Semi-Automatic High-Quality Reconstruction of the Heart Pathology from MRI Data by Soft Objects

Bo Li

Supervisors:

Burkhard Wuensche and Alistair A. Young

August 2004

Table of Content

Abstract

The most serious problem of the Heart Disease is that it will lead to Heart Failure, which is a serious condition in which the heart is not pumping well enough. Since the prognosis after Heart Failure is poor, people use computer visualization techniques to diagnosis and analysis the cardiac disease to prevent the happen of the Heart Failure. Over the last ten years, a variety of efficient and excellent visualization techniques for the heart are introduced. However, most of these techniques require either a relevant amount of user work or a default topology of a healthy heart. None of them is efficient enough for the diagnosis of congenital heart disease caused by the abnormalities of the heart pathology. Dr Alistair, Dr Burkhard and I try to develop a new heart-modeling technique which can be used to efficiently model the abnormal heart pathology from noisy MRI data. In this project, I improved the program from my previous projects [8, 9] so that right now my program works well for the noise heart MRI dataset and can be used to fast approximate the heart pathology. The feedback from the medical academic, Dr Alistair Young in the department of Anatomy with Radiology, about my program is displayed and relevant changes according to the feedback of my program will also be introduced in this paper.

Keywords: *Heart Failure, Congenital Heart Disease, Soft Object Modeling, Active Contour Model (B-Spline Snake), Radial Basic Functions (RBFs), Iso-surface, Marching Cube, Surface reconstruction, Skeleton.*

1 Introduction:

Nowadays heart disease remains the number one killer of the world. Because of the cause of the heart disease, our heart try to work header and header, which only makes the problem worse and worse. And finally lead to Heart Failure. Heart Failure (or Congestive Heart Failure) is defined as a serious condition in which the heart is not pumping well enough. Once the Heart Failure happens, it's very difficult to resume to the healthy status. How to effectively diagnose and treat heart disease becomes a popular problem in the medical area. Accurately constructing and modeling the pathology of the heart is recognized as an effective way to help the medical experts improve their diagnosis and treatments.

Over the last ten years, a variety of visualization techniques which either manually or semi-automatically visualize the ventricles of the heart are introduced [10, 11]. However, most of these techniques are based either on a relevant amount of user work or on a default topology of a healthy heart. For example, construct the shape of the heart by tracking the contour of the interest area in each MRI data slices by hand, and then integrate these sample data to derive a 3D topology of the heart. This process will take reasonably a mount of work for user to develop a heart model and it's never considered practical at all. This approach can be dramatically accelerated by make use of a sample healthy heart model, and then apply and adjust this healthy model to fit into the real

heart data. Since only a much fewer modifications are needed in this accelerated approach, it's widely used by medical experts.

However, none of above approaches is efficient enough for the diagnosis of congenital heart disease caused by the abnormalities of the heart pathology. The congenital heart diseases are recognized as having abnormal topology compared with healthy heart. For example, an abnormal connection of one or more arteries leaving the heart, an abnormal connection between two heart chambers, and one of the 4 heart valves may be deformed or absent and so on. Since the pathology of the abnormal heart is far away from the pathology of the normal healthy heart, previous accelerated approach is never effective in this situation.

Dr Alistair and Dr Burkhard come out an idea of fast and semi-automatically reconstructing heart pathology with Soft Object Modeling technique. This approach is very efficient and also very tolerate of the abnormalities of the heart. In this paper, I will introduce my implementation of this new approach of modeling heart pathology. With the help of my program, medical experts could easily and fast approximate a 3D structure of the abnormal heart pathology. So that Medical experts could either quickly diagnosis the congenital heart disease or apply this approximated model to develop a precise model with previous accelerated approach. The performance of my program have been tested by medical academic, Dr Alistair Young in the department of Anatomy with Radiology, a series of relevant changes according to the feedback have been made to my program, so that not only it's efficient to be used for heart modeling, it also very friendly and convenient for medical experts.

The rest of this paper is arranged as following:

- Background. Introduce in detail about how this new approached can be used to model heart pathology and how I implement it in previous projects [8, 9] and this projects. (**Section two**)
- Feedback and relevant changes in the user interface. Introduce the feedback about my program from medical academic and the modifications of my program according to the feed back. (**Section three**)
- Hierarchy of the program. Introduce the hierarchy and management of all the components mentioned in previous sections (**Section four**)
- Problems. Introduce the problem left (**Section five**)
- Conclusion. Conclude this paper by mentioning hardware environment and future improvement. (**Section Six**)

2 Backgrounds:

In this section, I am going to first introduce the Soft Object Modeling Techniques introduced by Wyvill in 1986 [1]. Then I briefly explain the main idea introduced in the project to efficiently model the pathology of the heart. Finally, I will mention the problems encountered while I was implementing the new approach and how I solve these problems.

Soft Object Modeling

Soft Object Models introduced by Wyvill in 1986 [1], also known as blobby objects, are a type of implicit modeling technique, are defined as Iso-surface $f(x) = v$ of a density field. The density field is created by a field function $D(r,R)$ which is defined as the sum of distance functions to simple geometric primitive. By simply summing the density value from each field function on a given point, we can get very smooth union of the models to which previous field functions belong. Rather than summing, there are also some other blending operators which could be used to achieve different results, e.g. differencing, to derive the intersection area of models. The field function used in my program is defined as:

$$D(r,R) = 1 - \frac{4}{9}\left(\frac{r}{R}\right)^6 + \frac{17}{9}\left(\frac{r}{R}\right)^4 - \frac{22}{9}\left(\frac{r}{R}\right)^2$$

Since Soft Object Models have the characteristics of easily defined, easily controlled by mouse, and smoothly blending together. My program provides users a 3D environment so that they can easily define and use Soft Object Models to produce new model, which can be used to approximate the 3D area of interest on MRI data slices. With the help of these Soft Object Models provided in my program, rather then tracing the contours on each slice manually, users can easily combine several Soft Objects to achieve the purpose of roughly approximation. There are basically three types of Soft Objects provided in my program:

- Soft Ball: The skeleton of a Soft ball is a single 3D point in space. I used the Euclidean distance function: $dis = \sqrt{(x-x_0)^2 + (y-y_0)^2}$ to find out the distance of any point in space to its skeleton. Users can adjust the size of the soft ball by changing the range of the density fields affected by this single point.

- Soft Cone: The skeleton of a Soft Cone is a 3D line segment. We used a simple method (as shown in **Fig.2**) in our program to find out the distance for a point to the line segment. Users can adjust the affected range at each end of the line segment separately to achieve different shape of Soft Cone: tube or a cone.

- Soft Curve: The skeleton of the Soft Curve is a Catmull-Rom Spline Curve [4]. The major characteristic of Catmull-Rom Spline curve is that it produces a smooth curve passing through all the given controlled points on the curve. This gains advantage of letting users more accurately represent the curve compared with other curves (B-Spline curve, Bezier Curve, etc). Since it's very difficult and time –consuming to work out the distance between a 3D point and a 3D curve segment, I used a fast method to approximate this distance. As shown in the curve in **Fig.1**, we can see that for every two control points of the Catmull-Rom Spline, the curve (black

curve) between them is very close to the straight line (pink line) connecting the control points of Catmull-Rom Spline. So I use these straight line segments to determine the distance between a point and Catmull-Rom Spline.

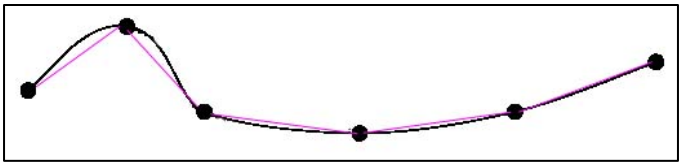


Fig 1: the Catmull-Rom Curve used for Soft Curve. The Pink lines are the lines which used to approximate the curve for while calculating the distance between a point to the curve.

The distance function between a 3D point and a 3D line segment used in my program is defined as shown in **Fig.2**:

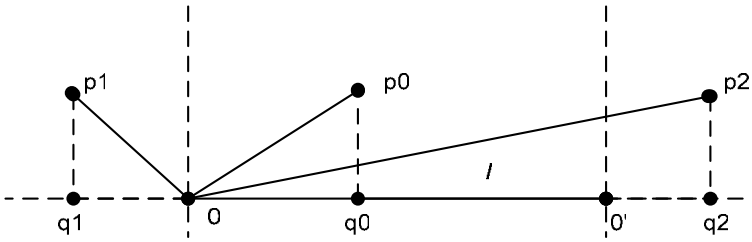


Fig 2: working out the distance between a 3D point and a 3D line segment.

In **Fig.2,** there are 3 3D points, *p0*, *p1*, and *p2*, locating in the different positions corresponding to the line segment, *l*. We work out the distance between point and line segment by finding the corresponding projected points of 3D points on the line segment, *q0*, *q1*, and *q2*. So that the distance can be derived by formula below:

$$dis = \begin{cases} |\, pq \,| \; if \; q > O \; And \; q < O' \\ |\, pO \,| \; if \; q < O \\ |\, pO' \,| \; if \; q > O' \end{cases}$$

The examples of above Soft Object models: Soft ball, Soft Curve, and Soft Cone are illustrated in the **Fig.3** in next page. A rough approximation of the heart pathology model made by generic soft objects is also shown in **Fig. 3**.

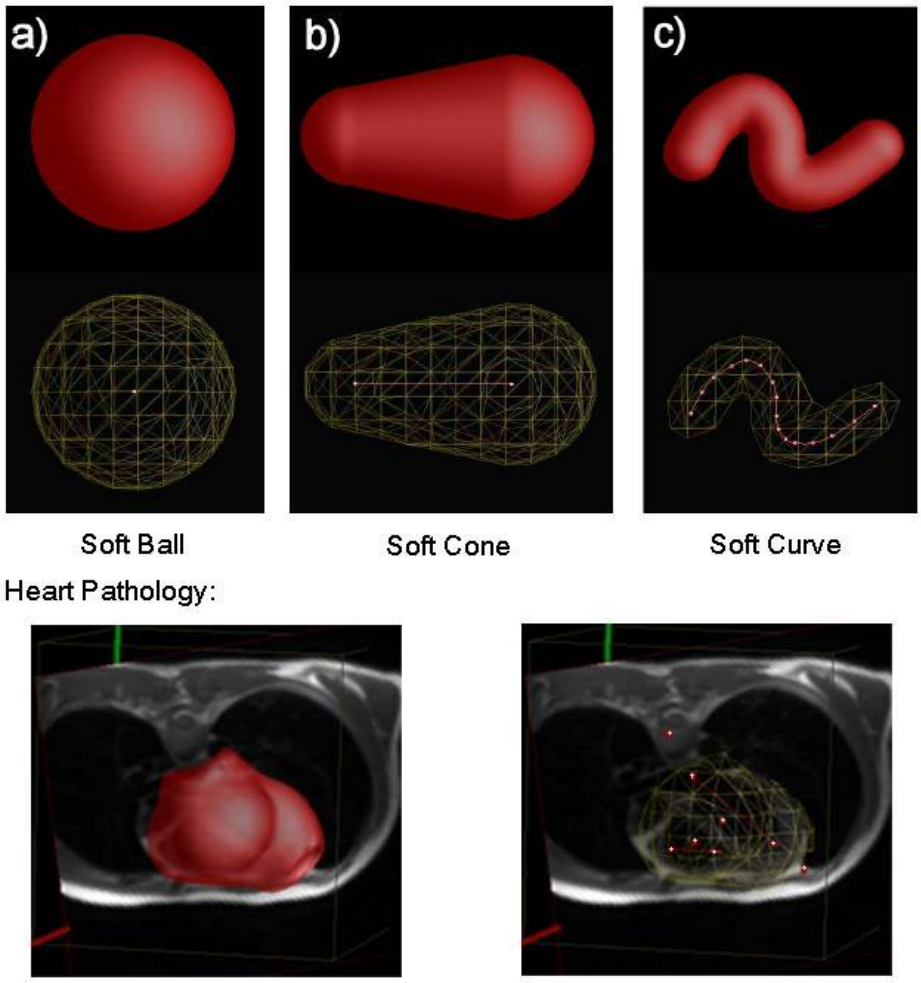


Figure 3: Three types of Soft Objects in Polygon and Wire-frame Mode, together with a rough approximation of heart pathology

As screen shots in **Fig.**3, we can clearly observe that there are basically two modes of Soft Objects existing in my program: Polygon Mode (View Mode) & Wire frame Mode (Modify Mode). View mode allows user have an entire view of the Soft Object Model made by them in polygon meshes, and Modify Mode enable users to drag or select the skeleton (in **Red** line) of the Soft Object.

For the model of heart pathology shown in the bottom, the contour of that model is created by combination of seven Soft Cones and four small Soft Balls. Since this model roughly goes through the heart area in the MRI data, it can be apply to further processes to derive more accurate heart pathology.

From the heart pathology model shown in **Fig. 3**, we can also seen that there is a concave shape on the bottom right, this concave shape is created by taking the difference between the entire density fields and the density field of a Soft Ball. Other than summing and differencing (Boolean operation) the density fields, there are also some other blending functions I have used in order to

model a complex heart shape (I will mentioned the one of these blending functions in the following).

Derive heart pathology from Soft Object Model

Until now, I assume users have already used Soft Object Modeling Technique [1] mentioned before and derive a rough model of the heart (as the last example shown in **Fig. 3**). In the rest of this part, I will introduce how the new approach in my project use this rough model to produce the pathology of the heart. Theoretically, this approach is very similar to the traditional ways mentioned in the Introduction section, except other than using a sample healthy heart model as the basic model. I used the rough model created by Soft Object Models [1]. The reason is that rough model is much more accurate than healthy heart in the case of abnormal congenital heart disease. The graphical processes of my approach and some results are illustrated in **Fig. 4** in the following:

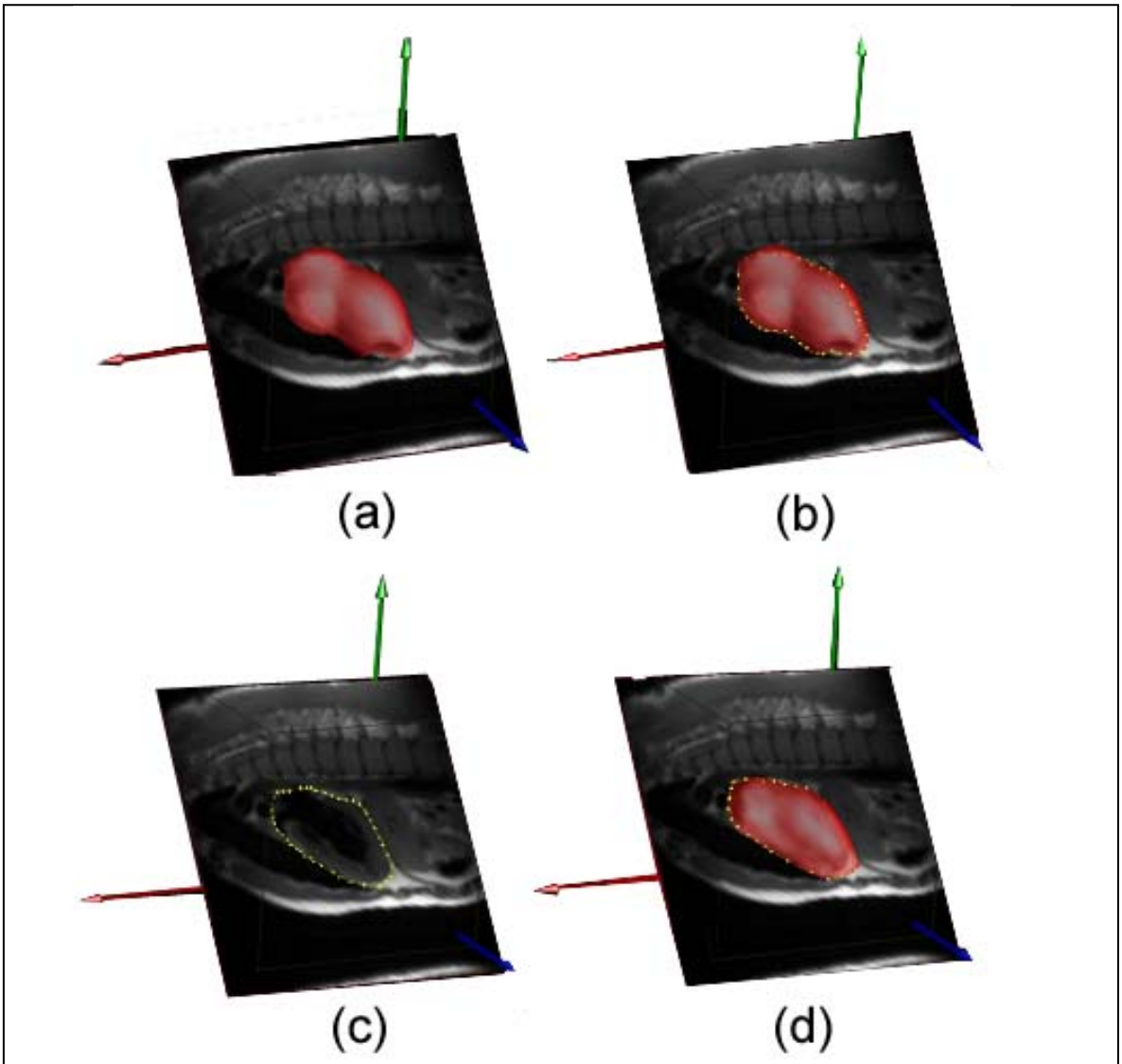


Figure 4: the processes of construct Heart Pathology with Soft Object Models

In my program, after get the rough model of the heart (shown in **(a)** in **Fig. 4**), user can start the processes of deriving accurate heart model by clicking the "*GetSnake*" button. This "*GetSnake*" button will:

- Firstly, start a simplified Marching Cube Technique [12], 2D Marching Square, to detect the intersection between the rough Soft Object Model and MRI data slice (the yellow points shown in **(b)** of **Fig. 4**). In my program, another step to sort the intersection points is also necessary in this process, so that it can use the sorted sample points to form a closed B-Spline curve.

- Secondly, apply the sorted intersection points to the B-Spline function to form a closed B-Spline curve. Since the closed B-Spline curves only approximate the edges on every MRI slice, my program then use the famous Active Contour Modeling (B-Spline Snake) technique from Kass in 1988 [2] to automatically adjust the closed B-Spline curves and make them accurately fit into the edges. The basic idea of the Active Contour Model [2] is that it provides an energy-minimizing function for every closed B-Spline curve according to the image slice relevant to the curve. This energy-minimizing function will iteratively adjust the B-spine curves until reach the situation of minimum energy, where every B-Spline curve accurately matches the edges on its relevant image. The corresponding result of this process is illustrated in **(c)** of the **Fig. 4.**

- Finally, after we got all the accurate B-Spline curves for all MRI slices. The Radial Basic Functions (RBFs) [3] is introduced to utilize the sample points on previous adjusted curves and integrate them to derive a smooth and energy-minimized 3D iso-surface. Since the basic characteristics of the RBFs, this iso-surface goes through all the deformed B-Spline curves, and holes from the noise of the source on this reconstructed 3D model are smoothly filled and surfaces are smoothly extrapolated (as shown in **(d)** of **Fig. 4).**

Encountered Problems

There are mainly two technical problems I have encountered when I was implementing above modeling method.

Technical problem 1:

The first problem comes from the characterization of the Soft Object Modeling technique. As I introduced before, one advantage of the Soft Object Modeling technique is that I can use some blending operations to combine two density field to achieve some complex Soft Object models, for example, using the Boolean operation: summing and differencing, to easily calculate the union and intersection of the density fields from every Soft Object in the environment, so that these Objects finally smoothly combine together to approximate the shape expected. These kinds of blending operation provide a much better result compared with other modeling technique, e.g. implicit function, in which it's very hard to derive the implicit function for the complex surface.

Although constructing complex Soft Object Models with blending operations is more efficient and effective than other approaches, blending operations are sometimes still not convenient enough to simulate some very complex objects, namely heart Pathology. The raise of this problem is that, when many previous Soft Object Models have already been used to join together to develop a complex model, it's very difficult to add another Soft Object Model to the previous density field and achieve the expected result. The more the Soft Objects used before, the harder to derive expected shape in the future.

Since the previous density field is created from applying a serious of summing and differencing blending operations, the density value in the final density field are not equally distributed. With this unequally distributed density field, it's very hard for user to estimate the distribution of the density values according to its surface and decide how to add another soft object to achieve the expected result. One simple example of this technical problem is that when user is using a differencing operator to try to create a hole on the previous model (As shown in **Fig. 5** in the following).

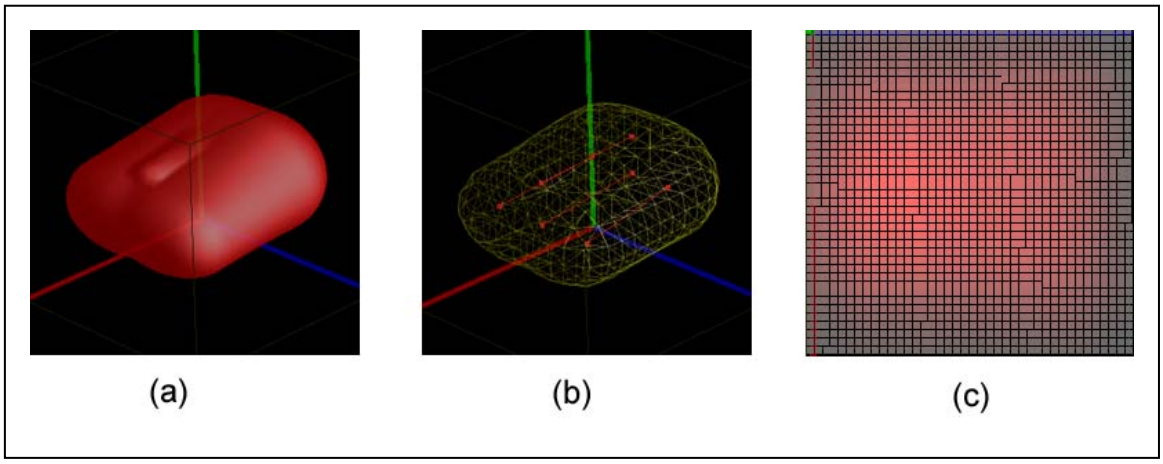


Figure 5: an Example of the problem from Soft Object Modeling

As shown in the **Fig. 5**, a very simple combination of Soft Object Models is presented (**(a)** in view mode and **(b)** in wire frame mode). According to the **(b)** in **Fig. 5**, we can discover that this model initially is made up of summing four Soft Cones with similar radius at each ends. Image **(c)** in the **Fig. 5** shows a slice of the density field created by these four Soft Cones along **Y** direction, the direction of the green axis in the image, and the density slice is taken from a horizontal plane which cut through the middle of the model. From the density slice in **(c)**, we can obviously find out that that density fields probably has two line-skeletons, and they are not quite parallel to each other. Right now, if user want to create a hole which passes through above model along the x-axis (the blue axis) corresponding to the Picture **(c)** in **Fig. 5,** simply differencing a new Soft Cone with the original density field will not be effective. The result from this differencing process can be seen in **Fig. 6** in the following.

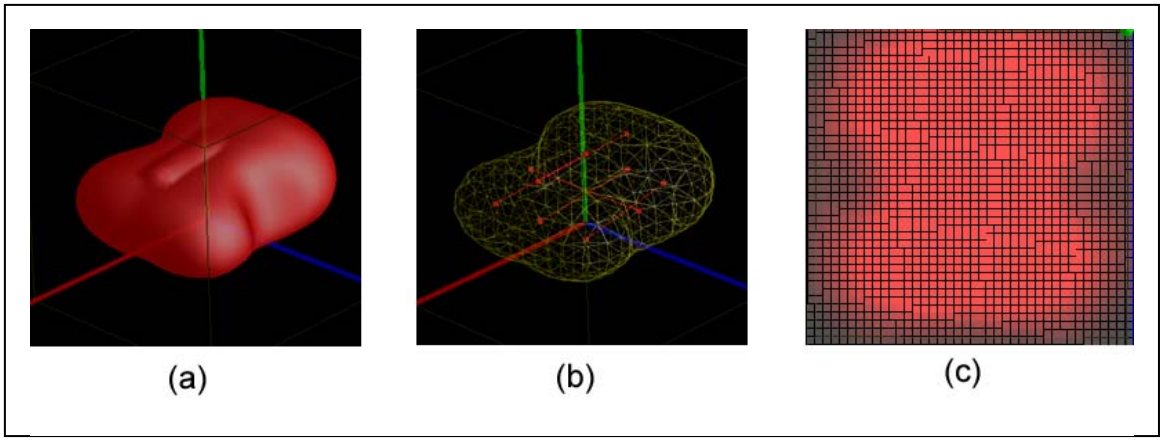


Figure 6: differencing a Soft Cone with previous density field.

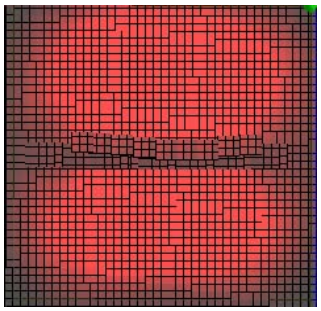


Figure 7: Expected Density Field.

As we can see in **Fig. 6**, creating a hole inside the model mentioned before gets very difficult, the final density field on the same slice shown in **(c)** just concaved a little bit in the area where we added a Soft Cone, but we were expecting to see some density field which looks like the one shown in **Fig. 7**. The reason for this un expected result that, since we have summed the density field before we added a new soft cone, the maximum field value in the old density field is much bigger than the maximum field value of the Soft Cone's density field, so when we take the difference between these two fields, the result after differencing are still too big to provide a hole shaped iso-value surface (as shown in **Fig. 7**).

Although it's very difficult to get ride of this problem mathematically, there are basically two ways we can consider to minimize the effect of the problem:

- Normalize the density field each time we add a new object. With this approach, we can control the maximum field value in the density field and probably we can get more expected result.

- Allow users to freely specify the range of each soft object they added. Users can specified of the range of the soft object by simply scaling the field range. With this approach, users can easily get the complex shape they expected, but similar problem may still happens when the number of Soft objects getting larger.

In my program, I mainly used the second approach to deal with this problem. A slider is provided in the user interface, and user can decide how they scale the density field by dragging the slider. I also provide a new blending operation with which user can hardly combine two groups of Soft Object Models.

In Wyvill's [1] formula (as shown in **Formula 1**), the range of the field values assigned to the density field is one of the factors which control the combination of Soft Object Models, as parameter *a* shown in **Formula 1**. Scaling the parameter *a* of each Object can directly affect the shape of the final model made out of them.

Formula 1: Wyvill's Field Function:

$$D(r) = a\,(1 - 4r^6/9R^6 - 17r^4/9R^4 - 22r^2/9R^2)$$

Note: *a* scales this function, each control primitive has no influence after a distance *R*.

As explained in **formula 1** above, parameter *a* control the range of the field made by Soft Object. For example, if *a* = 10, then the density field values are ranged between 0 and 10, the closer to the skeleton of the Soft Object, and closer to 10, otherwise, the closer to field value 0. We can also notice that, the bigger the value of *a*, the faster the field value distributed when getting away from the skeleton. With this property of the Soft Object model, when we are combining two Soft Object Models, we can control the curvature at the intersection by adjusting the value of parameter *a* of each model. Normally, the larger the differences between the values of *a* from two density fields, the harder these Soft Objects join together. If field-ranges of both Objects are the same, these two Objects are very smoothly joined together. This is clearly illustrated in **Fig. 8** in the following**.**

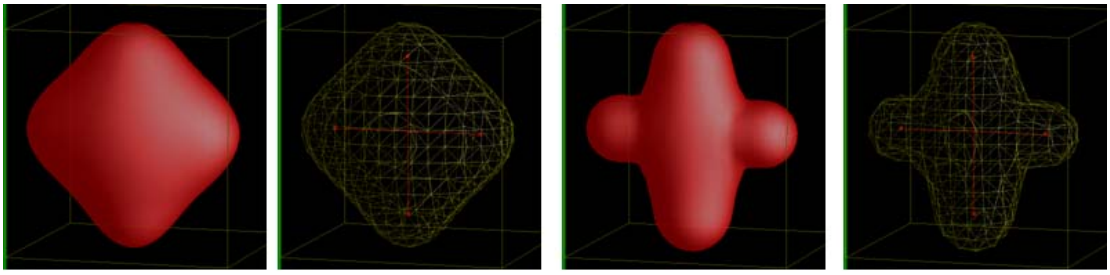


Figure 8: Adjusting the curvature at the intersection between 2 Objects

As shown in **Fig. 8**, we were combining two Soft Cones both of which has same radius at each end. In the left 2 pictures, the field-ranges and radiuses of both Soft Cones are the same. Since these two objects are too smoothly joined together, it very difficult to observe their skeletons according to the surface shown in the first left picture. For the two images on the right, I had adjusted one of the Soft Cone (the one which is along y axis) to make its field-range three times larger than before. Since the difference between two density fields is increased, it's much easier for people to determine its skeletons. Note: I have to decrease the radius of the soft cone which we scaled before. The reason is that,

after I scaled the density values to three times as large as before, the affect range is about three times larger, and we need to use radius property to minimize the range to the size we are interested in.

Although scaling the field values improves user's control over combining Soft Objects quite a lot, there are still some situations when it does not works (for example a lot of soft objects is joining together), and these situations happen while modeling heart pathology. When modeling the heart pathology, a reasonable amount of soft objects are needed, the density field out of these soft objects are large and various, it's vary difficult to get the shape we expect even we scale the field range. For example, it's very difficult to construct the left ventricle inside the existing heart model.

In order to build some holes which look like inter-ventricular and atrio-ventricular parts of the heart pathology inside the heart model, not only I need to consider how to solve the technical problem mentioned above, I also need to find a way to allow users to easily build this model without disturbed by the existing models. I approximate this goal by using the idea of Soft Object Groups. The definition of Soft Object Group in my report is that the Soft Object models will affect each other only if they are in the same group when users are building the models. A filter is used at last to combine groups of Soft Object to derive the final model.

In my program for modeling the heart pathology, I mainly define two groups of Soft Objects: one for approximating the outside boundary of the heart, including muscles and some aortas (red model in **(a)** of **Fig.9**), and the other for approximating the inter-ventricular and atrio-ventricular parts of the heart pathology (blue model in **(a)** of **Fig.9**).

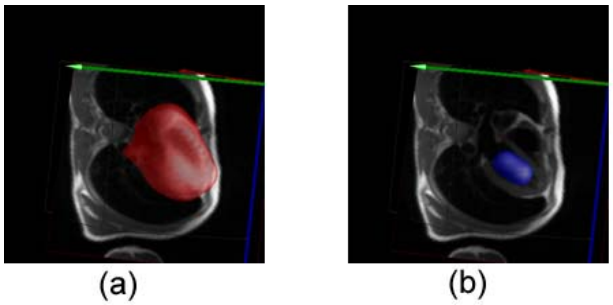


Figure 9: Two Kinds of Soft Object Groups

The filtering process for combining these two Soft Object Groups above is defined as: using the constant iso-surface of the blue model as the boundary, for every point inside the blue model, we replace field values at the corresponding positions in the red model by ($-f_{blue} * f_{blue}$). With this approach, for the density field of the red model, the positions which are not inside the boundary have positive field values, and the positions which are inside the boundary have very big negative field values. If we render this density field, the new red model will look like the original red model having a hole inside which is corresponding to the contour of blue model. Since the field values at the intersection area inside the final density field vary dramatically from great positive values to great negative values, an iso-surface could be constructed.

With the ideas of scaling field range and Soft Object Group, users can easily build the complex objects they want, and a approximated model of the heart pathology can be constructed easily.

Technical problem 2:

The second problem I meet when making this software comes from the process which reconstructs the 3D model described by the sample points from Active Contour Model [2]. Since the sample points are only a small part on the final surface, we need to interpolate these sample points to get a smooth and continuous surface. The Radial Basic Functions technique [3] is used to do this interpolation. A brief description about how Radial Basic Functions works is that, Radial Basic Functions uses the sample points (the points on the surface, and some points inside or outside surface) and sample values (0 for the points on the surface, positive values for the points inside the surface, and negative values for the points outside the surface) to derive the parameters for the final surface's implicit function. In our case, we can derive the sample points on the surface (zero-valued points) by using Active Contour Model [2] easily. We need to decide the sample points for inside and outside the final surface and their values.

Selecting the positions of the sample points for inside or outside the surface is not hard. I can either randomly select them from the environment or using some other approaches to regularly determine them. The hardest part is about how to decide the sample value accurately for the corresponding sample point. Although I can work out what the sign of the values (positive or negative) for the selected points by checking if they are inside or outside, I still need to determine what exact value I need to assign to them. These values have to be relative to each others, for example, the sample value of the point inside the final model need to greater than that of another point which is a little outside. If the sample values are not regularly assigned, a very different and messy result will be derived.

In my program, I used the final density field created in **Problem 1** as a library to approximate the sample values for the selected points. Although this density field has an area where the field values are not continuous, it's still good enough to help us decide the sample values. Another notice I have to point out is that when I was selecting the positions of the inside or outside points, I need to make it a little away from the boundary points. The reason is that the correct model is a little different from the model described in the density field. The one in density field exactly comes from the Soft Object Models, but the one we finally derived is adjusted according to the Soft Object Model by Active Contour Modeling Technique [2]. If we assign the sample value to a point which is close to the boundary point, that point might be inside the model in density field but out side the zero-valued boundary points, this will provide a very messy result. For

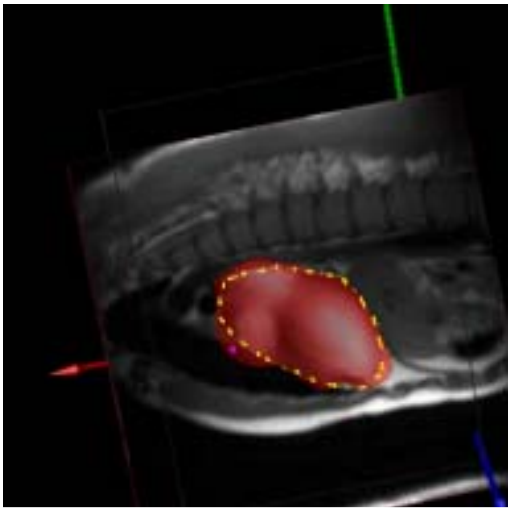


Figure 10: Sample Points from Active Contour Model, and the Soft Objects which derive it

example, as shown in **Fig. 10**, the yellow points are the sample points from Active Contour Modeling technique [2], they will be treated as zero-valued point, but these yellow points are a little inside the Soft Object Model (in red). If I try to work out the sample value for the pink point shown in **Fig. 10**, since the pink point is still inside the Soft Object Model, it will be treated as a inside point, but it is actually an outside point (outside the zero-valued points). So we have to select the points which are far from the boundary as inside or outside points. The process we used to select the positions for inside or outside sample points are:

- For each closed B-Spline on the data slices, find the centre point (P_{centre}) of the B-Spline by taking the average of every points on the B-Spline. Since the concave B-Spline is very rare in our case, this centre points are always in the middle of the B-Spline.

- For the **red** Soft Object Group, since zero-valued points are the points on the B-Spline, for each zero-valued point (P_{sample}). We work out the direction from the zero-valued point to the centre point (dir). Then the points inside and outside are:

$$P_{inside} = P_{sample} - a * dir$$

$$P_{outside} = P_{sample} + b * dir$$

  Where variable ***a*** and ***b*** are two constants to try to move the points P_{inside} and $P_{outside}$ away from the zero-valued points, the problem mentioned in last paragraph is dramatically avoid.

- For the **Blue** Soft Object Group, we only used the centre points as the inside sample points, and there is no outside sample points for them.

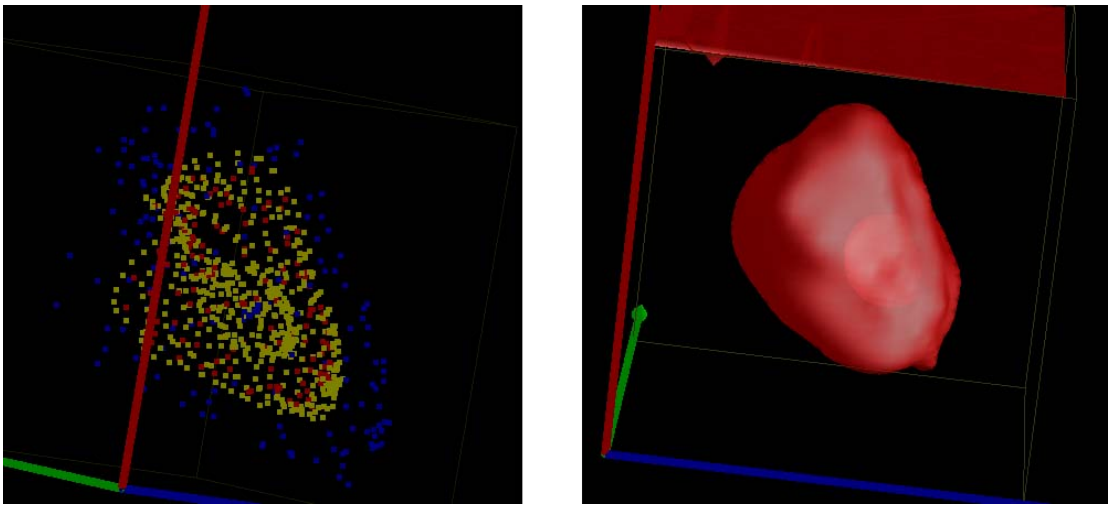


Figure 11: Sample Points for reconstructing the heart

The left picture in **Fig. 11** gives an example for the inside, outside and zero-valued sample points from our heart model: the yellow points represent

zero-valued points (boundary points), blue points represent outside sample points, red points represent inside sample points. We can also see from the left image of **Fig. 11** that there is another closed area of yellow points inside the outside boundary of the heart model. This area is made by negative Soft Object Group introduced before.

After pushing all the inside and outside sample points from both Soft Object group and their zero-valued sample points to the Radial Basic function [3], we can easily work out an iso-surface which passes though all the zero-valued points. This iso-surface for above sample points is shown in the right picture of **Fig. 11.**

Other problems:

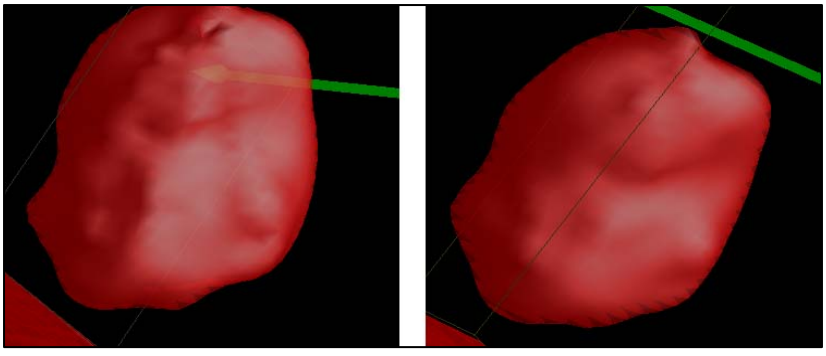
There are some other problems I have solved when developing the software. For example, since the MRI data has some noise in it, the sample points derived from it sometimes are not very related to each other, this unrelated sample points will produce a very unsmooth 3D surface when I reconstruct the model with Radial Basic Functions [3]. This is also because the Radial basic functions [3] are too strict to avoid the incorrect information from the noise.

I have overcome this problem by adding a constant into the equation of Radial basic Functions, which control the smoothness of the model developed by it. The new equation for Radial Basic Functions we used is given in **Formula 2:**

$$\begin{pmatrix} A-0.0015707K & P \\ P^T & 0 \end{pmatrix}\begin{pmatrix} \lambda \\ c \end{pmatrix}=\begin{pmatrix} f \\ 0 \end{pmatrix} \qquad \textbf{[Formula 2]}$$

Where K is an n × n matrix with all values 1, n is the number of the sample points.

Fig. 12 gives an example of comparing the result from the formal Radial Basic Functions and our Radial Basic Functions in **Formula 2.**



formal RBFs **Approximating RBFs**

Figure 12: Comparing the result from two kinds of RBFs

3 User Interface and the Toolkits of my program:

Since the users of my program are the medical experts with less computer background, they are not necessary to understand how the techniques which I used to model the heart pathology works, a very simple user interface which let medical experts easily understand and operate my program are necessary. Some small toolkits, which allow users efficiently control this program, are also created. Since the entire software is made in C++ and OpenGL only, my program also has the advantage of platform independent. In the following paragraphs, I am going to introduce the user interface and some small toolkits of my software with the screen shots. I am also going to mention some feedback from medical expert after he tested my program, and the relevant changes I have made in my program. There are basically three parts I am going to cover, they are:

- The user interface our software
- The small toolkits for MRI data management
- The small toolkits for Modeling

User Interface

Since I was required to develop the entire user interface without using any user interface developing tools, for example MFC, my program gain the advantages of portability, efficiency, and extensibility. All the components of the user interface are developed from scratch. They are very compatible with each others. I can also extend these components for further use. The entire user interface can be seen from **Fig. 13**:

As shown in the **Fig. 13** on the right, a 3D cube is mainly used to represent the user's working environment of my program. Since I used Marching Cube algorithm to reconstruct the Soft Objects, this 3D cube also used to represent the space of the Marching Cube algorithm available in my program.

A multi-level popup menu from OpenGL's GLUT library is used to allow users to efficiently control all the functionalities of my software. This Pop-up menu can be easily activated by clicking the **Middle button** of the mouse. In my program, I define that users use left button of the mouse to rotate the cube, and use the right button of the mouse to drag and add the Soft Object's skeleton.

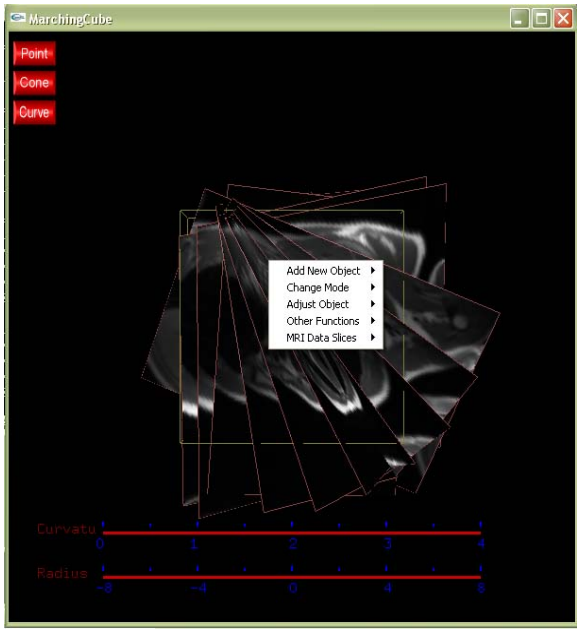


Figure 13: the entire User Interface

Two simple graphical sliders are used to allow users easily control the radius and the scaling parameter *a* of the selected soft object (I will give more detail of this in the following paragraph).

There are also three constant buttons: *Point*, *Cone*, and *Curve,* shown on the left side of my program, the functionalities of these buttons are allowing users easily create and modify Soft Ball, Soft Cone, and Soft Curve in my program.

- *Point*: when user click this button, *Point* button will be **down** and my program will automatically switch into modify mode (the wire frame mode). User can keep on adding new Soft balls into the environment by clicking the right button of the mouse. When users finish adding Soft balls, they can stop this functionality by clicking the *Point* button again (button will return back to **up**, and program switch back to polygon mode).

- *Cone*: The Control of the *Cone* button is almost the same as the *Point* button introduced above. Except that every two times users clicking the right button of the mouse, a new Soft Cone will be added to the environment.

- *Curve*: The Control of the *Curve* button is almost the same as the *Point* and *Cone* buttons introduced above. However, users can only add one Soft Curve into the environment every time they enable and disable the *Curve* button. The reason for this control is that, there are unknown numbers of the control points on the Soft curve object. Each time user click the right button of the mouse, my program will consider users add a new control points to the new Soft curve object. A new Soft Object can only be added to the program if user disable the *Curve* button and enable it again. Also, a new control point will only be added to either ends of the Soft Curve. My program will find out the distances between new point and previous ends of the soft curve. The new control point will be added to the end with minimum distance. As shown in **Fig. 14**, original curve is in **(a)**, a new control point will be added to the left end of the curve in **(b)** and to the right end in **(c).**

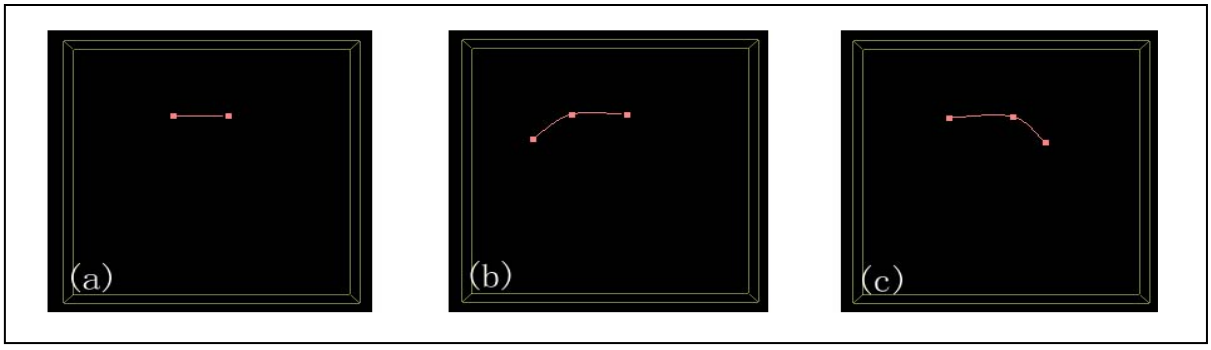


Figure 14: Adding a new point to left and right sides of Soft Curves.

In the feedback from Dr Alistair Young (medical expert who test my program), he suggested that since users will create the model according to the displaying MRI slices, it's better to add 3D new point on to the existing slice inside the cube. In my program, it allows user to decide whether to display one slice of MRI at a time and user can select which one to display or display all MRI data slices at the same time. If one slice is displayed each time, the new 3D control points of the soft objects will be added onto this slice. Otherwise, the new control point will initially be added to the centre of the volume.

Toolkits for MRI data Management

Clipping tools

Since my program allows users make 3D models in 2D environment (computer screen), how can I create tools which allow users easily and efficiently analysis and manage 3D MRI data slices is the problem I met when designing the interface. I have noticed that MRI slices are so annoying that when I display them in 3D, the front slice always cover the slices behind it. In order to deal with this problem, several kinds of clipping plane are adopted in my program. Users can easily use these clipping plane tools to clip the MRI data slices and get the information they want. There are two types of clipping plane provided to the users in my program.

The first clipping plane is called **static clipping plane**. According to the name of this clipping plane, this clipping plane will always facing a constant direction – user's position. After user active this clipping plan by "Other Functions->Enable/Disable static Clipping plane" option of the pop-up menu, no matter how the user rotate the working cube of our software, this clipping plane always face the user's view and it cuts the objects inside the cube which are in front of it (as shown in the first image of **Fig. 14,** the static clipping plane is described by green dashed lines). Users can move forward and backward this static clipping plane by using the up-arrow and down-arrow buttons on the keyboard.

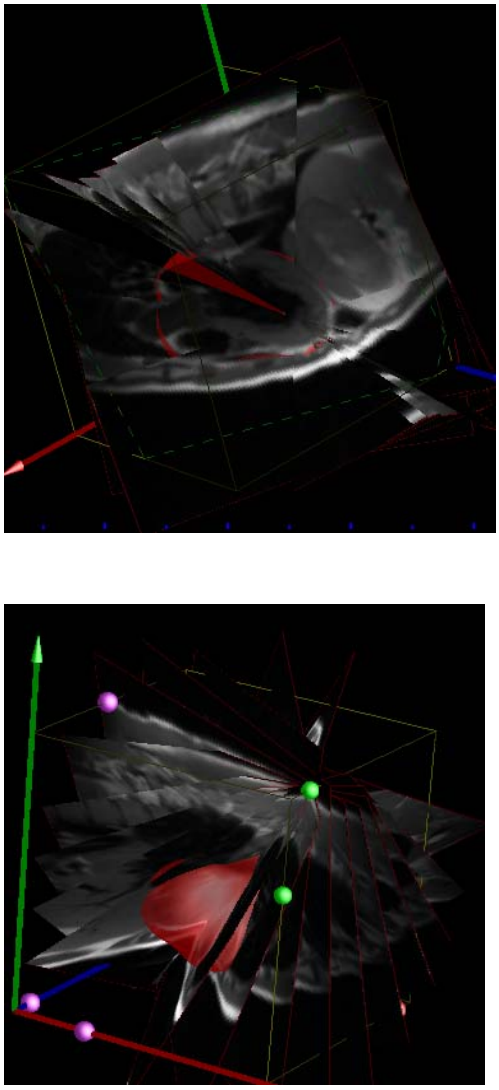


Figure 15: Static Clipping Plane & Normal Clipping Plane

Since the static clipping plane always facing to the user, it's quite annoying if user want to create a model when the clipping plane is facing to another direction. Users sometimes even feel more comfortable if they have the ability to define the direction and position of the clipping plane. Our software provide this ability to users through second clipping plane, called **Normal Clipping Plane,** which can be activated by "Other Functions->Enable/Disable Normal Clipping plane" option of the pop-up menu. This clipping plane is defined by three small balls, which can be moved only along the edges of the cube, the clipping plane is formed by this three balls, some green balls will be added when clipping plane intersect more than three edges. Users can change the position of the clipping plane by dragging these balls (except green balls) with the right button of the mouse.

From Dr Alistair, he has suggested that I should display the MRI slices in the preference way of medical expert, which are that:

- Human's body is normally horizontal to the view position.
- Human's feet are close to the view camera compared with the head.
- Human's face and Chest are facing upward.

According to the above suggestion, I have modified my program and add some more functionality into my program to make it convenient to be used:

- **MRI Data Slices** menu: there are five sets of heart MRI data slices from the source. Although we have clipping tools to clip these slices, if we display all of them at the same time, the working cube will look very messy. User can use **MRI Data Slices** menu in the pop-up menu to select which set of MRI data they want to display. Also, each slice of the MRI data is displayed following the rules mentioned above.

  Note: the Active Contour Model Technique will be implemented according to the current displaying set of MRI data.

- **Show/Hide MRI slices** submenu: users can use **Show/Hide MRI slices** submenu under **Other Functions** menu to display or hide the MRI data slices.

- **'S'** button: users can use the '**S**' button on the keyboard to ask my program to display one data slice at a time of the current selected set. Users can loop forward or backward to other slices of the same data set by using **'4'** and **'6'** button on the keyboard.

- **Space** button: When users are displaying one data slice at a time, sometimes they might prefer a situation where the data slice is paralleling to the screen and facing the users. They can achieve this by clicking the **space** button, and the current slice will be automatically adjusted to face the users.

Toolkits for Managing Soft Object Modeling

Some others options provided in our Software to manage Soft Object modeling process are listed in the following:

- **InputParser** class: this class is added in my program to increase the flexibility. With **InputParser** and its corresponding saving tool, users can save the existing soft objects into a file called ***input.data***. And ***input.data*** can also be loaded by my program in the future through the **InputParser**. Users can also manually create some soft objects through specifying the properties of Soft Objects in ***input.data*** file. The Format of the **InputParser** is descried in **Fig. 16.**

```
<Number of Soft Object Group>
// Group one
<Number of Soft Objects> <Soft Object Group type>
//Soft Object 1 in Group One
<Object's type> <Object's Mode: positive, negative> <Object's curvature>
<Positions>
<Radiuses>
//Soft Object 2 in Group One
…
//Group two
…
```

Figure 16: the format of *input.data* file

- **Add New Object** menu: under this menu of our pop-up menu, user can:
  - Switch between two kinds of Soft Object Groups by selecting **Switch Object Group** submenu
  - Determine the mode of the current group. Whether current is a negative soft object group (blue object), or a positive soft object group (red objects).

- **Change Mode** menu: users can decide the displaying mode of the current selected Soft Object through this menu.
  - The **Modify** mode displays the Soft Objects in wire-frame mode, so that users can drag and select the skeleton of Soft Object in order to modify them.
  - The **View** mode displays Soft Objects in polygon meshes. Users can not modify the Soft Objects in this mode, but **View** mode provides a nice 3D surface of the models, so that users can have an entire look at the models they created.

- **Adjust Object** menu: there are four functionalities provided under **Adjust Object** menu:
  - **Delete Object** submenu: after user selected the existing object which they want to delete, they can easily remove this object by clicking this submenu.
  - **Opposite Object** submenu: users can switch the selected Soft Object between positive and negative with this option. Positive Soft Object provide a positive density field and negative Soft object provide a negative density field.
  - **Run B-Spline Snake** submenu: after user has created a rough model of the heart Pathology, they use this menu to start the Active Contour Modeling (B-Spline Snake) technique [2].
  - **Re-Construct the Shape** submenu: Users use Radial Basic Functions [3] technique to reconstruct the heart pathology with sample points by selecting this menu.

- **Other Functions** menu: except the slices management functions mentioned before, there are three other functions provided under this menu.
  - **Show/Hide Model** submenu: with this option, users can hide the soft Object Models inside the cube. If the model shown in the cube is the final model from Radial Basic Functions, users can use this option to switch the final model between representation of polygon meshes and the representation of sample points (yellow for boundary points, red for inside points, and blue for outside sample points).
  - **Save Object** submenu: As the file mentioned in the **InputParser** before, this option allow users to save the Soft Objects made by them in a format which can be read by the parser. So that users are allow to save the models created by them for future use. The name of the file is ***input4.data.***
  - **Select Working Cube** submenu: sometimes the model users try to construct may just take a small part of the cube. It not necessary to calculate the density field for entire cube. This submenu allow user to limit their working space by inputting the start position and the size of the working space in console windows (As shown in **Fig. 17,** the small cube is the space where Marching Cube algorithm will be apply to). The advantages of this are: not only we can save the running time to calculate the field value of some useless position, we can also increase the resolution of the Marching Cube Algorithm, so that we can get a much better result.

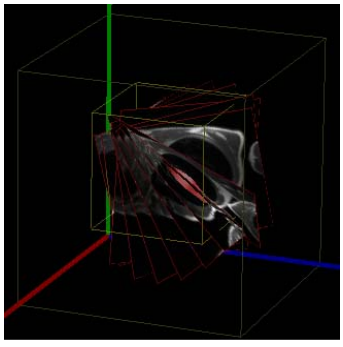


Figure 17: defining a working cube

- **Slider:** two slider Objects are added in the bottom of our software. One is used for adjusting the radius of the selected Soft Object, the other is used for adjusting the curvature of the selected Object (scale the field values of the selected soft object by adjusting the parameter *a* in **Formula 1**).

4 Component Management and hierarchy:

As mentioned in previous sections, there are several techniques I have used to achieve the purpose of modeling heart pathology. In order to simplify the process which users used to do the modeling, I also added a few toolkits into my program. All these things together with resources from the heart MRI data make the components arrangement and hierarchy of my program very complicated. I have to design a well-organized structure to make these components relatively works together. The content of my structure is explained in the following paragraphs.

Firstly I create a class called **Model,** which is in the top level of the hierarchy. Each instance of **Model** class represents a group of Soft Objects. Since there are two Soft Object Groups in our software, there are two **Model** instances created when our program starts. And user can change the properties of the group through user interface (as introduced before), for example, users can determine the mode of the selected soft object group. **Model** instance also contain a **SnakeCollection** instance, and use this **SnakeCollection** instance to manage all sets of MRI data from the resource and do the adjusting process.

As we mentioned, the **SnakeCollection** class contain all the information from the heart MRI dataset. The functionalities of the **SnakeCollection** class are:

- Store the image information for all sets of MRI data slices.
- Store the position of every MRI data slice.
- Store the index of the current displaying MRI slice.

After the rough model of the Soft objects combination has been created, **SnakeCollection** class can be used to detect the intersection between MRI data slices and Soft Object Models and to start Active Contour Modeling Technique [2] to do the edge capturing process. The **SnakeCollection** class achieves above purpose with the help of another class called **BsplineSnake.** A **BsplineSnake** instance defined in my program is used to represent a MRI slice. This **BsplineSnake** instance will implement Active Contour Modeling Technique [3] to minimize the energy on the B-Spline curves on the corresponding data slice. The entire hierarchy of our program can be illustrated in **Fig. 18.**

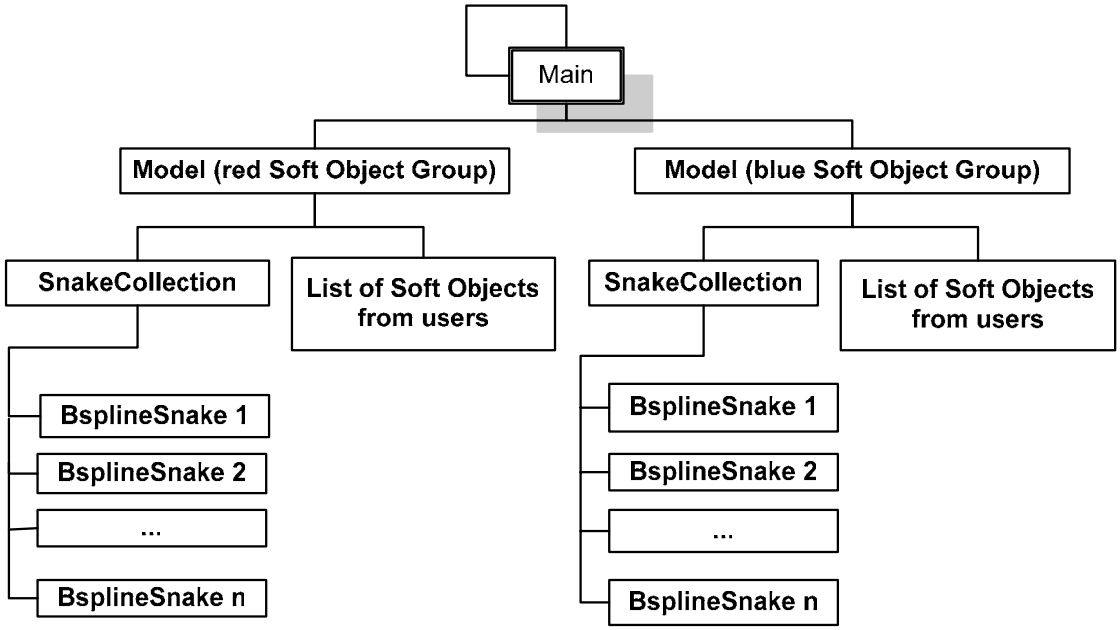


Figure 18: The Structure of our software

5 Problems:

Users probably sometimes will find is quite difficult to place the selected soft object into correct position when they are using my program to model the inter-ventricular and atrio-ventricular part of the heart pathology. The reason is that the heart MRI slices provided in our Program are too noisy to allow B-Spline Snake to detect the contour of the ventricular part of the heart. It's even difficult for users to detect this part by eyes. After users made a rough model for the ventricular part with negative Soft Object Group (blue), although the B-Spline Snake technique will provide some energy-minimized B-Spline for each slice, these B-Spline might not be continuous, and this will lead to a very poor result of the ventricular model. An example for the ventricular part of the heart and an adjusted B-Spline are shown on **Fig. 19.**

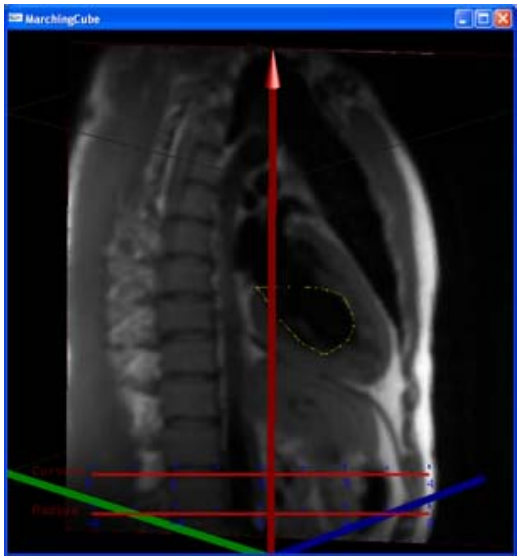


Figure 19: A MRI slice and an adjusted B-Spline curve

6 Conclusion and future improvement:

My program has been tested by creating the outside shape and an inside hole which similar to the right ventricle of the heart. The computation times for adjusting the closed B-Spline in red Soft Object Group is 29.5 seconds, the computation times for adjusting the closed B-Spline in blue Soft Object Group is 25.4 seconds, and the computation times for reconstruct the final model with Radial Basic functions is 31.2 seconds. This process is based on **long edge** data set of the heart MRI data. The hardware environment for this experiment is 2.8 GHz PC processor with 1G memory.

As introduced in this paper, Soft objects are ideally suited to approximate smooth organ shapes such as the heart. For example soft cones and soft curve are very suitable to model blood vessels such as the ventricles and the arteries and veins. Density field scaling can be used to model the important topological features of the heart. However, one problem I haven't solved yet satisfactorily is how to combine soft object models for the inter-ventricular and atrio-ventricular part of the heart. Also it is not clear how to best handle very thin anatomical structures such as the walls of the blood vessels. Modeling them with a single soft object is easy but is inconsistent with the rest of the model. A possible solution is to display the entire MRI data set using direct volume rendering and to adjust the opacity transfer function such that only selected features of interest are displayed during modelling. I could also consider using free form deformation technique to allow user to deform the soft objects to achieve particular topology of the heart instead of density field scaling.

Reference:

[1] G. Wyvill, C. McPhetters, B. Wyvill, *Data Structure for Soft Objects*, The Visual Computer, Vol. 2, pp. 227-234, 1986.

[2] M. Kass, A. Witkin, D. Terzopoulos, *Snake: Active Contour Models*, International Journal of Computer Vision 1 (4) (1988) 321 – 331.

[3] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, *Reconstruction and Representation of 3D Objects with Redial Basis Functions***.** Applied Research Associates NZ Ltd, University of Canterbury

[4] *Introduction to Catmull-Rom Spline Curve*, http://www.mvps.org/directx/articles/catmull/

[5] Burkhard Wunsche and Ewan Tempero, *A Comparison and Evaluation of Interpolation Methods for Visualizing Discrete 2D Survey Data.* Department of Computer Science, University of Auckland

[6] William H.Press, Saul A. Teukolsky, William T. Vetterling, Brian P.Flannery, LU *Decomposition and Its Application*, Chapter 2.3, Numerical Recipes in C.

[7] Liyan Zhang, *Active Contour Model, Snake*, Department of Computer Science, University of Nevada, Reno

[8] Bo Li and Burkhard Wunsche, *Modeling heart pathology with Soft Objects*, COMPSCI 380 Project Report, Department of Computer Science, University of Auckland, New Zealand, November 2003.

[9] Bo Li and Burkhard Wunsche, *A fast semi-automatic method for reconstructing heart pathology with implicit surfaces*, FoS Summer Scholarship Project Report. Department of Computer Science, University of Auckland, New Zealand, January 2004.

[10] A. A. Young, *Model Tags: Direct 3D tracking of heart wall motion from tagged magnetic resonance images*, Medical Image Analysis, 3 (4), (1999), 361-372.

[11] D. Wei, *Whole-heart modeling: progress, principles and applications*, Progress in Biophysics and Molecular Biology, 67 (1), (1997) 17-66.

[12] Lorensen, W. E. and Cline, H. E., *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*, Computer Graphics, Volume 21, Number 4, July 1987, pp. 163-169.