, 2000. ACM Press.

3. L. D. Davis and M. Mitchell. Handbook of genetic algorithms. *Van Nostrand Reinhold*, 1991.
4. H. De Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.
5. P. D'haeseleer, S. Liang, and R. Somogyi. Genetic network inference: from co-expression clustering to reverse engineeing. *Bioinformatics*, 16(8):707–726, 2000.
6. H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complexsimulation data. In G.-P. Bonneau, S. Hahmann, and C. D. Hansen, editors, *Data Visualization 2003 (Proceedings of the Eurographics/IEEE TCVG Symposimon Visualization)*, 2003.
7. C. C. Fowlkes, C. L. L. Hendriks, S. V. E. Keränen, G. H. Weber, O. Rübel, M.-Y. Huang, S. Chatoor, A. H. DePace, L. Simirenko, C. Henriquez, A. Beaton, R. Weiszmann, S. Celniker, B. Hamann, D. W. Knowles, M. D. Biggin, M. B. Eisen, and J. Malik. A quantitative spatiotemporal atlas of gene expression in the *Drosophila* blastoderm. *Cell*, 133:364–374, April 2008.
8. M. Fujioka, P. Miskiewicz, L. Raj, A. A. Gulledge, M. Weir, and T. Goto. *Drosophila* paired regulates late *even-skipped* expression through a composite binding site for the paired domain and the homeodomain. *Development*, 122(9):2697–2707, Sept. 1996.
9. D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung. WEAVE: A system for visually linking 3-d and statistical visualizations,applied to cardiac simulation and measurement data. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings IEEE Visualization 2000*, pages 489–492, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.
10. U. Grossniklaus, K. M. Cadigan, and W. J. Gehring. Three maternal coordinate systems cooperate in the patterning of the *Drosophila* head. *Development*, 120(11):3155–3171, 1994.
11. E. E. Hare, B. K. Perterson, V. N. Iyer, R. Meier, and M. B. Eisen. Sepsid *even-skipped* enhancers are functionally conserved in *Drosophila* despite lack of sequence conservation. *PLoS Genetics*, 4(6):e1000106., 2008.
12. C. Henze. Feature detection in linked derived spaces. In D. Ebert, H. Rushmeier, and H. Hagen, editors, *Proceedings IEEE Visualization '98*, pages 87–94, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.
13. M.-Y. Huang, O. Rübel, G. H. Weber, C. L. Luengo Hendriks, M. D. Biggin, H. Hagen, and B. Hamann. *Segmenting Gene Expression Patterns of Early-stage Drosophila Embryos*, pages 313–327. Springer Verlag, Germany, Jan 2008.
14. J. Jaeger, M. Blagov, D. Kosman, K. N. Kozlov, Manu, E. Myasnikova, S. Surkova, C. E. Vanario-Alonso, M. Samsonova, D. H. Sharp, and J. Reinitz. Dynamical analysis of regulatory interactions in the Gap gene system of *Drosophila melanogaster*. *Genetics*, 167(4):1721–1757, April 2004.
15. H. Janssens, S. Hou, J. Jaeger, A.-R. Kim, E. Myasnikova, D. Sharp, and J. Reinitz. Quantitative and predictive model of transcriptional control of the *Drosophila melanogaster even skipped gene*. *Nature Genetics*, 38, Sept. 2006.
16. C. Johnson, S. G. Parker, C. Hansen, G. L. Kindlmann, and Y. Livnat. Interactive simulation and visualization. *Computer*, 32(12):59–65, December 1999.
17. R. Kraut and M. Levine. Spatial regulation of the gap gene *giant* during *Drosophila* development. *Development*, 111(2):601–609, 1991.

18. D. T. Lang and D. F. Swayne. GGobi meets R: an extensible environment for interactive dynamic data visualization. In *Proceedings of the 2nd International Workshop on Distributed Statistical Computing*, Mar. 2001.
19. C. L. Luengo Hendriks, S. V. E. Keränen, C. C. Fowlkes, L. Simirenko, G. H. Weber, A. H. DePace, C. Henriquez, D. W. Kaszuba, B. Hamann, M. B. Eisen, J. Malik, D. Sudar, M. D. Biggin, and D. W. Knowles. Three-dimensional morphology and gene expression in the *Drosophila* blastoderm at cellular resolution I: Data acquisition pipeline. *Genome Biology*, 7(12):R123, 2006.
20. O. Rübel, G. H. Weber, M.-Y. Huang, E. W. Bethel, M. D. Biggin, C. C. Fowlkes, C. L. Hendriks, S. V. E. Keränen, M. B. Eisen, D. W. Knowles, J. Malik, H. Hagen, and B. Hamann. Integrating data clustering and visualization for the analysis of 3d gene expression data. *IEEE Transactions on Computational Biology and Bioinformatics*, 2008.
21. S. R. Russell, N. Sanchez-Soriano, C. Wright, and M. Ashburner. The *Dichaete* gene of *Drosophila melanogaster* encodes a SOX-domain protein required for embryonic segmentation. *Development*, 122(11):3669–3676, 1996.
22. F. Sauer and H. Jäckle. Heterodimeric *Drosophila* gap gene protein complexes acting as transcriptional repressors. *EMBO J*, 14(19):4773–4780, 1995.
23. S. Small, A. Blair, and M. Levine. Regulation of *even-skipped* stripe 2 in the *Drosophila* embryo. *The EMBO journal*, 11(11):4047–4057, Nov. 1992.
24. S. Small, A. Blair, and M. Levine. Regulation of two pair-rule stripes by a single enhancer in the *Drosophila* embryo. *Developmental Biology*, 175(2):314–324, May 1996.
25. S. Small, R. Kraut, T. Hoey, R. Warrior, and M. Levine. Transcriptional regulation of a pair-rule stripe in *Drosophila*. *Genes Dev*, 5(5):827–839, May 1991.
26. D. Thieffry and L. Sanchez. Dynamical modeling of pattern formation during embryonic development. *Current opinion in genetics & development*, 13(4):326–330, 2003.
27. E. P. van Someren, L. F. A. Wessels, E. Backer, and M. J. T. Reinders. Genetic network modeling. *Pharmacogenomics*, 3(4):507–525, 2002.
28. G. H. Weber, O. Rübel, M.-Y. Huang, A. DePace, C. C. Fowlkes, S. V. Keränen, C. L. L. Hendriks, H. Hagen, D. W. K. J. Malik, M. D. Biggin, and B. Hamann. Visual exploration of three-dimensional gene expression using physical views and linked abstract views. *IEEE Transactions on Computational Biology and Bioinformatics*, 6(2):296–309, April-June 2009.
29. D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, June 1994.
30. R. Yan, S. Small, C. Desplan, C. R. Dearolf, and J. E. Darnell Jr. Identification of a *Stat* gene that functions in *Drosophila* development. *Cell*, 84(3):421–430, Feb 1996.