UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2569

Spider Cursor: A Simple Versatile Interaction Tool for Data Visualization and Exploration

Rosane Minghim
Haim Levkowitz
Luis Gustavo Nonato
Lionis Watanabe
Veridiana Salvador
Hélio Lopes
Sinésio Pesco
Geovan Tavares

Nº 261

RELATÓRIOS TÉCNICOS

São Carlos – SP
Jun./2005

Spider Cursor: A simple versatile interaction tool for data visualization and exploration

Rosane Minghim[1], Haim Levkowitz[1,3], Luis Gustavo Nonato[1],
Lionis Watanabe[1], Veridiana Salvador,
Hélio Lopes[2], Sinésio Pesco[2], and Geovan Tavares[2]

[1] Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
{rminghim,haim,gnonato,lionis}@icmc.usp.br
[2] Pontifícia Universidade Católica do Rio de Janeiro
{lopes,sinesio,tavares}@mat.puc-rio.br
[3] also with University of Massachusetts Lowell, USA, haim@cs.uml.edu

Abstract. We present Spider Cursor, an exploration tool for geometric models that has proven useful in analyzing data coming from various sources. The Spider Cursor is simple to learn and use, and is useful to help extract various types of relationships present in the data. Additionally, it implements a dual presentation mode, in which visual and aural attributes are used to represent information. A cursor in the shape of a spider runs on top of a surface, helping location of information at neighboring points of the geometry. That can be used for data analysis as well as model manipulations (such as cuts and marks). Sound is employed to present complementary and supplementary information as well as to help define orientation. This paper describes the basic concepts of the Spider Cursor, and illustrates its use by giving examples of applications.

1 Introduction

Many interaction techniques have been developed that aim at supporting interaction, exploration and manipulation of complex data and 3D structures.

A very large number of applications relies on surface representation (or landscape plots) to represent objects, data and information of various kinds. However, whilst surfaces are very useful for understanding information and very convenient to represent objects, not many tools bring flexibility for exploration of information coded that way.

In this paper we introduce the Spider Cursor, an innovative, extensible and simple interaction tool that is capable of implementing many of these capabilities into a comprehensive "interaction" toolbox that can be used in a variety of ways to interact with visualizations.

The Spider Cursor is a probe that realizes constrained walks over a surface, displaying information at point- and neighborhood levels, while showing also an overall view of the object under study. It is implemented as an extension to a general-purpose open source visualization software, and can be added to it as a specialization of its interactor class. There is also a stand-alone application that is capable of delivering various kinds of visualization over surfaces and points with the Spider Cursor as an interaction tool.

The concept implemented by Spider Cursor can be extended to include most of the interface capabilities desirable in a data exploration process. The next section discusses that exploration process and the types of interactions useful to accomplish good data analysis. Section 3 presents details of the Spider Cursor. Section 4 exemplifies the use of the tool showing two applications. Conclusions are drawn in Section 5.

2 Previous work

Shneiderman's Visual Information Seeking Mantra (overview, zoom, filter, details-on-demand + relate, history, extract) has been considered the most succinct description of the menu of interaction techniques for visual information seeking [16]. The "Mantra" encapsulates a multitude of interaction techniques.

Interaction techniques have been traditionally classified into three higher-level categories, usually referred to as Data Transformation interaction, Visual Mapping interaction, and View Transformation interaction. Each one of these can be further divided to reveal several techniques that fall into the same group [3].

2.1 Data Transformation interaction techniques

Data transformation interaction techniques provide a variety of ways to select cases, objects, or variables to be included in or discarded from a visualization for the purpose of aiding in the analysis.

Dynamic queries Dynamic queries provide various means that allow the user to query a subset from a viewed visualization in order to seek more details on that subset. They are usually implemented with the aid of standard GUI widgets such as sliders and radio buttons. They provide visual means for the user to specify conjunctions of variables. For more details and examples of tools and technique in this category, the reader is referred to, e.g., [17, ?,?,?].

Direct walk A direct walk is a path of linkages from one node to the next one, which the user follows during the interaction process. A common example is following hyperlinks on the Web. See, e.g., [4, 7]for details.

Details-on-demand When a user is examining a large number of objects, s/he is usually limited to a low level of details about each object. Seeking more details about a smaller set of objects can provide, at the expense of losing sight of all other objects, an in-depth look at those that have been selected [16].

Attribute walk In an attribute walk, the user selects a case based on some attribute, and tries to seek other cases that share the same or similar attributes, based on some well-defined similarity criteria. It was developed for searching databases [19].

Brushing In a multi-view visualization of the same object, the highlighting of some object in one of the views will show the same object in all other views in a way that is clearly distinguishable from other objects. Simple examples include point selection in a visual presentation, table rows selection, or range selections with sliders, [5, 10]

Direct manipulation Direct manipulations are most commonly used to modify transformations, in which parameters can be adjusted directly in the visualization. See, e.g., [9].

2.2 Visual Mapping interaction techniques

These types of techniques provide means to modify the mapping from data items to visual attributes. One of the most common techniques is Dataflow, in which a visual map of the flow (often referred to as a "pipeline") is presented to the user. The user can then modify the data-to-visual mapping by interacting with the flow diagram. Some examples of this approach can be found in ConMan [8], AccuSoft's VisiQuest (originally Khoros), and AVS [20]. The latter introduced the approach whereby the flow diagram is displayed in a separate window, which has become a de-facto standard.

2.3 View Transformation interaction techniques

This group of techniques provides mapping from a given view to another view of the data, by interaction with the viewing parameters only.

Direct selection Highlight and select objects and groups. This can be used to enhance certain aspects of what is being visualized, and can be also used as a pre-step to choose input to subsequent processes or actions.

Camera movement Camera movement is a common way to describe the change of observer's viewing position to provide alternative views. Such techniques are usually most meaningful when dealing with three-dimensional data, or data that has been mapped to a 3D space.

Lenses The lens metaphor, sometimes called "see-through," has been used to describe interaction techniques that emphasize the view of a subset while leaving the rest of the objects viewable, but in an out-of-focus or out-of-center mode. A typical lens (at times referred to as "magic lens") selects objects based on their location in 2D display space. Objects that have been selected can then be subject to additional manipulation using other techniques as well as additional lens manipulations. See, e.g., [2, 1, 6].

Overview + detail Here, two levels of linked visualizations are provided to the user. One (the "overview") presents some high-level view of all objects involved, whereas the other one (the "detail") provides a more detailed view of a subset of the objects. The user can move the detailed region across the overview visualization [12].

Zoom A well known interaction technique, zoom provides a tradeoff between the number of objects that are viewed and the number of variables (or other attributes) that are being shown for each object. A zoom-in will reduce the number of objects but will increase how many variables of the objects will be visible; a zoom-out will do the opposite.

It is obvious that many of these techniques need to be employed in order to be able to follow Shneiderman's "Mantra." However, there is currently no existing "toolbox" where one can fetch interaction tools to accomplish all of these. Such capability would be most useful.

Recently, Salvador and others[13] revised the classification of interaction techniques regarding their relationship to information presented by sound. The Spider Cursor was first employed for visualization in the context of that work.

In the remainder of this paper we demonstrate how some of these interaction techniques have been implemented within Spider Cursor, and others, which have not yet but can be implemented, thus turning Spider Cursor into a "one-stop interaction tool box" that can be used in a versatile way to interact with visualizations. The tool developed is a prototype capable of providing interaction related to Dynamic queries, Direct walk, Direct selection, and Overview+detail. These are already implemented to some extent, as will be demonstrated in this paper. Others, such as brushing, direct manipulation, lenses and zoom can be implemented as direct extensions of the current version of the Spider Cursor.

3 Spider Cursor Tool

Spider Cursor is a probe-type tool capable of supporting a number of important tasks in visualization environments related to point identification, analysis of groups of elements and their relationships, association of properties at point and region levels, as well as marking and cutting.

It can be used to manipulate and explore data sets that have intrinsic geometric nature, such as objects reconstructed from MRI and CT scans, and meshes representing bodies such as airplanes and cars. It can also be used for generic data that have been mapped to geometric forms, such as texts, biomedical data, survey data and other data. One way that "generic" data can be mapped to geometry is by means of multidimensional projections (see[18]). An example of this transformation is given in Section 4.1

3.1 Spider Cursor Overview

In general, Spider Cursor is a powerful tool for exploring neighborhoods. This is a useful characteristic both for analysis of generic projections (for data exploration) and for modeling and 3D surface interaction (such as sculpting and simulated surgery).

Spider Cursor implements a constrained walking over a surface. Centered at a particular point, it shows the point's neighbors by highlighting the edges that connect these neighbors to the central point (see Figure 1). The center of Spider Cursor shows the target point, and additional information on that point is reflected in various ways. The

cursor is free to move anywhere on the surface under the user's guidance. Alphanumeric data can be displayed on an adjacent window (currently located at the top left corner of the cursor's window). In addition, numeric values associated with the viewed points can be mapped locally to sound, color or glyph shapes. By showing edges to the next neighbors of the focus point, the Spider helps selection of particular directions for the next movement. Also, when exploring 3D surfaces, its display improves the detection of slopes in the neighborhood of the point. That in itself is a helpful characteristic, since close examination of surfaces usually impairs orientation during exploration.

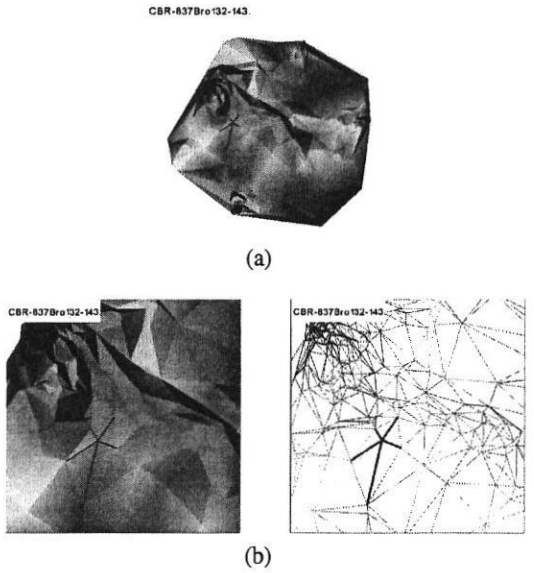


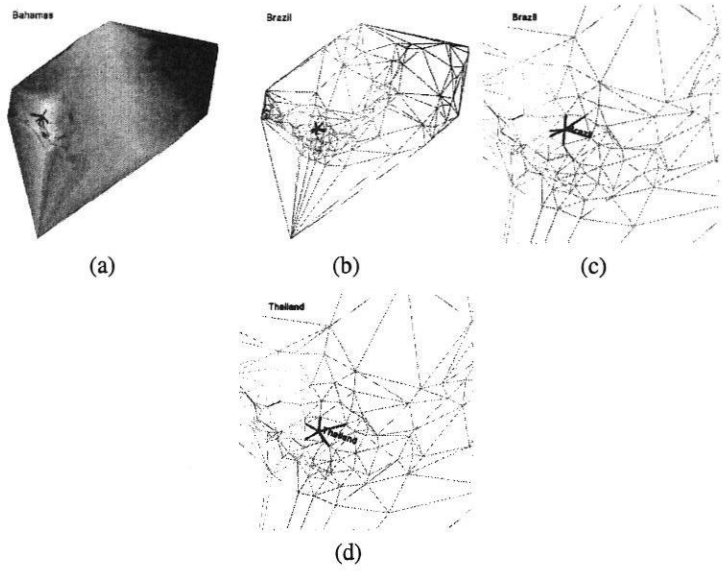
Fig. 1. The Spider Cursor (a) over a surface map (it may not appear in a black-and-white printout). (b) Zoomed in to the region of the located point. (c) Triangulation of (b).

A tool such as Spider Cursor can be used in a number of contexts. One particularly interesting ability is neighbor-identification in a neighborhood-based display.

As an example, Figure 2 shows a projection of the Human Development Index (HDI) data. The Quality of Life (QoL) data set provides the HDI of 174 countries listed by the United Nations Development Program (UNDP). It ranks countries according to the data about their quality of life. The projection was created by first mapping the multi-dimensional data into 2D using Nearest-Neighbor Projection (NNP)[18]. The projection was then improved by a force-based point adjustment (see details in Section 4.1). From the projection it can be clearly seen that the countries are separated into groups based on the variables reflected in the data set, identifying high, medium and low QoL values.

Suppose we want to explore, for example, the neighbors of Brazil in that pseudo-classification reflected by the projection. The Spider software allows us to place the cur-

sor on top of an element of interest, based on an associated label file (see Figure 2(a)). Then, from that point, one can explore all immediate neighbors of the current position and their neighbors' neighbors (see Figure 2(c) and 2(d)).



(a) (b) (c)

(d)

Fig. 2. Various levels of interaction with HDI data. (a) overview and search for Brazil. (b) triangulated view of the data set, after finding Brazil. (c) zomming into the region under analysis. (c) verifying the neighbors.

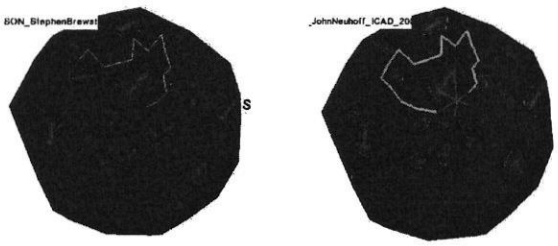
Spider Cursor's applicability can be extended by a variety of actions and the resulting events. Its current version is implemented as an extension of the interactor objects in the open source Visualization Toolkit, VTK [15]. VTK is a C++, object-oriented toolkit, based on the data flow visualization paradigm (that is, transformations with the interaction at the end, including feedback into the pipeline). Most of Spider Cursor's visualization features, therefore, are inherited from that software. So are the basic properties of scene zoom, and other viewing transformations. The interaction over the surface, the sound mappings, and the highlighting of attributes on top of the window are features added by the Spider Cursor implementation. Next section describes the intrinsic and inherited features of Spider Cursor.

3.2 Spider Cursor Functionality

The main intrinsic features of Spider Cursor are:

- constrained walking over the surface of an object, as illustrated in Section 3.1, meaning that once the Cursor locates a surface, mouse movements control position on top of that surface.

- path marking: By pressing the left button of the mouse, the walk is restricted to immediate neighbors of the current point; every time the button is clicked, the current point is appended to the path (see Figure 3(a)).
- closed path: by marking a closed path, a region can be selected for further examination (see Figure 3(b)).
- point mappings: values stored at the vertices can be displayed. In the current implementation numeric values can be displayed as sounds (to the pitch of a chosen musical instrument), and alphanumeric values can be shown in a window at the top of the scene window).
- flexible scene arrangement: to be able to function the Spider Cursor needs specification of adjacency relationships between points on the scene. In the current implementation this is provided by a triangulation of the points, which is the main input to the Spider Cursor. However, there can be other elements present in the scene. The triangulation itself must exist but needs not be displayed. Figure 4 shows an example of the Spider Cursor walking on top of scattered points and glyphs representing these points.



(a) (b)

Fig. 3. Path definitions with spider cursor

(a) (b) (c)

Fig. 4. Spider Cursor Walk. (a) over scattered points. (b) over glyphs representing the points with background surface (the cursor may not be visible in a black-and-white print out). (c) without background surface

With these features the Cursor offers additional cues for the user to explore his/her data as compared to other available techniques. Help in orientation is provided by the slope of the spider's legs as well as by sound. Overview is provided by the visual mappings while Cursor movement provides the detail. Point selection provides path and group focus.

A very expressive feature of the tool is its ability to map values (associated with the points or with the Cursor itself) to sound. Values are mapped to pitch in either direct (high pitch to high value) or reverse orders. The slopes of the Spider's legs can also be mapped to sound, as well as segment size, number of neighbors, or vector data. Sound mappings are very useful to confirm (or contradict) visual mappings when the same value is mapped to both. Sound is also useful to provide supplemental information, that is, one scalar field is mapped to sound, while another is mapped to color or another visual attribute. Additionally, under some circumstances Spider Cursor is very helpful for orientation (for instance, when the height of the point relative to a base plane is sonified).

These features make the Spider Cursor a unique tool for data exploration and manipulation.

The Spider Cursor can be added to a VTK program. Also a stand-alone application was developed to demonstrate a basic application using the Spider Cursor and provided means for interaction with it. Those forms of use are described in the next Section.

3.3 Spider Cursor Structure

As mentioned above, the Spider Cursor tool was implemented as an extension to VTK. As such, it is a class (called SV_SoundPath) derived from its vtkInteractor class. Being a specialization of it, the Spider Cursor inherits many of its functionalities, such as translation, rotation, track-ball interaction mode, and even point location via a cursor. The other features were added as new methods of SV_SoundPath.

In that context, any programmer can build a visualization using VTK and then add the Spider Cursor as its interaction tool. A VTK program works under a dataflow paradigm by which data sets are read then transformed until final visual display. At the end of the pipeline there is a renderer (responsible to generate the graphics related to the scene) and an interactor associated with it. The Spider Cursor can be added to such a program as an interactor assigned to a renderer.

The sample code in Figure 5 shows the part of the program responsible for hooking the end of the pipeline (the renderer) to the Spider Cursor and to set up its properties. After that, once the interaction starts, the Spider Cursor can be triggered by pressing the 'U' key on the keyboard.

The renderer passed as a parameter to the operation SetRenderer() (see Figure 5) is the end of the visualization pipeline in the VTK program. The operation associates that renderer with the Spider Cursor, so that when the renderer displays the scene, the Spider Cursor can be triggered.

In the SetDataFile() operation, it can be seen that the data file is output by another object (TriangFile). This is an instance of a VTK object that reads polygonal data from VTK files. This is used as a guide for the Spider Cursor walk, that is, the adjacency is given by the edges in the polygonal model, and the vertices are the focus points.

```
// Creating Spider Cursor Object
SV_Sound_Path *Sound_Path = new SV_Sound_Path;
// ren1 is the renderer created through a VTK program
Sound_Path->Set_Renderer(ren1);
// Sonification setup - chooses individual points
Sound_Path->Set_SonificationMethod(3);
// Sets Min. frequency, Max. frequency, and Timbre
Sound_Path->Set_SoundParameters(20, 120, 20);
// Sets model file, as output of a VTK reader object
Sound_Path->Set_DataFile(TriangFile->GetOutput());
// Draw over the cursor (yes = 1, no = 0)
Sound_Path->Set_DrawObject(1);
// Sets point identification file
Sound_Path->SetTextFile(File2Id);
// Initializes Interaction
Sound_Path->Start(1);
```

Fig. 5. Programming the Spider Cursor into a VTK program

The SetTextFile() operation is used to set a tag file, that is, the information related to the points (one per point) that will be shown on top of the scene window. This is just a conventional text file whose ID is passed to the operation. With that, any VTK program can make use of the Spider Cursor, provided that the data is coded into a triangular mesh.

As we were interested in using the Spider Cursor to explore data generated by multi-dimensional projections, an application was also built on top of VTK. It uses some of VTK's functionality and most of the Spider Cursor's functionality. That is a stand-alone application from which the user can make choices of visualizations and of operation of the Spider Cursor.

The interface of that application allows access to VTK's visualizations of multiple scalar fields, coding them by glyphs, height and color, as illustrated in the previous pictures. It also gives the user access to most of Spider Cursor's capabilities, including mapping data to sound, showing labels, as well as all other capabilities. One of the possibilities of the Spider Cursor offered by the interface is to place the Cursor at a specific point, identified by its label. From the list of point labels (given to the tool by use of the operation SetTextFile() above), the user can pick a point of interest and the Cursor will move to that point. At that point, the user can repeat the interaction.

The Spider Cursor has been employed successfully in various applications. To demonstrate its usability with more detail, we presented two such applications in the following.

4 Spider Cursor Sample Applications

Spider Cursor is capable of helping a number of user activities for data interpretation. We will give as examples two applications.

4.1 Analysis of Text Collections

The first example demonstrates Spider Cursor's help in the analysis of projected data used to interpret relationships amongst bodies of text.

In this application, maps are created from distance-based projections of the analyzed text (in our case, documents in a collection). A distance matrix is calculated from a data set and these distances are used to perform a point placement in 2D space. Multidimensional projection techniques attempt to keep in the same neighborhood those items that present high degree of similarity. After that a projection improvement scheme is applied to approach points deemed similar by means of the distance measurement. For documents, that distance can be calculated in a number of ways, such as latent semantic analysis, cosine distance from term attribute frequency, or even approximations of Kolmogorov complexity [11, ?]. The position of the projected points on a plane can be tiled (e.g., by means of a triangulation), and various attributes can be mapped on top of the resulting mesh to reflect additional properties. The complete description of what distance-based projections encompass in generic data visualization can be found in [18].

Figure 6 shows a text map. The text collection displayed here includes documents in four areas of research, *Case Based Reasoning* (CBR), *Inductive Logic Programming* (ILP), *Information Retrieval* (IR), and *Sonification*(SON). Figures 6 (a) and (b) show the flat map of the corpus. The colors in the picture represent pseudo-classification of the papers as belonging to one of those four areas. Figure 6(c) shows two views of a surface formed by the data mesh. In this case the height of the point represents the depth of the hierarchical clustering of the projected data. That means that points higher up are closer together (in terms of 2D distance) than points placed at the bottom. In Figure 6 (c), the Spider Cursor shows two related papers that have been mapped very close to each other. The name on top of the window shows the name of the file containing the document, coding its pertinent information (such as, class, author, year). Another map of the same data, with change of visual attributes, is shown in Figure 6 (e).

In that application it was possible not only to see overall views of the groups of documents but also to locate individual documents of interest, similar documents, subgroups of interest inside the corpus, and relationships between texts that had not been anticipated. During navigation sound has helped very much, particularly when redundantly mapping to it the same value as was mapped to height. Once the user zooms into a surface, the global sense of height is lost due to the difficulties imposed by perspective. By mapping the same value mapped as height also to sound, the user can tell whether s/he is going 'up' or 'down' from one point to the next.

One of our future goals is to build the Spider Cursor into a system for finding and browsing relevant documents recovered from databases.

In the following section we show the use of Spider Cursor for cutting geometric objects.

4.2 Marking and Cutting geometric forms

The second example shows how the analysis, interaction, and cutting of 3D object structures reconstructed from 2D images can be facilitated by the Spider Cursor.

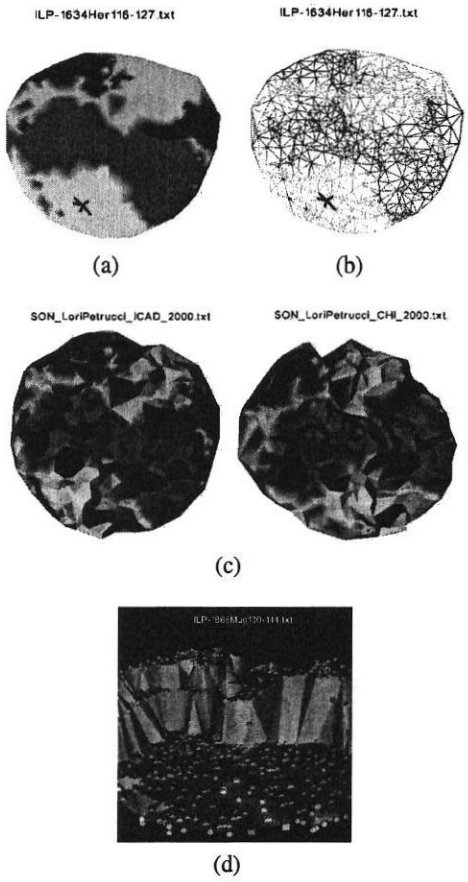
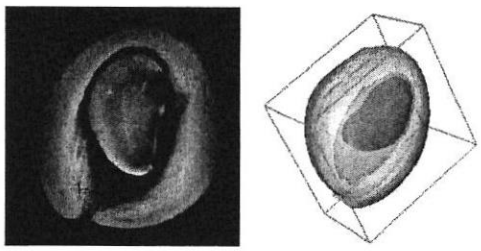


Fig. 6. Collections of document maps (a) flat map showing pseudo-classes. (b) triangulation of (a). (c) two views of the surface created by mapping hierarchical cluster to height. (e) map using height as class (each plateau is a class, isolated peaks and valleys are outliers), whilst color as well as glyphs shapes show levels of clustering (blues are grouped data, reds are isolated documents

Figure 7 illustrates a magnetic resonance (MRI) slice of a mango. The goal here was to analyze the fruit for the possible presence of a disease known as "internal collapse," which creates a "void" between the pulp and the pit of the fruit. Figure 7 shows the reconstructed mango from 36 MRI slices with the three main elements analyzed: the pit, the hole, and the pulp.

Figure 8 shows intermediate steps in the interaction with the model of the fruit. By cutting through specific parts of the fruit, it is possible to analyze its interior in ways not possible trough transparency, conventional walking or flying, or by analyzing each part of the model individually. The advantage of the Spider Cursor in these cutting activities is that once the surface is identified, the cursor 'sticks' to the it, avoiding the need for calculation of collision, which is necessary in other forms of the cutting processes. Additionally, if and when other data is mapped onto the surface, it is possible to use



(a) (b)

Fig. 7. Reconstruction from images. (a) image of one mango slice. (b) reconstructed mango parts.

this data to guide the cut. When cutting in detail, the slope of the Spider's legs also help guide the movement. The Spider Cursor cuts with precision up to the detail of the surface. In experience working on this case sound has also helped the user's orientation.

The Spider Cursor has evolved over the years into a nice tool for exploration of surface-based data.

5 Conclusion

We have presented Spider Cursor, a new tool for interaction with 3D surfaces that has proved very useful in a number of applications. The Spider Cursor offers flexibility and functionality that are different from other available tools, yet maintaining many interesting capabilities and lending itself to future adaptation to most styles of interaction.

Some of these extensions are planned in the near feature: attribute walking, zooming and various sorts of dynamic queries are in sight for implementation. A version of the Spider Cursor is planned in a generic neighboring search environment. Additionally, extra information on data sets can be shown changing the shape of the cursor. The Spider Cursor lends itself to hierarchical implementation.

The Spider Cursor is useful both in the Information Visualization and Scientific Visualization areas, as illustrated by the two case studies presented here. The additional capability of mapping data to sound has also shown advantages during the data analysis process. The aural capabilities of the Spider Cursor are to be expanded. Haptic feedback is to be added to the Spider Cursor in a near future.

In addition, we are planning objective evaluations to measure user performance while interacting with the Spider Cursor. Some user evaluations have already been conducted [14], suggesting possible advantages of the addition of sound in combination with visual mappings (particularly using color). Preliminary results of these studies suggest approximately equal performance with either mode.

6 Acknowledgements

This work has been funded by FAPESP, the São Paulo (Brazil) Research Funding Agency, São Paulo, Brazil proc. no. 04/12202-8, 98/13879-9 and 04/01756-2. Part of

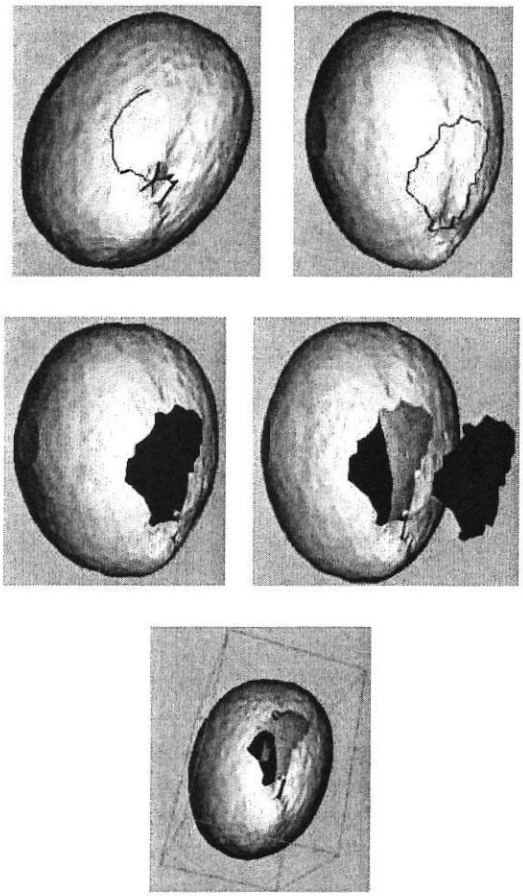


Fig. 8. Screen shots of interaction with 3D models

this work was done while Haim Levkowitz was a U.S. Scholar of the Fulbright Scholar Program in Brazil; he is grateful for its support. The mango images were courtesy of Embrapa, the Brazilian Agency for Research in Agriculture (São Carlos - SP, Brazil). We thank Prof. Alneu de Andrade Lopes for the corpus data used in the demonstrations, and to Renato Oliveira for his programming work.

References

1. E. A. Bier, M. C. Stone, T. Baudel, W. Buxton, and K. Fishkin. A taxonomy of see-through tools. In *Proc. CHI'94*, pages 358–364, Boston, MA, 1994. ACM SIGCHI.
2. E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. DeRose. Toolglass and magic lenses: The see-through interface. In *Proc. SIGGRAPH'93*, pages 73–80, Anaheim, CA, 1993. ACM SIGGRAPH.
3. S. K. Card, J. D. Mackilay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufman, 1999.

4. S. K. Card, P. Pirolli, and J. D. Mackinlay. The cost-of-knowledge characteristic function: display evaluation for direct-walk information visualization. In *Proc. CHI'94*, pages 238–244, Boston, 1994. ACM SIGCHI, ACM Press.
5. W. S. Cleveland and R. McGill. *Dynamic graphics for statistics*. Wadsworths and Brooks/Cole, Pacific Grove, CA, 1988.
6. K. Fishkin and M. C. Stone. Enhanced dynamic queries via moveable filters. In Card et al. [3], pages 253–259.
7. G. W. Furnas. Effective view navigation. In *Proc. CHI'97*, pages 367–374, Atlanta, 1997. ACM SIGCHI.
8. P. E. Haeberli. Conman: A visual programming language for interactive graphics. In *Proc. SIGGRAPH'88*, pages 103–111. ACM SIGGRAPH, 1988.
9. K. P. Hendon and T. Myer. 3d widgets for exploratory scientific visualization. In *Proc. UIST'94*, pages 69–70. ACM, ACM Press, 1994.
10. J. A. McDonald. Painting multiple views of complex objects. In *Proc. ECOOP/OOPLSLA'90*, pages 245–257, 1990.
11. R. Minghim, F. V. Paulovich, and A. Lopes. Fast content-based visual mapping for interactive exploration of document collections. Technical report, 2005.
12. C. Plaisant, D. A. Carr, and B. Shneiderman. Image browsers: taxonomy and guidelines. *IEEE Software*, 12(2):21–32, March 1995.
13. V. C. L. Salvador and R. Minghim. An interaction model for scientific visualization using sound. In *Proc. SIBGRAPI'03*, pages 132–139, 2003.
14. V. C. L. Salvador, R. Minghim, and H. Levkowitz. User evaluations of interactive multimodal data presentation. In *Proc. of the 9th Int. Conf. on Inf. Vis. IV05 (to appear)*. IEEE CS Press, 2005.
15. W. Schroeder, K. Martin, and W. Lorensen. *The Visualization Toolkit: an object-oriented approach to 3D graphics,*. Kitware Inc., 3 edition, 2002.
16. B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Procx 1996 IEEE Symp. on Visual Languages*, pages 336–343. IEEE CS Press, 1996.
17. B. Shneiderman. Dynamic queries for visual information seeking. In Card et al. [3], pages 236–243.
18. E. Tejada, R. Minghim, and L. G. Nonato. On improved projection techniques to support visual exploration of multi-dimensional data sets. *Inf. Vis. J.,*, 2(4):218–231, 2003.
19. F. Tou, M. D. Williams, R. A. Fikes, J. Henderson, and T. W. Malone. Rabbit: An intelligent database assistant. In *Proc. AAAI'82*, pages 314–318. AAAI, 1982.
20. C. Upson, T. Faulbel Jr., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. Van Dam. The application visualization system: A computational environment for scientific visualization. *IEEE Com. Grap. & Applic.*, 9(4):30–42, July 1989.