MRI Slice Picturing

Jiang Ni
Jun Chen
Ling Li
Tsinghua University
Beijing, China

Advisor: Jun Ye

Summary

We set up two coordinate systems, one in the object space and the other on the computer screen. We introduce six parameters to describe the slice plane, and we formulate the coordinate mapping from the screen to the object space.

We designed six alternative algorithms that use the given data to estimate the density at any location in space and produce a slice of a three-dimensional array. Some of the algorithms exploit global information and some are self-adaptive; all but one have advantages in certain circumstances.

We extended a well-known two-dimensional model of a human head model to build a three-dimensional model of a head, consisting of 10 ellipsoids of different size, orientation, and density. We produced the data sets by sampling in the object space (the head model) at evenly spaced intervals; the dimension of the data set is $128 \times 128 \times 128$.

We devised several test slices to test our model and algorithms. Some test slices have a complex shape, some are critical in their position, and some are really disasters to most algorithms. We also tried different sampling intervals to verify our ideas about the model.

Based on subjective and objective comparisons, we summarize the strengths and weaknesses of the algorithms. For common use, we suggest the gradient algorithm and our GNP-integrated algorithm. In most cases, both can render well slices with both sharp and smooth edges.

Facts about MRI

MRI has several features relevant to the problem:

The UMAP Journal 19 (3) (1998) 263–279.

- **High precision.** The scanning precision of MRI is about 1-3 mm. That is, MRI can easily distinguish nuances at the size of 1-3 mm. Commonly used MRI slices are no larger than 25 cm $\times$ 25 cm [Gao 1996].

- **High contrast.** One of the advantages of MRI is the high contrast of its images, which makes the boundaries of the organs sharp enough for diagnosis [Gao 1996, Frommhold and Otto 1985].

- **Long performance time.** The performance time of MRI is several minutes. For example, a typical scanning of a two-dimensional image ($128 \times 128 \times 256$) with pulse repeat time $T_R = 1.5$ s needs about 6 min [Gao 1996]. The time required is still one of the main drawbacks of MRI. Thus, we cannot expect that the given data set will be thorough enough to produce a good slice picture (that may require too much time). Our algorithms should not be too complex or time consuming.

- **Reconstruction Algorithms.** Two commonly used methods to reconstruct the three-dimensional information from the raw data produced by MRI are Projection Reconstruction (PR) and Fourier Transformation. They require that the data be evenly sampled through space, so we assume that.

Assumptions, Coordinates and Notations

Assumptions

From the request of the problem and some facts of MRI, we take the following assumptions:

- The dimension of the examined object is 256 mm $\times$ 256 mm $\times$ 256 mm, if not smaller; this is big enough in most cases. If a larger object is scanned, we can divide up the data into several cubes.

- The desired precision of pictured slices is 1 mm. We will picture the slices produced by our algorithms on the computer screen, using one pixel to present an area of 1 mm $\times$ 1 mm.

- The given data set is a three-dimensional array $A(i, j, k)$ sampled in the whole object space with evenly spaced intervals along the coordinate axes. Such intervals are about 2-4 mm and are big enough for MRI to scan in not too long a time. Later we discuss the case of the data not being evenly spaced.

- $A(i, j, k)$ takes an integer value from 0 through 255, indicating the water density, from high density to low density. On our screen, 0 is represented by black and 255 is represented by white.

- The examined object consists of several different components. We assume that the density does not change much within one component.

- The object is the body of some animal or of a human being. Since the organs or tissues in such body are likely to be tender, we can image that the boundaries are smooth and sharp in most cases. Exceptions will only happen between some kind of bones, such as the backbone (they are sharp but not smooth) or some sick tissues.
- The unknown density of a location is affected by all the given data. However, the distance between points plays an important role in this problem. Locations far away (for instance, 50 mm) from the unknown point are assumed to have little or no effect.

Coordinate Systems

We set up two coordinate systems, one in the data (or object) space and one on the computer screen, as presented in **Figures 1–2**. The units (pixel in the screen image) in these two systems are both 1 mm, for the sake of convenience. Since the object is of 256 mm $\times$ 256 mm $\times$ 256 mm, the data space is just $0 \leq x, y, z < 256$.

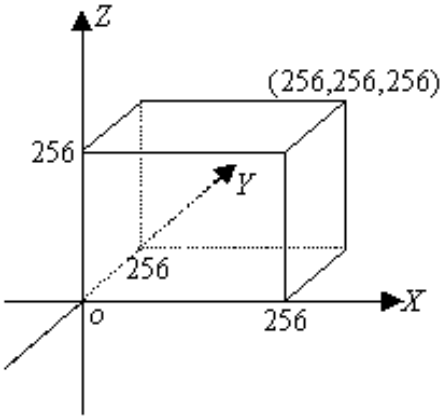


Figure 1. The data space coordinate system.

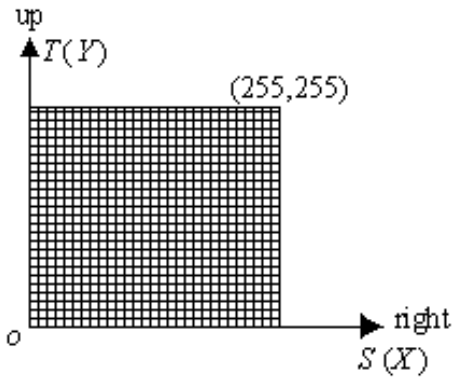


Figure 2. The screen image coordinate system. The origin O is the left bottom corner of the screen image.

Notation

"Density" indicates the water concentration in a small region of the scanned object at some location. The phrase "unknown point" or "unknown location" means the point (or the location) where the density of the object is not given as known data, hence needs to be calculated.

The symbols commonly used in this paper are:

$A(i,j,k)$	The given three-dimensional data indicating the density of a location. In some contexts, A also represents the location.
s_X, s_Y, s_Z	The three sampling intervals along the Cartesian axes. Thus, $A(i,j,k)$ is the density of location $(i \cdot s_X,\ j \cdot s_Y,\ k \cdot s_Z)$.
$\alpha, \beta, \gamma, x_0, y_0, z_0$	The six parameters to define a slice plane.
$D(x,y,z)$	The density of the object at location (x,y,z).

Analysis of the Problem

For a plane slicing the object, we want to know the density of the object throughout the plane. If we can convert the coordinates of the points in the slice plane to the real 3-D coordinates in the object, and calculate the corresponding densities, the problem is solved. The first step is simple, with some knowledge of the space geometry. But how about the second step?

Can the Unknown Density Be Known?

From the famous Nyquist sampling theorem, we know that to reconstruct the whole density information of the scanned object *exactly*, the sampling intervals must satisfy the inequality

$$\max(s_X, s_Y, s_Z) \le \frac{1}{2f_m}, \qquad \textbf{(1)}$$

where f_m is the upper limit of the spatial frequency of the density. In our problem, **(1)** would need to be satisfied if a slice is required to be pictured exactly; but we don't need to do that.

On the one hand, the inequality could never be satisfied, since the f_m in an object is always very large—infinity, in reality. No sampling intervals can satisfy such an inequality! On the other hand, to picture the slice we do not need to know *exactly* what the unknown is. Since the grayscale is from 0 to 255, an error of less than 1 grayscale unit is acceptable. In fact, a blur to some extent is always allowable and unavoidable. In this sense, we can know the unknown density.

How to Know the Unknown Density?

Since we cannot know the unknown density exactly, we must estimate it. We can choose from:

- **Simplicity and Complexity.** Our goal is to find an effective but simple algorithm to produce any slice of the object. We also believe that the real

object is too complex to describe or estimate by only one kind of algorithm. So our motto is "If it works, it's good enough," and we tried to find several different algorithms to deal with the different aspects of the real object.

- **Local and Global Information.** Global information is alluring but very difficult to use. As human beings, we can easily locate a vessel or a bone in an MRI image and outline it, using our global impression (thus we can do reconstruction). But it is difficult for computers to know a shape rather than a number. Current algorithms can outline an image, but the information used by computers is local (e.g., the difference between adjacent pixels) but not global. So we base our main idea on local information but remain alert to global information. Our experiments show that appropriately using even a little global information brings great benefit.

- **Static and Self-adaptive Algorithm.** There are several advantages to a static algorithm: it is fast and simple (in most cases), it is often designed by aiming at some aspect of the real application and may be effective in that aspect, and it is easily controlled and safe. Similar to global information, a self-adaptive algorithm is powerful but difficult to control.

Description of Our Model

Our model consists of three parts:

- the given data,
- the description of the slice plane, and
- the algorithm to estimate the density of the object at any location, whether or not this location is included in the given data.

The description of the slice plane is used to convert the coordinates of a point on the screen to coordinates in space, while the density-estimating algorithm obtains the density of the point from the known data. Thus, the slice can be easily displayed on the screen.

We propose six different algorithms to estimate the density, discussed in detail in the next section. The given data are already described in the problem and the subsection on **Assumptions**, so here we treat the mapping from the screen to space.

Slice Plane

In order to define a slice plane at any orientation and any location in space, we perform four steps to transform from the XY-plane to any other plane in the space:

1. Put a plane, say P, with its own coordinate system ST (the same as the screen coordinate system), onto the XY-plane and make the two coordinates exactly the same origin and orientation.

2. Rotate P around its normal line (i.e., the z-axis) by angle α to make the orientation of ST differ from XY.

3. Rotate the normal line of P around the origin to a prescribed orientation. In **Figure 3**, this orientation is defined by angles β and γ.

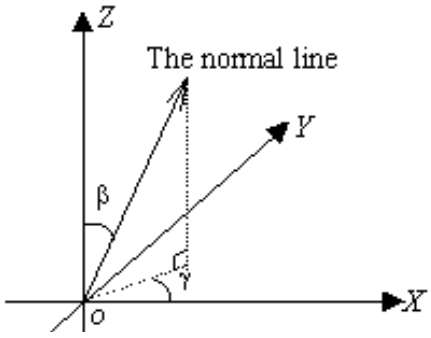


Figure 3. Rotate the normal line to a prescribed orientation.

4. Perform a translation to the plane P, moving the origin of P to some predefined point in the space, say (x_0, y_0, z_0).

Thus, using six parameters $(\alpha, \beta, \gamma, x_0, y_0, z_0)$, we can define a plane anywhere, with a coordinate system the same as the screen system.

Mapping from the Screen to the Space

Since the coordinate systems of the screen and of the slice plane are the same, we can change the screen coordinate to the slice plane and then use the the transformation in the last subsection to convert the slice plane to space coordinates.

Suppose a pixel in the screen is at position (s, t) and the corresponding point in the space is at (x, y, z). From the transformation, we get the mapping equation from the screen to the space and thereby solve the first step of the problem:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s \\ t \\ 0 \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}.$$

Density-Estimating Algorithms

Consider a pixel on the screen and the corresponding point U at location (x, y, z) in the object space. The task of the density-estimating algorithm is to estimate the density at U, which we denote by $D(x, y, z)$.

We tried five basic types of density-estimating algorithms. Based on experimental results, we designed an all-around effective method, which we call *GNP-integrated.*

Trilinear Interpolation

In general, linear interpolation can produce satisfactory results. In three-dimensional space, we use trilinear interpolation, which interpolates from the eight neighbors in three directions. That is,

$$\begin{aligned} D(x,y,z) &= A(i,j,k)\cdot(1-u)\cdot(1-v)\cdot(1-w) \\ &+ A(i+1,j,k)\cdot u\cdot(1-v)\cdot(1-w) + A(i,j+1,k)\cdot(1-u)\cdot v\cdot(1-w) \\ &+ A(i,j,k+1)\cdot(1-u)\cdot(1-v)\cdot w + A(i+1,j+1,k)\cdot u\cdot v\cdot(1-w) \\ &+ A(i+1,j,k+1)\cdot u\cdot(1-v)\cdot w + A(i,j+1,k+1)\cdot(1-u)\cdot v\cdot w \\ &+ A(i+1,j+1,k+1)\cdot u\cdot v\cdot w, \end{aligned} \quad \textbf{(2)}$$

with

$$i = \left\lfloor \frac{x}{s_X} \right\rfloor,\ j = \left\lfloor \frac{y}{s_Y} \right\rfloor,\ k = \left\lfloor \frac{z}{s_Z} \right\rfloor,$$
$$u = \frac{x}{s_X} - i,\ v = \frac{y}{s_Y} - j,\ w = \frac{z}{s_Z} - k,$$

where $\lfloor x \rfloor$ is the largest integer no larger than x.

This method uses the density values of eight neighbors to resolve the density at U. Discontinuity at boundaries can be mitigated by this approach in nearly all cases. However, this method tends to blur some sharp edges, because of its intrinsic low-pass filtering attribute.

Nearest-Neighbor

With the preservation of edge sharpness in mind, we tried the nearest-neighbor method, which assigns to U the density of its nearest neighbor in space.

This method is fairly simple and the computational load is very low. The effect produced by the method is quite unstable, although sometimes it really gives good results. Nevertheless, it partly preserves edge sharpness, and its power can be amplified if properly combined with other methods.

Median

The idea comes from the median filtering that is famous in signal and image processing. Median filtering can preserve the sharp edge of the signal from great damage while smoothing the signal. In our algorithm, we assign the median of the density of U's eight neighbors to U. The result of this algorithm, as expected, gives sharp edges but has obvious feather-out, which results in an unrealistic contour.

Power-Control

Since we believe that the distance between points is very important, we can conceive that each point within a reasonable distance from U has a "power" to control the density of U, forcing the density of U to be similar to its own, and that such power decreases with distance. The overall result should be the average of the densities of those points, taking their power into account.

We define the power of $A(i,j,k)$ over distance d as:

$$p = \frac{1}{1 + e^{5(d/d_0 - 1)}},$$

where $d = \sqrt{(x - i \cdot s_X)^2 + (y - j \cdot s_Y)^2 + (z - k \cdot s_Z)^2}$ and d_0 is a distance threshold (when $d = d_0$, the power is $\frac{1}{2}$). Then the density of U is estimated as

$$D(x,y,z) = \frac{\sum\limits_{d_\xi \le 2d_0} p_\xi \cdot A(i_\xi, j_\xi, k_\xi)}{\sum\limits_{d_\xi \le 2d_0} p_\xi}, \qquad \textbf{(3)}$$

with the summation over all the known points within a distance $2d_0$. We use $d_0 = 1$ mm when the sampling interval is 2 mm.

Though **(3)** has some similarity to trilinear interpolation **(2)**, the nonlinearity in the definition of power makes the edge produced by this algorithm smoother—but also more blurred.

Here we could adopt another type of power, called the *optimal interpolation function*:

$$p = \operatorname{sinc}(\pi d) = \frac{\sin(\pi d)}{\pi d}, \text{ where } d = \sqrt{\left(\frac{x}{s_X} - i\right)^2 + \left(\frac{y}{s_Y} - j\right)^2 + \left(\frac{z}{s_Z} - k\right)^2}.$$

This kind of power is famous, for it is an ideal low-pass filter in the frequency field, and it is used to reconstruct the original signal (from frequency information) in the sampling theorem. But since f_m is very large in this problem, we cannot expect that such a power will do a good job.

Gradient

The methods above are all based on the effect of one point on another. If the effect of a point-pair to one unknown point is considered, we can introduce the gradient method.

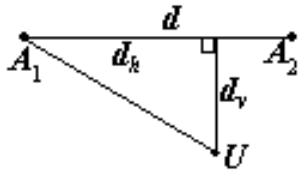


Figure 4. Gradient in a point-pair.

Figure 4 shows two given data values $A_1(i_1, j_1, k_1)$, $A_2(i_2, j_2, k_2)$ and the unknown point U. The distance between A_1 and A_2 is d, the projection of $\overrightarrow{A_1U}$ on $\overrightarrow{A_1A_2}$ is d_h (which is negative when the angle between $\overrightarrow{A_1U}$ and $\overrightarrow{A_1A_2}$ is an obtuse angle), and d_v is the distance from U to $\overrightarrow{A_1A_2}$. The density of point U, if only estimated by the gradient from A_1 to A_2, is

$$D(x, y, z) = A_1 + \frac{d_h}{d}(A_2 - A_1).$$

However, when other data-pairs in the neighborhood of point U are considered, the density D is a weighted average of all the effects, and the weight (similar to the "power") is defined as:

$$p = \begin{cases} e^{-d_v}, & \text{when } d_h \geq 0; \\ \frac{1}{4}e^{-d_v}, & \text{when } d_h < 0. \end{cases}$$

This algorithm exploits not only the density information around the unknown point U but also the tendency of the density in a local volume. This makes it self-adaptive to some extent. Further, we can add some global information to the algorithm. For example, in our implementation, when A_1 and A_2 are close enough ($|A_1 - A_2| < 20$), we multiply the weight p by 3; in such a case, A_1 and A_2 are deemed to be in the same component, which makes the probability that U is also in the same component very large. Similarly, when $|A_1 - A_2| > 80$, we multiply the weight p by 0.7, since A_1 and A_2 may be in different components.

GNP-Integrated

From experimental results (see the next section), we found out that the gradient and power-control methods are good at making smooth but slightly blurred edges, while the nearest-neighbor method always gives a high-contrast image with rough edges. In an attempt to combine their advantages, we integrated these three methods into one algorithm that we call *GNP-integrated*. It

can be described in brief as the combination of the gradient, nearest-neighbor, and power-control methods in the proportions $3:2:1$.

Test of the Algorithms

Data Sets: The Head Model

Suitable data sets are necessary for testing and demonstrating the algorithms, as well as for comparing different algorithms. Real MRI data would be best. However, besides the inconvenience involved in getting such data, another annoying problem is that we would have great difficulty comparing the pictured slices and the actual slices, since we actually can't have the latter!

Motivated by the widely accepted two-dimensional Sheep–Logan (S-L) head model [Gao 1996], where ten ellipses, different in location, shape, orientation and intensity, constitute an object representing a head section, we designed a 3-D head model made up of ten ellipsoids different in location, shape, orientation and density. The empty space inside the head model is filled with an ambient color that differs from that of the background outside the model.

We adopt ellipsoids for our data model because of their simplicity and because the combination of varied ellipsoids can imitate many real objects, such as a brain or a stomach. We designed three types of ellipsoids with different density distributions:

- Type 1: Uniform density.
- Type 2 : The density changes linearly from the center to the surface.
- Type 3: The same as type 2 but with additive random noise of a specific standard deviation. (In our experiments, we don't analyze this type, since noise filtering is beyond our concern in this paper.)

With the sampling intervals specified, data sets can be produced easily by determining in which ellipsoid a sample point lies. Such data sets are large; for example, when the intervals are all 2 mm, the data set is $128 \times 128 \times 128 = 2$ MB.

In addition, we can also compute the actual slice with our head model. The computation process is similar to the data set producing process.

Experiments and Results

An important issue is how to test the output of different algorithms. The two main objectives of this problem, maintaining the sharpness of the edges and the smoothness of the contours, are difficult to measure numerically, so we made comparisons by visual inspection (subjective as it is).

At the same time, although the RMS (root mean square) error cannot comprehensively and rationally reflect the quality of a pictured slice, it is still helpful to the assessment of a pictured slice. So we take the minimization of the RMS error as our third goal. (This makes sense only for our simulated data, since the actual slice is unknowable in the real world.)

We also must take the computational loads into consideration, since the data set is comparatively large.

We did a number of comparisons, from which, we present some representative slices. Typical slices are presented in **Figures 5–8**. We examine each in detail and then draw some conclusions. (In all cases, $s_X = s_Y = s_Z = 2$.)

In **Figure 5**, the slice is in the middle of the scanned object and parallel to the XZ-plane. In this case, the slice traverses all the ten ellipsoids, so the overall performance of each method is easy to evaluate. In **Figure 6**, the slice is oblique and all algorithms work well, except for the power-control using the sinc function. In **Figure 7**, the slice in **Figure 6** is translated by just a tiny distance, but the performances of some algorithms fall dramatically. In **Figure 8**, the slice plane is at an odd angle and in a critical position, which gives our algorithms a chance to show their performance in an awful situation.

Assessment of the Algorithms

Except for the power-control method (with the sinc function), which is clearly unsuited to this problem, each method has advantages and disadvantages.

Trilinear

The trilinear method works well in common cases. It usually has a small RMS error and takes a short time. But it has a tendency to blur the picture, and it disappointed us in the awful situation (see **Figure 8b**). When the contrast of different components in the scanned object is low, this method is not recommended.

Nearest-Neighbor and Median

Both the nearest-neighbor method and the median method preserve a sharp edge and take the least time. But they lose the smoothness of the contours and cannot discriminate small objects (see **Figures 6cd**). A small translation of the slice plane also causes them to produce many more zigzag contours (see **Figure 7cd**). Consequently, they have big RMS errors. In spite of these weaknesses, when the data set is very large and the time element is more important, or when the zigzag has little bad effect on the result, these methods are desirable.

a. Actual slice.

b. Trilinear interpolation (14.5).

c. Nearest-neighbor (20.0).

d. Median (23.9).

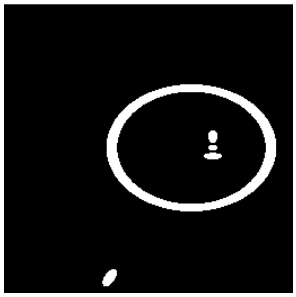
e. Power-control (15.3).

f. Power-control (sinc) (29.3).

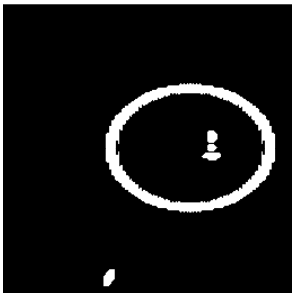
g. Gradient (12.9).

h. GNP-integrated (14.4).

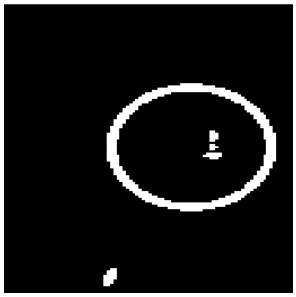
Figure 5. A slice in the middle of the object and parallel to the XZ-plane, with parameters $x_0 = 0$, $y_0 = 128$, $z_0 = 0$, $\alpha = 0$, $\beta = 90°$, $\gamma = 90°$. The number following each algorithm name is the RMS error.



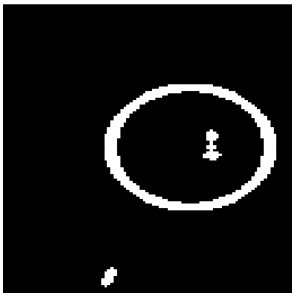
a. Actual slice.



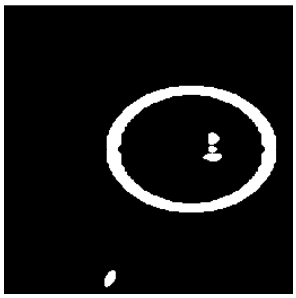
b. Trilinear interpolation (12.3).



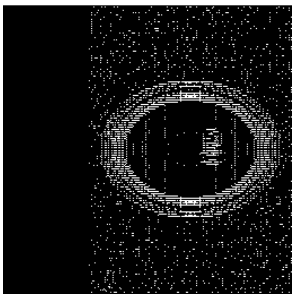
c. Nearest-neighbor (15.5).



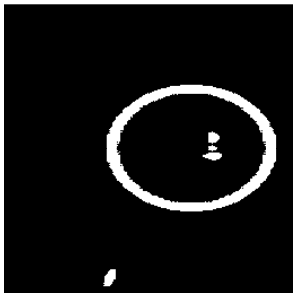
d. Median (18.7).



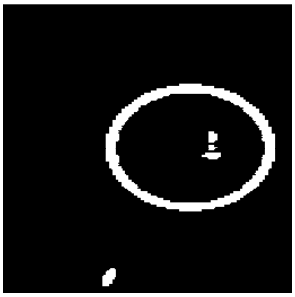
e. Power-control (13.7).



f. Power-control (sinc) (55.8).

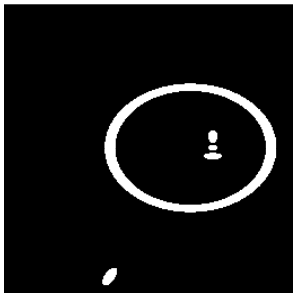


g. Gradient (11.3).

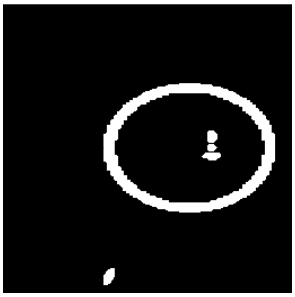


h. GNP-integrated (12.0).

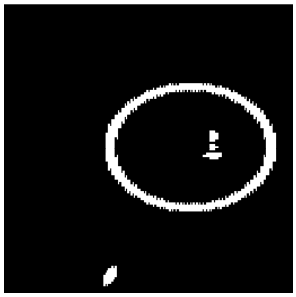
Figure 6. An oblique slice, with parameters $x_0 = 0$, $y_0 = 128$, $z_0 = 0$, $\alpha = 0$, $\beta = 45°$, $\gamma = 90°$. The number following each algorithm name is the RMS error. The black area on the left of each slice is outside the scanned object.



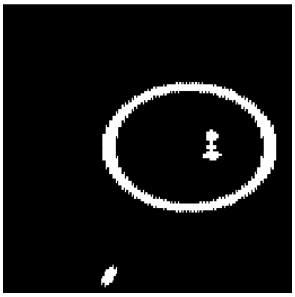
a. Actual slice.



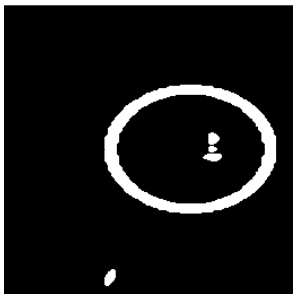
b. Trilinear interpolation (13.0).



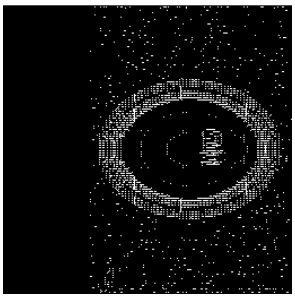
c. Nearest-neighbor (18.8).



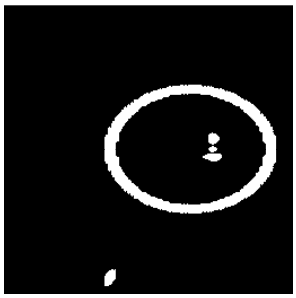
d. Median (20.6).



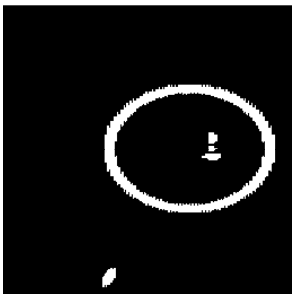
e. Power-control (13.8).



f. Power-control (sinc) (55.0).

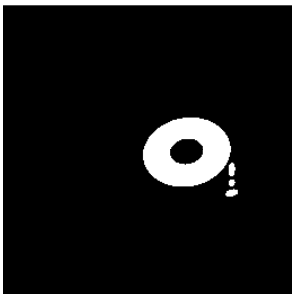


g. Gradient (12.3).

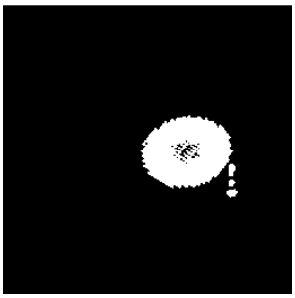


h. GNP-integrated (13.9).

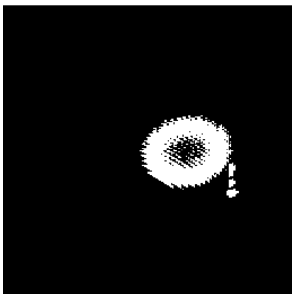
Figure 7. The oblique slice of **Figure 6** translated a tiny distance (one unit in the y-direction), with parameters $x_0 = 0$, $y_0 = 129$, $z_0 = 0$, $\alpha = 0$, $\beta = 45°$, $\gamma = 90°$. The number following each algorithm name is the RMS error.



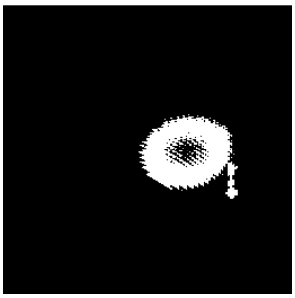
a. Actual slice.



b. Trilinear interpolation (12.3).

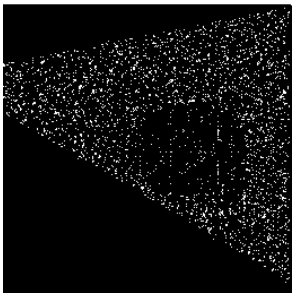


c. Nearest-neighbor (17.5).

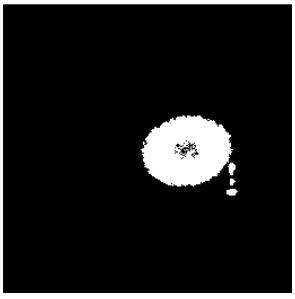


d. Median (18.0).

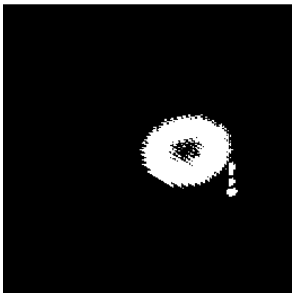
e. Power-control (13.8).



f. Power-control (sinc) (68.0).



g. Gradient (12.2).



h. GNP-integrated (13.4).

Figure 8. An oblique slice at an odd angle and in a critical position, with parameters $x_0 = 0$, $y_0 = 126$, $z_0 = 0$, $\alpha = 0$, $\beta = 70°$, $\gamma = 60°$. The number following each algorithm name is the RMS error.

Gradient

The gradient method has the amazing advantage that it has the minimal RMS error in all cases. More important, the slice as pictured by this method has rather satisfactory smoothness *and* sharpness, as is obvious in any of the figures.

GNP-Integrated

The GNP-integrated method has RMS error a little larger than that of the trilinear method; however it excels over any other algorithm when the sharpness and smoothness are taken into account.

Conclusion

The gradient and GNP-integrated algorithms are the most competent if the runtime (10-14 s on a Pentium 166 for our implementation) is not a serious consideration (the other algorithms take 2-3 s). They are especially powerful in awful situations when some critical oblique slice is desired.

What Happens When the Interval Is Too Large

To verify our discussion on the sampling intervals (see the section **Can the Unknown Density Be Known?**), we also tested the result when $s_X = s_Y = s_Z = 4$. As expected, the quality of the produced slices deteriorated. For example, some connected thin boundaries in **Figure 9** are broken because of insufficiency of the data.

Figure 9. The slice of **Figure 5a**, with sampling interval 4 mm, as rendered by the gradient method; compare with **Figures 5a** and **5g**.

If a data set is not sampled at evenly spaced intervals, or if the data are too scattered, the user should first use simple interpolation to construct a data set with evenly spaced sampling intervals.

Strengths and Weaknesses

- We present several good algorithms that can be selected by the user to fit different situations.
- We present a clear assessment of different algorithms, based on experimentation on simulated data for a head.
- We implemented all of the algorithms in a Windows95 user-oriented computer simulation with easy input, suitable for repeated experimental research.
- We tried to find objective measurements of sharpness and smoothness but time did not permit.

References

Frommhold, H., and R.Ch. Otto. 1985. *New Methods of Medicine Imaging and Their Application* (German). 1988. Chinese translation by Wang Zhen and Gu Ying. Beijing, China: Medician Technology Press.

Gao, S.K. 1996. Imaging System in Medicine. Beijing, China: Dept. of Electrical Engineering, Tsinghua University.