1994

A Distributed, Object-Oriented Framework for Medical Image Management and Analysis: Application to Evaluation of Medical Image Segmentation Techniques

James F. Brinkley
Digital Anatomist Program, Department of Biological Structure
University of Washington
Seattle, Washington

Abstract

A distributed framework for medical image management and analysis is described. Individual program modules interact over the network, creating and accessing a set of structural information resources that includes images, anatomic shape models, anatomic names, and contours which are the output of a knowledge-based segmentation program. Program modules are described that automatically evaluate the segmenter on large number of images, thereby allowing more robust testing of segmentation techniques than is currently possible in manual evaluations. It is suggested these kinds of programs will not only improve the evaluation of segmentation techniques, but will also enhance the overall utility of medical Picture Archiving and Communication Systems.

1 Introduction

Medical image segmentation involves identifying anatomic objects in medical images, and is an important step, not only for automatic image interpretation, but also for 3-D reconstruction and manipulation of anatomic objects for research, education and clinical applications such as surgery or radiation treatment planning [1,2]. General techniques for image segmentation are described in [3,4], and [5,6,7,8] provide a few examples of the many attempts to apply these techniques to medical imaging.

In spite of the number of attempts to solve this problem no satisfactory solution has yet been found. The main reason for this lack of success is the technical difficulty of teaching a computer to "see". However, an additional problem is that potential solutions are not adequately evaluated on a large enough sample of images to either show their utility or to provide insights that may lead to improved algorithms. When such solutions are brought to a clinical environment they often perform so poorly that they are discarded in favor of a manual method, and this poor performance may prejudice clinicians against trying other techniques. One reason for the lack of careful evaluation is that the automated techniques often require minutes to hours to run and must be manually started on images that are loaded from local disk. Thus, it is simply too daunting to consider evaluating more than a few images.

Fortunately, advances in computer hardware have created the opportunity for large scale medical image picture archive and communications systems, or PACS. Many such systems are currently under development [9], although the large number of images produced by typical radiology departments requires that most current research be directed towards providing large storage capacity, high network bandwidth and high resolution displays. Once functional PACS are in place, however, it will be possible to write computer programs that automatically evaluate different segmentation techniques on large numbers of test images obtained from a PACS archive.

In the University of Washington Digital Anatomist Program we are developing a distributed framework for creating, utilizing and disseminating structural information about the physical organization of the body [10], at levels ranging from gross anatomy to molecules [11]. As part of that framework we have developed a "mini" PACS for managing, displaying, and segmenting medical images. Each module in the framework is a stand-alone object-oriented computer program that communicates with other modules, including an image database, via remote procedure calls. The PACS modules are used to store and retrieve images in a remote archive, to display images of multiple types in a manner that is transparent to the user, to segment images using knowledge of anatomic shape, and to automatically evaluate multiple segmentation techniques on large numbers of images. Some of these capabilities have been described by others for managing or analyzing images in a research environment [12,13], but very few if any systems bring all these features together. This paper describes our PACS as part of a larger structural information framework, shows how it is used to evaluate medical image segmentation techniques, and suggests how it could be integrated with larger scale PACS when they become available at the University of Washington and elsewhere.

Reprinted from Proceedings of the IEEE Workshop on Biomedical Image Analysis,June 24-25, 1994, Seattle, IEEE Press.

2 System Design

2.1 The Digital Anatomist Structural Information Framework

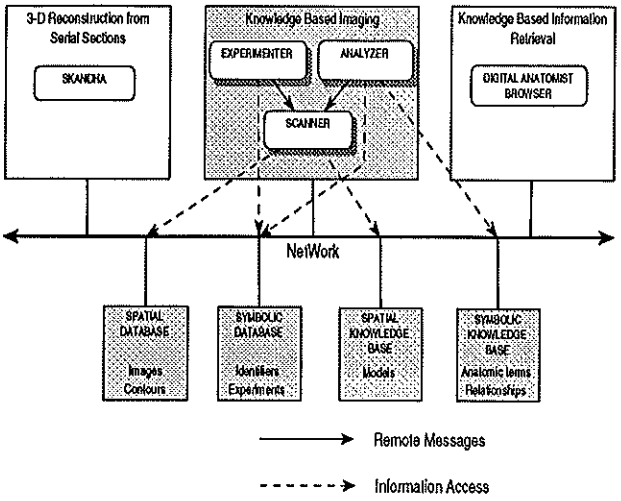

Figure 1. The Digital Anatomist Framework

Fig. 1 is an overview of our structural information framework, as well as the PACS components, which are shown shaded. The goals of the Digital Anatomist Program are 1) to develop a large knowledge and database of structural information ranging from gross anatomy to molecules, 2) to develop program modules which use this information to solve problems in medical research, education and clinical care, and 3) to disseminate the information as widely as possible. These goals have led us to design a distributed system, in which various modules, running on the same or different machines, communicate to solve problems that are larger than any module could solve on its own. The basic framework consists of a set of structural information resources shown at the bottom of Fig. 1 (the spatial database, symbolic database, spatial knowledge base and symbolic knowledge base) that are accessed by different interacting programs. The four structural information resources arise from our classification of structural information along two dimensions: data versus knowledge, and spatial versus symbolic [11]. Structural *data* is information about an individual structural object such as a patient, a kidney or a protein. Structural *knowledge* is information about classes of objects such as all normal kidneys. *Spatial* information is geometric information described in one, two or three dimensions, whereas *symbolic* information is textual in nature. These classifications give rise to the four kinds of structural information resources shown in Fig. 1.

Since structural information representation is a difficult problem, our research strategy is to choose several applications that not only drive the development of the structural information resources, but also lead to practical solutions that do not require all the representation problems to be solved at once. The current problem areas are shown at the top of Fig. 1, and include the SKANDHA program for three-dimensional reconstruction of anatomic objects from serial sections [1], the SCANNER, ANALYZER and EXPERIMENTER programs for knowledge based imaging, (the subject of this paper), and the Digital Anatomist BROWSER for knowledge based anatomic information retrieval [14,15]. Other problem areas will be added as we develop more sophisticated techniques for representing structural information.

2.2 The Imaging Framework

The modules comprising the imaging framework are shaded in Fig. 1. The structural information resources accessed by these modules contain information relevant to medical image management and segmentation. The *spatial database* contains 2-D *medical images*, as well as *contours* of individual structures that are seen on the images. The contours are the output of the segmentation module and are the basis for 3-D surface reconstruction from serial sections [1].

The *spatial knowledge base* (third box at the bottom of Fig. 1) contains geometric *models* describing the shape and range of variation for classes of structural objects. In the current system all models are two-dimensional. For example, one model describes the shape of a kidney cross section as seen on several images, and a second model describes the shape of a liver cross-section as seen on several images of the liver. Each model is determined from a series of similarly shaped contours stored in the spatial database, and is represented by a *radial contour model*, a type of *geometric constraint network* that in its most general form is proposed as a representation for capturing spatial knowledge at levels ranging from gross anatomy to molecules[16]. Details of the model representation are provided elsewhere [17,18].

The *symbolic database* contains *identifying* information about the images and contours, such as patient name and age, and image and contour filenames. It also contains identifying information about the models created in the spatial knowledge base, and *experiments* that are run to test various segmentation techniques.

The *symbolic knowledge base* contains *anatomic terms* derived primarily from the National Library of Medicine's Unified Medical Language System (UMLS) Metathesaurus [19], and augmented by terms from a glossary of neuroanatomical terms developed at the University of Washington [20] as well as our own additions. The terms are arranged in a semantic network that captures structural *relationships*

such as "the lateral ventricles are PART-OF the brain" [15]. In the current system the terms are used to index structures seen on medical images, and images are retrieved by querying the image database for specific structures. Later versions of the image management system will allow us to use the semantic network to formulate more intelligent queries, such as "Find all images containing any structure that is PART-OF the brain". The same symbolic knowledge base is used in our system for teaching anatomy to medical students [14,15].

Fig. 1 also shows the program modules comprising the knowledge based imaging project, as well as the structural information resources that each accesses. The non-shaded modules (SKANDHA and the BROWSER) are described elsewhere [1,14,15]. Each module is a separate computer program, with its own graphical user interface (GUI), that can run in stand-alone mode. SCANNER can also be accessed as a server by other programs, as indicated by the arrows leading to SCANNER from the other two modules. The GUI for each program is designed so that when the programs interact with each other the separate GUIs appear to the user as a single interface. The advantages of this approach over a single monolithic program are 1) modules can be developed and changed independently as long as the remote messages do not change dramatically 2) modules can be distributed over separate computers in order to take advantage of computational resources that may not be available at the local site, and 3) modules can be re-used to solve other problems. A potential disadvantage is that it may be difficult to keep track of multiple interacting modules, but this problem has not yet arisen in our application.

The central module in our imaging framework is called SCANNER. This module is used to display 2-D images, and to interactively segment them in order to produce 2-D contours. The segmentation process utilizes radial contour models that contain spatial knowledge of anatomic shape. SCANNER can also create new radial contour models from sequentially loaded contours of similar shape.

The images, contours and models that SCANNER accesses are all stored as files in the spatial database and spatial knowledge base. When SCANNER is used in stand-alone mode the user interacts directly with the program via menus. However this process can become very tedious when large numbers of images are used for segmentation experiments. In this case SCANNER can be called as a server by the other two modules.

ANALYZER is an image database manager which interacts with an image database (stored in the symbolic database). ANALYZER allows the user to search a database of images, organized by patient and image set. The user can also access the terms in the symbolic knowledge base to index the images by structure, and can retrieve images by various criteria including patient name, imageset, and structure name. Once a list of image filenames have been retrieved a single image is displayed by sending a remote message to SCANNER, which causes SCANNER to load and display the file containing the image. All interaction with ANALYZER is via point and click mouse commands.

EXPERIMENTER is used to create experiments allowing various segmentation techniques to be evaluated on large numbers of images stored in the spatial database. Images and contours selected in ANALYZER are placed in trialsets, experimental parameters are entered on forms, and experiments are run automatically. A single experiment may run overnight, and involves repeated remote calls to SCANNER, as well as recording of results in the symbolic database. Summary tables are generated at the end of the experiment.

The following sections describe the modules and structural information resources in more detail.

2.3 The SCANNER image segmentation program

SCANNER is an interactive knowledge-based program for two-dimensional medical image segmentation, as well as a test bed for the implementation of image segmentation techniques. The two basic notions behind SCANNER are 1) knowledge of anatomy is necessary for medical image understanding and 2) segmentation is a difficult problem, so any useful system must involve the human user. SCANNER is therefore designed to minimize the amount of work required of a human user, but not to completely automate the process of segmentation.

The anatomic knowledge employed by SCANNER is represented as 2-D *radial contour models* describing the expected shape as well as the range of variation for various anatomic objects, as they appear on cross-sectional medical images. The models are "learned" from training sets of similarly shaped cross-sectional contours, after which they are used to guide the search for contours from the same shape class on new images.

The input to SCANNER is an image and a radial contour model representing the external shape of the organ whose cross-section is to be found on the image. The output is a radial contour describing the location of the specified organ cross-section on the image, where a *radial contour* is a cross-section defined by the position of the organ boundary along a series of fixed radials emanating from the origin of a local coordinate system, as shown in Fig. 3. A radial contour model is created in SCANNER by manually segmenting a series of similarly-shaped radial contours from a particular shape class, then using these contours as a training set for the model.

A previously trained model is used in model based segmentation of a similarly-shaped anatomic object on an image that was not part of the training set. The long axis of the object is indicated by the user, after which the model guides the search for edges along one-dimensional radial lines. Radials in the model are searched sequentially. For each radial the image intensities along the radial are sampled, then passed to the edge detector. Edges found by the edge detector are used to update the model as to the location of the organ cross-section in the image, after which a new radial is examined. This process continues until all the model radials have been found. At this point the user can correct any radials that were incorrectly found. The hope is that very few radials will need to be corrected, thereby speeding up the segmentation process over a manual method. Details of the radial contour model, the SCANNER program, and results of an evaluation of its usefulness for medical image segmentation, can be found elsewhere [17,18]. However, these earlier reports only describe the SCANNER program, and do not describe the larger imaging framework which is the subject of this paper.

2.4 Structural information resources for model based image segmentation

The other modules of our imaging framework were designed to allow SCANNER to be run automatically, and to drive the development of our overall structural information resources. The result is a mini PACS, that not only facilitates the evaluation of the SCANNER program, but also can be used for other tasks such as image management and display.

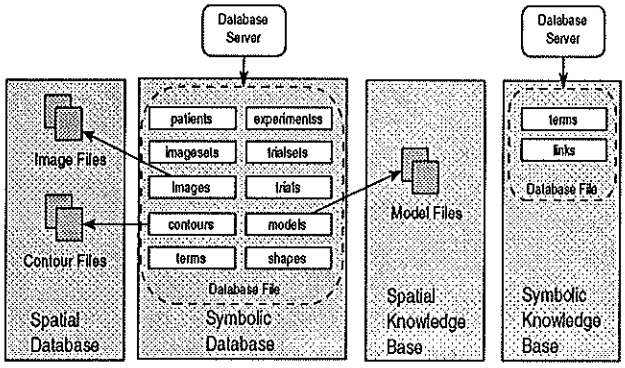

Figure 2. Structural Information Resources

Fig. 2 shows the implementation of the structural information resources that are used to support our mini PACS. This figure is a more detailed view of the corresponding boxes shown at the bottom of Fig. 1. The spatial database and spatial knowledge base consist of image, contour and model files stored in Unix directories. These are the files that can be accessed directly by SCANNER, as indicated in Fig. 1. The symbolic database and symbolic knowledge base provide additional mechanisms for accessing these files, and are implemented as extended relational databases [22], in which entries in relational tables point to image, contour and model filenames. The relational databases are implemented in Sybase on the NeXT.

The *symbolic database* is organized by individual *patients*, each of whom may have one or more *imagesets*, consisting of several 2-D *images*. Once manual or semi-automatic segmentation is complete each image may contain several *contours* delineating anatomic cross-sections. The contours reference a *shape* identifier, which delineates a unique 2-D shape class, as defined by the user. Each shape class is in turn identified by an anatomic *term* and a shape number. For example, an anatomic term might be "kidney" and three shape classes for the kidney might be kidney 1, kidney 2 and kidney 3. These shape classes refer to three different 2-D shapes that cross-sections of the kidney exhibit at different levels: the superior pole, the hilar region, and the inferior pole, and are the means for identifying the radial contour model files in the spatial knowledge base. A single *model* refers to a radial contour model file, and is derived from several contours.

Anatomic *terms* are copies of terms from the symbolic knowledge base, which include an identifier given by the UMLS Metathesaurus [19]. The terms are copied from the symbolic knowledge base to the symbolic database for reasons of efficiency. The symbolic knowledge base also includes *links* that capture semantic relationships between terms. The links are currently only used in the BROWSER [15].

The symbolic database also tracks *experiments* to measure the effectiveness of various configurations of the segmenter on large numbers of images. A single experiment consists of a set of global parameters defining the particular configuration of the segmenter being tested, *trialsets* consisting of manually segmented contours from various shape classes, and *trials*, which contain the result of running an experiment.

Many of these relational tables are created by the user with the ANALYZER program, or by the EXPERIMENTER program as it runs the SCANNER program as a server.

2.5 ANALYZER

The ANALYZER program is a front end for our image database. As such it forms the basis for a general image management and retrieval facility, as well as the specific application of setting up experiments for evaluating SCANNER. ANALYZER communicates with SCANNER by means of remote messages that allow SCANNER to be run as a server. These messages include "Image Open and Dis-

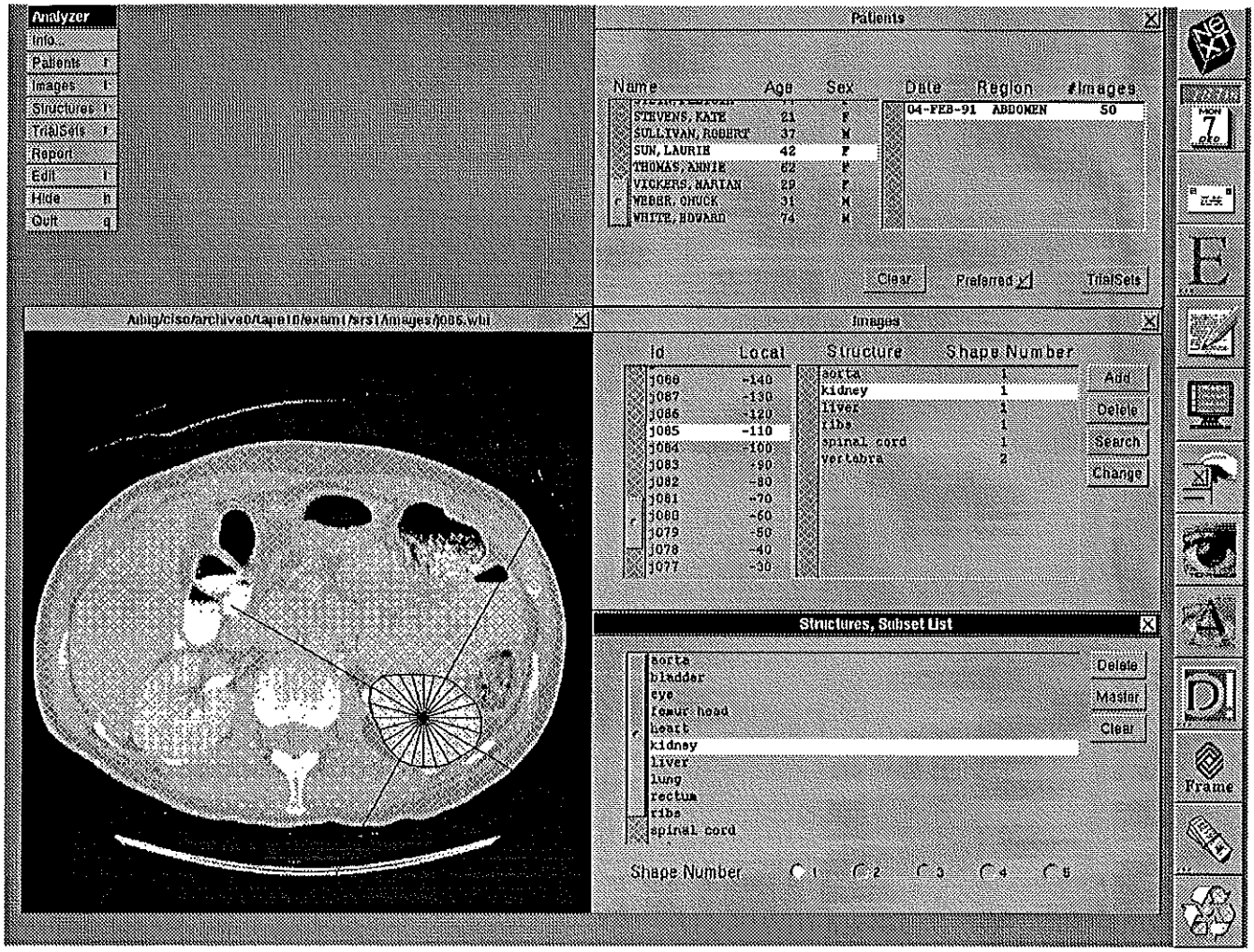

Figure 3. ANALYZER user interface

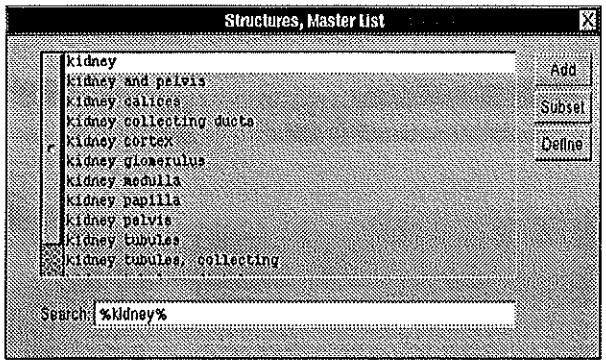

Figure 4. Master list of structures

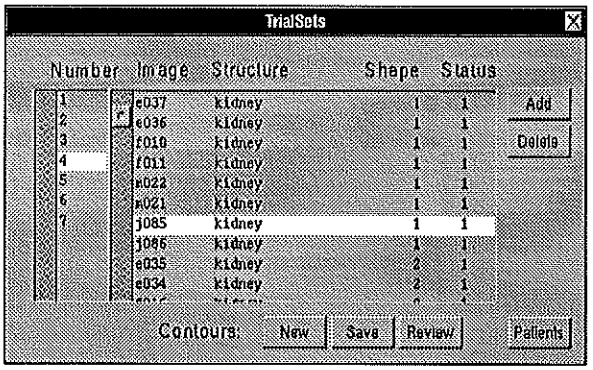

Figure 5. Trialsets for EXPERIMENTER

play", "Contour Open", "Contour Save", "Model Open", "Model Create", "Model Add Contour", and "Segment", among others.

ANALYZER allows images to be indexed according to the structures that appear on the images, and to be retrieved according to patient name, imageset, and structure. The indices are created by the user with point and click operations, and are stored in tables in the symbolic database.

The user interface for ANALYZER is shown in Fig. 3, and is designed to work with the SCANNER user interface, part of which is shown as the image window in the lower left portion of Fig. 3. The "Patients" panel in the top right of the figure allows images to be retrieved by patient name and image set, and the "Structures, Subset List" panel in the bottom right allows images to be retrieved by structure. In Fig. 3 identifiers for all images from the fictional patient "Laurie Sun", that contain kidney shape number 1, are shown in the lefthand browser of the middle "Images" panel. If no structure is selected in the bottom panel then all images from the selected patient and imageset are displayed in the "Images" panel. If no patient is selected then all images containing the selected structure are listed.

A particular image from the retrieved list is displayed by double clicking the identifier (j085 in the figure). This operation causes ANALYZER to query the images table in the symbolic database for the image filename, then send the "Image Open" remote message to SCANNER with the image filename as an argument. SCANNER loads the image and displays it in the SCANNER image window shown in the lower left part of Fig. 3. To the user this operation appears to involve only a single program.

Images are also indexed using the graphical user interface. Structure names are retrieved from the symbolic knowledge base by accessing the "Structures, Master List" panel (Fig. 4), adding a selected term to the "Structures, Subset List" browser in Fig. 3, then adding a term from the "Structures, Subset List" browser to a particular image. For example, in Fig. 3 image j085 has been indexed by shapes aorta 1, kidney 1, liver 1, ribs 1, spinal cord 1 and vertebra 2.

These operations demonstrate the primary use of ANALYZER, which is to visually analyze a set of patient images as to the structural shapes that they contain, and to record the classifications in a database that allows the images to be easily retrieved. This capability alone should be very useful for other applications that need to retrieve images by structure, and the modular nature of ANALYZER will allow it to be adapted for other uses.

In addition to this more general use, ANALYZER is also used to prepare the trialsets that are used by EXPERIMENTER for evaluating the segmenter. These capabilities are more specifically tuned to the evaluation of SCANNER, but the basic approach, as well as the program itself, is extendable to other evaluation procedures.

For the purpose of preparing trialsets the user accesses the "Trialsets" panel, Fig. 5. A trialset consists of a series of image-shape pairs, where each pair refers to an image and a shape class that appears on that image. For example, the highlighted rows in Fig. 5 delineate image j085 and shape class kidney1, from trialset 4. Each trialset in general contains several shape classes, and each shape class is represented by several image-shape pairs in the trialset. Each experiment utilizes two trialsets, each containing the same shape classes and the same number of image-shape pairs per shape, but the images are different in each trialset. It is up to the user to select the best images and shapes for each trialset.

In order for an experiment to be run each image-shape in each of the two trialsets must be represented by a radial contour. As more experiments are run many of the necessary contours will have been either manually or semi-automatically segmented by SCANNER or other programs, and will already be present in the spatial database as contour files. However, in the beginning each of the contours must be manually segmented. This is accomplished in ANALYZER for a selected image-shape pair in the Trialsets panel by clicking on the "New" button in the bottom line of buttons labelled "Contours". When this button is clicked a series of remote messages are sent to SCANNER, causing it to load the appropriate image and to initiate a new contour. The user then manually segments the contour by entering the long axis and the position of the contour along all radials, resulting in a radial contour similar to that shown in Fig. 3. When the contour is satisfactory the user clicks the "Save" button in the Trialsets panel, which causes SCANNER to save the contour in a filename generated by ANALYZER, to create an entry in the contours database table referring to the contour file, to link the image-shape pair in the trialset to the newly created contour, and to set the "Status" field of the selected image-shape pair to 1, indicating that the contour exists. If, when the image-shape is added to the trialset the contour already exists, then the Status field will already be 1.

The contour associated with an image-shape pair whose status field is 1 may be displayed by clicking "Review". If the contour is not satisfactory it may be edited using SCANNER, then saved again by clicking the "Save" button. The kidney contour shown in Fig. 3 is associated in the database with the highlighted image-shape pair in Fig. 5, and was displayed by clicking the "Review" button.

2.6 EXPERIMENTER

The main work in setting up a segmentation evaluation

experiment is selecting a set of images and image-shapes, and manually segmenting the contours associated with each image shape. The ANALYZER interface tools, combined with remote procedure calls to SCANNER, make this task much easier than it would be with SCANNER alone. As more images and contours are added to the database with ANALYZER and with other tools the trialsets will be useful for many other experiments. In the current work, however, the trialsets are used for the specific purpose of evaluating the SCANNER program, and these evaluations are embodied in the EXPERIMENTER module. Thus, this particular implementation of EXPERIMENTER is the least general of the three modules, but other EXPERIMENTER modules could be written to perform different experiments.

The current implementation of EXPERIMENTER contains only a minimal user interface controlled primarily by means of menus, and a single entry form that allows the global parameters of the experiment to be set up. The entry form has fields for the experiment number, the two trialset numbers, some global parameters, and a status field. These items are added as a row in the experiment table in the database. Once the experiment has been set up it can be run completely automatically, or can be run a step at a time by menu commands in EXPERIMENTER. EXPERIMENTER uses the information in the experiment and associated trialset tables to remotely request SCANNER to perform repeated model creation, segmentation, and comparison with gold standard contours, where in this case a *gold standard contour* is a contour that has been manually segmented by the author.

The basic experimental design embodied in this version of EXPERIMENTER is to use the contours in one trialset as a training set to build shape models, then to use these models to segment the corresponding shapes in the second trialset, using the contours in the second trialset as gold standards for comparison. The process is then reversed: contours in the second trialset are used as a training set to build models which aid the segmentation of shapes in the first trialset. The following paragraphs provide more details.

When the experiment is run the program first checks to be sure that each of the two trialsets is complete. A trialset is complete if each shape class is represented by at least three contours, and if each contour has been segmented. Both trialsets must also contain the same shape classes. If these conditions are not met the experiment is aborted. If the conditions are met shape models are created for each shape class in each trialset by sending remote messages to SCANNER to 1) create a new model with a system generated name 2) load contours in the trialset that represent the shape class (at least three per shape class) and 3) add the loaded contour to the shape model. Once the model is created an appropriate entry is made in the database.

Each trial created by EXPERIMENTER is a template for a single segmentation by SCANNER and the comparison of the resulting contour with a gold standard contour. Each trial specifies an image, a contour and a model. Two trials are created for each image-contour pair in trialset 1: a trial using a NO-SHAPE model that has no information about anatomic shape, to act as a control, and a trial specifying a SHAPE model representing the shape class of the contour, but learned from the corresponding contours in trialset 2. Similarly, two trials are created for each image-contour pair in trialset 2: a trial using a NO-SHAPE model that has no information about anatomic shape, to act as a control, and a trial specifying a SHAPE model representing the shape class of the contour, but learned from the corresponding contours in trialset 1. Thus, each contour in each trialset is used twice: as a member of a training set to build models that are used to segment the images in the other trialset, and as a gold standard that is used to judge the results of segmentation.

Once the models and trials are created each trial is run automatically. A single trial is run by remote messages to SCANNER that cause it to 1) load the image, contour and model files specified in the trial record 2) to obtain the initial contour axis from the loaded contour, thereby simulating the specification of the long axis by the user 3) to initiate model-based segmentation using the initial long axis and the model, and 4) to compare the segmented contour with the loaded contour treated as a gold standard. The comparison is made by counting number of radials in the segmented contour that would need to be corrected by the user in order that the segmented contour matches the gold standard contour within a global parameter EPSILON, which is set at the beginning of the experiment. The number of "user-defined" radials is returned by SCANNER and is recorded in the trials record in the symbolic database. The lower this number the more successful the segmentation is judged to be. The number can never be lower than two because the long axis, which is given by two radials, is always defined by the user. Additional details about this protocol can be found elsewhere [18].

Automatic runs can take several hours depending on the number of image-shape pairs in each trialset. Results of individual segmentation runs are recorded in the trials table of the database, and the trials are summarized in an output file. Thus, the primary work in setting up an experiment is in the ANALYZER program. Once the experiment is set up, the computer does all the work that would have taken many hours of manual loading of image files and setting up of segmentation runs.

Table 1 Usefulness of the SHAPE model for various SCANNER parameters[1]

Experiment	SCANNER parameters		Model Type	Trials	User-defined radials	
	Edge Type	Pixel Type		N	Mean	SD
2	First	Mapped	NO-SHAPE	240	18.1	6.8
			SHAPE	240	9.0	6.4
4	Last	Mapped	NO-SHAPE	240	23.8	1.3
			SHAPE	240	10.4	7.6
5	First	Original	NO-SHAPE	240	18.8	5.8
			SHAPE	240	10.2	7.2

1.Differences between mean user-defined radials are all significant at p<.05 for the SHAPE versus NO-SHAPE models within each experiment. Differences between Experiments 2 and 4 are significant at p<.05, differences between Experiments 2 and 5 are not.

3 Sample Experiments

This section describes three sample experiments that were run in order to show the utility of our framework for evaluating segmentation techniques on large numbers of images. One of the experiments, presented for comparison, was described previously [18]. The other two experiments are new and were designed to test different aspects of the segmenter on the same set of images as that used in the first experiment. The imaging framework allowed all three experiments to be set up in only a few minutes, after which each was run and analyzed automatically over three four hour periods. The following paragraphs describe these experiments in more detail.

SCANNER can be run with several user-settable parameters that control how images are sampled when looking for edges. These parameters include, among others 1) whether original 16 bit pixel values are sampled when searching for edges along radial lines, or whether 8 bit mapped pixel values are used, in which case the window width and level values become additional parameters, and 2) whether the first or last edge encountered along a radial line is used to update the model. In the previous report these parameters were held constant for all model comparisons as: Mapped pixels using a window and width level of 606 and 956 respectively, edge threshold set at 5 percent of the maximum gray value of 255, and the edge finder set to always choose the first edge it encountered that was over the threshold [18].

Results from one of these experiments is repeated as experiment 2 in Table 1, and shows that the addition of shape information increases usefulness by about a factor of 2 over a NO-SHAPE model (18.1 to 9.0), where a value of 24 is the least useful since all 24 radials would have to be entered by the user (all models in these experiments had 24 radials), and 2 is the most useful since only the axis radials would need to be entered. However, the usefulness of the shape model could also be influenced by the image sampling parameters, and the other two experiments (which were not described in the previous paper) examine the influence of two of these parameters.

Experiment 4 uses all the same parameters as Experiment 2, except the last edge was taken rather than the first edge. Experiment 5 uses all the same parameters as experiment 2, except the original pixels values were used. In all three cases the same two trialsets were used. Each trialset contained 120 contours representing 15 shapes from 8 patients, thereby generating 240 trials per trialset, or 480 trials per experiment. Since both trialsets had been previously prepared for experiment 2, and since all contours had previously been outlined, the time required to set up each experiment was only the few minutes needed to fill out the experiment form in Experimenter, and to set the parameters of SCANNER. The actual time to automatically run each of these experiments was about 4 hours.

The results in Table 1 provide some insights into the effect of various image sampling parameters on the usefulness of model-based segmentation. Comparison of mean user-defined radials for Experiment 2 and Experiment 4 shows that both the NO-SHAPE and SHAPE models do slightly better when the first edge along a radial is used, rather than the last, although the NO-SHAPE model does

much better. This result is reasonable since in general the first edge encountered is the contour boundary. However, this is not always the case, and more detailed analysis for individual structures, such as the thorax, suggests that the last edge is better in some cases.

Comparison of Experiment 2 and Experiment 5 shows that there is no significant difference between user defined radials when mapped pixels are sampled instead of original pixels. This result is somewhat surprising since the mapping parameters of 606 and 956 were chosen visually by the author to give the best contrast between the structures and the surrounding tissues, so it might be expected that sharper edges would be found when the contrast is better. This result suggests that, at least for these examples, the shape models limited the search region enough so that small edge changes in the original pixels were still found even though the contrast had not been enhanced by window and level mapping. On the other hand the mapping parameters were held constant for all shapes, and it may be that specific values of these mapping parameters for different structures will lead to better results.

4 Discussion

The results of both experiments 4 and 5 suggest that improvements in the segmentation program can be achieved by adding additional knowledge to the program which sets these parameters depending on the organ that is being segmented. The modular nature of SCANNER should make this relatively easy to do.

The main point of presenting these experiments in this paper was not, however, the small insights they suggest for ways to fine tune SCANNER. Rather the results are presented in order to show the utility of our framework for automatic evaluation of image segmentation programs. The execution time of each of these experiments was about 4 hours. If SCANNER were run in stand-alone mode each of these 3x480 trials would have had to be run manually by using the SCANNER menus to load images, contours and models, and to record the results of each segmentation. The extreme tedium of such a task makes it unlikely that these experiments would have ever been done, and in fact these kinds of experiments are almost never done in the evaluation of other image segmentation techniques reported in the literature.

Although a model-based program may require many more image samples than other image segmentation programs, it is still reasonable to assume that any robust evaluation of a segmenter should look at the effectiveness of the program on large numbers of images. Without some sort of automated framework this task has simply been too tedious to be performed except in exceptional cases.

This paper has shown that a combination of specialized experimenter and analyzer modules, with a mini PACS for image storage and retrieval, can greatly alleviate the tedium involved in evaluating an image segmentation program, and thereby can lead to much more robust image segmentation techniques for practical use.

Although the particular modules described in this paper have been tuned for evaluation of the SCANNER program for model based segmentation, the basic approach can be extended to handle other kinds of image segmentation techniques, and can be used to compare among them. SCANNER itself is a useful framework for substituting different image segmentation techniques. The object-oriented design allows new image and model types to be added, and the edge finder can be improved without changing the interface to other objects in the program. In addition, new objects can be added to perform more effective low level image processing than simple window and level controls.

The basic client-server architecture of the imaging framework also allows access to other image segmentation methods or to low level image processing modules, as long as they can be made into servers callable by other programs. As University-wide PACS become available evaluation programs can access the images over the network, thereby gaining access to a large storehouse of images to use in evaluating and comparing different image segmentation modules.

When a particular segmentation technique has been shown to be reliable (an elusive goal) it can be folded back into the framework and used as the basis for content-based image retrieval [23], which would allow new images to be automatically indexed by structure, without the need for the manual indexing methods utilized by ANALYZER.

The success of the object-oriented client server approach in this and the other two applications shown in Fig. 1 has suggested to us the development of a "toolkit" of independent but interacting modules for structural informatics. Each "tool" would be a stand-alone program that included its own user interface, and would be designed to perform a small number of tasks. Each tool would also be callable by other tools by adding a program interface that allowed it to act as a server to other programs.

The tools would access a common set of information resources. Currently the format and contents of the four structural information resources shown in Fig. 1 are slightly different for each of the three application areas. As the various representation issues are solved we expect these resources to converge to a form that will be re-usable for many applications. This convergence will allow modules developed for different applications to synergistically solve

problems that are larger than any module could solve on its own. It is at this point that the real power of the distributed approach will become apparent, not only for medical image management, but for management of other structural information as well.

Acknowledgments

This work was funded in part by National Library of Medicine grant LM04925, National Cancer Institute grant CA59070, National Science Foundation grant IRI-9116809, a gift from the F.S. Smithers Foundation, the Murdock Foundation Charitable Trust, and the University of Washington School of Medicine. The UW medical image archive is maintained in the Department of Electrical Engineering by Greg Zick and Eliot Lim. The images were obtained from the Radiology Department by David Haynor in that department. The imaging framework is one component of the Digital Anatomist Project in our department, which is directed by Cornelius Rosse. I would like to thank all the members of this project for their support. I would also like to thank John Prothero, Cornelius Rosse and Greg Zick for providing feedback on drafts of the manuscript.

References

1. Conley, D.M.; Kastella, K.G.; Sundsten, J.W.; Rauschning, W.; Rosse, C. Computer-generated three-dimensional reconstruction of the mediastinum correlated with sectional and radiological anatomy. Clinical Anatomy 5:1-17;1992.
2. Kalet, I.J.; Jacky, J.P. A research-oriented treatment planning program system. Comput. Prog. Biomed. 14:85; 1982.
3. Ballard, D.H.;Brown, C.M. Computer Vision. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.;1982.
4. Haralick, R.M.; Shapiro, L.G. Image segmentation techniques. Comput. Vision, Graph. and Image Proc. 29:100-132;1985.
5. Kobashi, M.; Shapiro, L.G. Knowledge-based organ identification from CT images. Proc. SPIE Medical Imaging VI:544-554;1992.
6. Moss, R.H.; Stoecker, W.V.;Lin, S-J.; Muruganandhan, S.; Chu, K.-F.; Poneleit, K.M.; Mitchell, C.D. Skin cancer recognition by computer vision. Comput. Med. Imaging Graph. 13:31;1989.
7. Waks, A.; Tretiak, O.J. Recognition of regions in brain sections. Comput. Med. Imaging Graph. 14(5):341-352; 1990.
8. Vannier, M.W.; Pilgram, T.K., Speidel, C.M.; Neumann, L.R.; Rickman, D.L.; Schertz, L.D. Validation of magnetic resonance imaging (MRI) multispectral tissue classification. Comput. Med. Imaging Graph. 15(4):217-223; 1991.
9. Huang, H.K, ed. Picture archiving and communication systems. Comput. Med. Imaging Graph. 15(3):133-203; 1991.
10. Brinkley, J.F. Structural informatics and its applications in medicine and biology. Academic Medicine 66:589-591; 1991.
11. Brinkley, J.F.; Prothero, J.S.;Prothero, J.W.; Rosse, C. A framework for the design of knowledge-based systems in structural biology. Proc. Symposium on Computer Applications in Medical Care 13:61-65;1989.
12. Robb, R.A.; Hanson, D.P.; Karwoski, R.A.; Larson, A.G.; Workman, E.L.; Stacy, M.C. Analyze: a comprehensive, operator-interactive software package for multidimensional medical image display and analysis. Comput. Med. Imaging Graph. 13(6):433-454;1989.
13. Leavitt, M.B.; Newell, J.B.; An image filing system for the implementation of image processing algorithms in a research environment. Comput. Biomed. Res. 21(2):174-185; 1988.
14. Eno, K.; Sundsten, J.W.; Brinkley, J.F. A multimedia anatomy browser incorporating a knowledge base and 3-D images. Proc. Symposium on Computer Applications in Medical Care. 15: 727;1991.
15. Brinkley, J.F.; Eno, K.; Sundsten, J.W. Knowledge-based client-server approach to structural information retrieval: the Digital Anatomist Browser. Comp. Meth. Prog. Biomed. 40:131-145; 1993.
16. Brinkley, J.F. Hierarchical geometric constraint networks as a representation for spatial structural knowledge. Proc. Symposium on Computer Applications in Medical Care. 16: 140-144;1991.
17. Brinkley, J.F. Spatial anatomic knowledge for 2-D interactive medical image segmentation and matching. Proc. Symposium on Computer Applications in Medical Care 15:460-464; 1991.
18. Brinkley, J.F. A flexible, generic model for anatomic shape: application to interactive two-dimensional medical image segmentation and matching. Comput. Biomed. Res. 26:121-142; 1993.
19. Tuttle, M.; Sheretz, D.; Olson, N.;Erlbaum, M.; Sperzel, D.; Fuller, L.; Nelson, S. Using Meta-1 -- the first version of the UMLS Metathesaurus, Proc. Symposium on Computer Applications in Medical Care 14:131-135; 1990.
20. Martin, R.F.; Dubach, J.; Bowden, D. Neuronames: human/macaque neuroanatomical nomenclature, Proc. Symposium on Computer Applications in Medical Care 14:1018-1019; 1990.
21. Yamashita, C.; Zick, G.L. Combined relational and textual retrieval in a medical image archive. SPIE Medical Imaging VII, Newport Beach, CA; 1993.
22. Cattell, R.G.G. Object Data Management: Object-Oriented and Extended Relational Systems. Menlo Park: Addison-Wesley; 1991.
23. Kofakis, P.; Karmirantzos, A.; Kavaklis, Y.; Petrakis, E.; Orphanoudakis, S. Image archiving by content: an object-oriented approach. SPIE Medical Imaging IV: PACS System Design and Evaluation 1234:275-285; 1990.