An object-oriented system for 3D medical image analysis

by P. J. Elliott
J. Diedrichsen
K. J. Goodson
R. Riste-Smith
G. J. Sivewright

As part of a European Commission-funded research project on medical image analysis, we have developed a system aimed at solving a real clinical problem: the outlining of the target volume and organs at risk for three-dimensional conformal radiation treatment planning. The clinical requirement is to make this process less tedious and time-consuming. We show how object-oriented design techniques can be used to good effect in a collaborative project such as this, so that image segmentation and other algorithms from a number of different partners in the consortium can be integrated into one system, for comparative evaluation by the clinical partners. We describe in some detail two of these algorithms, one a straightforward region and volume grower, in which particular attention has been paid to adapting the user interface to suit clinical needs, and the other an interactive tool using b-spline surface patches for direct three-dimensional segmentation. We report on the clinical feedback we have received, which indicates that the most technically advanced algorithms are not necessarily the most useful from a clinician's point of view.

Computer Vision in Radiology, or COVIRA, was a research project in the AIM (Advanced Informatics in Medicine) program of the European Commission. The main goal of COVIRA was to provide pilot application systems for multimodality image analysis in the fields of radiological diagnosis, neurosurgery planning, and radiation therapy planning. Based on a detailed analysis of the clinical requirements at the start of the project, computer science partners carried out research in areas such as image segmentation and image registration. The resulting algorithms were integrated into a number of pilot systems in order that the clinical partners could evaluate them to assess what advances had been made.

Each of the three industrial partners in the project developed its own clinical pilot system incorporating the relevant computer science algorithms. Philips Medical Systems produced a pilot system for general radiological diagnosis and neurosurgery planning, evaluated at the Gregorio Marañon Hospital in Madrid, Spain, and the Academic Hospital in Utrecht, Netherlands. Siemens Medical Engineering produced a pilot system for neuroradiological diagnosis and neurosurgery planning, evaluated at the University Hospital of Tübingen, Germany, and the University Hospital of Leuven, Belgium (near Brussels). The IBM UK Scientific Centre developed a pilot system for three-dimensional (3D) conformal radiation treatment planning using X-ray computed tomography (CT) and magnetic resonance images (MRI). The system was evaluated at the Royal Marsden Hospital and Institute of Cancer Research, Sutton, United Kingdom (near London), and the German Cancer Research Center, Heidelberg, Ger-

0018-8670/96/$5.00 © 1996 IBM

many. The design and implementation of that system is the subject of this paper.

In recent years new methods of planning and applying radiation treatment in 3D have been developed, but they are restricted to specialist centers at present because the planning stage is very time-consuming. Our requirement was to speed up the 3D planning process using the algorithms developed within COVIRA. The clinical application is the treatment of patients with tumors in the brain, breast, and pelvic areas.

Radiation treatment planning. The aim of radiation therapy is to provide a high dose of radiation to a tumor in order to eradicate it. At the same time the dose given to the surrounding healthy tissue, especially those organs that are particularly sensitive to radiation (the organs at risk), should be minimized. With the advent of CT and MRI, anatomical structures can be visualized in 3D, and new treatment planning tools have been developed to take advantage of this.[1] This enables the radiotherapist to find the optimum configuration of radiation beams for the treatment of each individual patient.[2]

One of the major bottlenecks in the treatment planning process is the outlining of the target volume and the organs at risk, a necessary precursor to finding an arrangement of radiation beams that gives a satisfactory dose distribution. Typically, the operator uses a manual drawing program to outline the appropriate parts in each slice of the data. With a 3D image containing 40 or more slices, this procedure can be tedious and time-consuming.

We therefore needed to speed up the process of extracting the relevant contours from the data. In image processing terms this is called *segmentation*. Many different methods of segmenting medical images have been reported over the years,[3–6] but in general the problem of automatic image segmentation is unsolved. Only in special cases is it feasible, so some form of user intervention is usually required[7,8] in order to achieve a clinically meaningful result. In fact, in the AIM exploratory action project COVIRA[9] we showed that automatic image segmentation is in any case not wanted by the clinical users. What they *do* want is a fast system (i.e., one that is faster than manual drawing) in which the user has complete control over the results. It is, after all, the clinician (not the computer!) who is responsible for the patient. The interface for such an interactive system should be designed so that the user's clinical knowledge and experience is complemented by the processing power of the computer in a way that is as efficient as possible.

Clinical pilot system. In building the radiotherapy pilot system for COVIRA, we had to satisfy the computer science partners on the one hand, and the clinicians on the other. This, of course, created a number of problems. As already mentioned, an essential feature of the COVIRA project was to base it soundly on clinical requirements, and everything was subjected to clinical evaluation. Not all computer science partners were used to working directly with clinicians, and they did not understand exactly what the needs were, or how computers were being used in the planning process. For the clinicians to be interested in the new system and find benefits from it, it had to be better (in terms of speed, functionality, ease of use, etc.) than their existing systems.

So, in order to achieve our end—to speed up the process of extracting the outlines of the relevant volumes of interest from the images—we also had to produce a complete, clinically usable system in which the various algorithms could be compared and evaluated. This took the form of TOSCA (TOol for Segmentation, Correlation, and Analysis), which constituted the first part of the radiotherapy pilot system. The other part, which is concerned with definition of the treatment plan itself, was produced by the German Cancer Research Center and is described elsewhere.[10]

A decision for UNIX** as the operating system was reached very early in the project. UNIX workstations are cost-effective and frequently used in this type of application, and all partners had previous experience with UNIX systems. It was, however, left to the individual partnerships between the industrial pilot system teams and their associated clinical evaluators to decide on a specific platform. For the radiotherapy pilot system we chose the IBM RISC System/6000*, running AIX* (Advanced Interactive Executive*) with the X Window System** (supported by the X Consortium) and OSF/Motif**.

To facilitate the incorporation of code from many different partners, it was decided to adopt an object-oriented approach. There were a number of reasons for this:

- The type of application seemed to be well suited for this approach.
- The geographical distribution of subproject teams

required a modular building block architecture, which is the natural object-oriented approach.
- The project was viewed as an opportunity to develop expertise in a new software development technique.

After agreeing in principle on an object-oriented approach to the project, suitable programming languages were evaluated. C++ was chosen for various reasons, influenced by the fact that all code-producing partners had experience in C and some even in C++. Other important advantages of C++ were:

- Ability to compile C code on C++ compilers
- Availability of C++ compilers on all relevant platforms and hence good portability of source code
- Support through various class libraries

The first point was particularly important in the light of the targeted windowing system. The X Window System and OSF/Motif both provide programming interfaces to C. Support for graphics programming under the X Window System is provided through C interfaces via products such as PHIGS (Programmers' Hierarchical Interactive Graphics System), an application programmer interface for 3D graphics, and PEX (PHIGS Extension for the X Window System).

In the next section we discuss the architecture of the system and some of the design considerations in using an object-oriented approach, together with the benefits and problems that arise. Following that, we briefly describe the TOSCA user interface, and the algorithms that have been integrated into TOSCA. We give more detailed descriptions of two of these algorithms, one relatively simple and one more complex, to illustrate some of the difficulties involved in producing a system that is effective and usable in a real clinical environment. Finally, we summarize our experiences, and bring out the lessons we have learned in the expectation that they will be of use to others working on similar problems.

Software architecture

As outlined earlier, the project involved collaboration between three types of teams:

- Algorithm providers who produced specialized image segmentation algorithms
- Pilot system makers who developed application frameworks in which to integrate the various algorithms
- Clinicians who were the actual end users and who specified what functionality they required

The design of our pilot system was influenced by two major considerations. On the one hand we had the clinical problem domain with its own vocabulary and requirements, for which we sought a flexible and extendable mapping onto an object-oriented architecture. On the other hand we were aware that not all of our partners were enthusiastic about the object-oriented design approach. In particular, some partners wanted to reuse and extend their existing C code. So our design had to be able to accommodate code that would merely compile with a C++ compiler, and could not be expected to fit any object-oriented design patterns. In this section we outline the overall object-oriented architecture of TOSCA, and show how code deliveries from multiple, geographically remote teams were integrated despite variations in design and implementation standards.

Top level design. The principal data input to our system is in the form of image data from CT and MR scanners. The data are organized as a set of two-dimensional (2D) image slices. The complete set of slices forms a 3D model of the scanned object. In addition to the pure pixel values, some structural data such as slice thickness, orientation, etc., are available.

A 3D-image and a 2D-slice are familiar terms for clinicians, so they were obvious candidates for classes in our object-oriented design. Other class candidates from the problem domain include contour, segmentation, region of interest, and volume of interest. Each of these represents something that the clinicians talk about.

In addition to these classes from the original problem domain, we introduced classes that are particular to the way we present information to the user and the way the user interacts with the system through a graphical user interface. This group of classes includes views, work spaces, action bars, etc. Additional classes were introduced to handle internal administration tasks where appropriate.

Integration of algorithms. One of the challenges within the system design was the integration of individual algorithms designed and implemented in a way that was definitely not object-oriented. The TOSCA framework had to be flexible enough to ac-

commodate such code deliveries as building blocks. This means that the framework must provide some basic services, such as loading and viewing images, without any of the algorithms or tools being integrated. Algorithms were required to be integrated via a general mechanism and using a defined interface. They also had to be removable without side effects on other algorithms or the pilot system as a whole.

In order to achieve this we introduced the concept of *communication objects* and their associated *interface objects*. A communication object typically provides an application programmer interface (API) to a set of services or an individual algorithm. It basically establishes a firewall between some source code, which may not have been designed with ease-of-use in mind, and an application that needs to use that code. For our algorithms, a communication object implements suitable constructors and destructors that hide the algorithm's internal initialization procedures. It also defines high-level routines that can be linked directly to interface objects, such as push buttons that allow the user to accomplish certain tasks.

The total set of generic objects required to integrate an algorithm into the TOSCA pilot system consists of:

- A communication object that provides an API
- A controller object that provides user interaction components such as push buttons
- A view object that can handle the display of algorithm-specific output

Communication objects are defined in algorithm-specific stand-alone classes. Controller objects and view objects, on the other hand, are defined in algorithm-specific classes that are embedded in a shallow class hierarchy. This allows the implementation of some algorithm-independent behavior in user interface objects.

Design considerations. One of the major design issues we encountered was how to organize the containment relationships of the objects described in the previous sections, and how the objects should communicate with each other. Published object-oriented design methodologies[11–13] provided us with some helpful rule-of-thumb guidelines, but no clear view of how to proceed in our particular case. So we experimented with a number of different approaches, based on inheritance, containment, and callbacks.

Interface propagation. Our initial iteration was influenced by a responsibility-driven design approach.[14] We made a list of all the things our system should be able to do. This included some general guidelines and constraints to allow for the flexible integration of algorithms. We invented a set of peer-level *controller objects* that could communicate with one another. Each object was specialized in a certain application area and offered its services to the rest of the world. In this spirit we had:

- A *Graphics Controller* that could handle color-allocation management according to the current hardware availability (i.e., 8-bit or 24-bit graphics cards)
- An *Image Controller* that would know how to read and write images from and to disk, making all the necessary conversions according to the information in the image headers
- A *View Controller* to organize the layout of image displays and the interaction with them, e.g., browsing through the image slices
- An *Algorithm Controller* to handle integration of the segmentation algorithms

Each of these controllers contained its own set of member objects, which in many cases were complex objects controlling further member objects. Inheritance was used to provide different flavors of the lower-level objects, e.g., different types of views within the View Controller. Virtual functions were used to implement the behavior of those objects accordingly. (Virtual functions are used in C++ to implement polymorphic behavior; a virtual function can be redefined in a descendant class.)

In our initial containment and inheritance hierarchies, the objects representing our problem domain were several levels deep. The obvious problem in this approach is: How does a low-level object within one containment hierarchy access information that is controlled by a low-level object within another containment hierarchy?

Making the low-level objects talk to each other directly introduces an undesirable tight coupling between those objects. Changes in the implementation of a single object can require changes in all the other objects that access it. A different approach is to propagate the required interfaces from the low-level objects through the containment hierarchy to the high-level objects. This reduces the coupling described above. In this scenario, the set of high-level objects, i.e., the controllers, really encapsulate their internal

details and all services are provided through a defined interface. The drawback is that this interface can quickly grow to enormous proportions. A controller's interface now becomes the union of all the interesting interfaces of all of its contained objects. An additional drawback is that the coupling between the objects still exists although it is reduced. A reuse of those objects in other systems would be impossible, since the objects rely on the existence of the controller objects.

Separating behavior from communication. One solution to this interdependence problem was to separate connecting mechanisms from the actual services provided by an object. For this we introduced the concept of *communicator objects*. Again, inheritance was used to implement the different requirements on the various communicator objects. The communicator base class defined the basic application message paths, whereas the subclasses added any further interfaces. This mechanism decoupled an object's behavior from the logistical tasks imposed on it by the system in which it was used. Reusing an object in a different system would now merely require rewriting its communicator object.

Message passing via callbacks. The final design approach we adopted was based on the full encapsulation of objects and the use of callback mechanisms to handle intercomponent communication. A callback mechanism allows a function to be registered somewhere against one or more specific events, then called back at the time that such an event occurs. This approach means that objects can be quite independent of the system as a whole and can easily be reused in other applications, but there is some overhead in programming effort to handle the callback mechanisms.

Lessons learned. The experience of developing TOSCA through these iterations highlighted a number of key design issues, and reminded us to keep the following principles in mind in future designs. It is of fundamental importance to:

- Be able to test and modify low-level objects rapidly
- Have a well-established containment hierarchy that maps neatly onto the problem domain
- Decouple contained and peer-level objects as much as possible
- Provide an efficient method for changes in very low-level objects
- Initiate appropriate and economic updates of the user interface
- Build objects, spanning from the lowest level to the highest levels, which may be reused in other applications
- Have a quick and easy method of integrating new tools

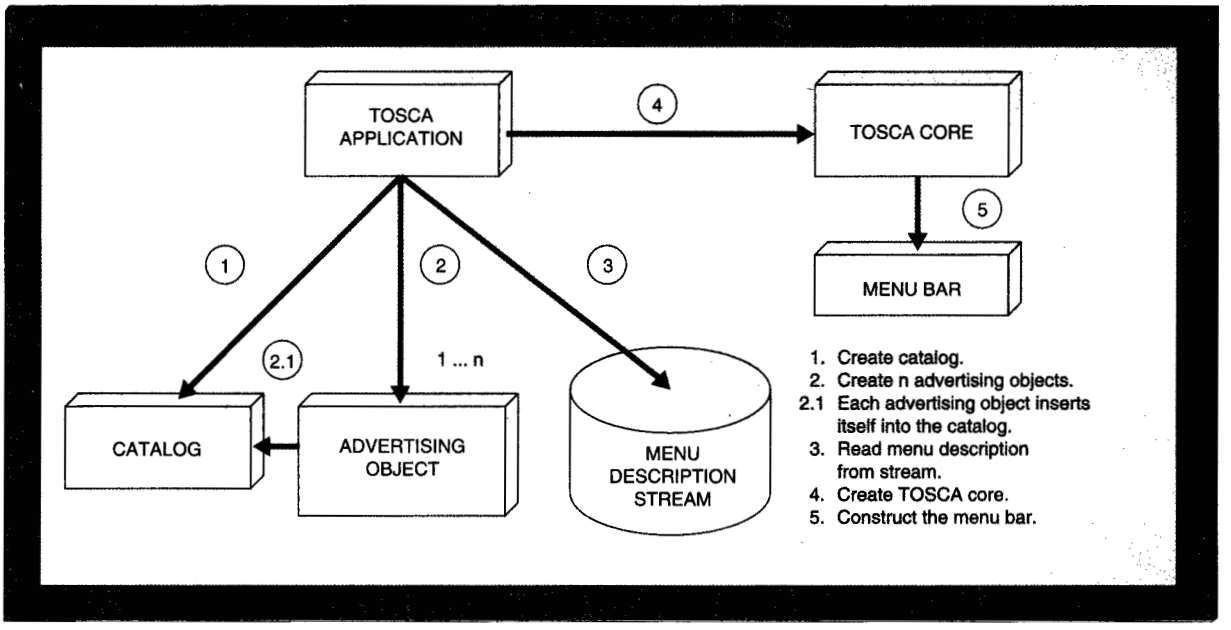
The lessons learned from our initial design approaches prompted us to find a more flexible mechanism for the integration of tools delivered by different partners. This lead to the redesign of TOSCA subsequent to the COVIRA project. This final design approach is best illustrated by the design examples given in the next two sections.

Tool integration via advertising objects. The new TOSCA application is developed as a derivation of a generic application class. The base class contains functionality to provide various services, such as the creation of the main menu bar. The desired menu bar is built up from a character stream supplied either from a file or from information built into the application. In addition the application creates a number of *advertising objects* that advertise various tools. They supply a name by which the tool is known and a global function that can be used to create an instance of the tool. These advertising objects are placed in a catalog that the application base class can refer to. At start-up time the application class uses the menu description and the tool catalog to build up the top-level menu (Figure 1).

Client/server design. The above technique introduced a good deal of flexibility in the configuration of TOSCA, but did not alone give us a complete, quick, and easy method of integrating new tools. Our main goal was to decouple the tools from the core of the application (the TOSCA core) in a form of *client/server* approach. A true client/server approach using serial communication streams or shared memory was not appropriate because of the high overhead in data communications it would introduce, and because we wanted the tools to make use of user interface windows created by the TOSCA core. So in our method both client and server are in the same address space with a single thread of execution.

A *ToscaClient* class was developed that requires a handle onto an object instantiated from an abstract application server class. The functions of ToscaClient use the services of the server object to affect or obtain information from the TOSCA core, and the server object sends application events to the set of

Figure 1 Tool integration via advertising objects



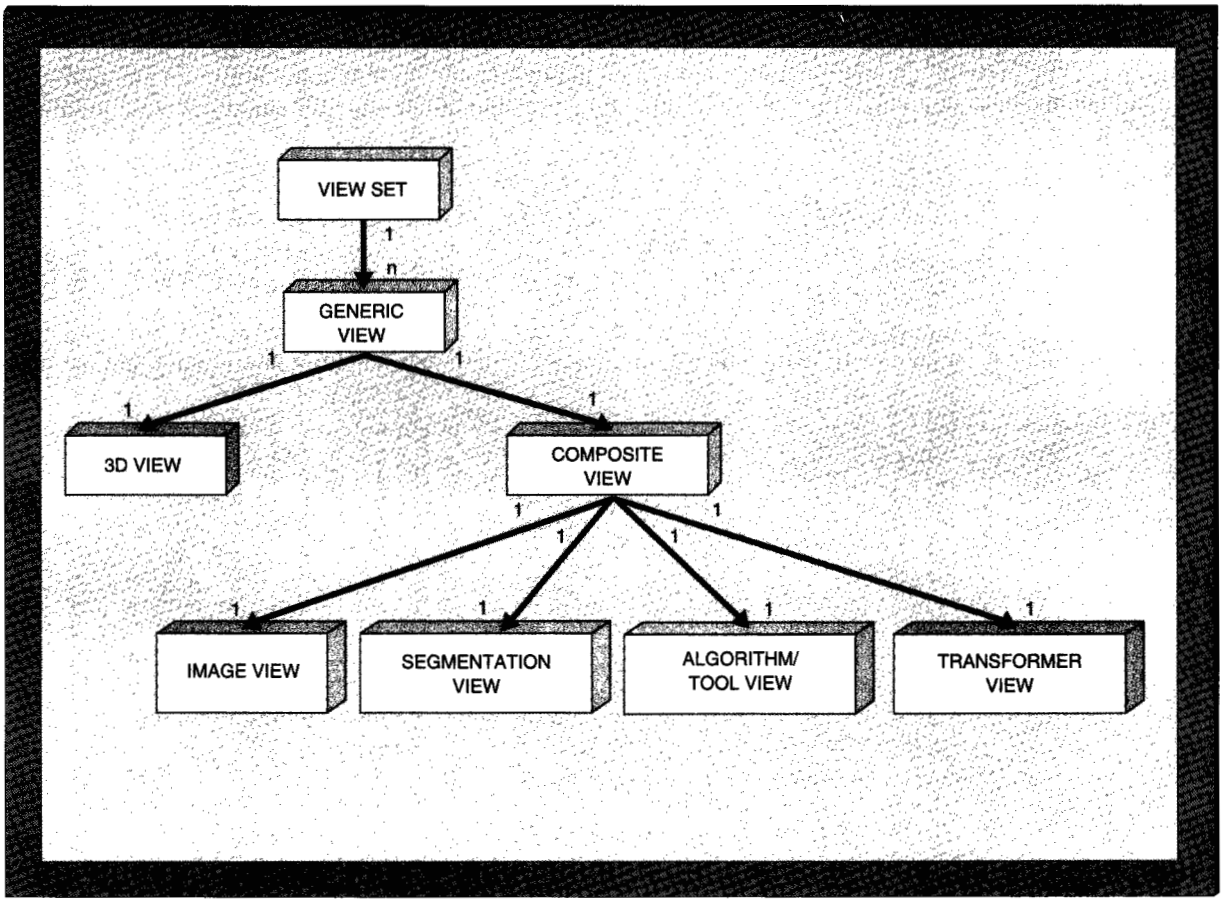
instantiated tool objects. To use this mechanism, the developer makes a subclass of ToscaClient to implement tools such as file readers and writers or similar simple tools that create and use their own user interfaces. For the more complex tools, such as image segmentation tools, the abstract class *ToscaTool* was developed. This class specifies behavior to be implemented in its subclasses. Objects instantiated from its subclasses must respond to requests, for example, to create interface components for insertion into standard places within the interface of the TOSCA core.

Flexible application development. Our design became a very powerful development method when coupled with AIX shared-library techniques for application building. When developing TOSCA a developer will create a source file of advertising objects representing the set of tools needed at that time. This source file is compiled and linked into a shared-library object. The advertising objects each reference a single global function that is itself in a shared-library object, together with the object code that implements that tool class. This means that although our tools share the same address space, adding a new tool or modifying an existing tool requires no changes to the TOSCA core. All that is required is the addition of a new advertising object followed by a quick link stage involving the resolution of less than two dozen symbols. Modification of a tool can be performed without any updates to the application infrastructure. This means that the application development process was made significantly easier and quicker than it had been before.

An additional benefit of this approach is that other developers will be able to write new functions using this derived tool method. The application server object insulates us (the TOSCA developers) from having to expose the higher level objects within our application to the new developers. Furthermore, the same basic set of objects and design methodology can be reused in other applications that have similar characteristics to TOSCA, i.e., the incorporation of a number of functions into a common framework.

Architecture of the TOSCA views. An area that required particularly careful attention in the core architecture of TOSCA comprised the various objects that together make up views onto the image and segmentation data (Figure 2). For a 2D view, the final image on the screen is composed of four view components:

Figure 2 Containment hierarchy of a view set



- The original *image* data
- The confirmed *segmentation* data
- The data provided by the current *algorithm*
- Some *transformer* data such as *zoom* information and *crosswires* to control which part of the image is displayed

Each component is managed by a separate object. For reuse purposes those objects should function efficiently as *stand-alone* service components as well as be able to cooperate with the other view objects to form the higher level view component that is required by the TOSCA application. An issue here is that each stand-alone object requires a set of instance data that describe the view state, but we did not want the awkward maintenance issue of keeping multiple copies of the same parameters in step when combining the individual components in a container *composite view*. We resolved this issue by using shared records.

Shared records. While working on the representation of individual values in the application, we developed a general *value class* that was associated with a name in a *record class*. The record class is held by objects in the application using reference-counted handles. Objects can register callbacks with the record class to observe when the value or the limits on the value change. The owner object of a value can register a RequestRedraw callback on the record. A whole set of user interface components had been developed to expose these values to a user.

This record class helped us as follows. A base-level view object would implement certain of its instance data as a derived class of a *shared record group*. For

example, an *image view* would expose, among other values, the four parameters that represent its zoom state as named records of the shared record group. When constructing an image view, an instance of the group of records is created. The image view itself is the only object with handles onto the instance data. The other basic views would have similarly named members in their group because they need to know the zoom state. Each view installs ValueChanged and RequestRedraw callbacks on its instance data, so that if, say, a zoom parameter changes, then the view would know that—when requested to do so—it must regenerate its window.

However, each view must be prepared to give up its instance data in favor of those generated by a container object. It must be prepared to look in some other record group for records of the same name, to create handles onto those records, and to reinstall its ValueChanged callbacks onto them. Now a *composite* container view can create a shared record group that contains at least the union of the sets of records of its four contained basic view objects. It then passes the record group to all of its contained objects, hence the low-level peer objects have the *same* instance data. This scheme is extended within TOSCA so that a container *generic view* installs its shared record group on both the composite view and a similarly constructed 3D view.

When a user interface component—for example a crosswire for pointing at an image—changes a value in a record, it will first set that value. This causes all ValueChanged callbacks to be called, allowing any object to mark itself as being out of date if it chooses to do so. The interface component will then cause the RequestRedraw callback on the installed record to be called. The object that receives the RequestRedraw message will in general see if it has a RequestRedraw callback installed on itself. If so, then instead of redrawing itself it will call its own RequestRedraw callback. This request propagates itself up through the containment hierarchy until it comes to an object that does not have a RequestRedraw callback installed on it. This object will then redraw itself and send the redraw message to all of its contained objects.

Crosswires example. Let us take the example of the crosswires to illustrate this mechanism (Figures 3 and 4). The actual value that is changed by moving the crosswires is a 3D point in space. The value is owned by a *ViewSet* object, which is an object that contains a number of *generic views*. The ViewSet installs the 3D point value on a selected set of views, which together will focus on that 3D point. When the value is changed, all of the views observing that point may mark themselves as out of date. The RequestRedraw message is sent to the 3D point that is finally received by the ViewSet, which initiates the actual redrawing process. Each view that is now out of date can update itself. Hence stand-alone low-level component objects can cause significant activity within the application. The result is a set of connected views that allow the user to navigate easily through the 3D image (Figure 5). The scheme ensures that the interface will update itself once and once only for each user interaction.

Summary of object-oriented design experience. Despite all the problems and uncertainties involved in undertaking object-oriented software development, it was felt by all team members that it was worth working through the initial painful period. The potential advantages in the later stages of the software life cycle made the investments in the early stages worthwhile. The powerful constructs promoted by object technology can yield elegant, quality code. However, potential users should keep in mind that while success is possible, the object-oriented paradigm itself does not guarantee this! Several iterations may be necessary, as in our case, before the final goal is achieved.

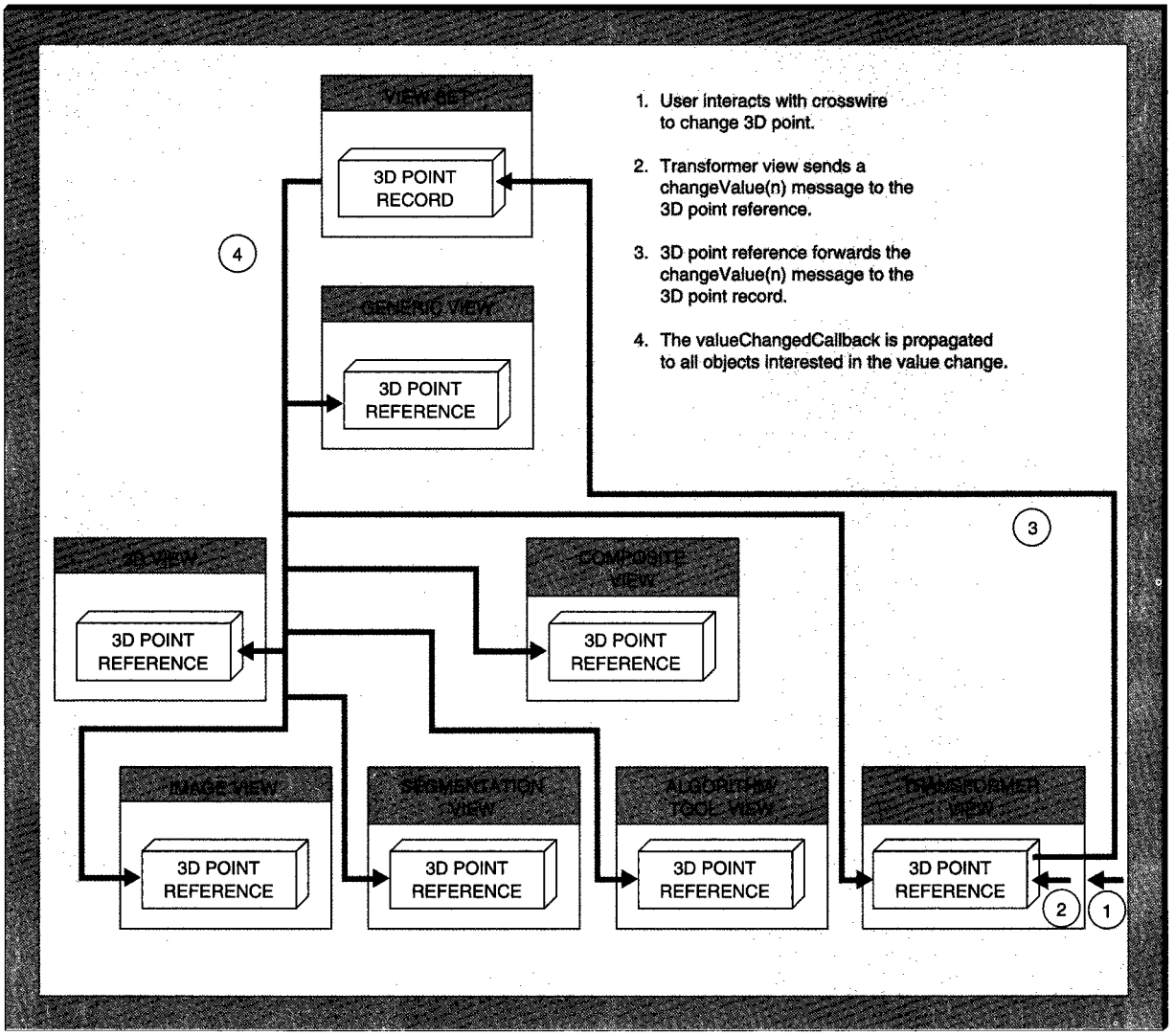
Functionality of the pilot system

In this section we briefly describe some of the concepts behind the design of the user interface for TOSCA, and mention the algorithms that we have integrated into the system from a number of partners in COVIRA. These algorithms represent different approaches to solving the problem of how to speed up the process of outlining volumes of interest in the image data, for subsequent analysis and processing. By integrating a number of these algorithms into the same system (TOSCA), comparative evaluation of their usefulness in a given clinical situation is facilitated. Two of these algorithms are then described in more detail, and we report on the clinical feedback we have received.

User interface design. To achieve clinical acceptance, TOSCA's user interface (Figures 5 and 6) was designed with three important factors in mind:

- Similarity to tools used previously
- Simplicity
- Style guide compliance

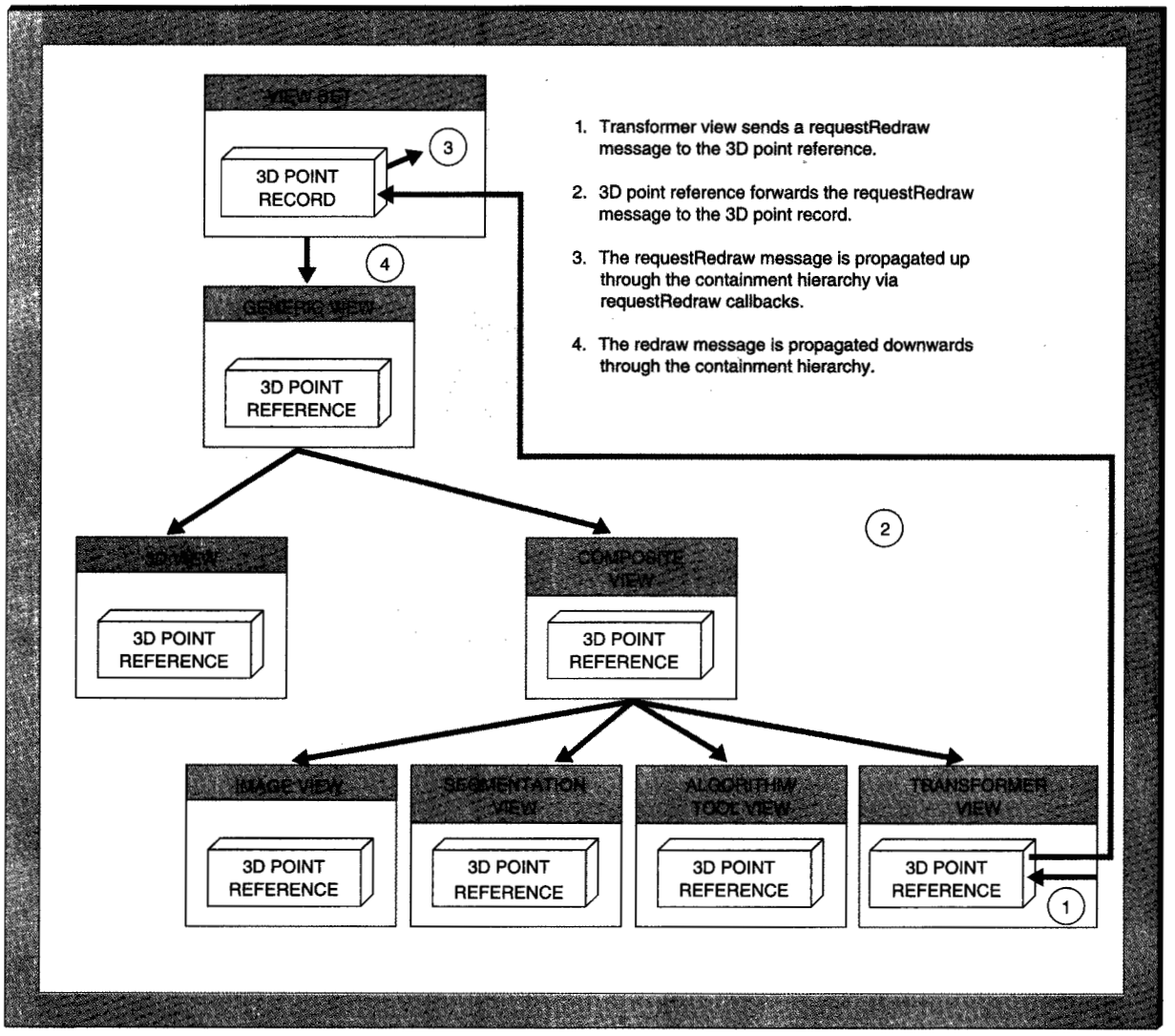
Figure 3 Propagation of a changeValue message after a user interaction



Similarity to tools used previously. TOSCA is a successor to a tool developed at the German Cancer Research Center called TOMAS (TOol for MAnual Segmentation).[15] It was felt to be good design policy to keep many of its good user interface features, so that clinicians familiar with TOMAS would immediately relate to TOSCA. They could thus concentrate on evaluating the benefits of the new algorithms without the confusion of a different interface. However, it is of course just as important to keep the user interface simple for the benefit of new users.

Simplicity. While some clinicians are computer literate, many are not. Graphical user interfaces can be confusing to computer novices—few hardened computer users can remember the first time they saw a windows-based application with pop-up windows appearing and disappearing all over the screen. So TOSCA has been given limited preset alternatives for its screen layout, enabling the users to concentrate on achieving their goals without the distraction of first having to learn too much about how the system operates. Similarly, whichever algorithm is invoked, the control panel appears at the same place on the

Figure 4 Propagation of a requestRedraw message after a value change

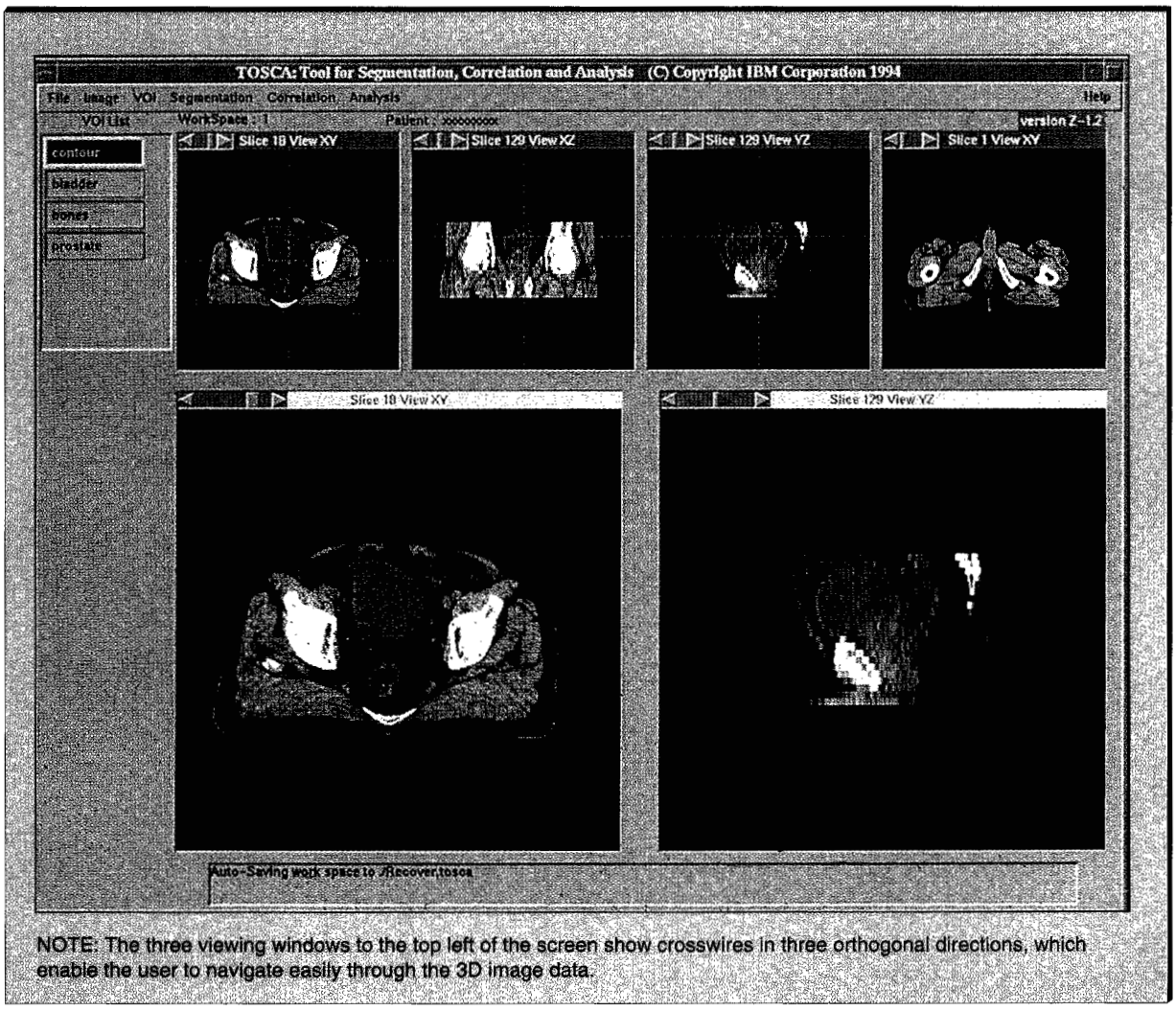


screen. We use pop-up windows only for functions that are transient or infrequently required, so that the screen does not get cluttered. Where choices need to be made, we always provide sensible defaults, which are easily configurable to suit a particular application. This ensures that the novice can get started with minimal tutoring, while complete flexibility is provided for the experienced user.

Style guide compliance. The *Motif Style Guide*[16] has been used whenever possible to help define, for example, which mouse button should be used for an operation (e.g., left button for selection, middle button for action, right button for pop-up menus). Since the function that will be selected or put into action depends on the algorithm that has been invoked, TOSCA provides a helpful reminder (at the bottom of the screen) of the interactions that are possible at that time.

Using TOSCA. TOSCA is essentially a toolbox for performing image segmentation, correlation (registration), and analysis operations, where in general each tool or algorithm corresponds to the work of a

Figure 5 TOSCA's graphical user interface, showing a set of pelvic CT images



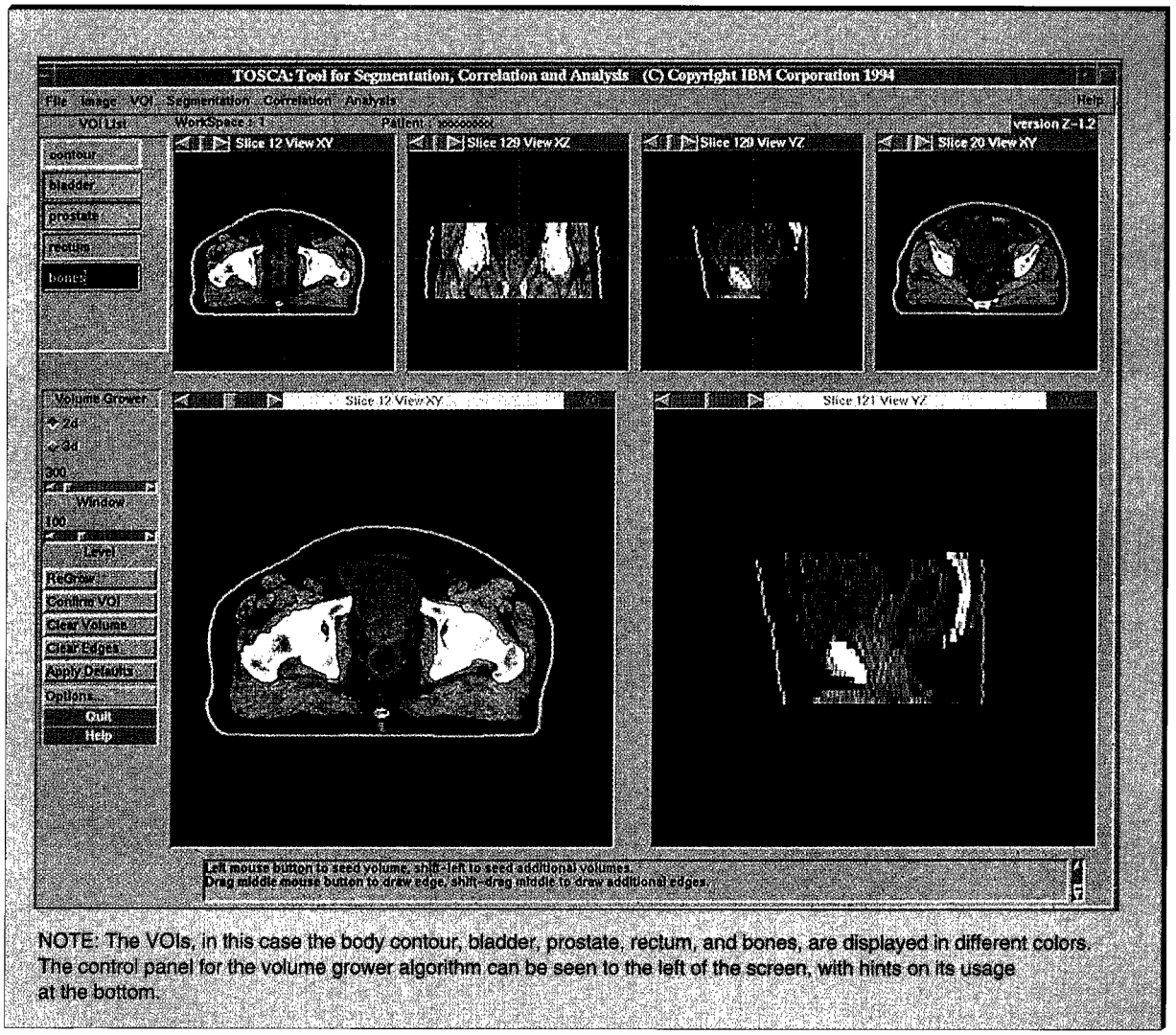
NOTE: The three viewing windows to the top left of the screen show crosswires in three orthogonal directions, which enable the user to navigate easily through the 3D image data.

COVIRA computer science partner. The algorithms are invoked as options from a first-level submenu of the main application window's menu bar.

Segmentation. The segmentation algorithms within TOSCA produce VOIs (volumes of interest), represented by stacks of contours that are parallel to the primary image slice orientation. Complex volumes can be built up, as the data model allows the user to define multiple outer boundaries for a given VOI within a given slice. In addition the data model allows regions within an outer boundary to be specifically excluded from the VOI by definition of one or more inner boundaries. This provides for descriptions of volumes that split and merge, or have disjoint parts, or are tubular, for example. There is a blend of leading-edge academic work along with some less sophisticated algorithms, which by virtue of their speed and flexibility lend themselves to clinical acceptance.

In all cases the objective is to produce a set of contours that accurately correspond to the boundary of the chosen structure (an organ or tumor, etc.), as shown in Figure 6. In some instances part of the boundary may not be visible in the image, but has

Figure 6 The result of using TOSCA to find the outlines of selected VOIs within the images shown in Figure 5



NOTE: The VOIs, in this case the body contour, bladder, prostate, rectum, and bones, are displayed in different colors. The control panel for the volume grower algorithm can be seen to the left of the screen, with hints on its usage at the bottom.

to be inferred by the users based on their clinical knowledge. If the algorithm itself does not provide the user with the flexibility to do this, a subsequent editing step will be needed.

Correlation. Sometimes magnetic resonance imaging (MRI) is used to scan the patient in addition to CT. CT is invariably used for radiotherapy planning because it is geometrically accurate (unlike MRI, which may suffer from image distortion), and also because it provides electron density information needed in the calculation of radiation dose distribution (this is done during the definition of the treatment plan, which is outside the scope of TOSCA). MRI has better soft tissue contrast than CT, and so can be useful in defining the exact position of the tumor or organ boundary. However, since the CT and MRI images probably have different slice thicknesses and resolution, with different patient positioning, it is first necessary to register or correlate the two sets of data before they can be used in conjunction with each other.[17] Some of the work in COVIRA was directed

toward this, and one such algorithm was installed in TOSCA.

Analysis. Finally, quantitative analysis of the VOIs is important in some applications. In radiotherapy planning, it is the boundary of the VOI that is needed in order to define the position, shape, and size of the radiation beams that are used to treat the patient. But in other cases it may be necessary to know the actual volume of the tumor, so that its response to treatment (by chemotherapy, for instance) can be measured. And there are many other applications outside the field of health care in which accurate measurement of volumes contained in 3D images is needed. Once the boundary of the volume is defined using the segmentation algorithms in TOSCA, it is of course relatively straightforward to calculate the volume or indeed any other measurement parameter that is required.

Installed tools. The following tools are part of TOSCA:

- The *2D edit* algorithm, produced by the IBM UK Scientific Centre, provides all the usual manual tools for creating and editing contours, and in addition some region-based tools for morphological editing operations. It is used to modify the results of the other segmentation algorithms when necessary, or to input complete contours manually (for example, with small organs that have poorly defined boundaries in the image, this may be the quickest solution).
- A *volume grower*, developed by the IBM UK Scientific Centre, provides fast creation of 2D regions or 3D volumes using sample image statistics from around a seed point selected by the user. A 2D region grown in one slice provides the basis for a region in the next slice.[18] This is described fully in the next section.
- *Snakes* (active contours) are usually the edge-seeking variety.[19] In COVIRA, we found better results using a statistical snake developed by the University of Sheffield, United Kingdom. Here the expansion or contraction of the snake boundary is controlled by the statistics of the image region it encloses (in a similar manner to the volume grower).[20] This algorithm also has features that allow interpolation of contours. So, for example, the user can define the boundary of a VOI on just a few of the slices in which it is present, and can then use interpolation to rapidly complete the VOI on all slices.
- An *iso-contour* algorithm from the University of Genoa, Italy, accepts a user-defined seed point to generate a range of suggested contours, which have varying degrees of similarity to the seed region. The user can quickly select the preferred contour by moving a slider in the control panel.[21] This is similar to the volume grower, but the method of providing user interaction is different, as is the way of generating the similarity statistics.
- A *contour refinement* algorithm (also from the University of Genoa) takes an existing contour and uses local edge information in the image to refine the contour to better fit the image data. The method is similar to the snakes algorithm, but with some modifications. This tool can also transfer contours to adjacent slices for successive refinement.[21]
- An *edge grouping* algorithm, developed at the Federal Institute of Technology in Zurich, Switzerland, uses edge information in a 2D image and groups the edge fragments together to produce a graph of connected regions. The user can select contours by pointing to parts of the graph in sequence. The optimum path to the next point is displayed, providing a quick and simple method of selecting between alternative edges. The system will also find the best path to close the contour back to its starting point, so that in many cases only two or three mouse clicks are needed to generate the complete contour.[22]
- The *3D edit* algorithm from the IBM UK Scientific Centre provides ways of creating true 3D surfaces directly. The surfaces can be scaled, rotated, shifted, and deformed. The user can interact with the cross sections of the surface models in any of three orthogonal directions, in order to match the model to the image data. Further details are given in a later section.
- The *template atlas* has its roots in radiotherapy. The German Cancer Research Center used surface models of organs at risk within the human head to generate contours that are overlaid on the 2D images. The models can be shifted, rotated, and scaled either *en masse* or individually until they conform to the patient data. This provides a rapid method for defining the complete set of organs at risk in the head.
- The *correlation* algorithm performs 3D image registration. Developed by the University of Leuven, it uses surfaces (defined by the segmentation algorithms), individual point landmarks (i.e., anatomical features that are visible in both images), or a combination of the two as references in the search for a 3D rigid transformation that brings one image into alignment with another. A chamfer distance transform method is employed.[23]
- The IBM UK Scientific Centre has also made avail-

able tools for quantitative *analysis* of VOIs. Distance, area, and volume can be measured, and histograms and statistics can be produced.

In the following sections, we give detailed descriptions of two of these algorithms (volume grower and 3D edit), together with the feedback we have obtained from the clinicians.

Volume grower. This algorithm is simple, robust, and fast. The arrangement of the user interface is important for the success of the method in a clinical application, and how this was achieved is described below.

Description. The method can be used in two modes: 2D region growth applied slice by slice or direct 3D growth applied to a stack of images. (Which mode is more useful depends on the image and the volume to be segmented from it.) Region growth starts from a seed pixel selected using a mouse. The four neighboring pixels (six in 3D) are added to the region (volume) if they meet the similarity criterion given below. Region growth typically takes a fraction of a second, making the method truly interactive (Figure 7A).

Controls. A pixel is included in a region if its intensity is within a window W, around a mean level L. The values for L and W are computed from the statistics of a small sample region (about 20 pixels in size) located centrally over the seed point. L is set to the mean of the sample region, and W is computed by multiplying the standard deviation of the sample region by a given scaling factor C (typically 2.0 for CT and MR images). Values for L and W can be adjusted manually, if needed, using sliders in the algorithm control panel. This means that the user can take control of the process if the computed values of L and W do not give the desired result. Another option is to use predefined values for each combination of image modality and organ type, if the user finds by experience that particular values give good results for that organ.

Since the growing region sometimes "leaks" into surrounding areas where there is insufficient contrast between adjacent organs (Figure 7B), it is possible to manually draw edges to inhibit region growth through these points. Drawing such a barrier will cause the region to regrow from its seed point up to the barrier, so that the effect for the user is to cut off the unwanted part of the region (Figure 7C). By providing this facility, the usability of the method is greatly enhanced.

3D growth. Region growth in 2D can be extended into neighboring slices by pushing an up- or down-arrow icon button. By successively repeating this procedure, a connected 3D volume can be computed. This approach allows the shape of the volume to be checked slice by slice (and hence corrected if necessary) but involves more user interaction than the direct 3D approach. In the latter mode, it is possible to start 3D growth from an existing 2D region, instead of just a seed point, which improves the initial statistics. The method is extremely fast, typically 10 seconds for the whole brain volume.

Starting 3D growth from an existing 2D region, instead of just a seed point, improves the initial statistics.

Upon completion of region growth, the boundaries are computed by tracking around regions marked in the work image. Small, spurious boundaries are discarded. Optionally, all interior regions may be removed, and a variable amount of boundary smoothing may be applied.

Clinical feedback. The region-growing method works well in those areas where there is sufficient contrast-to-noise ratio to discriminate between adjacent structures. Otherwise, the user's anatomical knowledge (and manual intervention) is needed to define a suitable boundary location. The method is well liked by our clinical evaluators because it is fast, easy to use, and easy to understand. Because they understand it, they can usually predict the result it will produce, which means they do not need several attempts to get what they want.

Taking pelvic CT data as an example (prostate cancer in men is a very common form of cancer, and often treated with radiotherapy),[24] the requirement is to outline the body contour, the prostate itself, and organs at risk such as the bladder, rectum, and pelvic bones (especially the femoral heads). The body

Figure 7 The volume grower algorithm being used to outline the bladder in slice-by-slice mode

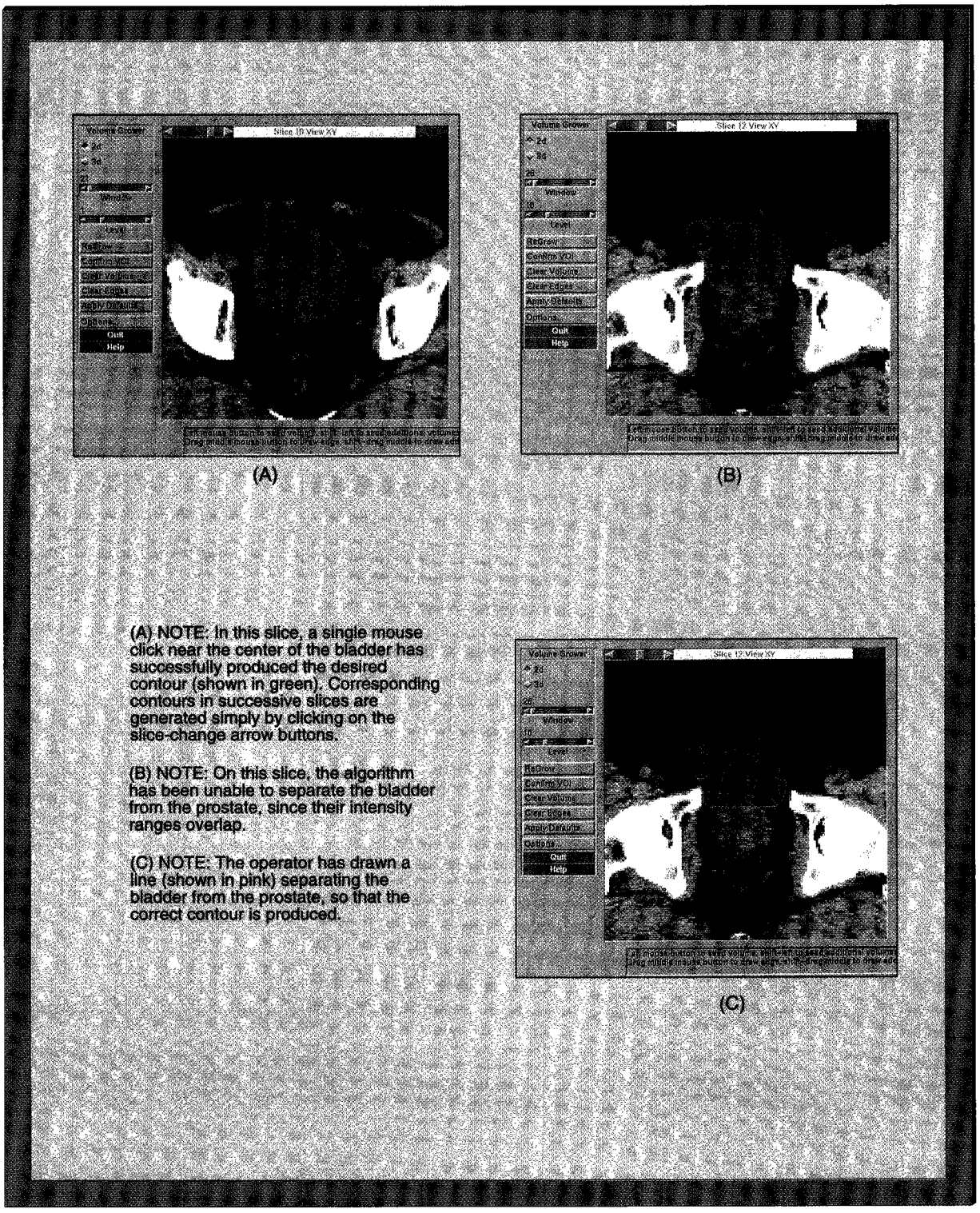
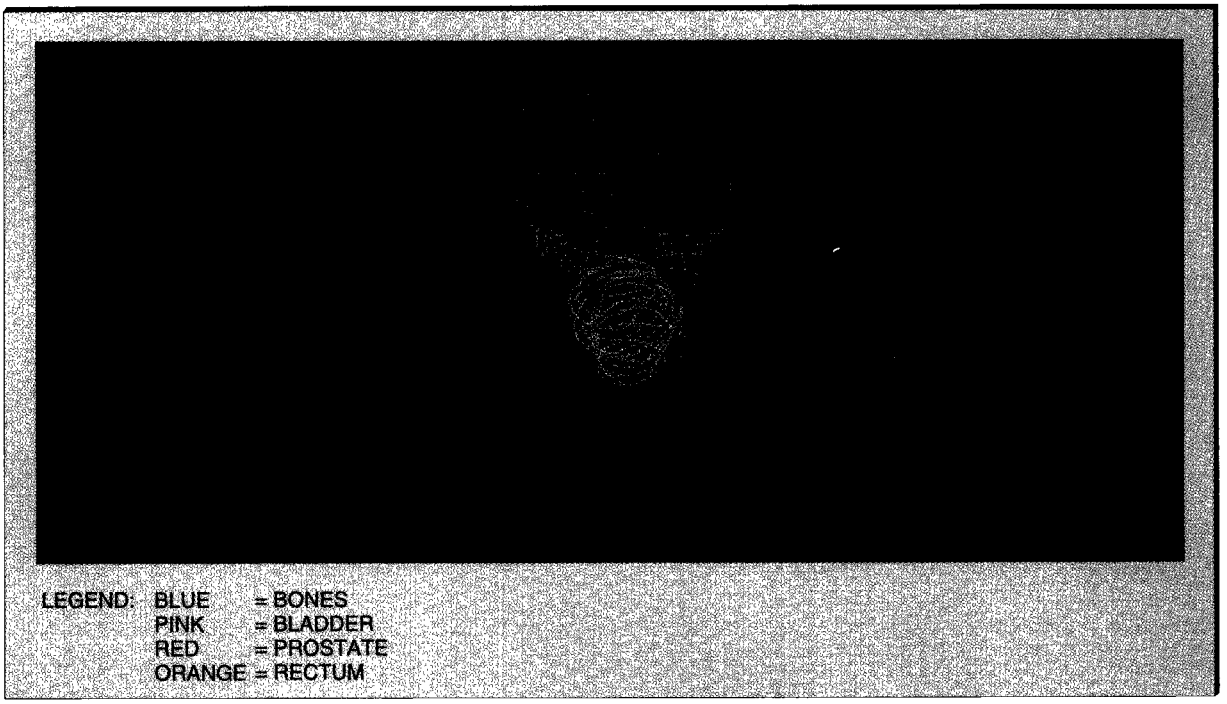


Figure 8 Volumes of interest from Figure 6 displayed as stacks of contours in 3D, using PEX to generate the 3D graphics



contour and pelvic bones have high contrast to their surroundings in CT images, so the 3D volume grower (with default settings) works very well in these cases. It is better than simple thresholding since only those pixels connected to the seed point are included in the VOI, and unwanted interior contours are automatically discarded (Figure 8). In many slices the bladder can successfully be outlined by the region grower, but occasional manual intervention is needed where it touches the abdominal wall or the prostate. So the slice-by-slice method is appropriate in this case, with smoothing automatically applied because the bladder always has a smooth outline (the CT scans are taken with the bladder full).

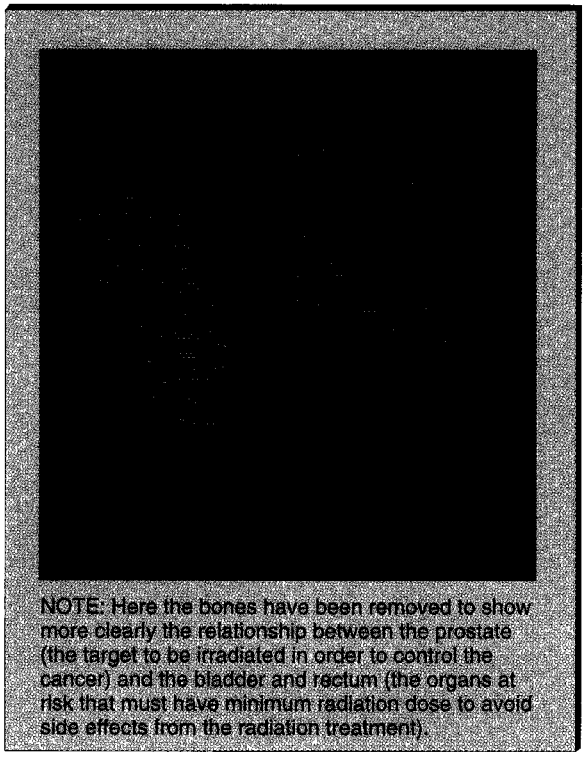
The prostate itself has similar intensity to the bladder and rectum, and touches or even overlaps them in places (Figure 9), so it is difficult for any algorithm to successfully segment this organ on all its slices. The rectum is also difficult, due to the presence of bowel gas, which has very different intensity to the rectal wall. However, these organs are quite small, so the clinical users can outline them rapidly, even with manual intervention. Taking all the organs into account, the time saved compared with the manual methods currently used in routine clinical practice is significant.

3D edit. This method enables direct 3D segmentation of image data by manual interaction with surface models using a b-spline parameterization.[25] The surface segmentation can be viewed in 2D as contours overlaid on the image in three orthogonal views (which also enable interaction with the model), or as a 3D rendered visualization.

Surface parameterization. The surfaces are formed from b-spline surface elements. This has a number of advantages from a user interaction perspective:

- It is conceptually straightforward, since the surface is determined purely by the positions of a set of control points.
- The surface is deformed simply by moving the control points.
- Moving a control point has only a local effect governed by the order of the surface.
- The surface can have varying degrees of smooth-

Figure 9 Volumes of interest as in Figure 8



NOTE: Here the bones have been removed to show more clearly the relationship between the prostate (the target to be irradiated in order to control the cancer) and the bladder and rectum (the organs at risk that must have minimum radiation dose to avoid side effects from the radiation treatment).

ness largely independent of the number of control points.

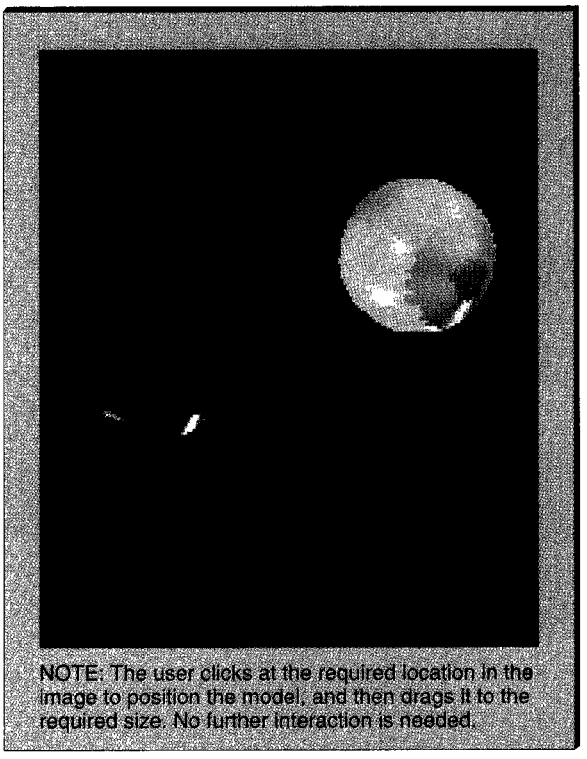
There is, however, a compromise between algebraic and geometric complexity. Although b-spline surfaces are geometrically "nice," it is more difficult to calculate intersections, for example, than with a simpler triangulation representation.

Model generation. A geometric model or object is made up of many b-spline surface patches joined together by sharing their common-boundary vertices. Each model is normalized by an associated reference datum point, which in the case of a sphere coincides with the center. This makes translation a trivial task. There are at present three methods used in the construction of surface models:

1. Simple geometric models such as ellipsoids and elliptic cylinders. The cylinders may be closed at both ends by either a flat plate or a dome termination. This is useful for rapid segmentation of organs that are nearly spherical, such as the eyes (Figure 10).
2. From an occluding boundary. A model is constructed by performing a surface of revolution (SOR) about an axis determined from the occluding boundary.
3. From a set of supplied surface data points, e.g., anatomical models obtained by prior manual segmentation of 3D images taken from a "typical" individual. This is particularly useful for structures that have poor contrast to the surrounding tissue, so that other segmentation algorithms (e.g., the volume grower) are of limited use. An example of such a model is the brainstem (Figure 11). The model is adapted to fit the patient data using the tools described below.

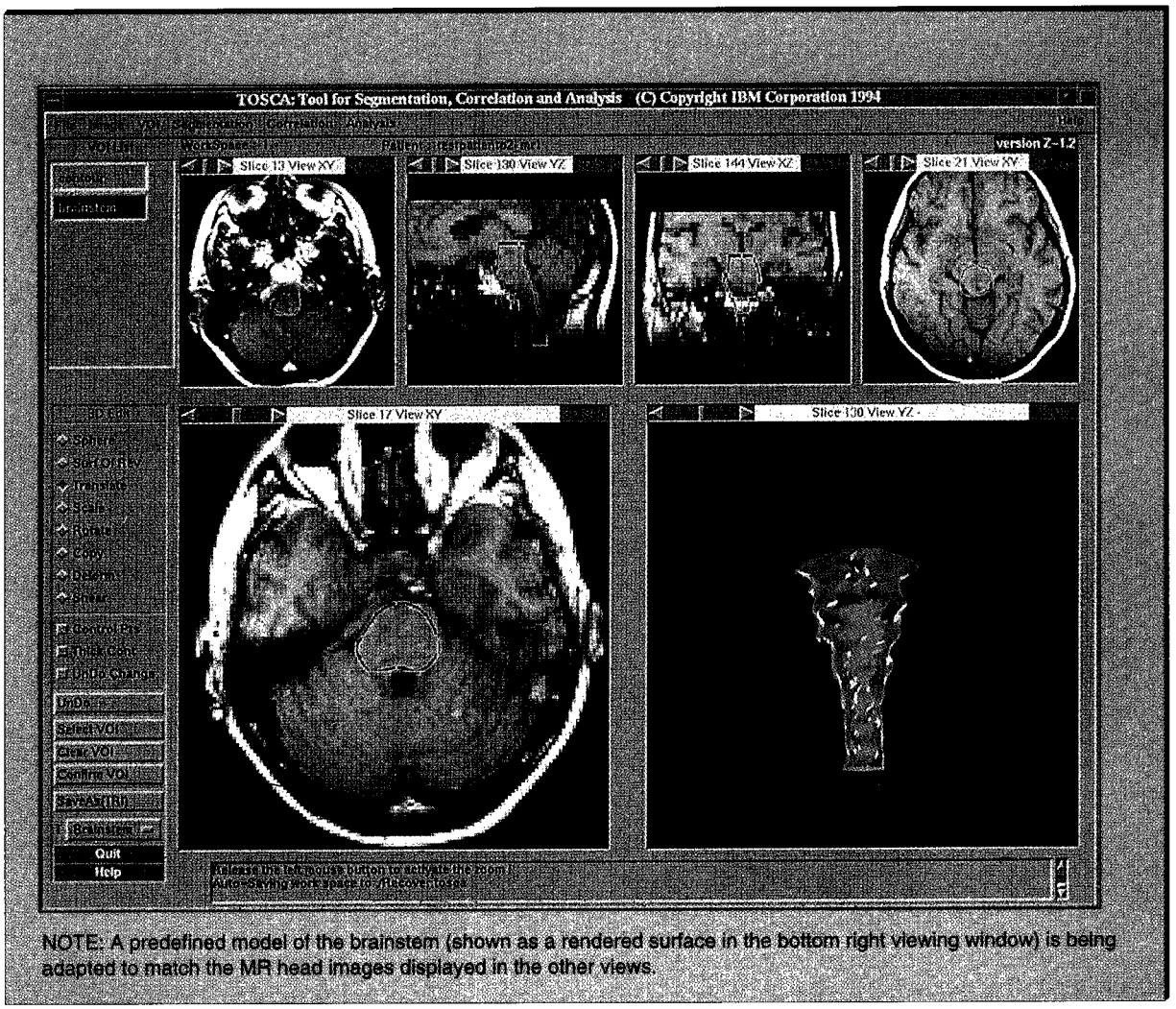
Model editing tools. Editing tools are provided to match the model to the image data. The tools use at least two orthogonal slices to control the manipulation. Tools are available for translation, scaling,

Figure 10 Surface models of the eyes produced simply by using the ellipsoid model in the 3D edit algorithm



NOTE: The user clicks at the required location in the image to position the model, and then drags it to the required size. No further interaction is needed.

Figure 11 The 3D edit algorithm in use



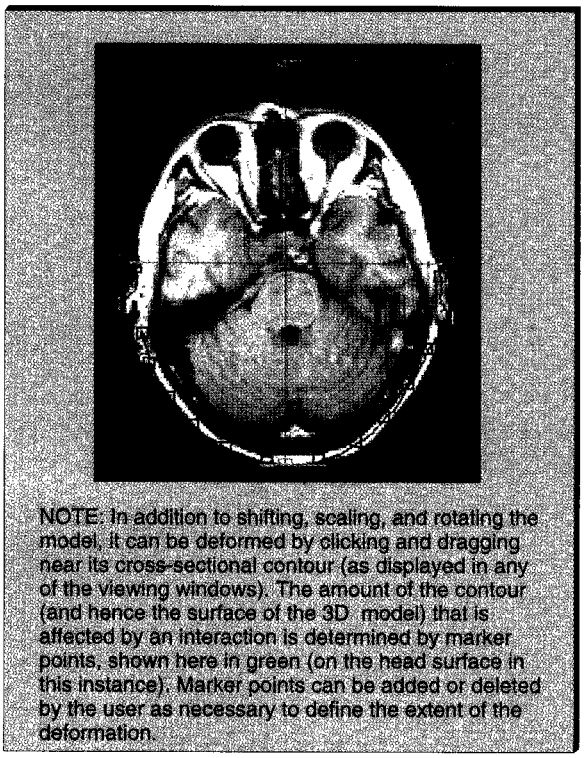
NOTE: A predefined model of the brainstem (shown as a rendered surface in the bottom right viewing window) is being adapted to match the MR head images displayed in the other views.

and rotation in 3D. The scaling can also be used to obtain a mirror image of a surface. Other tools allow shearing, surface deformation, copying, and deleting.

Translation applies to the whole surface, whereas shearing applies only to the surface portion intersected by a thin slab. For surface deformation, the user interacts with the model by dragging the surface in orthogonal planes. The model's shape is shown as 2D contours on three orthogonal views where the model surface intersects the volume data. It can also be visualized in 3D as either a shaded surface or as a wire frame. Surface deformation can be carried out at various scales, allowing from small scale up to global surface changes. This is done by allowing the user to add marker points on the viewed contours. These are used to define the affected surface area (Figure 12).

Application. Although we investigated methods of interacting with the 3D model directly, we decided that using a 2D interaction mechanism on the 3D surfaces was the most appropriate technique in our envi-

Figure 12 Manipulation of the model



NOTE: In addition to shifting, scaling, and rotating the model, it can be deformed by clicking and dragging near its cross-sectional contour (as displayed in any of the viewing windows). The amount of the contour (and hence the surface of the 3D model) that is affected by an interaction is determined by marker points, shown here in green (on the head surface in this instance). Marker points can be added or deleted by the user as necessary to define the extent of the deformation.

ronment (i.e., with relatively low-cost, standard workstation hardware). Simulating a 3D environment in the inherently 2D world of the computer screen, even when using aids such as rendering and stereography, is not at present a practical solution. The designer's problem is twofold: visualizing the tool and its relative position and orientation in a mass of 3D anatomical data, and controlling the tool's position, etc., with commonly available computer hardware. Virtual reality techniques may give an acceptable (and affordable) solution in the future, but not in the time scale of this project.

Surface contours can be viewed as either thin (single-pixel) connected contours, or as thick contours (regions). The latter highlight the partial volume effect, in which, due to the thickness of the slice, the surface leaves the slice at a different position from its entry point. The thick contours are obtained by the intersection of the surface model with the relevant slice, taking into account its thickness.

Clinical feedback. There are some instances of organs at risk that are nearly spherical (such as the eyes and the femoral heads), where the 3D edit technique has proved to be very useful. However, in general, the algorithm has had limited acceptance by the COVIRA clinicians. There appear to be a number of reasons for this.

1. Clinicians are used to thinking in terms of slices, and not in 3D. This means that slice-by-slice algorithms, for instance, are more accessible to them than true 3D algorithms.
2. It can be difficult to appreciate the effect that an interaction in a 2D window has on the 3D model, since each interaction affects several slices. Hence controlling the surface shape over adjacent slices can be difficult.
3. Because of this complexity, it may take longer to achieve the final objective than with a 2D method.
4. The clinicians always want to refer back to the original 2D image slices in order to check the segmentation results.

We started developing this algorithm on the assumption that segmentation in full 3D ought to be faster and more effective than working on individual slices. It was thought that the implicit slice-contour connectivity in 3D, and thick slice contours (which reveal surfaces tangential to the working slice such as optic nerves) would be of benefit. The clinicians have not yet reported that this is the case. The fact that a single interaction affects several slices is beneficial in some instances, but on other occasions may be counterproductive (e.g., when an already accepted contour is unwittingly changed).

Further work is required to provide more accessible interactive tools in order to make the method generally acceptable to the clinicians.

Other algorithms. The other algorithms installed in TOSCA are described in the references given for them. In terms of clinical feedback, it is generally true that the more complex an algorithm is, the less likely it is to be given a good rating by the clinical evaluators. This effect is modified by the amount of effort put into designing the user interface for the algorithm. It has to be remembered that the users are only aware of the interface—if it does not allow them to use the algorithm to solve (or at least reduce) their problems, then they will not be interested in that algorithm.

Conclusions

We have described TOSCA, the system we developed as part of the COVIRA project in order to address

clinical requirements in the field of 3D conformal radiation therapy planning. We have discussed the use of the object-oriented paradigm and the design iterations we went through before achieving a design that met our requirements, i.e., to produce a system in which image segmentation and other algorithms from a variety of partners in the project could be integrated successfully.

Two of these algorithms have been described in some detail, together with the feedback we received from the clinical evaluators. It is apparent from this that the most technically advanced algorithms are not necessarily the most appropriate in a real clinical situation where speed of operation and ease of use are of paramount importance. It is undoubtedly necessary to research new methods that will have potential when the technology has advanced beyond that currently available. However, it also seems to be important to work on relatively simple algorithms, and configure them in such a way that they can be used to solve clinical problems today.

Acknowledgments

We gratefully acknowledge the financial assistance given to this project by the European Commission under the AIM program, and the cooperation of those COVIRA partners whose work has been integrated into TOSCA. We are particularly indebted to our radiotherapists—Anthony Neal of the Royal Marsden Hospital and Cajus Seyfried of the German Cancer Research Center—for their patience, understanding, and encouraging comments.

Participants in the COVIRA consortium were as follows.

Industrial partners: Philips Medical Systems, Best, Netherlands (prime contractor) and Madrid, Spain; Philips Corporate Research, Hamburg, Germany; Siemens AG, Erlangen, Germany, and Munich, Germany; and IBM UK Scientific Centre, Winchester, United Kingdom.

Clinical partners: Gregorio Marañon General Hospital, Madrid, Spain; University of Tübingen, Neuroradiology and Theoretical Astrophysics, Germany; German Cancer Research Center, Heidelberg, Germany; University of Leuven, Neurosurgery, Radiology, and Electrical Engineering, Belgium; University of Utrecht, Neurosurgery and Computer Vision, Netherlands; Royal Marsden Hospital and Institute of Cancer Research, Sutton, United Kingdom; and National Hospital for Neurology and Neurosurgery, London, United Kingdom.

Academic partners: Foundation of Research and Technology, Crete, Greece; University of Sheffield, United Kingdom; University of Genoa, Italy; University of Aachen, Germany; University of Hamburg, Germany; and Federal Institute of Technology, Zurich, Switzerland.

Cited references

1. W. Schlegel, "Computer Assisted Radiation Therapy Planning," *3D Imaging in Medicine*, NATO Series F60, Springer-Verlag, Berlin (1990), pp. 399–410.
2. H. Suit and W. Du Bois, "The Importance of Optimal Treatment Planning in Radiation Therapy," *International Journal of Radiation Oncology and Biological Physics* **21**, 1471–1478 (1991).
3. P. Pavlidis, "Image Analysis," *Annual Review of Computer Science* **3**, 121–146 (1988).
4. M. Morrison and Y. Attikiouzel, "An Introduction to the Segmentation of Magnetic Resonance Medical Images," *Australian Computer Journal* **26**, No. 3, 90–98 (1994).
5. M. Bomans, K-H. Hoehne, and M. Riemer, "3D Segmentation of MR Images of the Head for 3D Display," *IEEE Transactions on Medical Imaging* **9**, No. 2, 177–183 (1990).
6. A. Simmons, S. Arridge, G. Barker, A. Cluckie, and P. Tofts, "Improvements to the Quality of MRI Cluster Analysis," *Magnetic Resonance Imaging* **12**, No. 8, 1191–1204 (1994).
7. K-H. Hoehne and W. Hanson, "Interactive 3D Segmentation of MRI and CT Volumes Using Morphological Operations," *Journal of Computer Assisted Tomography* **16**, No. 2, 285–294 (1992).
8. H. Sekiguchi, K. Sano, and T. Yokoyama, "Interactive 3-Dimensional Segmentation Method Based on Region Growing Method," *Systems and Computers in Japan* **25**, No. 1, 88–97 (1994).
9. M. Kuhn, I. Carlsen, S. Dellepiane, P. Elliott, F. Galvez-Galan, H. Neumann, and H. S. Stiehl, "Computer Vision in Radiology (COVIRA): Knowledge Based Segmentation and Interpretation of Cranial Magnetic Resonance Images," *Proceedings of the AIM Euroforum*, Seville, Spain, December 1990, pp. 169–180.
10. R. Bendl, J. Pross, M. Keller, J. Burkelbach, and W. Schlegel, "VIRTUOS—A Program for VIRTUal Radiotherapy Simulation," *Proceedings of the Computer Assisted Radiology Conference*, Springer-Verlag, Berlin (1993), pp. 676–682.
11. D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, and P. Jeremes, *Object-Oriented Development: The Fusion Method*, Prentice Hall, Englewood Cliffs, NJ (1994).
12. G. Booch, *Object-Oriented Analysis and Design with Applications*, Second Edition, Benjamin/Cummings Publishing Co., Redwood City, CA (1994).
13. G. Booch, *Object-Oriented Design with Applications*, Benjamin/Cummings Publishing Co., Redwood City, CA (1991).

14. R. Wirfs-Brock, B. Wilkerson, and I. Wiener, *Designing Object-Oriented Software*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1990).
15. P. J. Elliott, J. M. Knapman, and W. Schlegel, "Interactive Image Segmentation for Radiation Treatment Planning," *IBM Systems Journal* **31**, No. 4, 620–634 (1992).
16. Open Software Foundation, *OSF/Motif Style Guide*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1990).
17. M. Kessler, S. Pitluck, P. Petti, and J. Castro, "Integration of Multimodality Imaging Data for Radiotherapy Treatment Planning," *International Journal of Radiation Oncology and Biological Physics* **21**, 1653–1667 (1991).
18. G. Sivewright and P. Elliott, "Interactive Region and Volume Growing for Segmenting Volumes in MR and CT Images," *Medical Informatics* **19**, 71–80 (1994).
19. M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Proceedings of the First International Conference on Computer Vision*, London, UK, June 1987, IEEE Computer Society Press, pp. 259–268.
20. J. Porrill and J. Ivins, "A Semiautomatic Tool for 3D Medical Image Analysis Using Active Contour Models," *Medical Informatics* **19**, 81–90 (1994).
21. F. Fontana, A. Bonomi, and G. Vernazza, "Innovative Interactive Methods for Image Segmentation," *Proceedings of the Computer Assisted Radiology Conference*, Springer-Verlag, Berlin (1993), pp. 321–327.
22. R. Ogniewicz, T. Vehkomaki, and G. Szekely, "Extraction of Closed Boundaries from Fragmented Edge Maps Using Shape-oriented Grouping Procedures," *Proceedings of the Computer Assisted Radiology Conference*, Springer-Verlag, Berlin (1993), pp. 334–340.
23. A. Collignon, D. Vandermeulen, P. Suetens, and G. Marchal, "An Object Oriented Tool for 3D Multimodality Surface-based Image Registration," *Proceedings of the Computer Assisted Radiology Conference*, Springer-Verlag, Berlin (1993), pp. 568–573.
24. A. J. Neal, G. J. Sivewright, and R. Bentley, "Evaluation of a Region Growing Algorithm for Segmenting Pelvic Computed Tomography Images During Radiotherapy Planning," *British Journal of Radiology* **67**, 392–395 (1994).
25. D. Rogers and J. Adams, *Mathematical Elements for Computer Graphics*, McGraw-Hill, Inc., New York (1990).

Accepted for publication September 25, 1995.

Peter J. Elliott *IBM UK Scientific Centre, IBM UK Laboratories, Hursley Park, Winchester, Hampshire S021 2JN, United Kingdom (electronic mail: pjelliot@winvmd.vnet.ibm.com).* Dr. Elliott graduated with first-class honors in physics from the University of Sussex, UK, in 1968. He joined the IBM UK Laboratories at Hursley Park to work on thin film magnetic recording heads for disk files. Sponsored by IBM, he carried out research on amorphous semiconductor thin films at the Cavendish Laboratory, University of Cambridge, leading to his Ph.D. degree in 1975. He subsequently worked on the development of disk storage devices, for which he holds four patents. In 1989 he joined the IBM UK Scientific Centre, to work on medical image analysis. He was instrumental in setting up the European consortium for the main phase of COVIRA, project A2003 of the AIM research program, and was joint project manager of COVIRA. Dr. Elliott is active in a number of image analysis and data visualization projects, and is the author of several publications in the field.

Jens Diedrichsen *IBM UK Scientific Centre, IBM UK Laboratories, Hursley Park, Winchester, Hampshire S021 2JN, United Kingdom (electronic mail: jens@hursley.ibm.com).* Mr. Diedrichsen graduated as Diplom-Ingenieur der Technischen Informatik at the Fachhochschule Hamburg, Germany, in 1992 with the equivalent of a first-class honors degree. During his studies he took part in a European student exchange program and received an upper-second-class honors degree in computer science from the University of Hertfordshire, UK. Mr. Diedrichsen then joined the IBM UK Scientific Centre as a research fellow to work on an object-oriented system for medical image analysis. In 1994 he worked on the development of IBM's CICS™ Systems Manager for AIX—a distributed, object-oriented product. He is currently studying part-time for a Ph.D. in object-oriented tools at the University of Hertfordshire.

Kelvin J. Goodson *IBM UK Scientific Centre, IBM UK Laboratories, Hursley Park, Winchester, Hampshire S021 2JN, United Kingdom (electronic mail: kelvin@vnet.ibm.com).* Dr. Goodson graduated with first-class honors in photographic sciences from the Polytechnic Institute of Central London, UK, in 1984, and received a Ph.D. in document image analysis in 1988 while working at the Dorset Institute, UK, in collaboration with the University of Southampton, UK. After a post-doctoral research fellowship at the University of Southampton, he joined the IBM UK Scientific Centre in 1990 as a research fellow, working on the extraction of camera motion from sequences of images of static objects. He then worked on medical image analysis and was responsible for creating a clinical pilot system for radiotherapy planning as part of the COVIRA project. During 1995 Dr. Goodson has adapted the pilot system for commercial use in medical, pharmaceutical, and materials science disciplines.

Robert Riste-Smith *IBM UK Scientific Centre, IBM UK Laboratories, Hursley Park, Winchester, Hampshire S021 2JN, United Kingdom.* Dr. Riste-Smith graduated with first-class honors in electrical and electronic engineering at the University of Strathclyde, UK, in 1979. He worked for General Electric Company as a systems designer for a digitally encrypted mobile radio system. In 1986 he went to Portsmouth Polytechnic, UK, to carry out research on image processing of medical X-ray films and was awarded a Ph.D. in 1990. He then joined the IBM UK Scientific Centre as a research fellow to investigate methods of 3D interactive segmentation and visualization of medical data under the COVIRA project. He is currently working on an object-oriented front office treasury system for a major financial institute based in London.

Gordon J. Sivewright *IBM UK Scientific Centre, IBM UK Laboratories, Hursley Park, Winchester, Hampshire S021 2JN, United Kingdom.* Dr. Sivewright graduated from the University of East Anglia, UK, with an honors degree in biophysics in 1986. After entering the University of Manchester, UK, he carried out research into quantitative magnetic resonance imaging within the department of Medical Biophysics and Diagnostic Radiology, obtaining his Ph.D. in 1990. In 1989 he worked as a research associate at the Wolfson Image Analysis Unit of the University of Manchester on a European Commission-funded research project. He joined the IBM UK Scientific Centre as a research fellow in 1990, and as a member of the COVIRA team developed algorithms and systems for performing interactive image segmentation. In 1995 he became an independent software contractor and is currently helping to develop a new trading system based on distributed object-oriented technology for a large bank.

Reprint Order No. G321-5592.