AN AUTOMATED SOFTWARE SYSTEM FOR BRACHYTHERAPY SOURCE LOCATION

By

ISMAIL AHMET NALCACIOGLU

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2000

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr.Sartaj K. Sahni, my supervisory committee members, Dr.Sanjay Ranka and Dr.Zuofeng Li, for giving me an opportunity to work on this challenging topic and for providing continuous guidance and feedback during the course of this work and thesis writing.

I also wish to take this opportunity to thank my parents for their emotional support and encouragement throughout my academic career.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure page

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

AN AUTOMATED SOFTWARE SYSTEM FOR BRACHYTHERAPY SOURCE LOCATION

By

Ismail Ahmet Nalcacioglu

May 2000

Chairman: Sartaj Sahni
Major Department: Computer and Information Science and Engineering

Permanent implant of the prostate using I-125 and Pd-103 seeds is a popular choice of treatment for early stage prostate cancer in the United States. The evaluation of quality of implant is best done based on calculated dose distribution from post-implant CT images. This task however has been time-consuming and inaccurate. We have developed an algorithm for the automatic source localization from post-implant CT images. The only requirement of this algorithm is the knowledge of number of seeds present in the prostate, thus minimizing the need for human intervention. The algorithm processes volumetric CT data of the patient, and pixels of higher CT numbers are categorized into classes of definite and potential source pixels. Multi-thresholding technique is used to further determine the number of seeds and their precise locations in the CT volume data. A graphic user interface is developed to facilitate operator review and intervention of the calculation and the results of the algorithm. This algorithm was tested on two phantoms containing non-radioactive seeds, one with 20 seeds in discrete locations and another with 100 seeds with close distances between seeds. The tests show that the algorithm was able to identify the seed locations to within 1 mm of their physical

locations for discrete seed locations, and was able to separate seeds at close proximity to each other while maintaining an average seed localization error of less than 2 mm, with no operator intervention required.

CHAPTER 1
INTRODUCTION

Prostate cancer is the most common type of cancer in man in the United States, excluding skin cancer. In the treatment of prostate cancer, the role of interstitial permanent prostate brachytherapy has gained great importance.

In this treatment of prostate cancer, radioactive seeds are implanted permanently in the tissue. Before implants are done, a pretreatment dosimetric planning is done to determine how much dose the patient needs and where the seeds (radiation sources) should be implanted. Implants are planned to deliver an optimum dose. However, since patients differ in anatomy and seed placement accuracy depends upon the skills of the practitioner, seed placement errors can occur and planned dose is rarely achieved. To evaluate the success or the failure of the operation, a post implant dosimetric evaluation is needed as an indicator of the dose produced on the prostate and surrounding organs. This analysis can be done by performing a CT-based dosimetric evaluation, which allows prostate, and other organs as well as the seeds to be visualized. Using CT scans, dose-volume histograms (DVHs) are generated to analyze the dose coverage and quality of implants. CT-based dosimetric evaluation is important for a healthy and successful treatment program.[1]

CT-based evaluation requires analysis of axial image sets which display slices of the patient's anatomy. These slices represented by axial images usually have widths of 3,4, or 5mm. The sources used (usually 4.5 mm in length), rarely end up precisely where

they were planned to be in the pretreatment plan. The displacements may change the intended dose coverage considerably. Therefore, from the axial images, radioactive seeds have to be identified and the resulting dose coverage has to be recomputed. The problem that arises here is the identification of the seed locations with exact precision. Slices can be thinner than the seed length or can be oriented in such a way that a single seed can appear in more than one slice. A physicist has to look at the image set and make up a data set of seed locations, then he/she has to eliminate the redundant locations due to a single seed's appearing on more than a single slice. This is a tedious and time-consuming process, which bears an incentive for automation.

In the remaining parts of this paper, we describe a completely automated algorithm to locate seeds from CT images. This algorithm has been implemented and tested using both phantom and human data. Our seed location software system and experimental results are also described.

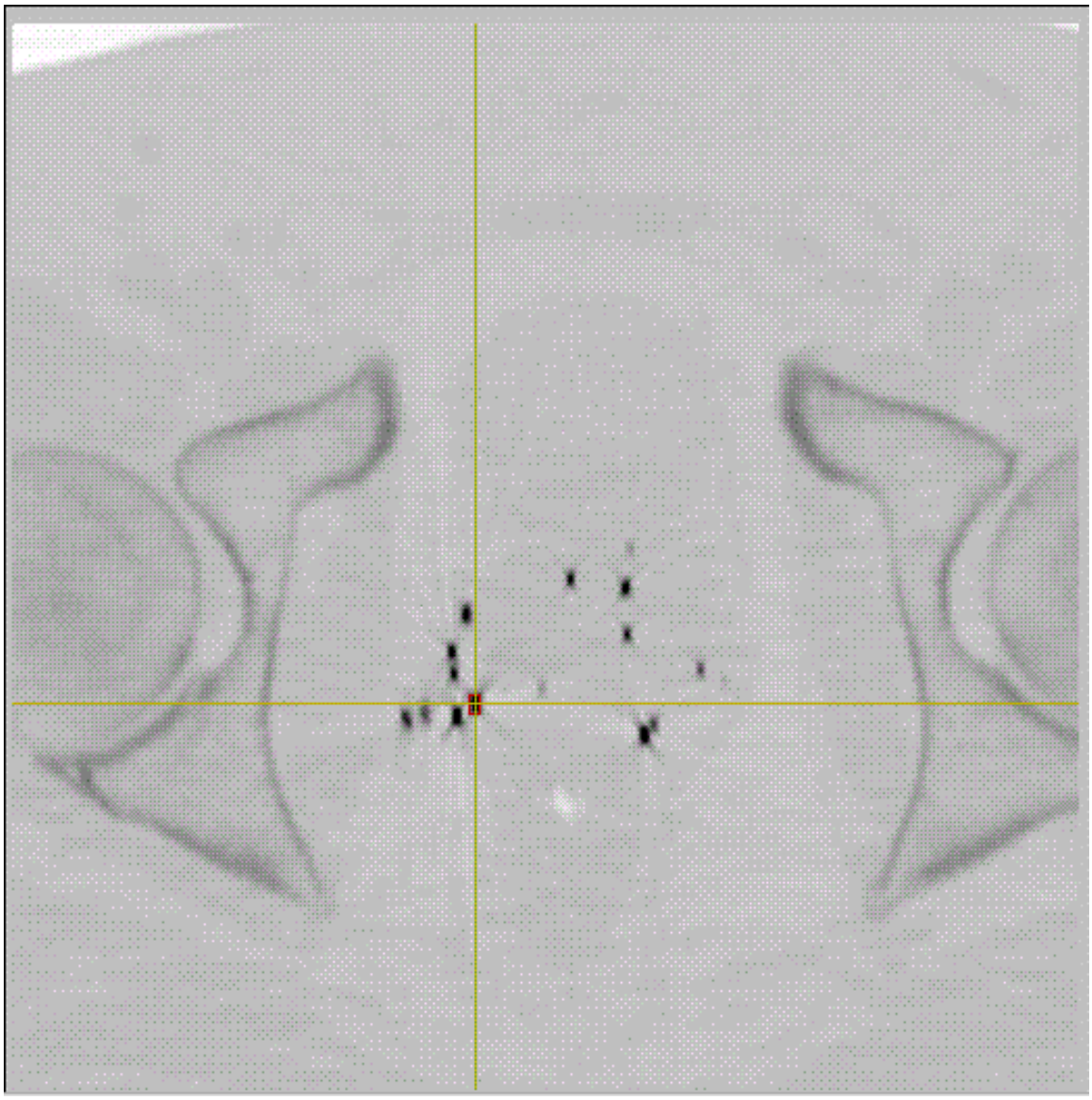

Figure 1-1: A real patient’s CT-scan, inverted in grayscale for image quality

CHAPTER 2
OVERVIEW

Several approaches have been made to automate the seed locating process. Most of the existing methods are based upon the nearest neighbor criterion and they are semi-automated. The user must first enter source locations in *(x,y,z)* coordinates in each slice. There are always more source locations derived from these axial images then the actual number of seeds implanted. The actual number of implants is retrieved from patient records. Then these methods try to reduce the number of entered locations to the exact number of seeds. These methods usually use the fact that sources on abutting slices, which have less than a certain distance can be considered the same seed. In their paper published in 1993, Roy *et al.* introduce such a technique for reducing the number of source locations.[2] In 1995, Feygelman *et al.* describes a spreadsheet technique as a modification of the Roy *et al.* technique, to interactively determine physical seed locations from the post-implant CT images.[3] In 1999, Bice et al. published another version of automating the source location reduction process, using the nearest neighbor concept.[4] Their algorithm iteratively modifies the nearest neighbor criterion, making multiple passes during the same iteration to reduce the number of source locations.

Our method, described in the next section, does not require any user intervention. Instead of asking the user to input the locations of the sources on the slices, it takes in the whole CT image as raw input and processes it to yield a list of source locations. However, through the visualization tool provided, users can make manual changes to the

results of the program. Our method consists of two parts. In the first part a sophisticated algorithm processes the CT image, filters noisy and redundant pixels from the image and feeds the necessary input to the second stage. In the second part an algorithm works on the provided input, and generates a list of source locations. We use a different approach than is used in the previous work of Roy *et al.* and Bice, Jr. *et al.*. Instead of treating the slices individually, we process the slices in 3-D and perform a 3-D clustering on the images. Using the number of seeds and the volume of clusters, we define an expected seed volume, and then we distribute the sources to clusters in such a way that bigger clusters get more seeds.

CHAPTER 3
METHOD

3.1 Initial Filtering Algorithm

Patient data, which consist of CT slices, is stored in a binary file. The data consist of ω Q×Q matrices, each of which constitutes one slice of the CT data. Pixels of the slices are represented by elements of the matrices and each has a value between 0 and 4095 (COLOR_RANGE), 0 being black and COLOR_RANGE being white. Radioactive seeds tend to have pixel color values in the upper part of this spectrum. We first read this data into a 3D array.

In a CT scan, along with seeds, the bone structures also show up as color contrasts. The main challenge faced in automatic seed detection is filtering the bone structure from the images. Using a single threshold to filter the bones doesn't work, because some pixels, which actually belong to a seed, can have pixel color values equal to that of pixels that belong to a bone. Setting up a very high threshold to filter bones may cause loss of data, as some of the pixels belonging to seeds will also be filtered. Setting the threshold to a low value may cause relatively bigger error, as bone structures can be erroneously classified as seeds. Note that, filtering out some pixels belonging to the seeds is to be favored over retaining pixels that belong to bones.

As a result of our observations, we made the following assumptions, which led us to a two level thresholding:

1-) A seed's image has at least one very bright pixel. Therefore all seeds have at least a pixel that has a color value as high as some threshold.

We call this threshold as MAGIC_NUMBER. This assumption is a fair one, as we know that seeds are the brightest elements in the images, and usually have the highest pixel values. In our implementation, in a spectrum of 0-4095, we have chosen our MAGIC_NUMBER as 3000. According to our analysis of data sets available to us, all seeds in the images had a pixel, which had a color value greater than or equal to this number. Usually all of them had a pixel with a value greater that 3500, and most of them had a pixel with a value between 4000-4095, which is bright white.

2-) Bone structures cannot have a pixel, which has a color value as high as MAGIC_NUMBER.

If bones had as bright pixels as the brightest pixel in a seed, even human experts would mistakenly identify bone fragments as radioactive implants.

3-) Bones tend to have pixel color values less than some secondary threshold.

We call this threshold in the program as THRESHOLD. For our spectrum of 0-4095, 2600 is used for this purpose. There might be bone structures that have pixel values higher than this THRESHOLD value.

Our filtering algorithm, in effect clusters all pixel groups, that have color values greater than THRESHOLD (2600) and have at least one pixel that has a color value greater than or equal to MAGIC_NUMBER (3000).

Using only one threshold, MAGIC_NUMBER, to filter out bones is not enough. As there are pixels both in bones and seeds which have color values within the range TRESHOLD-MAGIC_NUMBER(2600-3000), using the single threshold MAGIC_NUMBER would also filter out most of the pixels from the image that belonged to a seed. These erroneously filtered seed pixels are needed to predict the number of seeds belonging to a cluster. We use the second threshold THRESHOLD so as to reasonably include more pixels into our clusters.

The algorithm first filters out all pixels that are less than MAGIC_NUMBER. What are left, according to our assumptions, are the pixels that strictly belong to a seed (Type 1 pixels). Type 1 pixels are used to obtain the initial clusters.

The time complexity of this initial filtering is $O(Q \times Q \times \omega)$.

After the first clustering, pixels that are adjacent to the boundaries of the initial clusters are marked as boundary pixels (Type 2 pixels).

The algorithm then marks all pixels that are within an EPSILON-1 neighborhood of a boundary pixel (Type 3 pixels). The purpose in identifying these pixels is to detect the pixels that actually belong to a seed but have color values not high enough to beat the harsh filter threshold, MAGIC_NUMBER. This way, we may come to a closer approximation of number of pixels that belong to seeds.

Identification of Type 2 and Type 3 pixels also takes $O(Q \times Q \times \omega)$ time.

EPSILON is the length of the longest dimension of a seed in pixels. Our program computes EPSILON as, the length of the longest dimension of a seed divided by the length that a single pixel represents. Our system also allows reading in an EPSILON value from a configuration file. As the same type of seeds can show up in different

lengths in different images due to CT positioning, this allows getting a better EPSILON set by an expert. Picking an EPSILON too high may result in loss of performance, since we would be including pixels, which do not belong to seeds, which are going to be filtered in the next filtering stage.

Now we have marked pixels that have values greater than MAGIC_NUMBER (type 1) as well as those that have values less than MAGIC_NUMBER but being within an EPSILON neighborhood of a pixel of type 1 (type 2 & 3).

We now can make the following reasoning:

1-) if a pixel is of type 1, it belongs to a seed.

2-) if a pixel is of type 2 or type 3 and has color value greater than THRESHOLD, it belongs to a seed.

The second reasoning above lets us include pixels, which most likely belong to a seed even though they have a color value comparable to that of a bone. Looking at a neighborhood of a type 1 pixel as a requirement to belong to a seed, helps us filter out noisy pixels which happen to be either an exceptionally bright bone part, or some other noise in the image. If such pixel groups do not have a very bight pixel (i.e., color value ≥ MAGIC_NUMBER) among them (in their neighborhood) that can qualify being an exact seed pixel (a type 1 or type 2 pixel), then this pixel group can’t represent a seed.

After the filtering algorithm runs, pixels that survive the filtering are clustered into groups that represent at least one seed.

3.2 Assigning Seeds To Clusters

The next challenge is to predict the number of seeds lying within a cluster. In cases where multiple seeds are very close to each other, the image has a single big cluster and human experts have to decide the number of seeds contained in it. In our case, we use a simple algorithm that uses the volume of the clusters after filtering.

For this algorithm to work correctly, the number of seeds implanted is needed as input. Since surgeons count how many seeds they implant during the operation, this data is available to the program.

The algorithm is as follows:

We determine the total number of pixels that remain after the filtering steps. These are the pixels that belong to seeds. We divide this number by the total number of seeds implanted to get the expected single seed pixel volume(ESSV). Using ESSV and the cluster volume, we can estimate the number of seeds in a cluster.

Given the accumulated total volume of pixels (**V)** and the number of seeds (**N**), **ESSV** is computed as:

ESSV=V/N

3.2.1 First Phase of Seed Assignment

Let k denote the number of clusters, and let V_i be the volume of cluster i.

Note that,

$$V = \sum_{i=1}^{k} V_i$$

Let $m_i.d_i=x_i=V_i/ESSV$, where $m_i=\lfloor x_i \rfloor$ and d_i is the decimal part of x_i.

In the first phase of seed assignment, cluster i is assigned $s_i=max\{1,m_i\}$ seeds.

3.2.2 Second Phase of Seed Assignment

Let

$$N' = N - \sum_{i=1}^{k} s_i$$

be the number of seeds that remain unassigned. Let A be the set of clusters for which $m_i > 0$ and let B be the set of the remaining clusters. Since,

$$\sum_{i=1}^{k} x_i = \sum_{i=1}^{k} m_i . d_i = N$$

$$N = \sum_{i=1}^{k} m_i + \sum_{i=1}^{k} 0.d_i = \sum_{i=1}^{k} m_i + \sum_{i \in A} 0.d_i + \sum_{i \in B} 0.d_i$$

So,

$$\sum_{i \in A} 0.d_i = N - \sum_{i=1}^{k} m_i - \sum_{i \in B} 0.d_i$$

$$> N - \sum_{i=1}^{k} \max\{1, m_i\}$$

$$= N'$$

So, the number N' of seeds yet to be assigned is fewer than the number of clusters in A. The N' seeds are assigned to the clusters of A, one seed per cluster, in descending order of d_i. The time required for this assignment is $O(k \log k)$.

3.2.3 Third Phase of Seed Assignment

The heuristic described above assigns seeds to clusters. These clusters usually consist of multiple seeds. After phase II.B.2, we are left with clusters that have one or

more seeds assigned to them. In order to specify each seed's location more precisely, we need to further separate the big clusters into smaller ones and re-assign the seeds contained in each initial big cluster to their smaller subclusters.

As we know the number of seeds contained in the initial big cluster prior to separation, we can filter out the pixels in this cluster until the cluster separates into smaller clusters. This process is nothing but a reuse of the whole algorithm described in this paper, with a threshold set to a higher value.

3.2.3.1 Separating The Big Cluster

The initial domain is a 3-D cube in which the big cluster best fits. In this cube, lies a cloud of pixels, condensed in some areas and with lighter valued pixels in some areas. If there are two different groups of seeds lying in this cluster, the pixels, which connect these two, will have lighter pixel values than those that belong to the center of the groups. Filtering out those pixels yields the separate clusters we are looking for.

We start a series of filtering steps, using threshold values between THRESHOLD (set to 2600 in our case) and a maximum value MAX_THRESHOLD, which is close to the maximum possible pixel value(4095). In our data sets, all clusters had values greater than 4000, but not all of them had the maximum brightness(4095). So we have chosen 4000 as the MAX_THRESHOLD. In order to speed up the process, we increase the filtering threshold used in this process in steps of 200, which can be changed for better precision.

When a cluster partitions, in i^{th} iteration, we update a variable max_partition to the maximum number of subclusters ever found during this and the previous iterations,

and save the filter threshold for the maximum partitioning case. We stop the iteration on two conditions:

a) We have reached the MAX_THRESHOLD, i.e. we have executed all possible filtering modes.

b) The cluster has partitioned into more subclusters than the number of seeds assigned to it. This means that there has been excessive filtering which caused a seed to separate into more than one piece.

In this phase, the time spent is proportional to:

$$V \times \text{Iteration Count}$$

$$= O(\text{Number of Seed Pixels} \times \text{Iteration Count})$$

So, Phase 3.2.3.1 has a time complexity of O(*N*).

3.2.3.2 Distribution of Seeds to Subclusters

This phase is an exact repetition of B.1 and B.2. We divide the volume of the cluster with the number of seeds assigned to it and find **ESSVC** (Estimated Single Seed Volume in Cluster). Dividing the volumes of subclusters by ESSVC, we find the integer and decimal components of the subclusters. We first assign as many seeds to a subcluster as its integer component. Then we set up a priority queue and distribute the remaining seeds of the big cluster to its subclusters.

This phase, since bounded above by the number of seeds implanted, requires O(*NlogN*) time.

3.3 Setting Coordinates For The Seeds

After the completion of phase 3.2, we have a list of subclusters with seed counts associated with them. The output of the algorithm is defined as seed coordinates in 3-D space.

For clusters which have multiple seeds associated with them, more then one seed will appear to have the same coordinates in the output. If we could have separated them, they would appear in different clusters. This, however, means that these seeds are very close to each other, and the error we get from declaring them to exist in one single coordinate will be very small.

Picking a point in a 3-D cluster to represent the location of the seed most precisely is the next step of the algorithm. The program represents clusters as 3-D rectangular prisms, identified with one corner of the prism and length of its dimensions. Our first approach was to associate the seeds within a cluster with this corner point of the prism. However, we decided to use another approach, which yielded superior precision. We search for the point in the cluster that has the highest pixel color value. This pixel surely belongs to a seed within the cluster. We associate the x and y coordinates of this pixel with the seeds belonging to this cluster. Then we evenly distribute the seeds along the z dimension of the cluster. In x-y dimensions this brings very little error since pixels of a slice represent the x-y plane with much higher resolution than consecutive slices' representing the z dimension. A pixel length usually represents 0.039 cm., while slice thickness can be 0.150-0.500 cm.. When we go by one pixel in x-y plane, we only cover

0.039 cm. while we may cover up to 0.500 cm. when we go to a next data point in x-z or y-z plane.

Finding the highest pixel color value in a seed requires time proportional to maximum possible number of pixels in a Seed, which is a changing constant for different image sets. At most we can have N seeds, so this phase requires $O(N)$ time.

CHAPTER 4
TEST RESULTS

We have implemented the algorithm that is described above. For the test runs, we prepared two different wax phantoms. The first phantom contained 20 radioactive seeds. There was enough spacing between the seeds so that it was designed not to challenge the accuracy of the algorithm in a situation where the seeds are very close to each other. The program found all of the seeds and reported their exact locations.

In the second phantom, there were 100 seeds. Some of the seeds were placed close to each other to form clusters that contain multiple seeds. From this phantom, two different data sets were generated. One of them consisted of 1.5 mm. thick CT-scan slices, and the other one consisted of 3 mm. thick slices of the same phantom.

We ran the program on both data sets to test the program for situations where seeds are placed very close to each other, and also to analyze the effect of slice thickness on the program's performance. We then compared the location sets generated by the program with the location sets manually generated by 2 different experts.

In order to measure the error distances between any two location sets, we select one set to be the reference set (set A) and the other as set B. For all locations in set B, we find a location in set B which minimizes the distance in 3-D. This process assigns more than one location to some locations in set A, leaving some locations in set A unassigned. Then, we manually distribute extra locations from set B that are assigned to locations in set A, to unassigned locations of set A if the distance between them is under 6mm. In test

phantoms that we have used, 6mm corresponds to 15 pixels in x and y dimensions, 4 slices in 1.5 mm thick sliced phantom and 2 slices in 3mm sliced phantom. If two locations from different sets have that much distance, they may not denote to the same seed. Locations in set A that go unassigned are the locations that set B misses to match (undiscovered seeds), and the extra locations assigned are the erroneous extra locations. The slice thickness effects the number of seeds discovered by the program. When slice thickness is 3 mm., moving one pixel in x-z or y-z dimension covers twice the volume as it does when slice thickness is 1.5 mm. This precision loss effects the total volume computation, which is used to decide the number of seeds that a cluster contains. Test runs prove that our program performs better with thinner slices.

For the 1.5 mm. thick sliced phantom, 95 out of the 100 seeds that the program has reported matched with findings of the experts', leaving 5 seeds undiscovered. As for the 3 mm. thick sliced phantom, the program was successful in identifying 91 seeds exactly, leaving 9 seeds undiscovered.

We also used two other measures to evaluate the performance:

Let (x_1,y_1,z_1) and (x_2,y_2,z_2) be two different locations proposed for the same seed by two different agents (program or an expert). N is the number of seeds.

$\Delta x=x_1 - x_2, \Delta y=y_1 - y_2, \Delta z=z_1 - z_2, \Delta r=(\Delta x^2+\Delta y^2+\Delta z^2)^{1/2}$

Maximum Distance(MD): $\text{Max}\{\Delta r\}$

Average Distance(AD):

$$AD = \frac{\sum_{i=1}^{N} \Delta r}{N}$$

The following tables display the differences between the data sets generated by the program and the experts.

Table 4-1: Maximum Error Distances

MAXIMUM DISTANCE	1.5 MM. SLICES	PHANTOM 3 MM. SLICES
Program and Expert A	0.33 cm.	0.35 cm.
Program and Expert B	0.31 cm.	0.34 cm.
Expert A and Expert B	0.19 cm.	0.48 cm.

Table 4-2: Average Error Distances

AVERAGE DISTANCE	1.5 MM. SLICES	PHANTOM 3 MM. SLICES
Program and Expert A	0.14 cm.	0.19 cm.
Program and Expert B	0.14 cm.	0.17 cm.
Expert A and Expert B	0.08 cm.	0.09 cm.

In 3 mm. thick phantom, expert B also failed to locate 2 seeds out of 100.

Maximum distance between the findings of the program and the experts' can be up to 3.5 mm. for seed locations that are matched with those of the experts'. The findings of the experts among themselves can have a distance of up to 5 mm.

As seen in the tables, the average distance between the location sets did not increase as much as we expected with the increasing slice thickness. When the slice thickness is 3mm, the precision in assigning seed coordinates decreases compared to the precision when slice thickness is 1.5 mm (Average Distance Table). In 3 mm. case, if a

cluster shows up with bright pixels in 3 consecutive slices, the program decides that the cluster spans 9 mm. in z dimension. However, it usually is not the case that the cluster actually spans through the whole 3 mm. of the top end and the bottom end slices. This small deviation in the assignment of the coordinates is not the big problem that is brought by increasing slice thickness. As the same error happens both on the top and the bottom end slices, the seed center assignments do not deviate a lot.

CHAPTER 5
THE PROGRAM

Program is executed from the command line by

needle <header file name> <number of seeds in the data>

The input data consists of 2 files, which has to be placed in *data* directory. These files have the patient's name, and an extension to specify their contents.

"John_Smith.img", will hold the image data file, which is a binary file, containing the color values of pixels in a row-wise order, slice by slice.

"John_Smith.header", contains the information regarding the image data file.

The program can be run on this sample data set by

needle John_Smith 102

assuming that 102 seeds were implanted in this patient.

The following items describe the contents of this file, along with their explanations:

vol_min = 0.000000;

vol_max = 4095.000000;

These two values state that color value of a pixel may lie within the range 0-4095.

x_dim = 512;

y_dim = 512;

z_dim = 39;

These three lines mean that, the image file consists of 39 512×512 slices.

dim_units : cm

This line tells the unit of the dimensions that are given in the following lines.

x_pixdim = 0.039062;

y_pixdim = 0.039062;

z_pixdim = 0.150002;

Each pixel is actually representing 0.039 cm in x and y dimensions, and the slice thickness is 0.15 cm.

x_start = 0.000000;

y_start = 0.000000;

z_start = 127.599998;

These three lines give the starting coordinates for the image slices according to a global origin.

db_name : John Smith

date : 9/24/1998

medical_record_num :

These lines contain information about the patient that the CT-images belong to.

The output of the program is a special file, which can be imported by *ADAC Pinnacle* software system, a commercial treatment-planning tool. It has the same name as the input file, and has the extension *.plan*.

The program also writes out a log file for the user to trace the execution process. It has the same name as the input file and has the extension *.log*.

All output files are kept in *output* directory.

The following is a list of files with the directory structure for a sample patient, "John_Smith":

Table 5-1: Files Associated With The Program

File Name and Path	Description
needle	Executable file(The program)
data/John_Smith.img	Image data file in binary format
data/John_Smith.header	Information file for image data
output/John_Smith.plan	Output file, ready for post-implant treatment planning.
output/John_Smith.log	Log file for the program execution

CHAPTER 6
THE USER INTERFACE

When the program finishes determining the seed locations, it pops up the user interface. It chooses the first cluster that is identified as cluster 0 to display first. The user later can change the user interface's focus to other clusters.

The user interface consists of many panels that help the user visually analyze the image data. It allows navigating through cross section images of the 3-D structure, zooming in and out, and getting information about the clusters that the program has identified. The user is provided with information on the number of assigned seeds and is allowed to make changes to it. If the user decides that a certain cluster contains more seeds than the program has assigned, or vice versa, he or she can change it manually through the user interface.

With a click of a mouse, he or she can jump through the clusters and visually analyze the program's output, incorporating the given measures of the cluster, its volume and its image with his or her expertise.

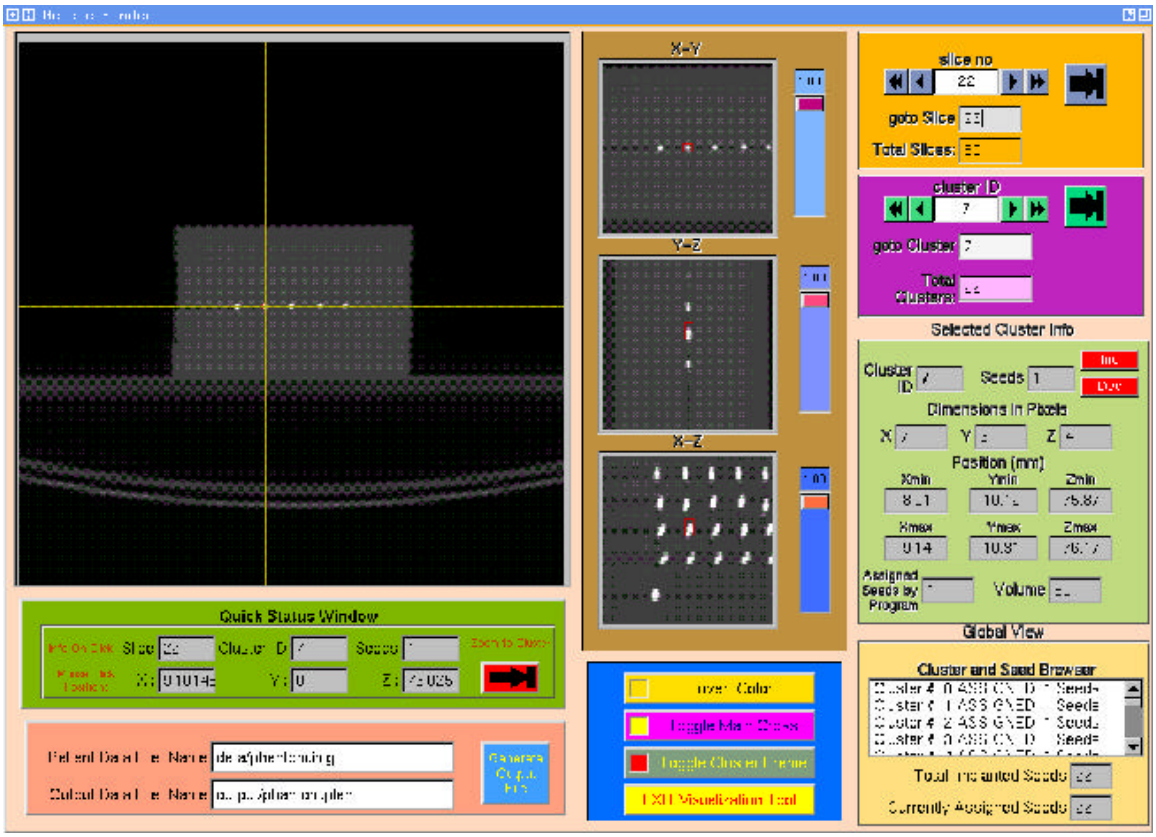

Figure 6-1: The User Interface

6.1 Main Window

This window displays the whole slice that the user is currently interested in. On this main window, one horizontal and one vertical, two yellow lines mark the focus of the user interface at their intersection point. The focus of the user interface is always some cluster that is being viewed. These lines help the user distinguish the current cluster from others that are visible on the slice. The user interface uses a current slice notion to display the visual data. When the user decides to view cluster number *n*, the focus of the user interface becomes cluster *n*'s center and the current slice is the first slice in the +Z

direction that this cluster shows up. After this, using the control buttons provided, the user can navigate through the slices to see how the cluster extends in the slice sequence. The user interface also uses a current cluster notion to display the data and related information. When the current cluster is *m*, the yellow marking lines show the center of cluster *m* on the main window. On other panels, information regarding the current cluster, in this case cluster *m*, is displayed.

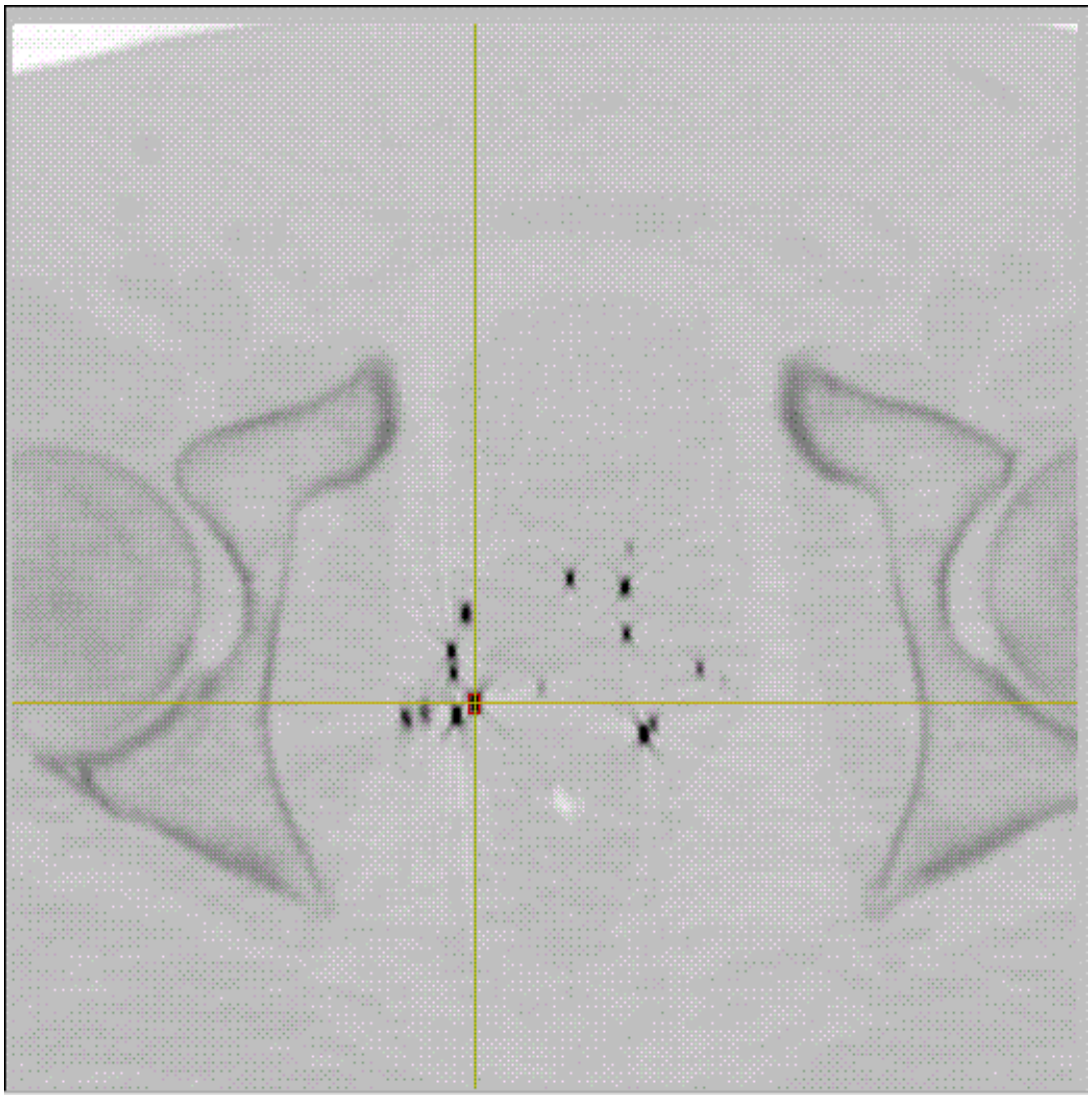

Figure 6-2: Main Window, inverted in gray scale.

6.2 Cross Sectional Display Windows

These three windows display the current cluster and its neighborhood in X-Y, Y-Z and X-Z planes respectively. When the user changes the current cluster, the image of the new cluster is brought to the center in these windows, encircled with a rectangular red frame to distinguish it from the other clusters seen in the window. The main property of these cross sectional display windows is their capability to zoom in, up to 10 times the original image shown in the main window. The sliders placed next to each window allow the user to zoom in and zoom out on a scale between x1 to x10. Being able to see the layout of clusters in X-Z and Y-Z planes, the user can have a better idea on the size and the contents of the clusters, so that he or she can manipulate the program's results when appropriate.

6.3 Quick Status Window

This panel gives instant information on the points that the user is interested. Whenever the user clicks on a point on the main window or on one of the cross sectional display windows, information regarding that point is displayed on this panel. This window displays the slice's number that the point is contained in, as well as giving the metric information on the location of the point, in X, Y and Z dimensions. If the point clicked is within an identified cluster, the cluster's identification number and the number of seeds assigned to it are displayed. If the user wants to navigate to

that cluster by making it the focus of all windows, he or she simply clicks on the "zoom to cluster" button, and the cluster, in which the previously clicked point lies, becomes the "current cluster". If the clicked point is not contained in any defined cluster, this panel shows it as if it belongs to cluster number -1, which is undefined.

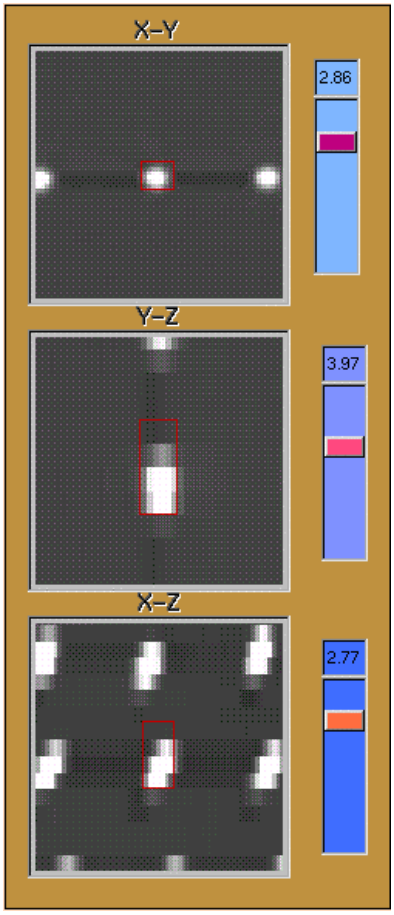

Figure 6-3: Cross Sectional Display Windows

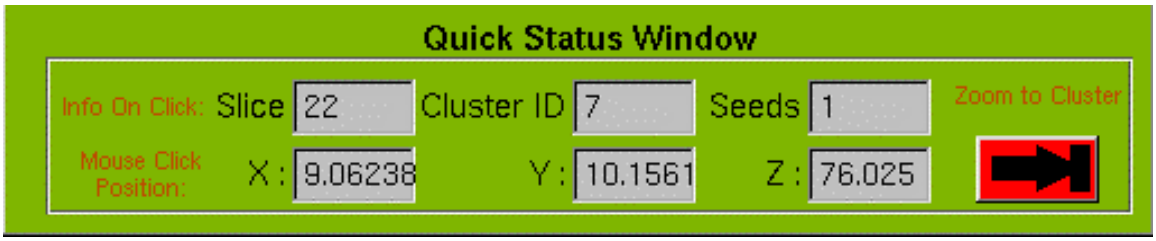

Figure 6-4: Quick Status Window

6.4 Slice Navigation Panel

This panel lets the user change the displayed slice on the main window and the X-Y cross sectional window. When the current slice is changed, the images displayed in X-Z and Y-Z windows are also changed and updated, as their Z components, which denote the slice number, are changed as well.

The single arrows let the user navigate slice by slice. The double arrows let the user increase or decrease the current slice number by 5 slices, allowing a faster navigation.

The user is also given the option to enter the slice number he or she wishes to see in the "goto Slice" edit box. The big button distinguished with a special sign takes the user to that particular slice. At the bottom of this panel, the total number of slices in the image data is provided to the user.

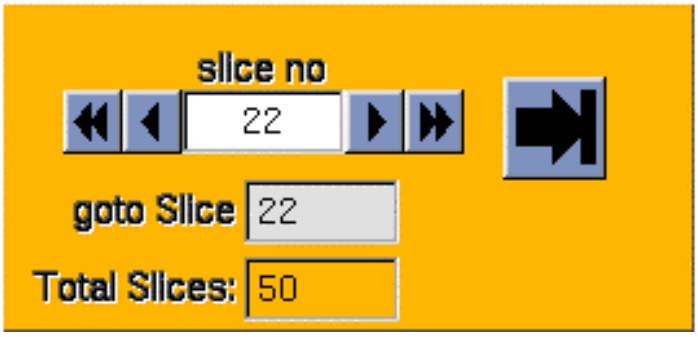

Figure 6-5: Slice Navigation Panel

6.5 Cluster Navigation Panel

This panel lets the user change the current cluster that the attention of the user interface is focused on. When the current cluster is changed, the main window displays the first slice that this new cluster shows up in +Z direction, and all cross sectional windows display this cluster in the center of their views.

The single arrows let the user navigate cluster by cluster. The double arrows let the user increase or decrease the current cluster number by 5 slices, allowing a faster navigation through the clusters.

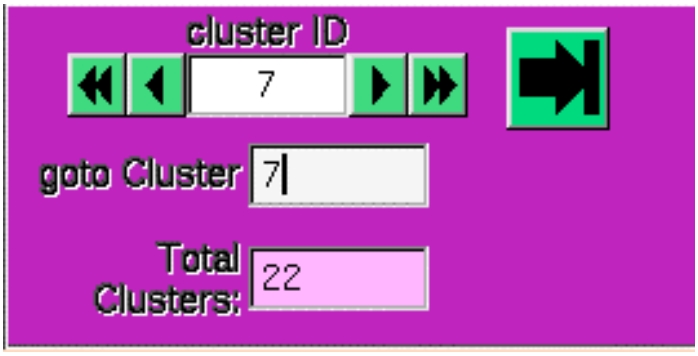

Figure 6-6: Cluster Navigation Panel

The user is also given the option to enter the cluster's identification number he or she wishes to see in the "goto Cluster" edit box. The big button distinguished with a special sign makes that particular cluster the center of focus in all windows. At the bottom of this panel, the total number of clusters identified is presented to the user.

6.6 Selected Cluster Information Panel

This panel displays information about the current cluster. It displays the cluster's identification number and number of seeds in it. If the user has made a change in the number of seeds that this cluster contains, it displays the manually set number. Next to the "Seeds" text box which displays the number of seeds assigned to this cluster, there are two red buttons, marked with "Inc" and "Dec". Using these buttons, the user can increase or decrease the number of assigned seeds within the current cluster.

In the program's memory, each cluster is represented by a rectangular prism of pixels. This prism is the best fitting smallest rectangular prism that contains all the pixels within the cluster. In this panel, the dimensions of this rectangular prism are also displayed. If the dimensions are X:7 Y:6 Z:4, then this cluster spans 7 pixels in X dimension, 6 pixels in Y dimension and it is contained in 4 consecutive slices.

The dimensions of this rectangular prism are also given in metric numbers which represent the exact location of the cluster within the 3-D space whose CT-images were used as input to the program.

At the bottom the volume of the cluster is given to the user, so that he or she can reflect on the number of seeds assigned to this cluster.

This panel also displays the initial assignment of number of seeds made by the program for this cluster.

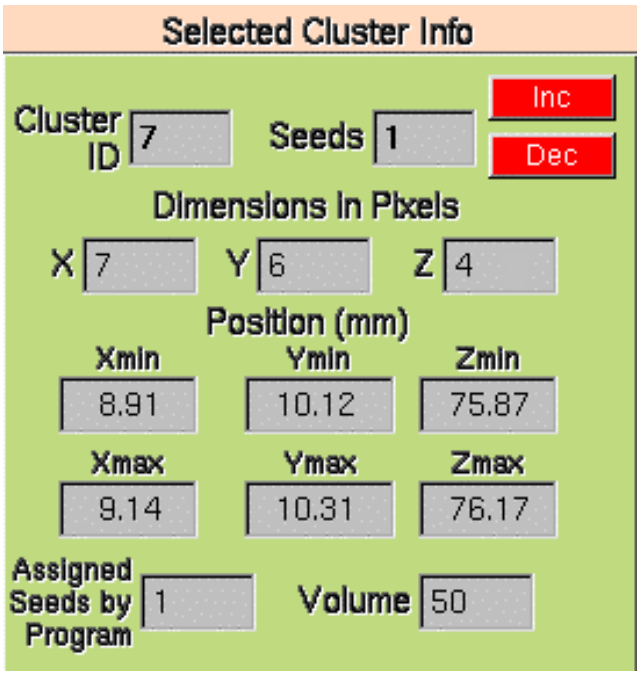

Figure 6-7: Selected Cluster Information Panel

6.7 Global View Panel

The "Cluster and Seed Browser" located in this panel shows the current status of seed assignments to the clusters. It contains a list of clusters and seed counts assigned to them. If a cluster is assigned more seeds (by user intervention) than the program has decided, then that cluster is displayed in red. In case the user decides a cluster should have fewer seeds than the program has assigned to it, the cluster is displayed in blue. This browser is useful in the sense that it gives an overall view on the assignments made throughout the session.

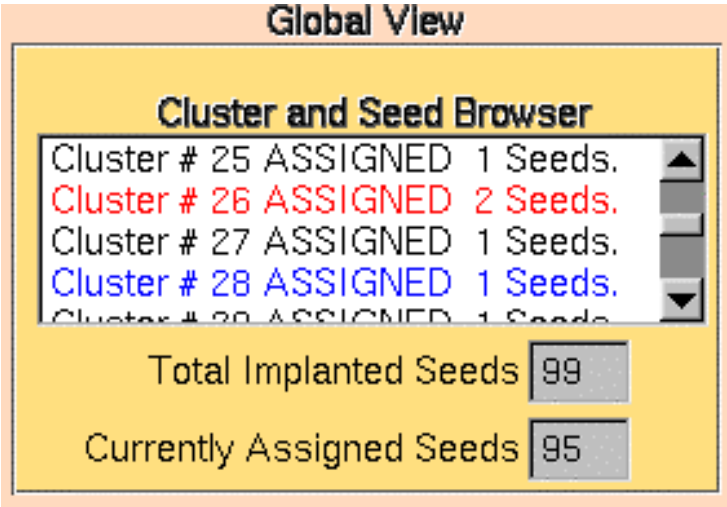

Figure 6-8: Global View Panel

In two text boxes, this panel displays total implanted seeds and the total number of seeds currently assigned. This information helps the user decide whether he or she has assigned correct number of seeds in total.

6.8 Toggle Buttons

In this panel there are 4 buttons that allow the user to change the user interface settings.

Invert Color button reverses the color spectrum of images. Color value of all pixels in the image windows changes to (4096-Color Value).

Toggle Main Cross button toggles the view of the main cross in the main window on and off.

Toggle Cluster Frame button toggles the red frame surrounding the clusters in the cross sectional view windows.

Exit Visualization Tool button saves the locations of the seeds in a special output file and terminates the execution of the program.

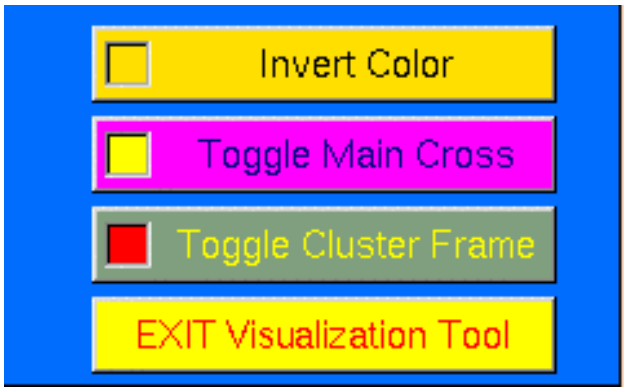

Figure 6-9: Toggle Buttons

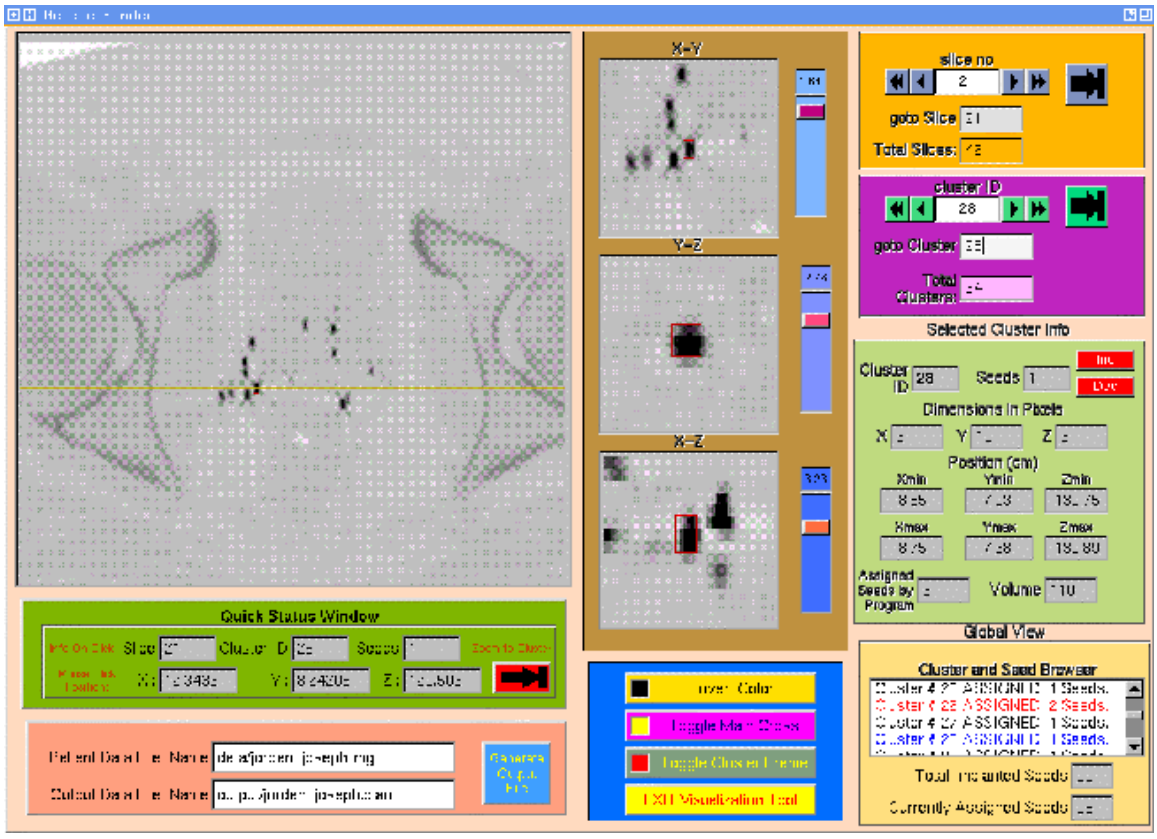

Figure 6-10: User interface, when image data colors are inverted in grayscale.

6.9 File Operations Window

This window displays the input data file name and path, as well as the output file name and path, in which the seed locations are being written.

The program writes the locations of the seeds in a specially formatted file, so that the file can be imported to *ADAC Pinnacle*, which is a commercial visualization tool used in brachytherapy treatment planning.

Generate Output File button re-writes the output file on demand, as user is allowed to change the number of assigned seeds in clusters, to account for a different output file.

Figure 6-11: File Operations Window

CHAPTER 7
CONCLUSION AND FUTURE WORK

In this paper, we have described a fully automated post-implant seed detection algorithm. To the best of our knowledge, this is the only algorithm that uses image processing to detect the locations of the seeds directly from the image slices in a fully automated way. The test results demonstrate that when we compare the locations found by the program and the experts in pairs, the average difference in reported locations is less then 2 mm. in all cases. The maximum error distance between experts A and B can be as big as 4.8 mm. This maximum distance is at most 3.5 mm when the comparison is done with one of the experts and the program.

We can conclude that the program reports the locations precisely as long as it can identify the correct number of seeds in the correct clusters as the average error distances are below 2 mm. in both cases. The success rate of the program depends on the slice thickness, as slice thickness has a major effect on the precision of the images. When taken into account that expert B also failed to locate 2 seeds out of 100, the decrease in the performance of the program is expected, and can be bound to this fact. The only drawback in the performance of the program is that, when it fails to find a seed in its correct place, it assigns an extra imaginary seed to some bigger cluster, in order to match up with the total number of seeds implanted. This means that, every time it fails to identify a seed, the failure percentage increases twice, as the extra assigned seed should

not be there. However, through the user interface, a single pass over the results can be done and this error can be voided.

As further research, the algorithm described in this paper can be used together with existing nearest neighbor search algorithms to yield a hybrid and more robust algorithm. Instead of a 3D clustering, a 2D clustering followed by 2-D source location can be done on each slice. These 2-D source locations can then be used as inputs to other existing 3-D source location algorithms [2,3]. By doing this, we eliminate the need for a human to provide source locations in each slice as is required by those algorithms[1,3].

REFERENCES

[1] Y. Yu, L. L. Anderson, Z. Li, D. E. Mellenberg, R. Nath, M. C. Schell, F. M. Waterman, A. Wu and J.C. Blasko. Permanent prostate seed implant brachytherapy: report of the American Association of Physicists in Medicine Task Group No. 64. In Medical Physics, 26(10), 2054-76, 1999.

[2] J. N. Roy, K. E. Wallner, P. J. Harrington, C. C. Ling and L. L. Anderson. A CT-based evaluation method for permanent implants: application to prostate. Int. J. Radiat. Oncol.. Biol. Phys., 26, 163-169, 1993.

[3] V. Feygelman, B. K. Noriega, R. M. Sanders, J. L. Friedland. A spreadsheet technique for dosimetry of transperineal prostate implants. Medical Physics, 22(1), 97-100, January 1995.

[4] W. S. Bice, Jr., D. F. Dubois, J. J. Prete and B. R. Prestidge. Source localization from axial image sets by iterative relaxation of nearest neighbor criterion. Medical Physics, 26(9), 1919-1924, September 1999.

BIOGRAPHICAL SKETCH

Ismail Ahmet Nalcacioglu was born on September 6th, 1976, in Ankara, Turkey. He received his Bachelor of Science degree from Bilkent University, Ankara, in July 1997, majoring in computer and information science and engineering. He joined the University of Florida in August 1998 and started to pursue a master's degree in computer and information science and engineering. He conducted research as a research assistant for Dr. Sanjay Ranka and Dr. Sartaj K. Sahni., and worked as a teaching assistant during his master's degree.

His research interests include algorithms, data mining and data warehousing.