

AN ALGORITHM FOR FINDING PARAMETER – DEPENDENT CONNECTED COMPONENTS OF GRAY IMAGES

Yang Wang

Computer Science Dept.
Southwest Missouri State University
Springfield, MO 65804

Prabir Bhattacharya

Dept. of Computer Science and Engineering
Univ. Nebraska-Lincoln
Lincoln, NE 68588

ABSTRACT

In a previous work we have introduced the concept of a parameter-dependent connected component of gray-scale images that takes into account both the gray values of the pixels and the differences of the gray values of the neighboring pixels. This concept is a convenient tool to analyze or understand images at a higher level than the pixel level. In this paper, we describe an algorithm for finding the parameter-dependent components for a given image. We discuss different strategies used in the algorithm and analyze their effects through the experimental results. Since the proposed algorithm is independent of the formation of the images, it can be used for the analysis of many types of images. The experimental results show that for some appropriate values of the parameters, the objects of an image may be represented by its parameter-dependent components reasonably well. Thus, the proposed algorithm provides us with the posibility of analyzing images further at the component level.

1. INTRODUCTION

The theory of digital connectivity was first studied by Rosenfeld [1], and since then a number of subsequent papers, e.g., [2, 3, 4, 5, 6, 7, 8, 9, 10], have appeared. Rosenfeld [9] gave a definition of connected components in which the pixels of a connected component have the same gray value. In a binary image, the objects are usually represented by either black pixels or white pixels and a connected component may be defined as a maximal black (or white) area. In this case, the properties of the connected components, such as connectedness, correspond to the properties of the objects in the images. In a multi-gray-value image, an object usually contains pixels which have different gray values. If we limit each connected component to have only one gray value, an object might be "cut" into several pieces while each piece belongs to a different connected component. In this case, the properties of connected components do not necessarily correspond to the properties of the objects in the images. However, in practical situations, for comprehensive image understanding we would like to know the inter-relationship among the objects, rather than among the "parts" of the objects. So, we feel that it would not be convenient in practice to constrain each connected component to have the same gray values. Instead, it may be more useful to relax this condition to allow the pixels in the same an image corresponds to a connected component.

Recently ([11]), we have introduced the concept of (ϵ, δ)-components of gray images that takes into account both the gray values of the pixels and the differences of the gray values of the neighboring pixels, where ϵ, δ are two parameters. Each such component may contain the pixels which have different gray values. We have discussed [11] some properties of (ϵ, δ)-components which may help us to analyze and understand the structure of an image at a higher level. The experimental results have shown that for some appropriate parameter values, an (ϵ, δ)-component may represent an object of an image reasonably well. So, the properties of the (ϵ, δ)-components describe the properties of the corresponding objects of the image. In this paper, we shall discuss an efficient algorithm to find the (ϵ, δ)-components for a given image and some strategies used in the algorithm.

2. PRELIMINARIES

A gray image Σ is represented by a set of points (also called pixels) each of which has a certain *gray value* representing the intensity or brightness of the point. We shall use $\sigma(p)$ to denote the gray value of the point p. Although, theoretically, $\sigma(p)$ could be any number,

Support by the US Air Force Office of Scientific Research under Grant AFOSR 9620-92-J-286 and Grant AFOSR F49620-94-1-0029 are gratefully acknowledged, and also matching funds from the Center for Communication and Information Sciences, University of Nebraska-Lincoln.

we shall take it, for convenience, to be a non-negative integer. The *spectrum* of the gray values of a set of pixels C is defined to be $max_C - min_C$, where min_C, max_C denote the minimum and the maximum of the gray values in C respectively. Two points in a gray image Σ are called *adjacent* (or *neighbors*) if they share either a vertex or an edge. Our treatment does not depend on the grid system chosen to represent an image and the way in which the points share vertices or edges. Instead of considering whether the points are 4-, 6- or 8-neighbors (see [9] or [12] for definitions), we shall simply consider here only whether two points are adjacent or not.

A *path* between two points p_0 and p_n in a gray image Σ is a sequence of points $p_0, p_1, \ldots, p_n$ such that $p_i \in \Sigma$ and p_i and p_{i-1} are adjacent for all $1 \leq i \leq n$. Given nonnegative integers ϵ and δ, we say that two distinct points p, $q \in \Sigma$ are (ϵ, δ)-*connected* if there exists a path $p = p_0, p_1, \ldots, p_n = q$, such that the maximal variation of the gray values of the points on the path is less than or equal to ϵ, and the maximal variation of the gray values of any two adjacent points along the path is less than or equal to δ. Such a path will be called an (ϵ, δ)-*connected path* between p and q. A subset of Σ is called an (ϵ, δ)-*connected set* if each pair of points of the subset is (ϵ, δ)-connected. The definition of (ϵ, δ)-connectedness gives us a convenient tool to study the variation of gray values in an image. By varying the parameters ϵ and δ, we may investigate the diverse distribution of gray values. Given a point $p \in \Sigma$, any maximal (ϵ, δ)-connected set containing p is called a *related* (ϵ, δ)-*connected component* (in short, (ϵ, δ)-*RCC*) of p, and is denoted by C_p^{Σ}; the point p is called the *seed point* of its (ϵ, δ)-RCCs. (By a *maximal* (ϵ, δ)-connected set, we mean an (ϵ, δ)-connected set S such that there exists no other (ϵ, δ)-connected set which contains S properly, *i.e.*, for any point $p' \notin C_p^{\Sigma}$, there is at least one point $q \in C_p^{\Sigma}$, such that p', q are not (ϵ, δ)-connected.) If we fix the value of ϵ as the spectrum of Σ, then the concept of (ϵ, δ)-RCC has similarity with the concept of G-neighbors [13]. For a given image, each (ϵ, δ)-RCC obtained by Algorithm 3.1 (see Section 3) is called an (ϵ, δ)-component of the image.

3. AN ALGORITHM TO FIND (ϵ, δ)-COMPONENTS

Algorithm 3.1 Let Σ be a gray image. The steps of the algorithm are as follows:

0. input a set Σ, integers $\epsilon \geq 0$ and $\delta \geq 0$;
1. $buffer = \Sigma$; $i = 0$;
2. *while* ($buffer \neq empty$) *do*
   {
3. select a point $p \in buffer$ as the seed point of the next (ϵ, δ)-RCC; $i = i + 1$;
4. find an (ϵ, δ)-RCC C_p^{buffer}; $S_i = C_p^{buffer}$;
5. $buffer = buffer - C_p^{buffer}$;
   }

First, we make some general remarks about the algorithm and then in the following subsections we shall discuss in detail certain steps of the algorithm. Up to this point, the algorithm does not depend on the grid system used to represent an image. Also, it does not depend on the manner (among the various possible ways) in which two points are defined as *adjacent*. Since for any point p, its (ϵ, δ)-RCCs are non-empty, we always have $| C_p^{buffer} | \geq 1$, therefore

$$| buffer - C_p^{buffer} | < | buffer |.$$

So, we see that the algorithm will definitely terminate after a finite number of steps. Also the algorithm gives a partition of Σ: This is because according to the algorithm, for some k, $1 \leq k \leq | \Sigma |$, we have $\Sigma = S_1 \cup S_2 \cup \ldots \cup S_k$, and $S_i \cap S_j = \emptyset$ if $i \neq j$. Each partition S_i obtained by using Algorithm 3.1 is called an (ϵ, δ)-*connected component* of Σ. For convenience, we shall refer to an (ϵ, δ)-connected component as an (ϵ, δ)-*component*. Since the variable *buffer* in the algorithm is a set of points, it is clear that we could always find a point p in *buffer* in Step 3 of the above algorithm as long as *buffer* is not empty. This procedure can be done in $O(| buffer |)$ time. During each iteration, the content of the variable *buffer* in Step 4 of the algorithm is a subset of Σ and the content of *buffer* in the ith iteration is a proper subset of that in the $(i-1)$th iteration. So, an (ϵ, δ)-RCC of the point p in the buffer is not necessarily an (ϵ, δ)-RCC of p in Σ. (Later in this section, we shall give an algorithm for finding an (ϵ, δ)-RCC of a point p in *buffer* in $O(| buffer |)$ time.) Since $| buffer | \leq m \times n$, Steps 0, 1, 3, 4, 5 can be done in $O(m \times n)$ time. From the above discussion, it is clear that the iteration can be repeated at most $m \times n$ times. Therefore, Algorithm 3.1 has an upper bound run–time $O((m \times n)^2)$.

3.1. Strategies for choosing a seed point

Now, we shall discuss more details about Step 3 of Algorithm 3.1. During the iteration, there would be often more than one point in *buffer*. So, a natural question is which one should we select as the seed point p for the next (ϵ, δ)-RCC? In principle, any one of them could be selected. There are many possible strategies to select it. (For example, we may choose p randomly, or as the point that has the smallest gray value, or the greatest gray value, or the average gray value, in *buffer*.)

If we have some pre-knowledge of the gray values of the objects we are interested in, and hope that each potential object to be partitioned has one component, then we may make a more specific choice for selecting p. For example, in the image at the left side of the second row of Figure 1, the objects are darker than the background, so we may select the darkest point in *buffer* as the seed point p. The image at its right side shows the corresponding (ϵ, δ)-components with $\epsilon = 46$ and $\delta = 10$. For comparison, the image below it shows the (ϵ, δ)-components with the same ϵ, δ values but here we select the brightest point in *buffer* as the seed point during each iteration. As we can see, in the first image the chromosomes are represented by the (ϵ, δ)-components very closely, but in the second image the pixels of each chromosome have been grouped into several (ϵ, δ)-components. If we decrease the value of ϵ, then before the pixels of a chromosome been grouped into an (ϵ, δ)-component (see the chromosomes at the top left and top right of the image at the right side of the third row of Figure 1) the range of the gray values of the pixels within another chromosome may already exceed the value of ϵ (see the chromosomes at the middle of the right side of the same image). So, if we select the brightest point as the seed point, it is unlikely to have the values of ϵ and δ such that the chromosomes could be represented by the (ϵ, δ)-components. In a more general situation, we may use a histogram to find the distribution of the gray values of the objects and the background. Then, we may choose a point which has a peak or a valley value accordingly as a seed point. Note that, for a given image, different strategies for selecting the seed point may induce different partitions but each member of the partition will still be an (ϵ, δ)-RCC of the seed point of the member. In this paper, for convenience, we shall refer to the strategy as $strategy_1$ if we select a seed point p that has the minimal gray value in *buffer*, and as $strategy_2$ if we select p that has the maximal gray value. It is clear that we could find the point which has the lowest or highest gray value in *buffer* in $O(|\ buffer\ |)$ time. Many other strategies, such as finding the point which has the average or medium gray value, could also be adopted with similar run-times.

3.2. Strategies for finding an (ϵ, δ)-RCC

Let p be a given seed point of *buffer* and C_p^{buffer} represent an (ϵ, δ)-RCC of p in *buffer*. We may find C_p^{buffer} as follows: At the beginning, we set C_p^{buffer} as $\{p\}$ since p belongs to C_p^{buffer}. Then, starting from the point p, we shall find all points q in *buffer* such that $q \in C_p^{buffer}$ by using either the *breadth-first search* or *depth-first search* algorithms (see e.g., [14, Chap. 23]). In a breadth-first search, we search the neighbors of only the points that are currently in C_p^{buffer}. After finding all the new points belonging to C_p^{buffer}, we add them into C_p^{buffer} and search the neighbors of the new points, and continue in this manner till no more points can be added. In a depth-first search, we start from a point in C_p^{buffer}; once we find a neighbor of the point which belongs to C_p^{buffer}, we add the neighbor into C_p^{buffer}, keeping track of the other neighbors of the point, and then we turn to search the neighbors of the newly added point. This procedure is repeated till no more points can be added into C_p^{buffer}. Then we select the last added point which has unchecked neighbor point, and repeat the searching procedure as above, until all the points in C_p^{buffer} have been searched. Using the following result (for details see [11]), we can decide whether or not a point q belongs to C_p^{buffer} for a specific point p: *If S is an (ϵ, δ)-connected set of Σ, and p, q are two adjacent points of Σ such that $p \in S$, $q \notin S$, $|\ \sigma(p) - \sigma(q)\ | \leq \delta$, and also any one of the following conditions be satisfied: (1) $max_S \geq \sigma(q) \geq min_S$; (2) $\sigma(q) > max_S$ and $\sigma(q) - min_S \leq \epsilon$; (3) $\sigma(q) < min_S$ and $max_S - \sigma(q) \leq \epsilon$, then $S \cup \{q\}$ is also an (ϵ, δ)-connected set of Σ.*

We shall only give here an algorithm which uses a breadth-first search for finding an (ϵ, δ)-RCC of a seed point p. For implementation purposes, we choose a quadrate grid system to represent an image and use 8-neighbors for considering adjacency of two points (it is straightforward to modify the algorithm for other grid systems or other kinds of adjacency). We shall use two markers for each point q in *buffer*. The marker "$check_q$" represents whether the eight neighbors of the point q have been checked or not. The marker "$included_q$" represents whether the point q has been included in C_p^{buffer} or not. Initially, C_p^{buffer} is empty and both $check_q$ and $included_q$ are FALSE for each q. The basic idea of the algorithm is that, first, we put p in C_p^{buffer}, change $included_p$ to TRUE; while there is an unchecked point q ($check_q$=FALSE) in C_p^{buffer}, then we change $check_q$ to TRUE, and check its eight neighbors (if q is a boundary point, it will have less than eight neighbors). Among the eight neighbors, we add those not-yet-included points (their *included* marker being FALSE) that belong to the considered (ϵ, δ)-RCC into C_p^{buffer}; repeat checking the points in C_p^{buffer} until there is no unchecked point. For convenience, we represent the eight neighbors of a point q by $q_1, q_2, \ldots, q_8$

starting from the upper-left corner of q in a clock-wise manner.

Algorithm 3.2 Let *buffer* be a connected set of points, p be a seed point in *buffer*, ϵ be a given range-parameter and δ be a given adjacency-parameter. The steps of the algorithm to find an (ϵ, δ)-RCC of p are as follows:

0. $C_p^{buffer} = \{\}$;
1. For all $q \in buffer$, $check_q$=$included_q$=FALSE;
2. $C_p^{buffer} = \{p\}$; $included_p$=TRUE;
   $min_{C_p^{buffer}} = max_{C_p^{buffer}} = \sigma(p)$;
3. while ($\exists q \in C_p^{buffer}$ and $check_q$==FALSE)
   {
4. $check_q$=TRUE;
5. Put q's neighbors $q_1, q_2, \ldots, q_8$ in a certain order, say $q_{i_1}, q_{i_2}, \ldots, q_{i_8}$, according to their gray values;
6. for (j=1; j<8; j++)
   if ($included_q$==FALSE)
   if ($max_{C_p^{buffer}} \geq \sigma(q) \geq min_{C_p^{buffer}}$ or
   $\sigma(q) > max_{C_p^{buffer}}$ and $\sigma(q) - min_{C_p^{buffer}} \leq \epsilon$
   or
   $\sigma(q) < min_{C_p^{buffer}}$ and $max_{C_p^{buffer}} - \sigma(q) \leq \epsilon$)
   {
   $C_p^{buffer} = C_p^{buffer} \cup \{q_{i_j}\}$;
   $included_{q_{i_j}}$=TRUE;
   if ($\sigma(q_{i_j}) > max_{C_p^{buffer}}$)
   $max_{C_p^{buffer}} = \sigma(q_{i_j})$;
   if ($\sigma(q_{i_j}) < min_{C_p^{buffer}}$)
   $min_{C_p^{buffer}} = \sigma(q_{i_j})$;
   }
   }

Since $| C_p^{buffer} | \leq | buffer |$ and we change the state of a $check_q$ in each while loop, the while loop in Step 3 can be repeated at most $| buffer |$ times. So, we see that the algorithm will terminate after a finite number of steps. In order to check all the neighbors of q and decide whether they belong to C_p^{buffer}, we need to examine them one by one in a certain order. A seed point p often has more than one (ϵ, δ)-RCC. For a point q, it is possible that two of its neighbors belong to its two different (ϵ, δ)-RCCs respectively, but not belong to a same (ϵ, δ)-RCC. Different methods used to order the neighbors of a point q in Step 5 reflect the different strategies for selecting a specific (ϵ, δ)-RCC of the point q in *buffer*. Thus, different types of order may induce different (ϵ, δ)-RCCs. It would induce a more meaningful result if we make a decision that takes into account the method of selecting the seed point p. For example, since we hope that each object in the image would be partitioned into one (ϵ, δ)-RCC; intuitively, if p has the minimum gray value in *buffer*, we order the neighbors of q by their gray values increasedly. If p has the maximum gray value in *buffer*, then we order the neighbors of q by their gray values decreasedly, and so on. Since there are only a constant number of neighbors for any point q, we can sort them in $O(1)$ time. Steps 0, 2, 4, 5 and 6 can be done in $O(1)$ time and Steps 1 and 3 can be done in $O(| buffer |)$ time. Therefore, Algorithm 3.2 has $O(| buffer |)$ run-time.

4. EXPERIMENTAL RESULTS

We have implemented Algorithm 3.1 by using Algorithm 3.2 in Step 4 of Algorithm 3.1. The experiments on a variety of images were carried on a SUN 4 workstation using the standard *Khoros* image processing environment (see [15] for an overview of Khoros). In our implementation, we use a rectangular grid to represent an image which is an 300×300 array and use the 8-neighborhood of the points to consider the connectivity. Although for randomly chosen ϵ and δ, the (ϵ, δ)-components may not represent the objects in an image, the experimental results show that we could adjust the values of ϵ and δ such that the (ϵ, δ)-components represent the objects reasonably well in many kinds of images. Figures 1 and 2 show some experimental results for different kinds of images. In each output image the white colored curves correspond to the boundaries of the (ϵ, δ)-components. From the experimental results, we infer that if we select the strategy and the parameters ϵ and δ appropriately, the (ϵ, δ)-components of an image may segment the required objects from the background reasonably well. Of course, the appropriate ϵ, δ values would change with the different kinds of images, objects and applications.

5. CONCLUSIONS

In this paper, we have discussed an effective algorithm which partitions a gray image such that each member of the partition is an (ϵ, δ)-component of the image. A practical advantage of the concept of (ϵ, δ)-components of an image is that we may understand the structure of an entire image through the (ϵ, δ)-components, the inter-relationships of the (ϵ, δ)-components, and the relationships of the (ϵ, δ)-components with different values of ϵ and δ, rather than that only of the pixels of the image. Another benefit is that we may understand the content of an image by extracting the target objects in

a certain problem through the (ϵ, δ)-components.

Since we do not make any assumptions about the formation model of the image data, our approach could be applied to different kinds of images formed by optical or non-optical sensors. It is also not difficult to modify the algorithm for images which use different grid systems or use different kinds of adjacency to consider connectedness. Results of experiments on a variety of images have been given. Our technique provides a possible method of transition from low level computer vision to a higher level vision. Since the latter level describes also the inter-relationships of the objects in an image, it helps us to understand the image macroscopically in an easier manner than only through the study of the relationships of the pixels of the image. To summarize, the concept of the (ϵ, δ)-components is a convenient methodology for helping us to analyze a gray image effectively.

6. REFERENCES

[1] A. Rosenfeld. Connectivity in Digital Pictures. *Journal of the Association for Computing Machinery*, **17**:146–160, 1970.

[2] S. B. Gray. Local Properties of Binary Images in Two Dimensions. *IEEE Transactions on Computers*, **c-20**:551–561, 1971.

[3] L. Janos and A. Rosenfeld. Digital Connectedness: An Algebraic Approach. *Pattern Recognition Letters*, **1**:135–139, 1983.

[4] J. P. Mylopoulos and T. Pavlidis. On the Topological Properties of Quantized Spaces (I and II) . *Journal of the Association for Computing Machinery*, **18**:239–254, 1971.

[5] A. Rosenfeld. Adjacency in Digital Pictures. *Information and Control*, **26**:24–33, 1974.

[6] A. Rosenfeld. *Picture Languages. Ch. 2.* Academic Pr., New York, 1979.

[7] A. Rosenfeld. Fuzzy Digital Topology: Introduction and Survey. *Information and Control*, **40**:76–87, 1979.

[8] A. Rosenfeld. Digital Topology. *American Math Monthly*, **86**:621–630, 1979.

[9] A. Rosenfeld. On Connectivity Properties of Grayscale Pictures. *Pattern Recognition*, **16**:47–50, 1983.

[10] S. Yokoi, J. I. Toriwaki, and T. Fukumura. An Analysis of Topological Properties of Digitized Binary Pictures Using Local Features. *Computer Graphics Image Processing*, **4**:63–73, 1975.

[11] Y. Wang and P. Bhattacharya. On Parameter–Dependent Connected Components of Gray Images. *Pattern Recognition*, **29**:1359–1368, 1996.

[12] T. Y. Kong and A. Rosenfeld. Digital Topology: Introduction and Survey. *Computer Vision, Graphics, and Image Process.*, **48**:357–393, 1989.

[13] T. Boult, R. A. Melter, F. Skorina, and I. Stojmenovic. Applications of G-Neighbors to Image Processing and Morphology. *Machine Graphics and Vision*, **4**:39–58, 1995.

[14] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[15] K. Konstantinides and J. Rasure. The Khoros Software Development for Image and Signal Processing. *IEEE Trans. Image Process.*, **3**:243–252, 1994.

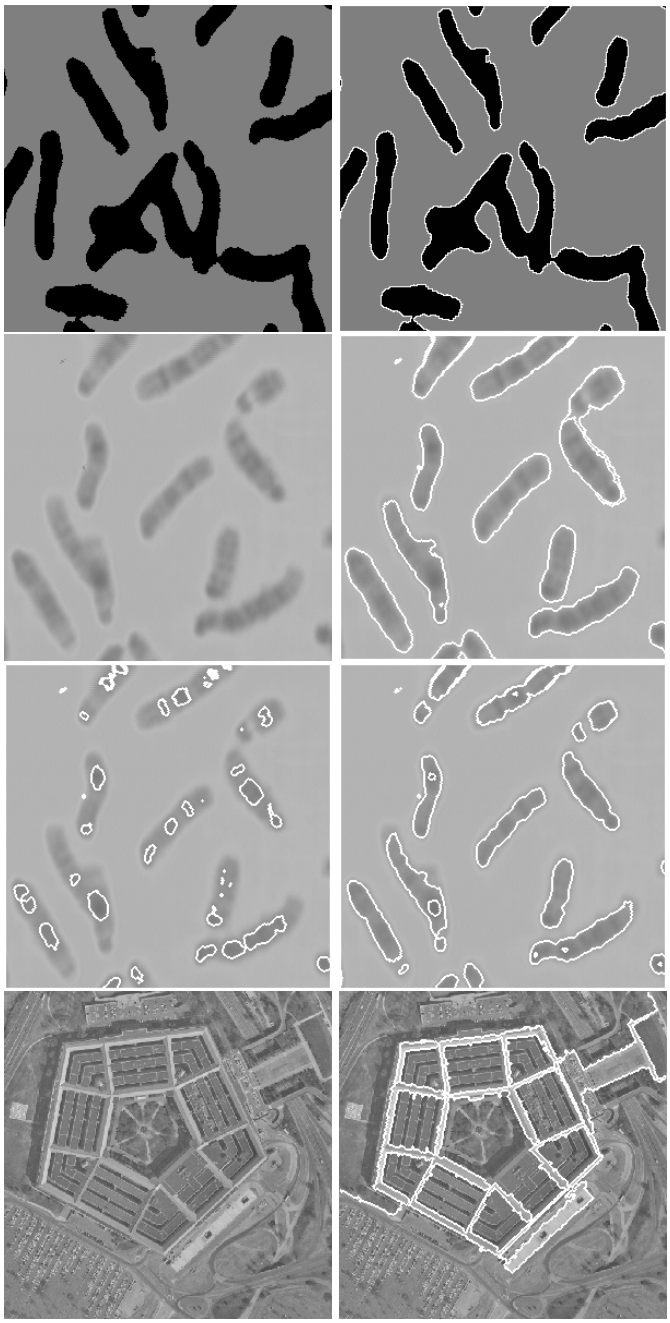

Figure 1: Some examples of (ϵ, δ)-components. Frome top to bottom and left to right: A binary image; Its (ϵ, δ)-components with $\epsilon = 1$, $\delta = 0$. A chromosome image; Its (ϵ, δ)-components by using $strategy_1$ with $\epsilon = 46$, $\delta = 10$; Its (ϵ, δ)-components by using $strategy_2$ with $\epsilon = 46$, $\delta = 10$; Its (ϵ, δ)-components by using $strategy_2$ with $\epsilon = 30$, $\delta = 20$. An original image; Its (ϵ, δ)-components by using $strategy_1$ with $\epsilon = 80$, $\delta = 80$.

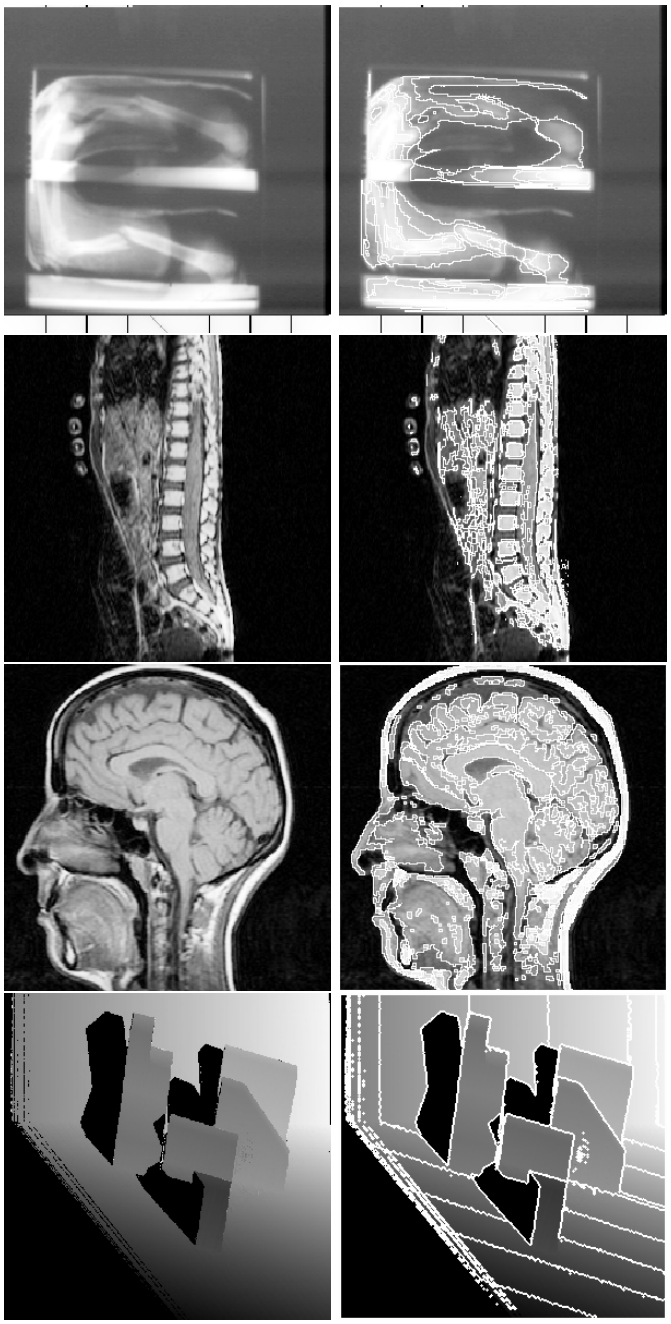

Figure 2: Frome top to bottom and left to right: A X-ray image; Its (ϵ, δ)-components by using $strategy_2$ with $\epsilon = 60$, $\delta = 5$. A magnetic resonance image; Its (ϵ, δ)-components by using $strategy_2$ with $\epsilon = 50$, $\delta = 20$. Another magnetic resonance image; Its (ϵ, δ)-components by using $strategy_1$ with $\epsilon = 50$, $\delta = 20$. A range image; Its (ϵ, δ)-components by using $strategy_1$ with $\epsilon = 30$, $\delta = 15$.