5.3 参赛论文二

A Quick Algorithm for MRI Problem

Xu Dongming Gu Nan Liu Guangyi
Advisor Zhou Yicang

Summary

In this paper, an algorithm is developed to picture slice through the selected plane in any orientation in space. Testing on a personal computer, our algorithm gives satisfying results.

Our algorithm consists of three main aspects: determining the shape of the cross-section through the selected plane, determining the positions of selected pixels on the cross-section, and calculating the gray-scale value of selected pixels.

We adopt three methods to calculate the gray-scale value of pixels: Nearest Method, Average Method and Linear Interpolation Method. There is some difference among the three methods. Nearest Method can preserve sharp boundary information well while the smoothness of the recovered picture is not very good; Linear Interpolation Method tends to blur sharp boundaries slightly while the smoothness of the recovered picture is high. As to the Average Method, its performance is between the two methods mentioned above.

Our algorithm has been implemented on a 486DX2/66 personal computer and tested by designed data sets. A distinguish character of our algorithm is that its processing time is short even on a 486 per-

sonal computer. The algorithm also accommodates various resolutions of original data sets and recovered pictures. We evaluated our algorithm by three indexes: processing time, the smoothness of recovered picture and the ability of preserving sharp boundary information. With these indexes, we come to a conclusion that Linear Interpolation is a better method for common use. However, when sharp boudary information is concerned, we suggest the Nearest Method and Average Method.

Restatement of The Problem

Industrial and medical diagnostic machines known as Magnetic Resonance Imagers(MRI) scan a three-dimensional object and deliver their results in the form of a three-dimensional array of pixel. Each pixel consists of one number indicating a color or a shade of gray that encodes a measure of water concentration in a small region of the scanned object at the location of the pixel. Typically, the value of the number range from 0 to 255.

Algorithms for picturing slices through oblique planes are proprietary and current algorithms have many constraints and are implemented only on heavily used dedicated workstations. So the problem is to design and test an algorithm that produces sections of three-dimensional arrays by planes in any orientation in space, preserving the original gray-scale values as closely as possible. To design such an algorithm, one can access the values and locations of the pixels, but not the initial data gathered by the scamper.

Assumptions

- The size of the three-dimensional array $A(i,j,k)$ is $L \times M \times N$. (i ranges from 0 to $L-1$, j ranges from 0 to $M-1$ and k ranges from 0 to $N-1$)

- The three-dimensional array of pixels form a cube in space.
- The distance between adjacent pixels is equal in the direction of each Cartesian coordinate axis.
- Each pixel consists of one number indicating a color or a shade of gray that encodes a measure of water concentration in a small region.
- The orientation and position of the oblique plane are determined by user.
- The data set delivered by MRI machines has enough information of scanned object and the original gray-scale values are assumed to be exact.

Definitions of Constants and Terms

P(x, y, z): an arbitrary pixel on the cross-section

$G(P)$: the gray-scale value of pixel P

Selected Pixels: Pixels selected from the cross-section to reconstruct its picture

Resolution: the density of original data set sampled from scanned object

Analysis

The original three-dimensional array of pixels form a cube in space, and the oblique plane determined by user may intersect with it and produce a cross-section (see Fig. 1).

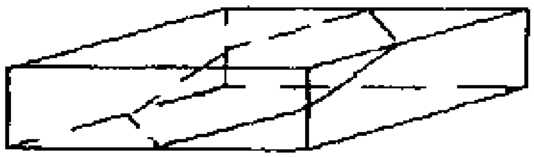

Fig. 1

Our task is to picture the cross-section. If we can select enough pixels from the cross-section and calculate their gray-scale values, we are able to reconstruct the picture of the cross-section clearly and display it on the screen. In this way, we developed our algorithm that can be separated into following aspects:

- **Determine the shape of the cross-section**
- **Determine the positions of pixels selected from the cross-section**
- **Calculate the gray-scale values of selected pixels**

In this paper, we are going to discuss each respect respectively and give our entire algorithm.

Design of Algorithm

In order to describe our algorithm clearly, we have to set up the three Cartesian coordinate system (as it is shown in Fig 2).

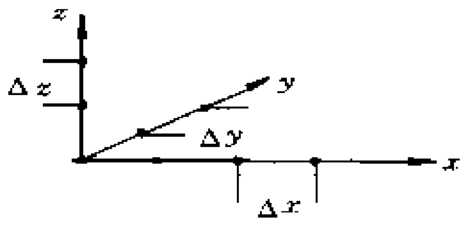

Fig. 2

According to assumption 3, we are able to set up the relationship between (x, y, z) and (i, j, k):

$$(x, y, z) = (i \times \Delta x,\ j \times \Delta y,\ k \times \Delta z)$$

Where Δx, Δy, Δz respectively indicate the distance between adjacent pixels in the direction of coordinate axis x, y, z.

Determine the shape of cross-section

As the cross-section is a polygon, so in order to find its shape we only have to know the vertices of the cross-section that are the intersection points of the plane and the cube.

Let $l(=(L-1)\times\Delta x)$ to be the length of the cube, $w(=(M-1)\times\Delta y)$ to be the width of the cube, and $h(=(N-1)\times\Delta z)$ to be the height of the cube.

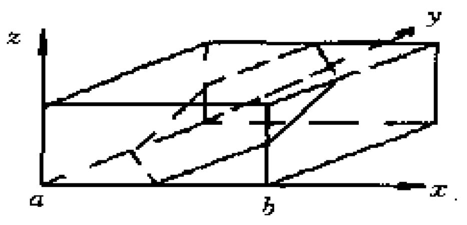

Fig. 3

Thus each edge of the cube can be expressed by parametric equations. For instance, edge ab (see Fig. 3) is expressed by following equations:

$$\begin{cases} x = t, \ 0 \leqslant t \leqslant l \\ y = 0 \\ z = 0 \end{cases} \qquad (1)$$

If (n_x, n_y, n_z) is the normal vector of the plane determined by user and (x_0, y_0, z_0) is a point in the plane, the equation of the plane is:

$$(x - x_0)\times n_x + (y - y_0)\times n_y + (z - z_0)\times n_z = 0 \quad (2)$$

From equations (1) and (2), we can eliminate variable t:

$$t = \frac{(n_x \times x_0 + n_y + y_0 + n_z \times z_0)}{n_x}$$

If $t > 1$ or $t < 0$, side ab has no intersection point with the plane; otherwise the position of the interseciton point is:

$$(x, y, z) = \left(\frac{(n_x \times x_0 + n_y \times y_0 + n_z \times z_0)}{n_x}, 0, 0\right)$$

After we have determined all the intersection points, we can joint the points in turn and then the shape of the cross-section is determined.

Determine the positions of pixels selected from the cross-section

To convert an analogous picture to a digital picture, the picture has to be divided into two-dimensional array of pixels. Usually, the pixels form square mesh array or triangle mesh array or regular hexagon mesh array (Fig. 4). Because square grid array is standard and easy to be implemented, we choose this method in our algorithm.

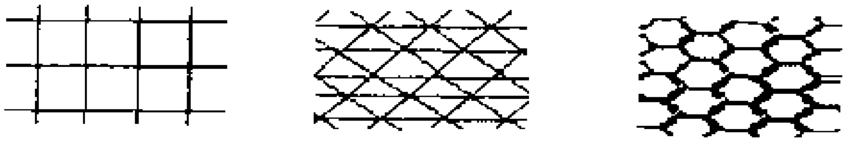

Fig. 4

Before we introduce our algorithm, we have to determine the orientations of mesh lines. After we have determined the shape of the cross-section, we choose a side ab (see Fig. 5), and let horizontal mesh lines are parallel to side ab and vertical mesh lines are perpendicular to side ab.

Now, we give our algorithm in following steps:

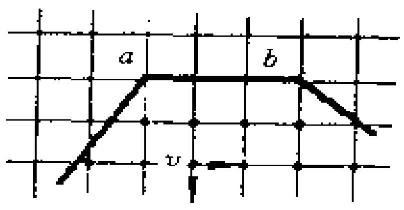

Fig. 5

1. Let vertex v to be a mesh point, start from vertex v, step along the horizontal mesh line for a distance Δh to get a new mesh point until the new mesh point is out of the cross-section.

2. Start from v, step along the vertical mesh line for a distance Δv to get a new mesh point and let v indicate this mesh point. Start from this point, do 1. on both sides. If v is out of the cross-section, repeat 1. until the new mesh point is in the cross-section. But

if the mesh point is still not in the cross-section after having stepped Max (see formula (3)) times, turn to 4.

3. Repeat 2.

4. End.

In this algorithm, Δh and Δv respectively indicate the horizontal and vertical mesh spacing. And Max is determined by following formula:

$$\text{Max} = \left[\sqrt{l^2 + w^2 + h^2}/\Delta h + 0.5\right] \qquad (3)$$

Where $\sqrt{l^2 + w^2 + h^2}$ is the length of the longest line segment in the cube.

Calculate the gray-scale value of selected pixels

After we have divided the cross-section into mesh array, we still have to calculate the gray-scale value of selected pixels. In this section, three methods to do this work are discussed: Nearest Method, Average Method and Linear Interpolation Method.

Nearest Method

The distinguish character of this method is its simplicity.

The idea of Nearest Method is to let the gray-scale value of the pixel equal to the gray scale value of the nearest vertex (as it is shown in Fig.6). If $G(P)$ indicates the gray scale value of pixel P and A_i is nearest vertex to P, $G(P)$ is determined by following formula:

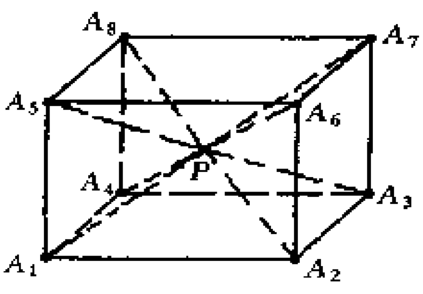

Fig.6

$$G(P) = G(A_i) \qquad (4)$$

But the Nearest Method has a weakness that if distances between P and A_i are approximately equal, the error of computation will exert great influence on the result. However, the Nearest Method can not avoid it. So we modify this method and get the Average Method.

Average Method

Based on this concept, we give an improved average method. We noticed that the gray scale value of a pixel indicates a measure of water concentration in a small region of the scanned object at the location of the pixel, so we assumed the small region to be a spheroid whose center is the pixel (see Fig. 7).

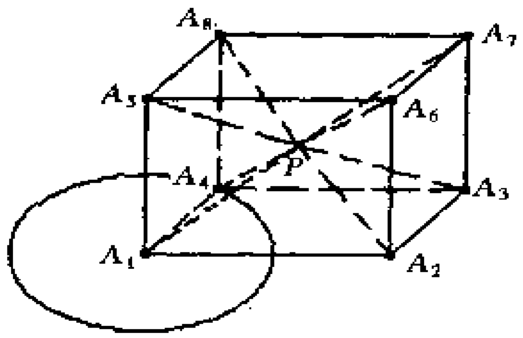

Fig. 7

We think this assumption is reasonable, because according to the mechanism of MRI machines, the gray-scale value actually indicates the intensity of signal in a small region. So if P is in a spheroid whose center is vertex A_i, we let $G(P)=G(A_i)$ and if P is not in any spheroid, we let

$$G(P)=\frac{\sum_{i=1}^{i=8} G(A_i)}{8}$$

Linear Interpolation Method

As one can see, the algorithm mentioned above is just a rough expression of the relationship among pixels that are located closely. So sometimes the smooth surface of planes may be coarsely presented. To solve this problem, we consider using the interpolation models.

The most simple one is linear interpolation model. Firstly, we take the example of plane pixels. If the gray-scale value of a pixel is not one of the original data, we must determine its gray-scale value according to the four pixels that are most close to it. The computing method is shown in Fig. 8.

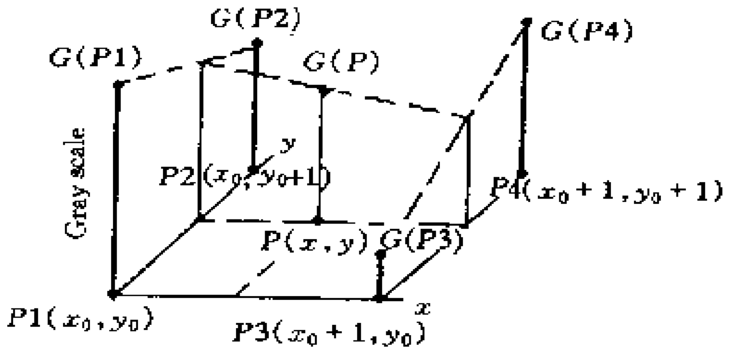

Fig. 8 The Linear Interpolation Method

The formula is:

$$\begin{cases} \alpha = x - x_0 \\ \beta = y - y_0 \\ G(P) = [G(P_3) - G(P_1)] \times \alpha + [G(P_2) - G(P_1)] \times \beta + \\ [G(P_4) + G(P_1) - G(P_2) - G(P_3)] \times \alpha \times \beta + G(P_1) \end{cases} \tag{5}$$

Similarly, when we deal with a pixel $P(x, y, z)$ in space, we take following steps:

1. Locate the pixel $A_1(i_0, j_0, k_0)$ in the three-dimensional array: $i_0 = [x/\Delta x]$, $j_0 = [y/\Delta y]$, $k_0 = [z/\Delta z]$($[x]$ is the maxium integer smaller than x), then we will use eight pixels to determine the gray-scale value of $P(x, y, z)$. The eight pixels are

$$A_1(i_0, j_0, k_0),\ A_2(i_0, j_0 + 1, k_0),$$
$$A_3(i_0 + 1, j_0, k_0),\ A_4(i_0 + 1, j_0 + 1, k_0),$$

$$A_5(i_0,j_0,k_0+1),\ A_6(i_0,j_0+1,k_0+1),$$
$$A_7(i_0+1,j_0,k_0+1),\ A_8(i_0+1,j_0+1,k_0+1);$$

2. Use the interpolation method for plane points (see formula (5)) to determine the grayscale value of $P_1(x,y,z_0)$ by $G(A_1)$, $G(A_2)$, $G(A_3)$ and $G(A_4)$; also to determine the gray-scale value of $P_2(x,y,z_0+1)$ by $G(A_5)$, $G(A_6)$, $G(A_7)$ and $G(A_8)$;

3. thus we obtain:

$$\begin{cases}\alpha = x/\Delta x - i_0\\ \beta = y/\Delta y - j_0\\ \gamma = z/\Delta z - k_0\\ G(P) = G(P_1) + \gamma \times [G(P_2) - G(P_1)]\end{cases} \tag{6}$$

Other Consideration

Assuming the selected pixel on the cross-section is also on the horizontal slice of the origihal three-dimensional array. If the original data on the horizontal slice satisfy the two-dimensional Nyquist principle, then we can recover the gray-scale value of any pixel on the slice as following:

$$G(P(x,y,z_0)) = \sum_{i=-\infty}^{+\infty}\sum_{j=-\infty}^{+\infty} G(A(i,j,k_0)) \cdot \frac{\sin\frac{\pi}{\Delta x}(x-i\times\Delta x)}{\frac{\pi}{\Delta x}(x-i\times\Delta x)} \cdot \frac{\sin\frac{\pi}{\Delta y}(y-j\times\Delta y)}{\frac{\pi}{\Delta y}(y-j\times\Delta y)} \tag{7}$$

There are aloso some algorithms to reconstruct the continuous gray-scale value in space as precisely as possible. However, as we will discuss following, these algorithms will cost a very long time and require huge memory in personal computer. Moreover, we find out through the testing of model that in normal conditions the simple algorithm is not inferior to these ones.

Test and Evaluation of the Algorithm

Test

Programming with Borland C+ +3.1, we have implemented our algorithm and tested them by self-designed data set. Here is a picture to give a first impression of our algorithm (from left to right: Nearest Method, Average Method, Linear Interpolation Method).

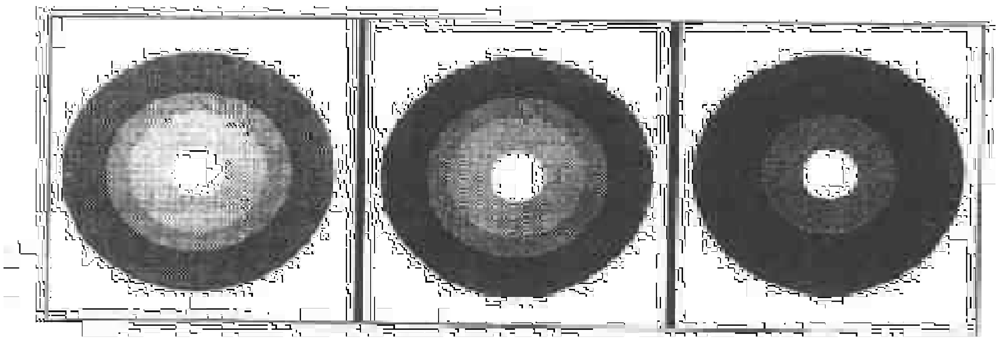

Fig. 9

Some typical results are also shown below to give clear demonstration of the effect. Each figure, from left to right, was respectively gotten by: Nearest Method, Average Method, Linear Interpolation Method and directly calculating from the original function (see Fig 10 and Fig. 11). In all the pictures, the cross-sections are the same: going through the point (2.56, 2.56, 2.56), having a normal vector (−1, −1, −1). And in all pictures, $\Delta h = \Delta v = 0.1$.

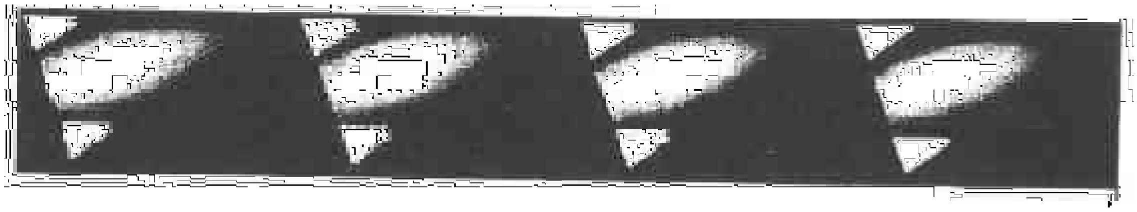

Fig. 10 Example of a cross-section picture(256 scales of gray, screen resolution 640 × 480) : original three-dimensional data array is sampled from function 255 × sin $(x/1.5)$ × sin $(y/1.5)$, Resolution: 512 × 512 × 256 $\Delta x = \Delta y = 0.25$, $\Delta z = 0.5$

Fig. 11 Example of a cross-section picture(256 scales of gray, screen resolution 640 × 480) : original three-dimensional data array sampled from function 255 × sin $(x/1.5)$ × sin $(y/1.5)$, Resolution: 1 024×1 024×512, $\Delta x = \Delta y = 0.125$, $\Delta z = 0.25$

We have also changed Δh and Δv, without changing other parameters in the Fig. 10, to change the number of pixels in cross-section and got a table of processing time of each method (our machine is a 486DX2/66 personal computer):

Table 1: Algorithm processing time

Number of pixels in cross-section picture	Nearest Method	Average Method	Linear Interpolation Method
7 856	0.9s	1.50s	1.71s
12 172	1.24s	1.68s	2.61s
18 536	1.36s	2.62s	3.56s
76 680	3.82s	6.79s	12.73s
307 200(640×480)	13.20s	26.55s	46.99s

Evaluation

Generally speaking, the three methods can reconstruct the picture of the cross-section well, what is more, they cost little time even running on our 486DX2/66 PC. We evaluate their performance with following indexes:

- T—Processing time

- S—The smoothness of reconstructed picture
- P—The ability of preserving sharp boundary information in pictures, that is, on what degree they avoid blurring the sharp boundary

Our evaluation of the these methods is given in table 2:

Table 2: Evaluation of methods

	Nearest Method	Average Method	Linear Interpolation Method
T	very fast	fast	fast
S	medium	medium	good
P	very good	good	medium

As we can see, index S and P are conflicted. One can not improve the smoothness of a picture while not blur its sharp boundaries between pixels when the resolution of the picture is determined. Under most circumstances, our algorithm can produce a useful compromise between these two indexes. A more complicated method to caculate the gray-scale value may produce a smoother picture while tends to blur the sharp boundary of the picture more seriously and cost more time. In our model, we focus our work on obtaining the information from original data. So we prefer methods that can preserve more boundary information, which happen to be easy in mathematics. After all, in practical situation, there are many mature methods to process an exist picture. One can use them to enhance the visual effect of a picture after obtaining it by our methods.

Conclusion

Finally, as a conclusion, we suggest that one adopt the Linear Interpolatin Method in common use because this method presents a good compromise in index S and P in most situation. If you feel the sharp boundary, you had better adopt Nearest Methoel and Average method.

Choosing Resolutions

In our model, two kinds of resolutions may influence the reconstructed picture:

- **The resolution of original data set**
- **The resolution of selected pixels on the cross-section**

Of course, the higher resolution the original data set has, the better the algorithm will perform. As you can see in Fig. 10 and Fig. 11, when the resolution of original data set increased from $512 \times 512 \times 256$ to $1\,024 \times 1\,024 \times 512$ (note that the actual size of object did not change), the quality of pictures has an obvious improvement. But when the resolution of selected pixels is high, it is not necessary that all algorithm will perform well. Here is another figure that can show it.

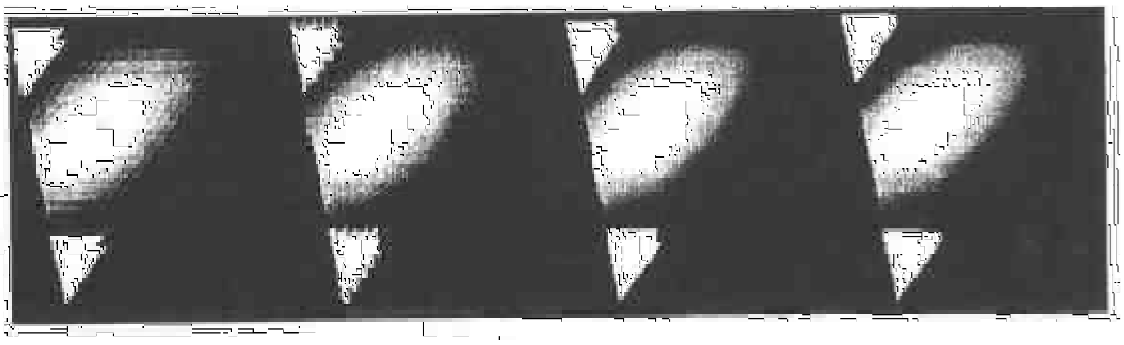

Fig. 12 Another presentation of Fig. 10 (Changing the Δv to 0.05)

In fact, in the first two methods, higher resolutions of selected pixels only enlarge the picture to be shown and will not get any more information. This is demonstrated by further analysis of the two methods. In the first two methods, space is divided into small regions. The gray-scale values of pixels in the same small region are equal, so the information we can get is limited. Therefore you can not increase information by selecting more pixels. Too more pixels can only show the shape of each region in the space more clearly. The third method, however, could provide smoother and larger picture when the resolution of selected pixels is high. But here exist a problem: when the original data set is given, the effectiveness of methods to reconstruct original picture is undetermined if we only have the original data without any extra information. So sometimes simple methods are also very practical and useful.

But what resolutions on earth are good ones? Our conclusion is : the higher the resolution of original data set , the better the final result; as to the resolution of selected pixels, it is more flexible and hard to determine we can try a set of values and find a satisfying one based on the visual effect.

Strengths and Weaknesses

Strengths

- Our model can reconstruct any oblique plane in space and can amplified the cross-section by increasing the resolution(selecting more pixels). It can also preserve original information of sharp boundary very well while still has a good visual effect.
- As we have analyzed before, in our model, the quality of the pictures is influenced by the resolution of original data set. So one can calibrate the MRI machines according to the required quality of pictures.
- The most valuable strength of our model is that it works very fast. Even in our 468DX/66, we need only 13.2 seconds to process (select and calculate pixels) a full screen picture(640×480) using the Nearest Method, less than 50 seconds using Linear Interpolation Method. We believe it can fit real-time use in a faster PC. If our algorithm is adopted in practical use, the processing of picture can be

implemented by software, which will save much money.

Weaknesses

• We have not test our algorithm by typical medical data for lack of access to real medical data.

• We could not determine the optimal parameter Δh and Δv for practical use.

References

• Zhu Xiaoping and Su Xuezeng. ABC of Magnetic Resonance Imaging. Shanghai: Tongji University Press, 1987.

• Zhao Rongchun. Introduction of Digital Image Processing. Xi'an: Northwest Polytechnic Univ. Press, 1991.

• Zhou Xiaokuan. Practical Computer Image Processing. Beijing: Beijing Areospace Univ. Press, 1994.

论 文 点 评

本篇论文给出了一种针对 MRI 问题的快速算法。算法包括了三个方面,即通过所选平面确定横截面的形状,确定横截面在所选坐标轴的位置,以及计算所选轴的灰度值。在计算灰度值时,文中提供了三种不同的方法,即最近法、平均法及线性插值法,通过对具体的实例的实现,分析了三种方法各自的适应情况及表现,总结了使用三种方法的一般原则。本论文的算法可实现性较好,处理图像的速度非常快,有着较好的应用前景;论文层次结构较为清晰,语言简洁流畅。不足之处,未能结合具体的医学实例,实现所给的算法。

论文获得美国 1998 年大学生数学建模竞赛一等奖。

implemented by software, which will save much money.

Weaknesses

- We have not test our algorithm by typical medical data for lack of access to real medical data.
- We could not determine the optimal parameter Δh and Δv for practical use.

References

- Zhu Xiaoping and Su Xuezeng. ABC of Magnetic Resonance Imaging. Shanghai: Tongji University Press, 1987.
- Zhao Rongchun. Introduction of Digital Image Processing. Xi'an: Northwest Polytechnic Univ. Press, 1991.
- Zhou Xiaokuan. Practical Computer Image Processing. Beijing: Beijing Areospace Univ. Press, 1994.

论 文 点 评

本篇论文给出了一种针对 MRI 问题的快速算法。算法包括了三个方面，即通过所选平面确定横截面的形状，确定横截面在所选坐标轴的位置，以及计算所选轴的灰度值。在计算灰度值时，文中提供了三种不同的方法，即最近法、平均法及线性插值法，通过对具体的实例的实现，分析了三种方法各自的适应情况及表现，总结了使用三种方法的一般原则。本论文的算法可实现性较好，处理图像的速度非常快，有着较好的应用前景；论文层次结构较为清晰，语言简洁流畅。不足之处，未能结合具体的医学实例，实现所给的算法。

论文获得美国 1998 年大学生数学建模竞赛一等奖。