An interactive interface for velocity optimization using geological constraints

Jos van Trier

ABSTRACT

I have started implementing an interactive interface for a structural-velocity optimization method that features interactive input and modification of structural velocity models, geological constraints, and optimization parameters. The interface is written in C++, an object-oriented programming language, and uses InterViews, a library of C++ subroutines on top of the X Window system.

INTRODUCTION

In SEP–57 (Van Trier, 1988a) I presented a velocity analysis method that determines structural-velocity models. The input of the analysis consists of an initial velocity model, a migrated image (obtained from migration with the initial model), and if available, geological information coming from well logs, dip measurements, or general geological knowledge.

Although velocity optimization methods such as the one above ideally are supposed to produce results without any human intervention, reality frequently shows the opposite: during the optimization decisions have to be made about the feasibility of the velocity model. These decisions are often based on the effect of the optimization on the seismic data. Also, merely the input of a starting model and geological constraints can be cumbersome and time-consuming.

To overcome these problems, I have designed an interactive interface to the velocity optimization that facilitates easy input of a structural velocity model and geological information, and that displays rays used in the inversion, seismic data and inversion results. The user is allowed to interact during all the stages of the optimization, e.g., to delete "bad" rays, to pick events from the data, or to adjust inversion results. The interface is written in an object-oriented programming language, C++, and uses a graphical toolkit, called InterViews (see Dulac et al., 1988, for an introduction of InterViews).

Before describing the interactive interface, I briefly review the velocity optimization method. I illustrate the interface by displaying results from the part that is currently implemented, the interactive input and modification of the structural-velocity model and the display of the raytracing results.

REVIEW OF VELOCITY INVERSION METHOD

The structural-velocity estimation method uses a structural depth image of the subsurface. The image is obtained from migrating the seismic data with a smooth-velocity model, which can be found with a global optimization method (for example, see Fowler, 1988; Sword, 1987). The assumption is that, although the smooth-velocity model is not necessarily geologically relevant, it correctly models cumulative traveltimes to reflectors.

The structural-model inversion starts with converting the smooth model to a structural model using the above observation. For this purpose, rays are traced upwards from the reflectors to the surface, and their traveltimes are compared with those of rays traveling through the smooth model. The structural model consists of different structures, where the structural boundaries are picked from the seismic image. The velocity in each structure is modeled independently from the other structures with 2-dimensional B-splines, resulting into smoothly varying velocities in the structures with sharp velocity contrasts at the boundaries. The spline coefficients are found using standard traveltime inversion techniques. I intend to damp the inversion with geological constraints from well logs, dip information, etc.

After the smooth model is converted into a structural one, the migrated seismic data are used to verify or refine the model. In this part of the optimization, reflection events in the constant surface location (CSL) gathers need to be examined for residual curvature, or perturbations from the horizontal. These residual effects can either be found from picking events or from semblance calculations (see Van Trier, 1988b), and can be inverted to give local perturbations of the structural model.

WHY AN INTERACTIVE INTERFACE?

While modeling synthetic seismic data from structural-velocity models, I found that just building a structural model is time-consuming and cumbersome if done with a standard "batch" program. Each time the model is modified the program has to be rewritten, and usually many iterations (compilations) are necessary before a satisfactory model is found, only to discover in the modeling stage that additional changes are necessary. The same applies to the input of geological information.

Apart from these practical reasons, an interactive interface is necessary for the picking of structural boundaries from the seismic image: although picking algorithms can be partially automated, they generally need some interactive guidance

to avoid mispicks. This is even more true when events are picked in the unstacked migrated data, where the lower signal-to-noise ratio introduces an higher risk for mispicks.

Finally and most importantly, since the goal of the optimization is to find a geologically relevant model, it is necessary to be able to check the progress of the inversion and make corrections to the model if results are not consistent with geological knowledge. It would be desirable of course to automate this decision-making process, but it is not (yet) possible to eliminate the need for good interpreters. The estimation problem is generally underdetermined, and, rather than heavily damping the inversion and automatically finding a meaningless result, I think it is better to guide the optimization process with the help of an interactive interface that provides easy access to the velocity model, seismic data, and geological information, and that indeed uses this information in the inversion.

CHOICE OF SOFTWARE

I have written the program in C++ with the help of InterViews, a graphics toolkit on top of the X window system. Dulac et al. (1988) give an introduction to InterViews. Rather than repeating their arguments for using this object-oriented toolkit, I want to discuss some additional reasons why I found InterViews and C++ useful for my particular application.

"Inheritance" (see Dulac et al., 1988) is one of the basic properties of object-oriented languages. All parts of the velocity model are derived from the same basic InterViews object, a so-called **Picture**, and they inherit all the properties of that object. The combination of the inheritance property and basic tools and operations in InterViews provide an easy way for building a velocity model.

For example, let us consider the example of representing a structural boundary. A boundary segment is a picture that contains a line segment and a filled square at one of the endpoints of the line. A boundary is just a picture containing several boundary segments. Each boundary is inserted in the "world" picture of the subsurface. All coordinates are in geophysical coordinates, only the world picture needs to be transformed to screen coordinates. For example, changing the zoom only involves a simple transformation of the world picture; all the subpictures know how to redraw themselves accordingly. Similarly, to find all the boundaries in the model, the world picture only needs to be searched for other pictures, one of the basic operations provided by InterViews, and then each picture needs to be checked to see if it is a boundary picture. This means that there is no data managing problem of storing and manipulating arrays of boundaries, which themselves are arrays of coordinate pairs. Also, a picture knows if it intersects another picture, and thus it is easy to check if boundaries cross, an operation that would require a lot of programming effort if implemented from scratch.

DESCRIPTION OF INTERFACE

In this section I describe the interactive interface as I plan to implement it. The next section discusses the part that is actually implemented.

Input of velocity model

The optimization requires a starting structural model. The boundaries in this model are picked from a seismic image. The simplest implementation of picking just manipulates line segments on the top of the image, where the endpoints of the segments can be later interpolated to give continuous boundaries. Similarly, velocities can be interpolated in 2 dimensions from velocity values specified at one or more points in each structure. This implementation can also be used when no image is available, for example for building a structural model to be used in the modeling of synthetic data. Therefore, I have started with this approach; later the implementation will be combined with an automatic picking procedure. The model has to make geological sense, i.e., boundaries can not cross each other, and must have endpoints that are attached to either another boundary or the edge of the model.

Geological constraints

The most important geological constraint in the optimization method is the structural map, the input of which has been described in the previous section. Additional constraints may come from well logs, which can be displayed on top of the model. The manipulation of the logs is basically the same as the manipulation of the boundaries in the velocity model (described in the example presented in the last section), since both are 1-dimensional functions. In my last report (Van Trier, 1988a) I discuss the use of dip information to constrain the velocity model. The dips can be represented by little line segments that are drawn parallel to the local dip. The dip lines are also drawn on top of the model; they can be changed by rotating them around their center point.

Inversion

In the inversion rays need to be traced from selected reflector points to the surface. The reflector points can either be specified at regular intervals on the boundary, or defined by the user. After calculating the rays, they are are displayed in the model, and the user can decide to delete possible "bad" rays, or trace extra rays through parts of the model that are not well covered. Traveltimes and other properties of rays can be displayed in separate windows.

Inversion results

The results of the inversion are the updated velocity model and adjusted traveltimes. The updated traveltimes can be compared with the traveltimes in the data

by overlaying them on selected gathers. Parts of the velocity model may have to be edited, for example to correct side effects at the edge of the model. If a non-linear inversion is performed, in which the data are remigrated, the new seismic image may dictate repicking of the boudaries.

CURRENT STATUS AND FUTURE WORK

The main drawback of InterViews is that it currently does not support byte maps for raster images, and thus seismic images can not be displayed in InterViews. For now, we plan to write our own local implementation of byte maps; future releases of InterViews will probably support raster images. Therefore, it is not yet possible to pick boundaries from seismic images in the first part of the optimization, nor to display seismic data in the last part. Apart from the picking, I have finished the implementation of the input of the model. I did not yet implement the input of geological constraints, but since most of the operations needed for this part are similar to the ones used in the model input, I do not expect much problems with the implementation. Except for the display of traveltimes, the implementation of the display and manipulation of the ray tracing results is also finished.

EXAMPLE

To demonstrate the interface I will show several screendumps (Figures 1–8) of a typical session in which a structural model is built and rays are traced from some reflector points to the surface. As I go along, I will explain different aspects of the interface in the captions of the figures.

ACKNOWLEDGMENTS

I thank Jean-Claude Dulac and Dave Nichols for many interesting discussions on object-oriented programming.

REFERENCES

Dulac, J., Nichols, D., and Van Trier, J., 1988, An introduction to InterViews: SEP–**59**.

Fowler, P., 1988, Seismic velocity estimation using prestack time migration: Ph.D. thesis, Stanford University.

Sword, C., 1987, CDR tomographic velocity analysis: Ph.D. thesis, Stanford University.

Van Trier, J., 1988a, Geological constraints in velocity inversion: SEP–**57**, 117-138.

Van Trier, J., 1988b, Velocity inversion of migrated data after structural interpretation: SEP–**59**.

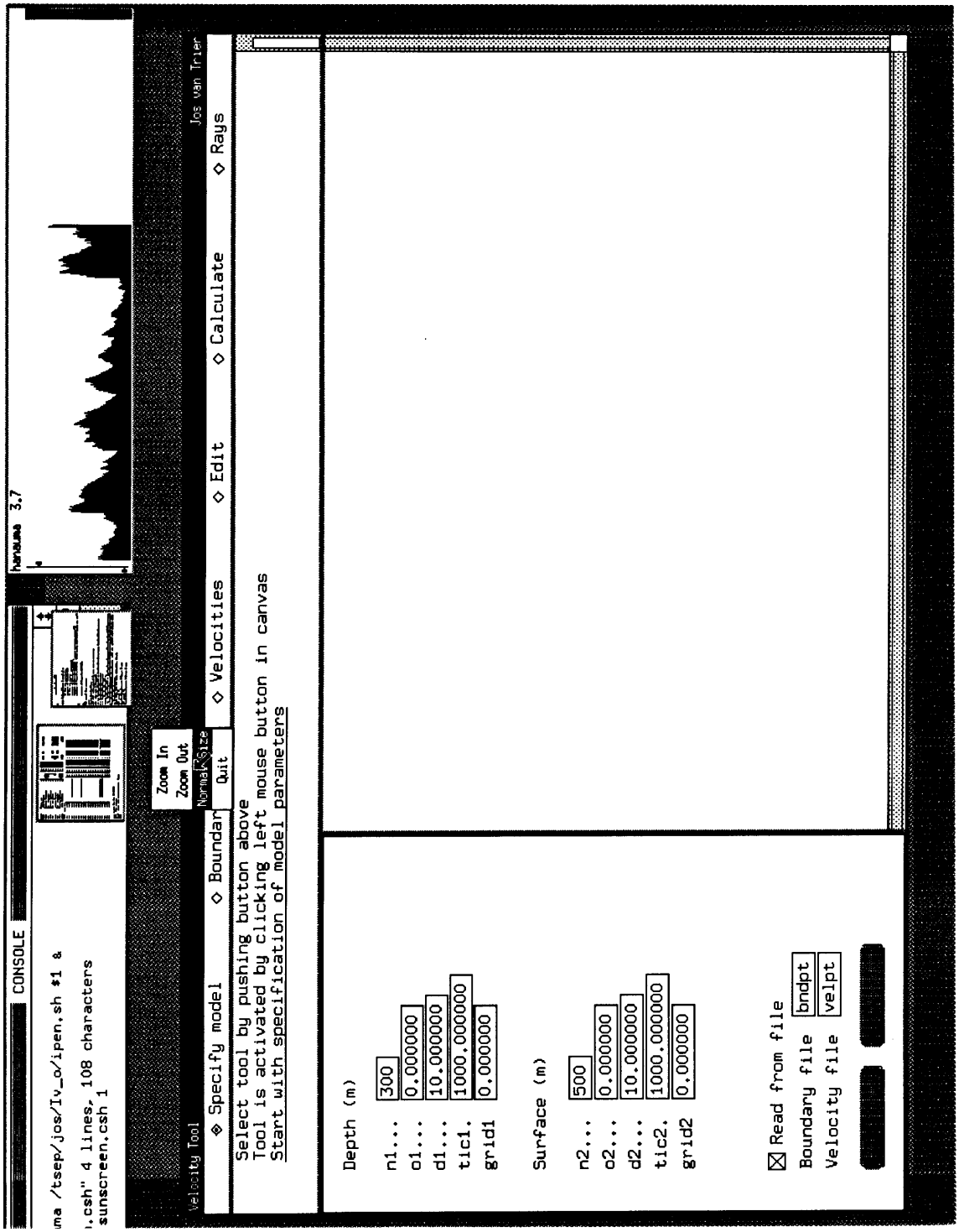


FIG. 1. The display consists of 5 parts: the canvas, on which the model will be drawn, a popup menu for performing view operations, a scrollable information window for displaying information and error messages, buttons, for different operations or tools, and tool-specific windows for numerical input. The information window displays information that I will not repeat here. The same applies to subsequent figures.

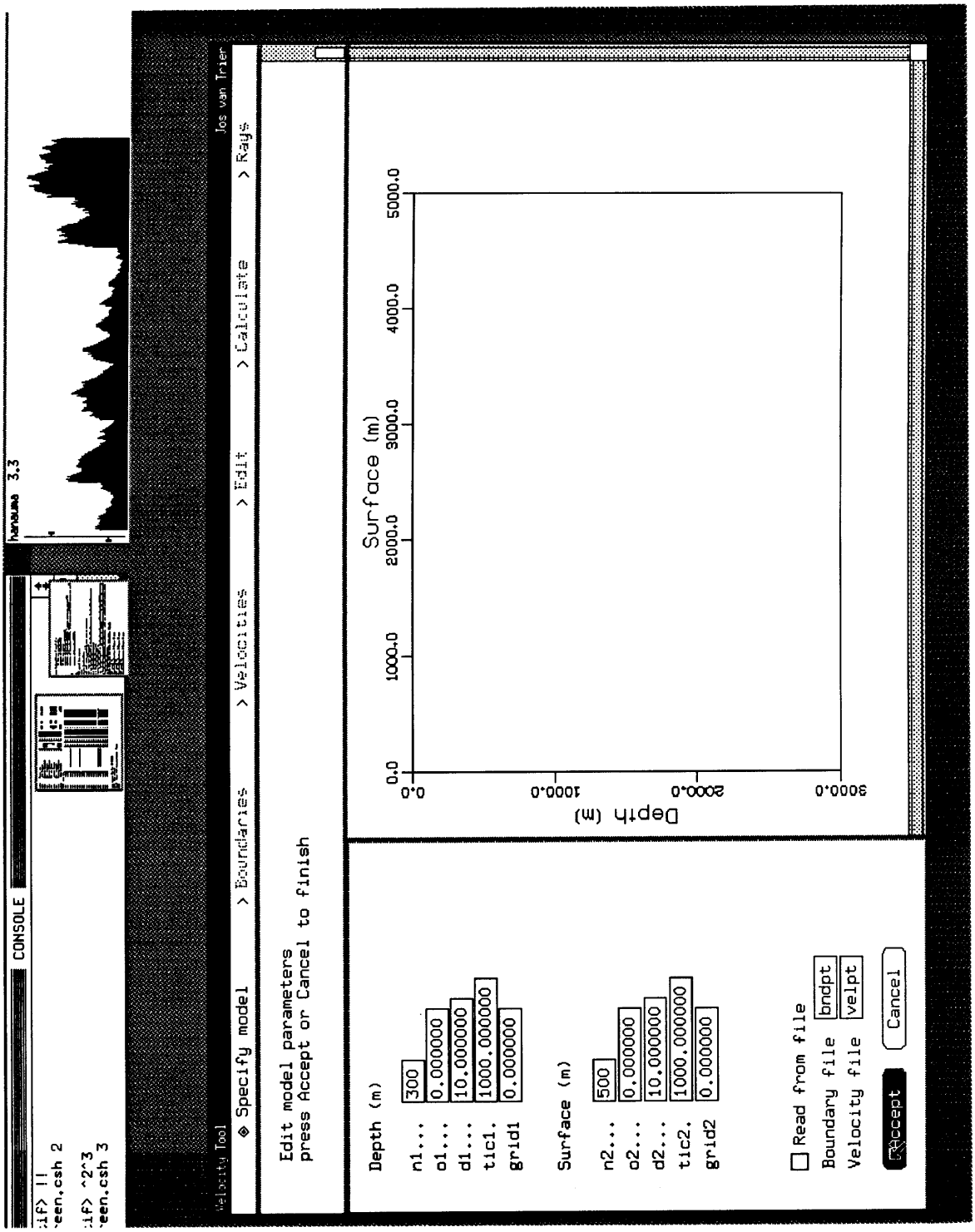


FIG. 2. Specify model. General model parameters can be edited or read from file. When the "accept" button is pushed, a pair of axes is drawn. The model parameters can be changed any time in the digitization process, even after boundaries and velocities have been put in.

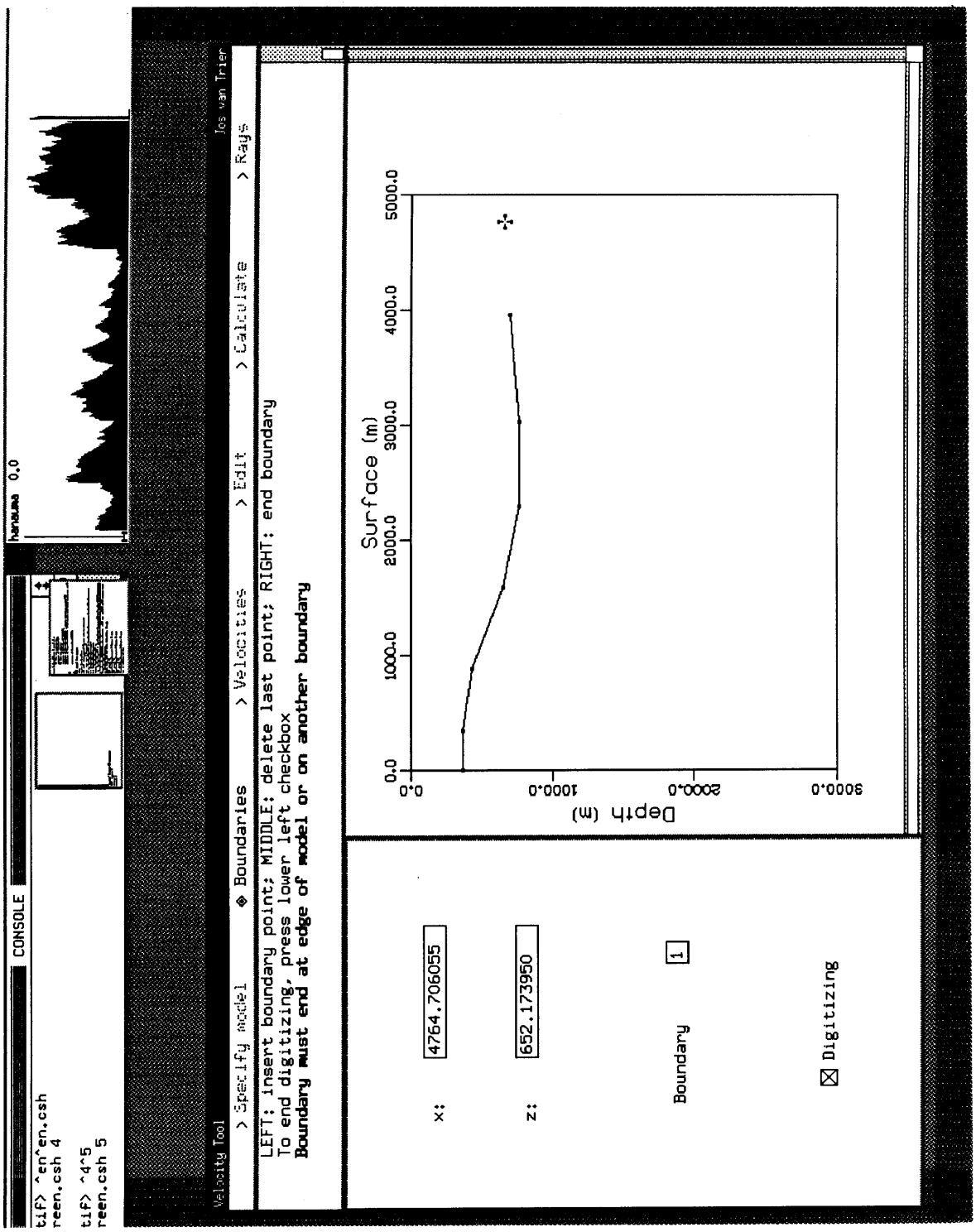


FIG. 3. Digitize boundaries. The position of the mouse is displayed in the two fields on the left. Boundaries must start or end at other boundaries or on the edge of the model, and can not cross.

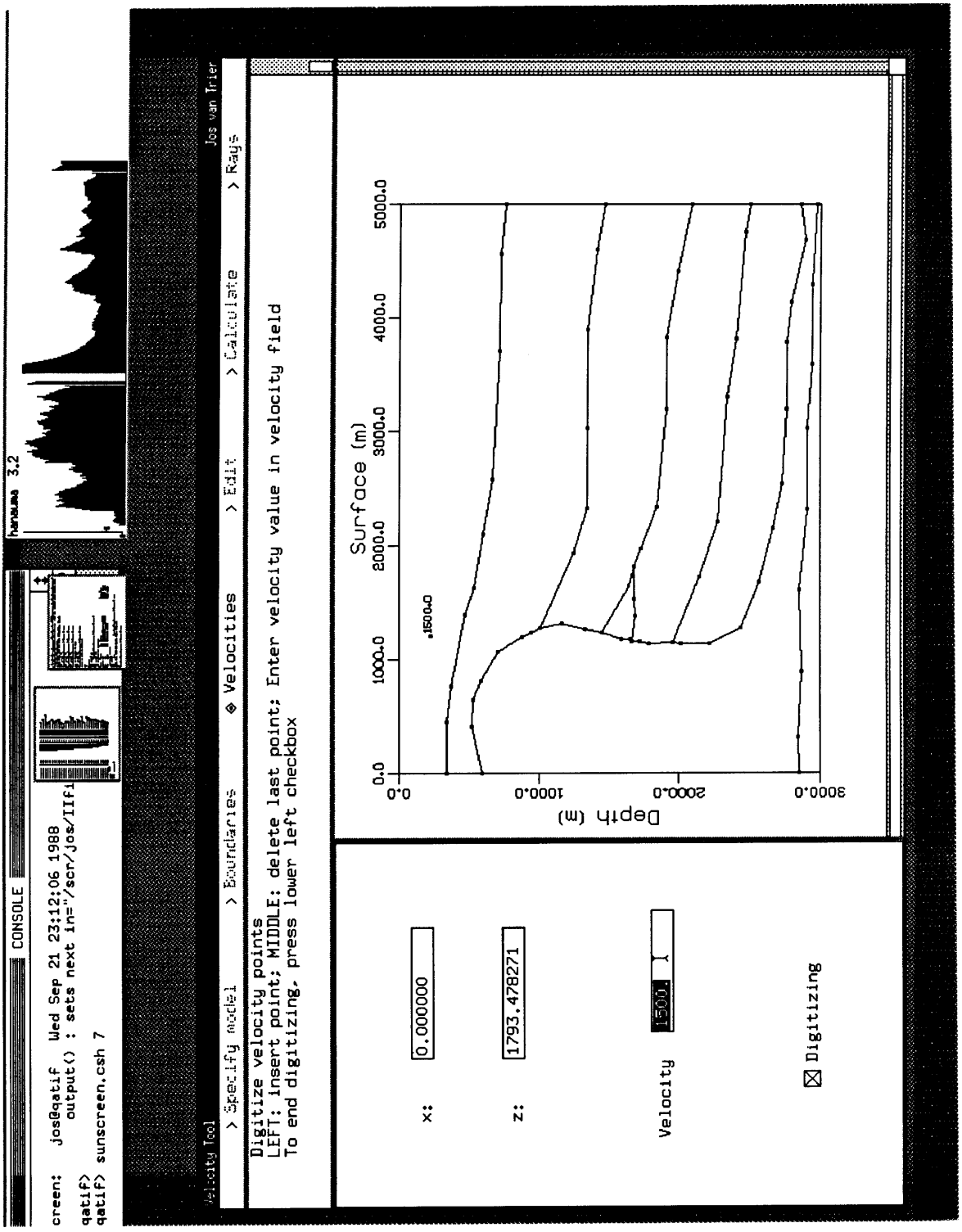


FIG. 4. Digitize velocities. The velocity value is entered in the field on the left.

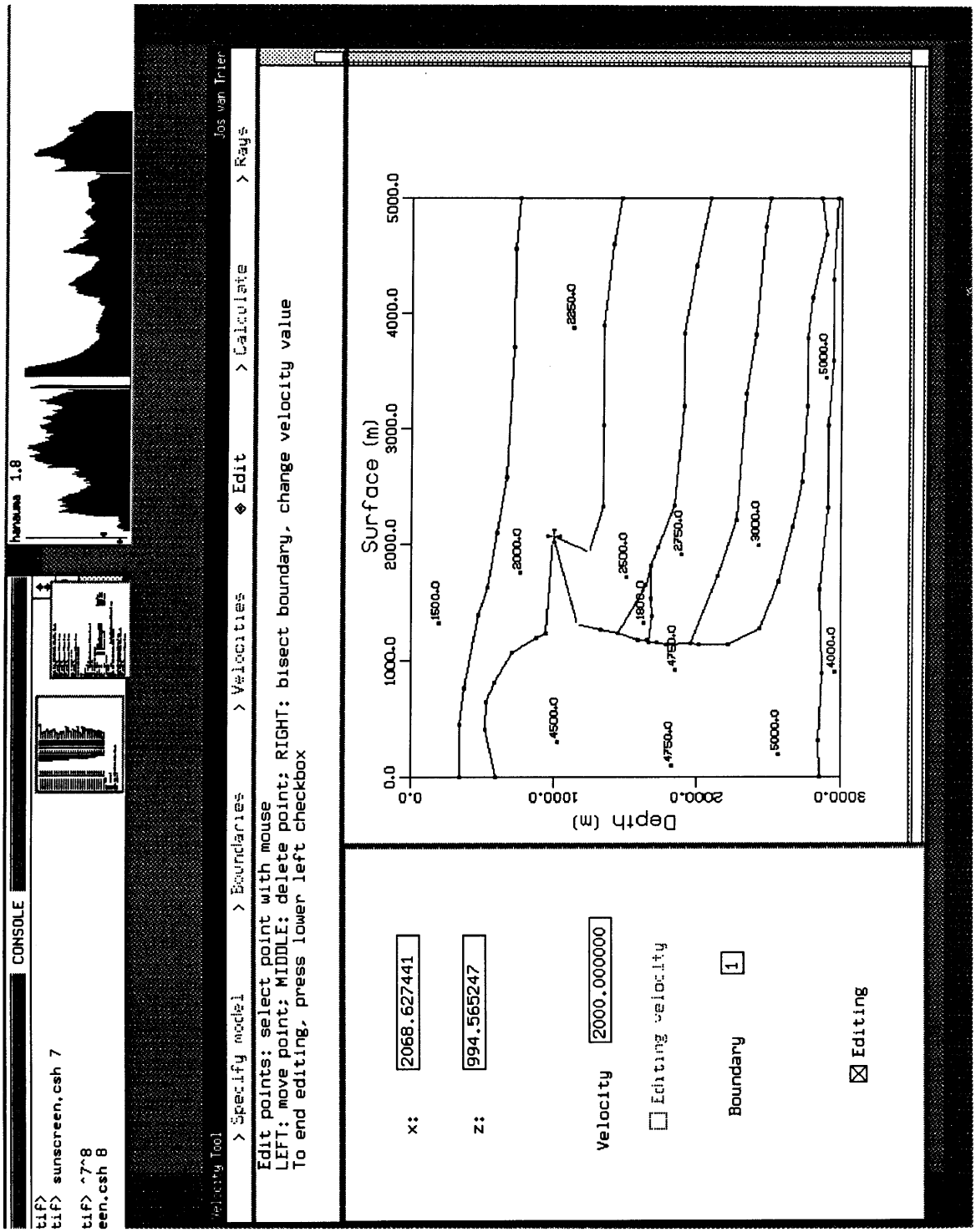


FIG. 5. Edit model. Depending on whether the selected point is a boundary or velocity point, different operations are performed (see information window). In this picture the moving of a boundary segment is shown: when a segment is selected it will change into a "rubberline", as will any segments that are connected to the selected segment. Moving the mouses drags the rubberlines around until the mouse is released.

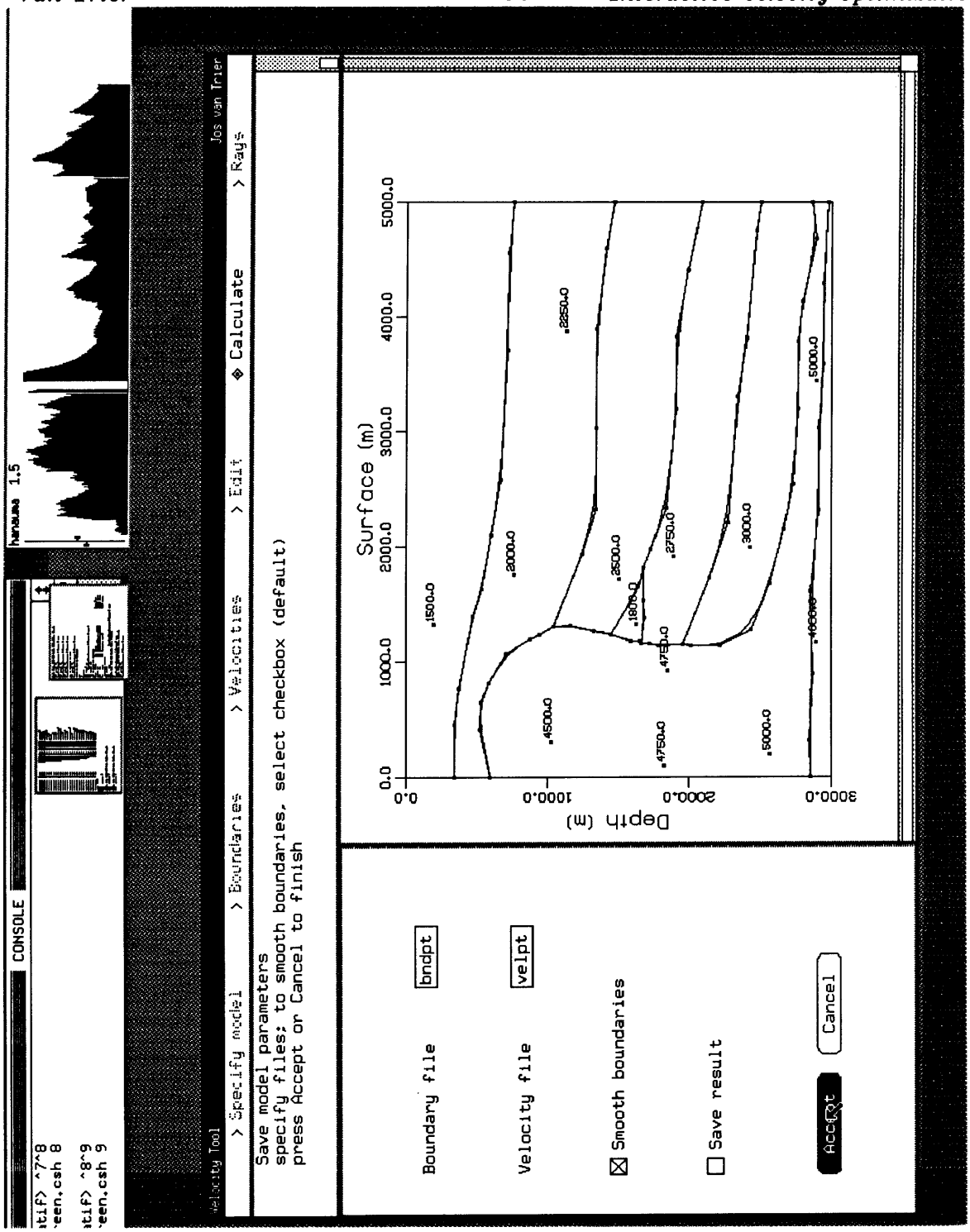


FIG. 6. Calculate smooth boundaries and save results. The boundaries are smoothed using B-splines and are drawn on top of the line segments in a different color. The line segments are kept to provide handles for moving the splines in the editing stage.

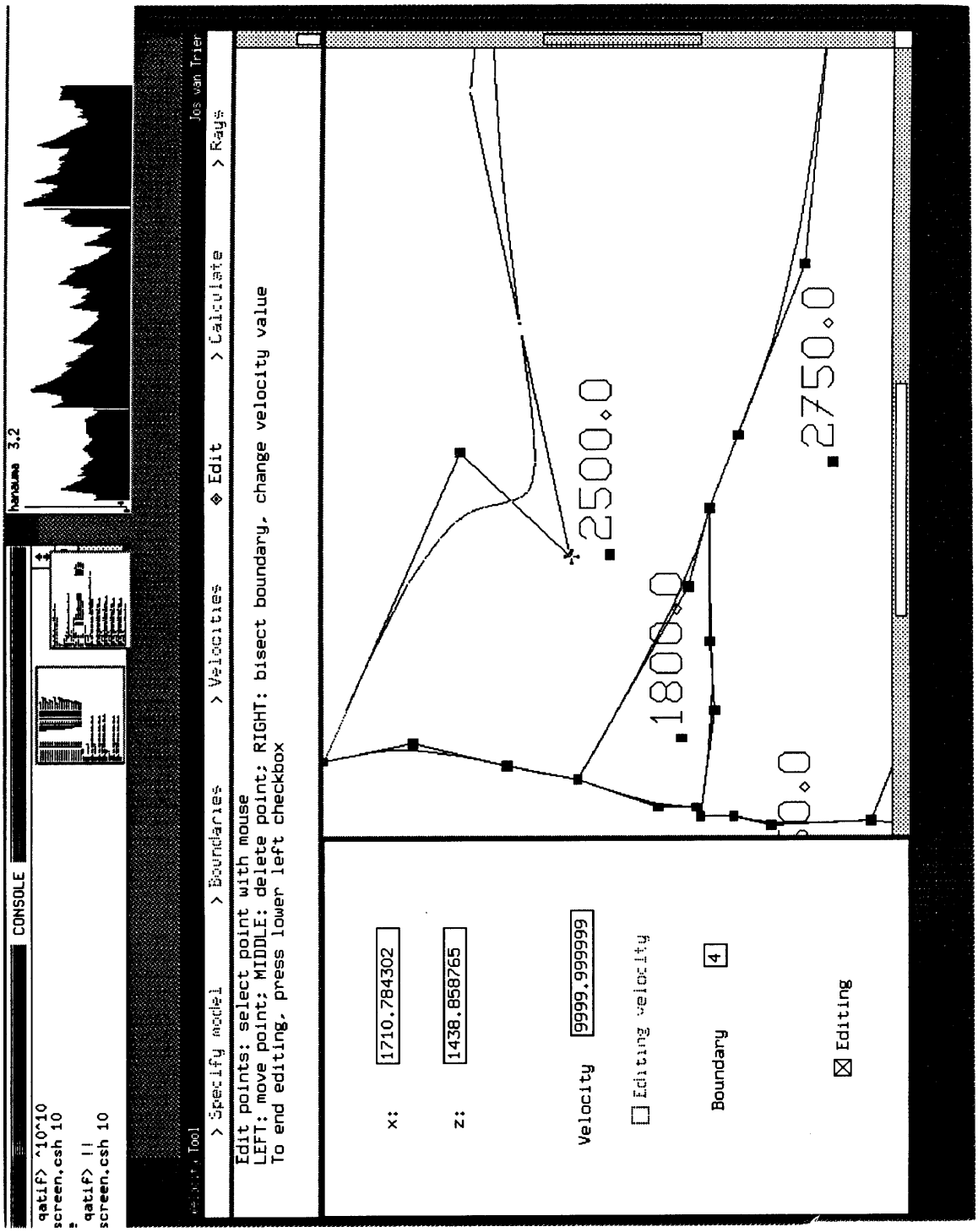


FIG. 7. After smoothing the boundaries, the model can be refined. One can zoom in and re-enter the editing phase. When a boundary point is moved, a rubberspline appears that changes as the mouse is moved.

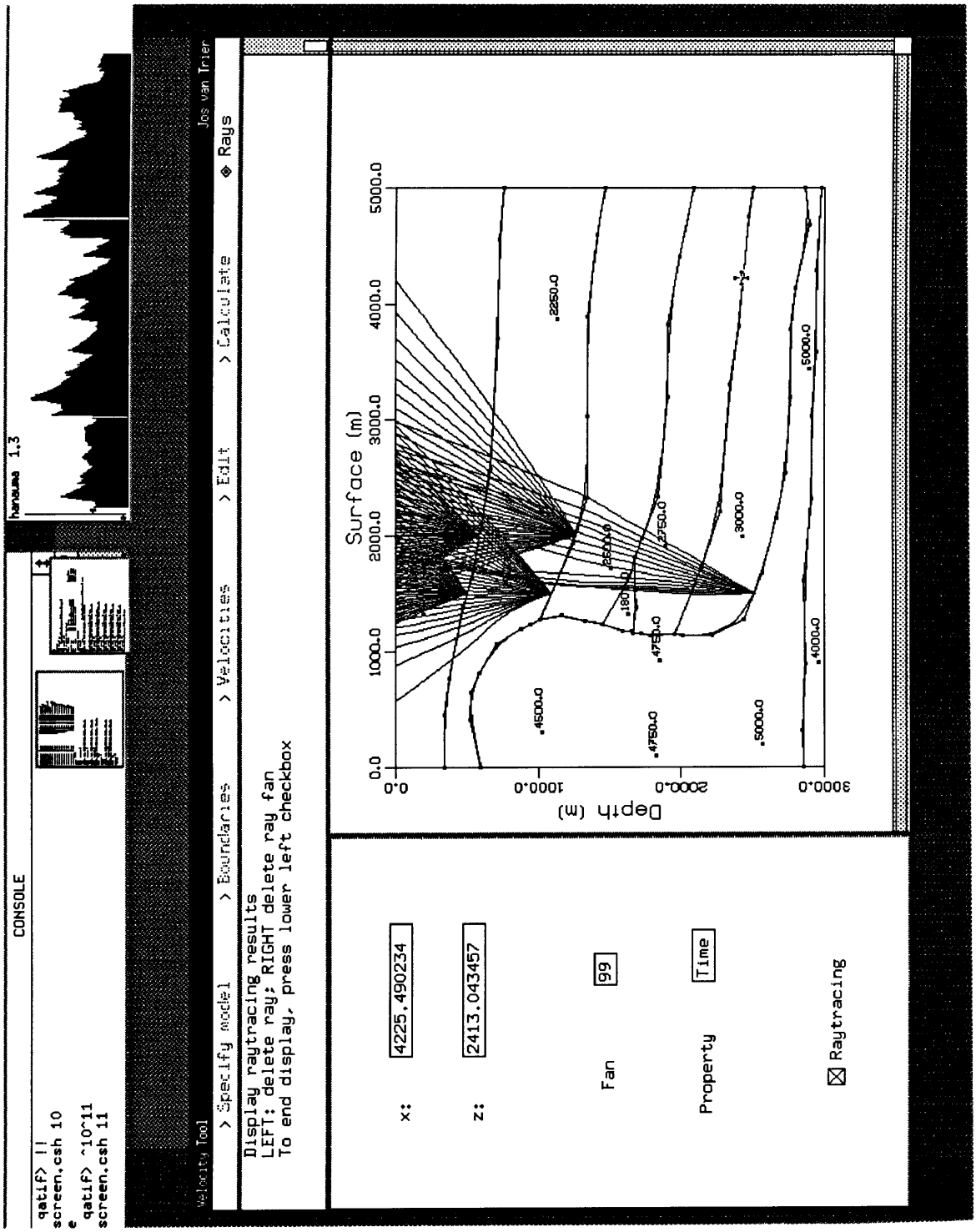
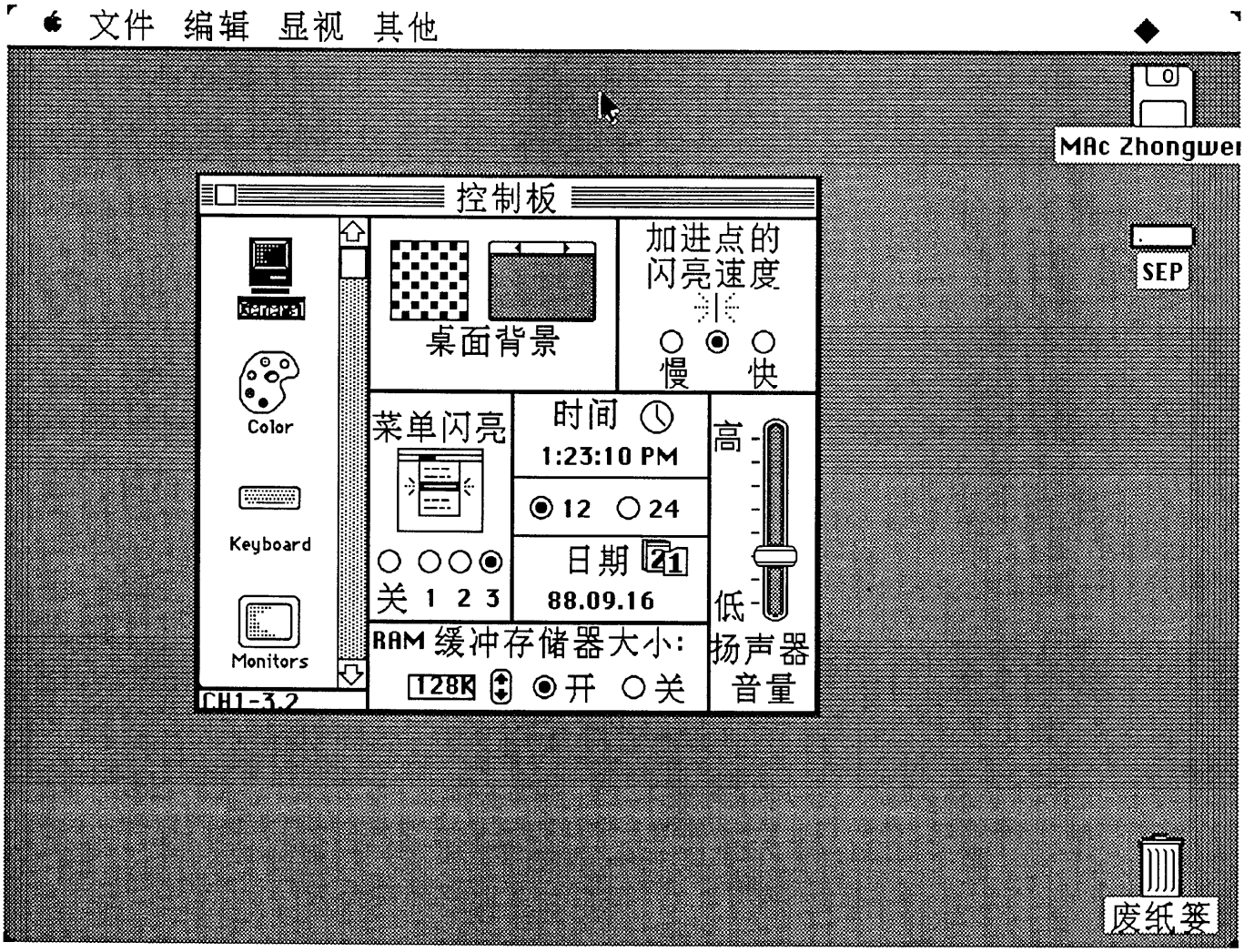


FIG. 8. Ray tracing results. Rays are plotted on top of the model, and can be deleted if one does not want to include them in the inversion. Some work still needs to be done for this tool: for example, the traveltimes or other properties of the ray need to be displayed in a separate window.


文件 编辑 显视 其他
MAc Zhongwei
SEP
控制板
General
Color
Keyboard
Monitors
CH1-3.2
桌面背景
加进点的
闪亮速度
慢
快
菜单闪亮
关 1 2 3
时间
1:23:10 PM
12 24
日期 21
88.09.16
RAM 缓冲存储器大小:
128K
开 关
高
低
扬声器
音量
废纸篓