Pacific Symposium on Biocomputing 5:203-214 (2000)

MAPPETSHOW: NON-LINEAR VISUALIZATION FOR GENOME DATA

F. GUYON
GIS Infobiogen
7, rue Guy Môquet – BP 8
94801 Villejuif cedex, France

G. VAYSSEIX, E. BARILLOT
GIS Infobiogen

Généthon
1bis, rue de l'Internationale
91000 Évry, France

The genome mapping projects now produce very dense maps with up to several thousands of markers per chromosome. Besides synteny plays a increasing role in mapping: enrichment of poor maps from the maps of close genomes (in terms of evolution) is a high-reward task. We propose a map viewer adapted to this situation: MappetShow gives a clear view of very dense maps and compares efficiently several maps. MappetShow is based on non-linear viewing and is written in Java™. A map description language isolates the software from the data sources. This software was easily used on data coming from as different sources as an Object Request Broker, an Object-Oriented Database, or a flat data stream. MappetShow can be browsed at the URL http://www.infobiogen.fr/services/Mappet. More generally we discuss how to use the non-linear viewing concept in molecular biology data visualization.

1 Introduction

Since the beginning of the human genome mapping project, a lot of softwares have been developed to visualize the genome maps. In the first phase of the mapping projects, the genome viewers were generally developed by the genome centers. These early attempts were mainly written in C or C++ and used X-window libraries[1,2,3]. More recently, the data produced by the genome centers have been widespread among the scientific community, and very logically new genome viewers have been developed that are based on a distributed object technology: the most popular viewers are now written in Java™ [4]. They include for example Anubis[5], Genome Navigator[6,7], Jade[8], the Cyanobase Map viewer[9], Mapview2[10], ZoomMap[11] and the Oxford Molecular/Glaxo Wellcome interface[12].

Recently, there has been an important modification in the strategy of the mapping community. First, the Radiation Hybrid Consortium[13], the White-

head-MIT Center for Genome Research[14] and Généthon[15] released very dense maps with several hundreds of markers per chromosome; also single nucleotide polymorphism maps are arriving that will contain several dozens if not hundreds of thousands of polymorphic sites. Secondly and as a consequence of the availability of these dense maps, research groups now use more and more the high level of synteny, for example between mammalians, to enrich the map of map-poor genomes (such as porcine, bovine or ovine genomes) from the map-rich genomes such as those of Mus Musculus or Homo Sapiens Sapiens. This new situation poses new problems of visualization, for example: how to visualize in an intelligible way maps of several hundreds if not thousands of markers? How to navigate in such maps? How to compare two or more maps in a readable way? How to cope with the comparison of maps with very different densities of markers?

Most of the genome map viewers do not address satisfactorily all the points listed above. In some cases, the dense map representation is unreadable because the marker names overlap each other in the region the user is visualizing. In other cases, the comparison of two maps is unintelligible because when zooming in on a given region of one map, the corresponding region of the other map is shifted out of the visualization window. Some map viewers display static images (.gif) in which navigation is very limited and that are disconnected from genome databases. Starting from these observations, we decided to implement a genome map viewer, MappetShow, that overcome all these limitations.

In part 2, we explain the visualization concept we have used in MappetShow to address the problem of visualizing very dense maps. Then we present the functionalities and implementation of MappetShow. In part 4, we detail the architecture of the software.

2 The Concept of Visualization in MappetShow: Non-Linear Views

2.1 Principles of Non-Linear Views

The purpose of non-linear views is to provide the user with a context and thus to avoid him/her getting lost in the mountain of data. The idea is to give him/her a view of all the data but with a focus on his/her area of interest. The view is thus distorted so that the area of interest is represented centered and at large scales while the rest of the data is shrunk at lower scales to fit in the limited space of the screen. The scale of the representation is therefore not constant and the mapping of the data world to its representation on the screen is given by a mapping function or a magnification function.

Mapping Functions

The mapping function is used to map an image to a distorted view. It is simply represented by a two variables or complex function m which maps each point (x, y) of the original image to a new location $m(x, y)$.
It is easy to see that the magnification factor is given by the slope of the mapping function. Distortion characteristics directly depend on the mathematical properties of the function m. For examples, continuity of m means that the view is just stretched or compressed and not split or torn; positive derivatives of m imply that there is no overlap or folding of the view; holomorphic functions locally preserves angles.

Magnification Functions

The magnification function z represents the local zooming factor applied to the original image. At position (x, y) the factor $z(x, y)$ can be defined as the ratio of a mapped surface to the original surface (infinitely small and centered at (x, y)) (see Appendix).

Examples

Different types of distorted views include :

- fisheye views [16].
- perspective walls [17].
- hyperbolic views [18,19,20]. Hyperbolic views are well suited for graph visualization and have already been used in Biology for that purpose [21].
- rectangular mapping view.

Examples of distortion with their mathematical definition and a figure of their effects are given in Appendix.

2.2 Implementations of Non-Linear Views and Performance Issues

Distortion techniques can requires a large amount of computer time to generate and update images. This can render the user interface unusable. Performance during interactions (zooming and scrolling , when screen needs to be updated) must be taken into account, which can be achieved by discrete mapping.

Discrete Mapping

The idea of discrete mapping is to avoid mapping the view pixel by pixel, and to map only the important nodes of the view and interpolate the other points. For example to produce the mapping of a straight line one would only compute the mapping of its extremities and link them by another straight line. In the general case, this would create artificial discontinuities in the distorted view because mapping function preserves angles only locally and therefore do not transform straight lines in straight lines.

Two solutions address these problem:

- to use mono-dimensional or rectangular mappings (see figure 3) which preserve continuity and inclusion.

- for a general mapping function, a solution is to make the problem discrete instead of continuous, by organizing nodes on a grid, defining the figure to be displayed as straight lines between nodes, and mapping the grid alone.

Density Control

By definition, context regions are represented at lower scales than the focus zone and they should therefore present less details to maintain a good readability. This leads to the idea of constant information density on a view: by indexing the level of details represented at a given point of the view on the current zoom value at this point, one ensures that all part of the view have an approximately equal density of information. This is also a way of ensuring better display performance by limiting the global complexity of the drawing.

Color can be used to give an indication of the information density where details have been removed.

Local Mapping

It is often interesting to apply different mapping functions to different parts of the view, for example to different objects. This allows a local distortion of the view which may be more adapted to the visualization problem and less demanding in terms of computer resources.

Performance issues are addressed by the use of discrete mapping, density control and local mapping which reduce largely the computation time and allow a real-time navigation with maps of several thousands of markers.

3 Map Visualization

3.1 Problems Arising from Representing Dense Maps

Several problems of readability occur when representing dense maps:

1. the map is too dense and marker position and names are overlapping and undiscernible (see figure 1).
2. marker names are shifted out of the visualization window when zooming (see figure 1).
3. when comparing two maps, the problem is worse because it suffices that one of the map have one of the two problems listed above to make the drawing unreadable (see figure 1).

3.2 Designing Views for Genome Visualization

To address the problem of readability, MappetShow implements non-linear viewing using a one-dimensional hyperbolic mapping function. Since maps are essentially one dimensional objects, there are no need to use two dimensional mapping functions. Nevertheless for displaying a lot of maps on the same view, rectangular mapping views can be used. An example of improved view is given figure 1. The marker names are perfectly readable in the focus area and all the map context is visible.

To address the problem of marker name layout (case 2), we compute dynamically their positions after each scroll using a mapping function as for the general view. This can be achieved of course on linear or non-linear views (see figure 1).

To address the problem of comparing maps (case 3), we defined in MappetShow local mapping functions that can be applied to each compared map. This allows the user to get a readable comparison of several maps. An example with two maps is given on figure 1.

3.3 Density Control

Graphical objects in MappetShow are organized in a hierarchy: basic components, which may be for example lines, circles, rectangles or text, and containers which contains components or other containers. We define two levels of representation for a container: coarse and a fine level or representation.

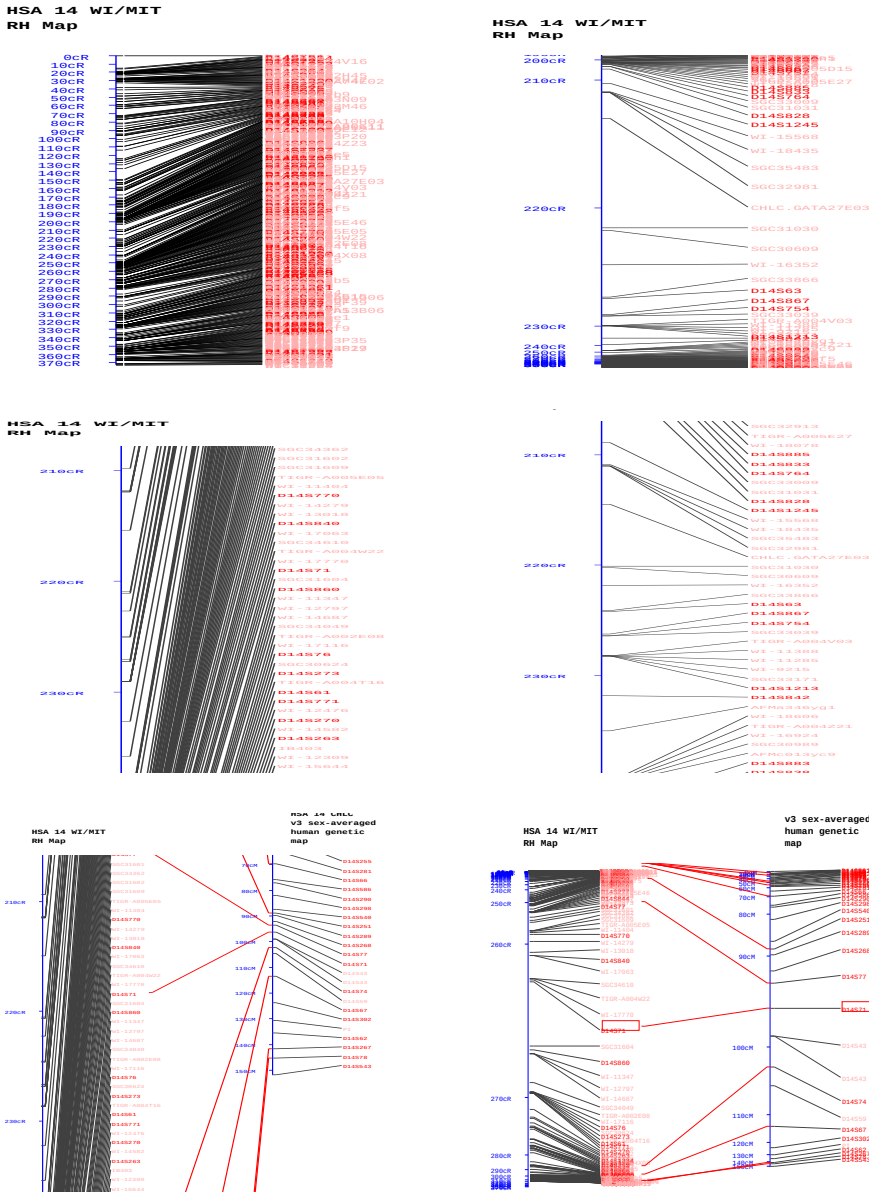

Figure 1: Example of unreadable (left) map representations and the corresponding readable output of MappetShow (right). The two figures at the top correspond to the case 1 stated subsection 3.1, the two in the middle to case 2, and the two in the bottom to case 3.

The representation of each graphical object depends on its distance to the focus point of the picture. Graphical objects whose center point is close to the focus point are represented finely, which means that all their components are displayed. Graphical objects which are more distant are represented in the coarse mode. This representation rules are recursively applied to the containers included in a graphical object.

3.4 User Control

When starting MappetShow, the user has to choose a chromosome or a marker of interest. Then the list of maps of this chromosome or the list of maps containing the marker is proposed; the user selects the maps to display and the visualization applet is started with these maps. In the applet, the user has the control on:

- the zooming factor with a scrollbar.
- the position of the focus point with a scrollbar.

Each marker in MappetShow is also linked to a WWW textual browser.

4 Architecture

MappetShow is written in Java™ and its architecture consists of four main components as shown on figure 2:

- the broker: it can fetch the data from as disparate data sources as a CORBA (common object request broker architecture) server, remote procedure calls that query the HuGeMap database, a stream of data. The interface layer queries objects (map, marker ...) by name and returns an object description string. The CORBA server implements a standard interface for genome maps that is described elsewhere [22]. The stream of data is structured in a language of graphical object description that can be used for maps but also for any other drawing purposes. The resource file contains graphical parameters such as colors, fonts ...
- the drawing layer: it parses the map description string and builds graphical objects (a container).
- the viewing layer: it ensures the distortion and selection of the view. It draws a graphical object to the screen using a non linear view at the

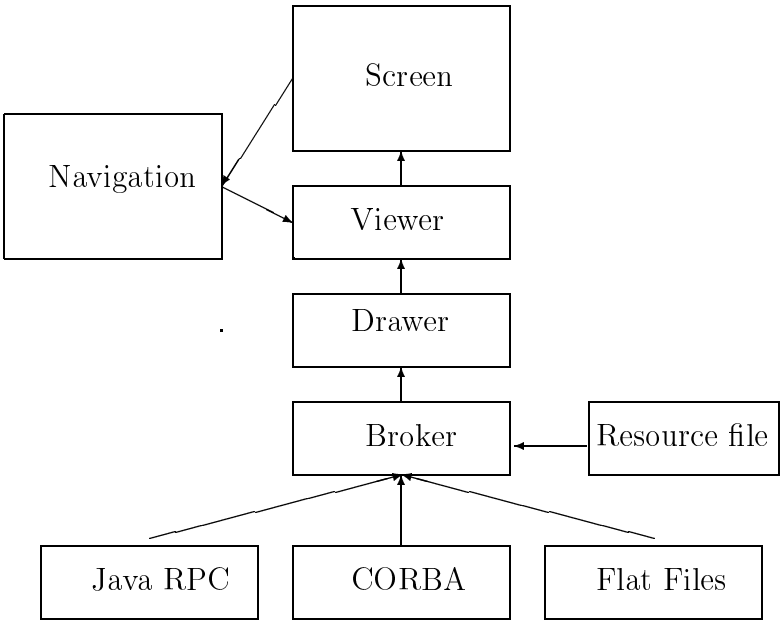

Figure 2: Architecture of MappetShow.

appropriate level of representation depending on the local magnification factor.

- the navigation layer: it takes care of the interactions with the user who can modify the viewing layer parameters;. The navigation layer manages scrollbars and mouse events. It updates the viewing layer parameters : focus position, scale parameters (mapping functions parameter).

5 Conclusion

Non-linear viewing is a solution for providing context to the user navigating in large data sets. In genome map browsing and comparison, rectangularly mapped views can be used to improve the readability of the drawing. Such capabilities are necessary today to cope with very dense maps and with maps build from rich maps of other species. MappetShow achieves such capabilities in a flexible way and can fetch data from a CORBA server, an object-oriented database or a flat data stream.

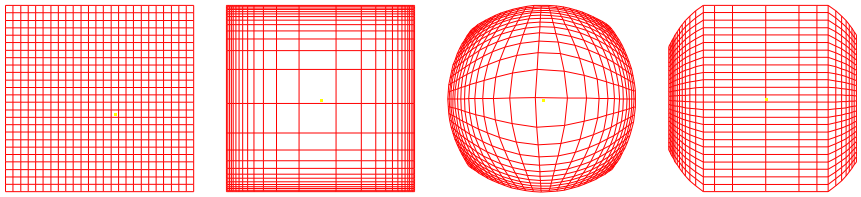

Figure 3: Simple grid before any non-linear transformation (left). Example of rectangular mapping view of the simple grid (middle left). Same with polar mapping view (middle right). Same with perspective wall (right).

Thanks to its low level language of drawing description, MappetShow can be used for other drawing purposes than maps: it has been used for example to visualize large pedigrees.

Acknowledgments

This work was supported in part by the European Union contracts BIO4-CT95-0037 and BIO4-CT98-0030.

Appendix

Examples of Mapping Functions

Mapping functions are constructed from normalized one-dimensional mapping functions, which are generally continuous, increasing and anti-symmetric (which gives a symmetric zooming function).

An common example of one-dimensional mapping function is the hyperbolic mapping function μ:

$$\mu(x) \;=\; \frac{\sigma x}{\sigma|x|+1}$$

From this one-dimensional hyperbolic mapping function, one can define several types of two-dimensional non-linear views, whose distortions are illustrated figures 3 and 4:

- Rectangular mapping : each coordinates is mapped separately using the single variable function μ.

$$X \;=\; \mu(x) \qquad Y \;=\; \mu(y)$$

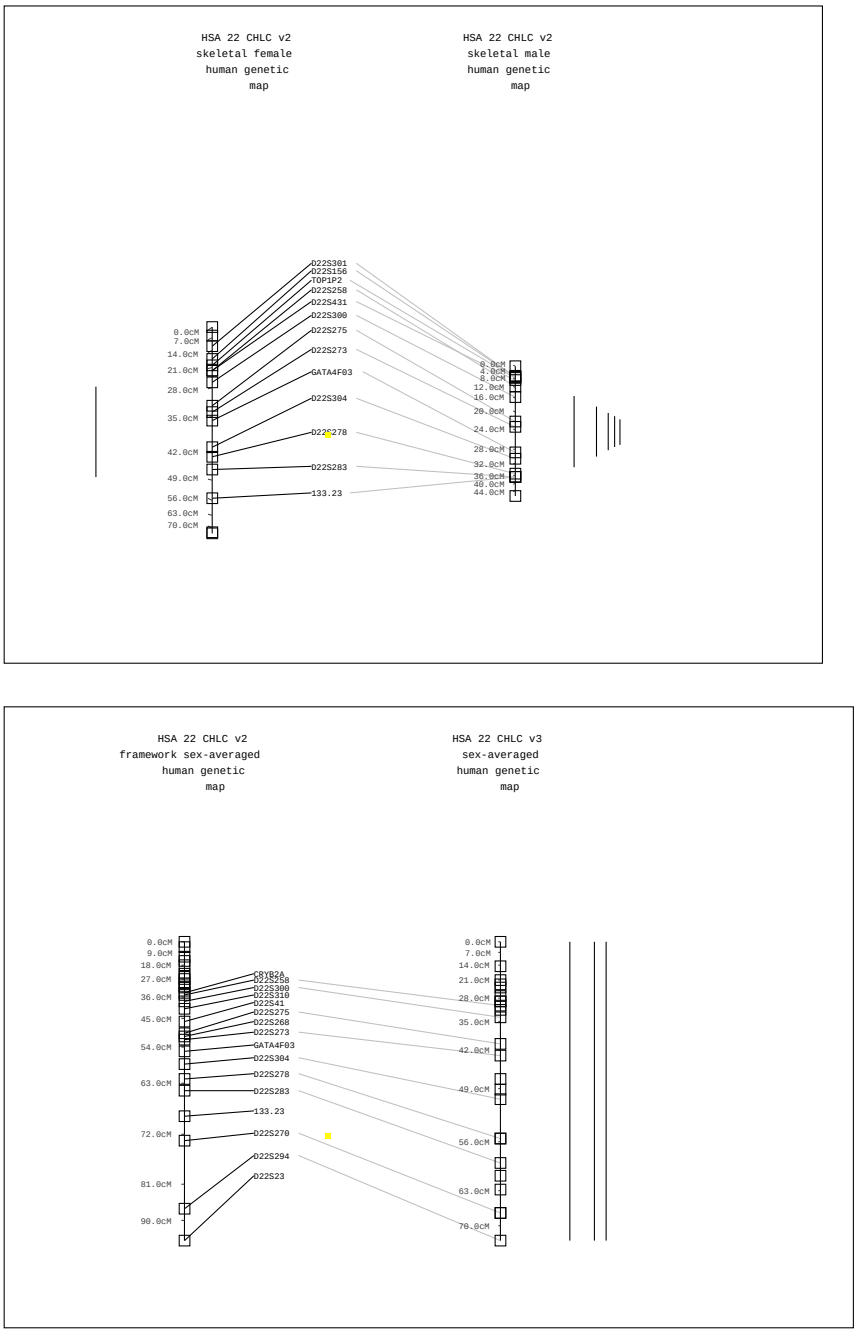

Figure 4: Perspective wall (top) and rectangular (bottom) view of maps with density control.

- Polar mapping : r is the radius (distance to the focal point).

$$X = \frac{\mu(r)}{r}x \qquad Y = \frac{\mu(r)}{r}y$$

- Perspective wall mapping : The image fades away in the y-direction as it goes far from the focus point. In that case, the magnification factor is constant along a y-direction.

$$X = \mu(x) \qquad Y = \frac{\mu(x)}{x}y$$

References

1. Richard Durbin and Jean Thierry Mieg. A C. elegans Database. Documentation, code and data available from anonymous FTP servers at lirmm.lirmm.fr, cele.mrc-lmb.cam.ac.uk and ncbi.nlm.nih.gov. 1991.
2. M.J. Cinkosky, J.W. Fickett, W.M. Barber, M.A. Bridgers, and C.D. Troup. SIGMA: A system for integrated genome map assembly. *Los Alamos Science*, 20:267–269, 1992.
3. Stuart Leonard Pook, Eric Viara, Bruno Lacroix, Guy Vaysseix, and Emmanuel Barillot. The GenomeView project. *Généthon technical report*, 1994.
4. Sun Microsystems. The source for Java^{tm} technology. http://java.sun.com.
5. J. Hu, C. Mungall, D. Nicholson, and A. Archibald. Design and implementation of a CORBA-based genome mapping system prototype. *BIOINFORMATICS*, 14:112–120, 1998.
6. Andy Grigoriev. Genomes with a view. *Trends Genet.*, 13:499, 1997.
7. Andy Grigoriev. Genome Navigator. *Trends Microbiol.*, 6:184, 1998.
8. LD. Stein, S. Cartinhour, D. Thierry-Mieg, and J. Thierry-Mieg. JADE: An approach for interconnecting bioinformatics databases. *Gene*, 209:39–43, 1998.
9. Kazuo DNA Research Institute. The Cyanobase map viewer. http://www.kazusa.or.jp/cyanobase/.
10. Fasman K.H, Letovsky S.I., Robert W.C., and Kingsbury D.T. The GDB human genome database anno 1997. *Nucleic Acids Research*, 25:72–81, 1997.
11. Stuart Leonard Pook, Guy Vaysseix, and Emmanuel Barillot. Zomit: biological data visualisation and browsing. *BIOINFORMATICS*, 14:807–814, 1998.

12. Oxford Molecular/Glaxo Wellcome. Graphical interface to public genome maps. http://www.oxmol.com/biolib/map/.
13. G.D. Schuler and al. A gene map of the human genome. *Science*, 274:540–546, 1996.
14. T.J. Hudson and al. An sts-based map of the human genome. *Science*, 270:1945–1954, 1995.
15. C. Dib and al. A comprehensive genetic map of the human genome based on 5,264 microsatellites. *Nature*, 380:152–154, 1996.
16. George W. Furnas. Generalized fisheye views. In *CHI '86. Conference proceedings on Human factors in computing systems*, pages 16–23, Boston MA, USA, April 1986. ACM Press.
17. Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: Detail and context smoothly integrated. In *CHI '91. Human factors in computing systems conference proceedings on Reaching through technology*, pages 173–176, New Orleans LA, USA, April 1991. ACM Press.
18. John Lamping and Ramana Rao. Laying out and visualizing large trees using a hyperbolic space. In *UIST '94. Proceedings of the ACM symposium on User interface software and technology*, pages 13–14, Marina del Rey CA, USA, November 1994. ACM Press.
19. John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI '95. Conference proceedings on Human factors in computing systems*, pages 401–408, Denver CO, USA, May 1995. ACM Press.
20. John Lamping and Ramana Rao. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing*, 7(1):33–35, March 1996.
21. Alan J. Robinson and Tomas P. Flores. Novel techniques for visualising biological information. In *ISMB 97*, pages 241–249, Halkidiki, Greece, 1997. AAAI Press, Menlo Park, California.
22. E. Barillot, U. Leser, P. Lijnzaad, C. Cussat-Blanc, K. Jungfer, F. Guyon, G. Vaysseix, C. Helgesen, and P. Rodriguez-Tom. A proposal for a CORBA interface for genome maps. *BIOINFORMATICS*, 15:157–169, 1999.