A novel and efficient implementation of the marching cubes algorithm

K.S. Delibasis, G.K. Matsopoulos*, N.A. Mouravliansky, K.S. Nikita

Institute of Communication and Computer Systems, National Technical University of Athens, 9 Iroon Polytechniou Street, Zografos, 15773 Athens, Greece

Received 16 June 2000; accepted 16 October 2000

Abstract

In this paper, a novel and efficient implementation of the marching cubes (MC) algorithm is presented for the reconstruction of anatomical structures from real three-dimensional medical data. The proposed approach is based on a generic rule, able to triangulate all 15 standard cube configurations used in the classical MC algorithm as well as additional cases presented in the literature. The proposed implementation of the MC algorithm can handle the Type A ‘hole problem’ which occurs when at least one cube face has an intersection point in each of its four edges. Theoretical and experimental results demonstrate the ability of the new implementation to reproduce standard MC results, resolving Type A ‘hole problem’. Finally, the proposed implementation was applied to real medical data to reconstruct anatomical structures. The output of the proposed technique is in WWW compliant format.

Keywords: 3-D reconstruction; Marching cubes algorithm; Isosurface; Medical imaging

1. Introduction

Three-dimensional (3-D) medical images are routinely produced in current clinical practice. Anatomical structures can be segmented and identified as a stack of intersections with a number of parallel planes, corresponding to the 3-D image slices. Viewing these slices leaves too much to the viewer’s imagination, if the shape and morphology of the object is to be comprehended. After image segmentation and reconstruction, the structure should be visualized and further manipulated in such a way that realism is maximized. This can be considered as the first step towards Virtual Reality, which is the current trend in medical imaging systems.

The techniques available for reconstructing a structure from 3-D images are based on the following principles:

1. Produce contours of the required object and then solve the triangulation problem using De Launey triangulation—Voronoi diagrams [1] or graph techniques [2].
2. Produce the appropriate triangles from the binary image, working in cubic neighborhoods of (typically) eight voxels (e.g. the marching cubes technique [3]).
3. Face the problem as a functional minimization procedure [4,5].

The first and third techniques require closed, non intersecting contours from a gray scale image, a problem that can be quite difficult to handle, especially in complicated 3-D images where many structures may exist.

2. The standard marching cubes (MC) implementation and the ‘Type A’ hole problem

Given a gray scale 3-D image, the MC algorithm produces an isosurface of value \( t \) [3]. The algorithm operates on a standard length (usually one voxel) cubic region of the image, which occupies eight adjacent voxels. This cubic region will be called ‘the cube’ in this paper. The vertices of the cube are set to \( I \), if the value of the corresponding image voxel is greater than or equal to the threshold \( t \) and \( 0 \) otherwise. The pattern produced in this way will be called ‘cube configuration’ whereas the points that belong to the isosurface will be called isopoints in this paper. The triangles that constitute the isosurface for every cube configuration are predefined.

There are a number of 256 possible cube configurations in each of which, the isosurface is triangulated. The weaving wall algorithm is proposed, in which all 256 cube configurations are explicitly defined [6]. This method, however, is tedious and error prone. In the standard MC implementation, the use of symmetry reduces the number of cases to 15. The complementary symmetry is defined as the equivalence between complementary configurations [3]. Two configurations are defined as complementary, if the action of the logical NOT operator on one of them generates the other.

* Corresponding author. Tel.: +30-1-772-2285; fax: +30-1-772-3557. E-mail address: gmatso@esd.ece.ntua.gr (G.K. Matsopoulos).
The MC algorithm can be summarised in pseudocode as follows:

```
FOR each image voxel
  a cube of length 1 is placed on eight adjacent voxels of the image
  FOR each of the cube’s edge{
    IF (the one of the node voxels has value greater than or equal to \( t \))
    AND the other voxel has value less than \( t \) THEN
      [calculate the position of a point on the cube’s edge that belongs to the isosurface, using linear interpolation]
  }
  FOR each of the predefined cube configurations{
    FOR each of the eight possible rotations{
      FOR the configuration’s complement{
        [compare the produced cube configuration of the above calculated isopoints to the set of predefined cube configurations and produce the corresponding triangles]
      }
    }
  }
}
```

In the case of a binary image with threshold \( t \), the isopoints necessarily lay on the middle of the cube’s edges, for which the two voxel-nodes have values of 0 and 1. When the input to the algorithm is a gray scale image, and the two cube vertices (image voxels) have values \( v_1 \) and \( v_2 \) satisfying the inequality: \((t - v_1)(t - v_2) < 0\), the location of the isopoint is calculated using linear interpolation, along the edge connecting the two voxels. Non linear interpolation models can be also used at the expense of execution time.

The essence of the standard MC algorithm is that for each cube configuration encountered in the image, a match has to be found in the predefined cube configurations, so that the surface triangles can be determined. In order to decide which cube pattern matches the cube in the current image position, the cube is extracted, rotated in both directions by step of 90° and compared to every one of the predefined cube configurations. Fig. 1 shows the 15 cube configurations with the predefined triangulation of the isosurface (Case 0 at the top left and Case 14 at the bottom right) as defined in Ref. [3]. The spheres represent image voxels with values above threshold and the lines the produced triangles.

The classical MC algorithm implementation drawbacks include the occasional ‘hole problem’ [7], as well as great number of produced triangles and computational overhead imposed by the cube rotations. From the programming point of view, this method of predefining cube configurations and their connected triangles is tedious and error prone.

The use of symmetry that reduces the number of cube configurations can produce topologically incoherent surfaces, or ‘holes’ in certain cases of two adjacent cubes. This is indicated as Type A ‘hole problem’ [3,7]. A certain topology problem with an ambiguity in the surface connection is presented in Ref. [7] causing Type A holes to appear in certain pairs of cube configurations. A modified MC algorithm has been developed in Ref. [8] in order to tackle the Type A ‘hole problem’. This approach introduced the problem of ‘inconsistent surface construction’ [9]. Furthermore, a total of 84 cases (Table 1, in Ref. [10]) and six additional cube configurations have been introduced in order to tackle the ‘hole problem’ [7].

3. The proposed algorithm

A novel implementation of the classical MC algorithm is proposed based on a single generic rule capable of generating the triangulated isosurface for all predefined cube configurations. The new MC algorithm generates the resulting triangles in every possible cube configuration, without resorting to any predefined cases. A lookup table of predefined cube configurations is the common element of every variation of the standard MC algorithm. The proposed approach orders the isosurface points directly in polygons rather than triangles, thus produces less triangles.

The isosurface intersects a cube edge only if the two end voxels of the edge have values above and below the value \( t \) of the isosurface. Therefore, an isopoint can only exist on a cube edge, which contains one and only one voxel with gray value above the threshold \( t \). We define this voxel as the isopoint’s associated voxel.

The proposed algorithm starts by finding all the isopoints lying on a cube according to the above criterion and stores them into a list. These isopoints can form up to four different polygons (Case 13, Fig. 1). However, each isopoint has to belong to only one polygon. The isopoints are ordered into a polygon according to the following function, which accepts as argument the current isopoint (\( ip_1 \)) and returns the next isopoint (\( ip_2 \)):

```
function next_isopoint(ip_1) {
  for all unmarked isopoints
    IF isopoint \( ip_2 \) exists such that
      Condition 1: lies on the same cube face with the current isopoint \( ip_1 \) AND
      Condition 2: the current \( ip_1 \) and the next isopoint \( ip_2 \) have the same associated voxel OR
      Condition 3: its associated voxel shares a common cube edge with the current isopoint’s \( ip_1 \) associated voxel OR
      Condition 4: the cube face containing the current and the next isopoint \( ip_1 \) and \( ip_2 \) contains 3 voxels with values above \( t \)
      THEN return \( (ip_2) \)
    ELSE return (null)
}
```
The polygon is traced with the above rule until the initial isopoint is found. The polygon isopoints are marked and stored in a VRML file. The same procedure is repeated until every isopoint of the list is placed into a polygon. Since a triangle is the simplest polygon, the polygon tracing rule always returns more than two isopoints.

The implementation of the proposed algorithm can be described in pseudocode as follows:

Step 0 FOR each image voxel
A cube of length 1 is placed on eight adjacent voxels of the image
FOR each of the cube’s edge
IF (the gray value of one of the edge node voxels is above t and the other below t) THEN

Fig. 1. The triangulation of the 15 predefined cube configurations (Case 0 top left up to Case 14 bottom right), as employed by the standard MC algorithm [3]. The spheres represent image voxels with values above threshold and the lines the produced triangles.
Step 1 (calculate the position of the isopoint on the cube’s edge using linear interpolation; place the isopoint into a list)

\[ p = 0 \]

Step 2 (Scan the list of isopoints until the first unmarked isopoint is found)

IF no unmarked isopoints exist in the list THEN GO TO Step 0
ELSE

\[ p = p + 1 \]
set the first unmarked isopoint as current
mark current – isopoint as belonging to polygon \( p \)
new – isopoint = next – isopoint(current – isopoint)
WHILE (new – isopoint is NULL)
new – isopoint = next – isopoint(current – isopoint)
current – isopoint = new – isopoint
mark current – isopoint as belonging to polygon \( p \)
END
store the polygon into the VRML file
GO TO Step 2
END FOR (each image voxel)

The above algorithm generates all the 15 cases that are predefined by the standard MC algorithm, as it will be shown in the next section [3,7]. Let us verify the proposed algorithm using a rather complicated standard cube configuration, the Case 12, in Fig. 2, as it is predefined in the standard MC implementation. According to Step 1 of the algorithm, the coordinates of the isopoints are calculated and stored in a list, labeled with numbers 1–8. Note that there are no isopoints on cube edges with two or none voxels with values above the isovalue \( t \) (blue spheres). At this step, all isopoints are unmarked. According to Step 2, isopoint 1 is selected randomly from this list. Isopoints 2 and 5 fulfil Condition 1 (lie on the same cube face with isopoint 1) and Condition 2 (the associated voxels of isopoints 1 and 2 and 1 and 5 lie on the same cube edge) of function next – isopoint. It is therefore assumed, without any loss of generality, that isopoint 2 will be the next point of polygon 1. Isopoints 1 and 2 are marked. From the rest unmarked isopoints, only isopoint 3 fulfil the criteria of function next – isopoint, with respect to isopoint 2 (specifically Conditions 1 and 2). Isopoint 3 is marked and fed to function function next – isopoint, which returns isopoint 5 (it complies with Conditions 1 and 3). This isopoint is also marked. Function next – isopoint with isopoint 5 as argument will return NULL, since there are no more isopoints that satisfy the relevant conditions. Therefore, the first polygon is saved into the VRML file as consisting of points \( \{1,2,3,4,5\} \) and the execution of the algorithm jumps to Step 2. The list of isopoints still contains unmarked isopoints (specifically 6,7,8). Therefore the index \( p \) of the polygon is increased by one. Selecting any of them at random as the initial isopoint (e.g. isopoint 6), it can easily be verified that these three isopoints form a triangle (Conditions 1 and 2 of function next – isopoint) and are also marked. For polygon 2 the points \( \{6,7,8\} \) are stored into the VRML file. The next time algorithm execution jumps to Step 2, there are no unmarked points in the list of isopoints. This would mean that there are no more polygons to be drawn at this cube configuration, therefore the algorithm execution jumps to Step 0, or equivalently, the cube marches to the next position in the image and the same process is repeated. The algorithm terminates when the cube has covered the whole image.

It becomes evident that the output is produced in polygons instead of triangles, but with their points equivalently ordered. For instance, the outputs of the standard MC algorithm and the proposed method, for the Case 9, are shown in Fig. 3. The standard MC algorithm produces four triangles whereas the suggested implementation produces a hexagon. Since all the four triangles are coplanar, it is much faster to treat them as a hexagon rather than individual triangles.

The feature of triangle reduction through polygonization is not to be confused with the well established algorithms of triangle/polygon decimation, which are applied after the isosurface has been triangulated [11]. These algorithms are also applicable, after the proposed technique has been completed. The fact that the surface produced by the proposed algorithm comprises of polygons, rather than triangles, has no effect on the decimator, since any triangle merging by vertex or edge elimination produces polygons.

Fig. 2. Verification of the proposed MC algorithm, generating all the necessary polygons of Case 12, of the standard configuration of Fig. 1.

Fig. 3. Output of the standard MC algorithm (a) and the proposed implementation (b) for Case 9 cube configuration of Fig. 1. Note the replacement of the four coplanar triangles by an equivalent hexagon.
In certain cases, the produced polygons may be non co-planar. There is no inherent restriction in VRML specification to handle non co-planar polygons, most VRML viewers can also visualize non co-planar polygons, as it can be seen in Fig. 4, Case 5, 11, 12, 14 etc. (these surfaces were produced by the proposed algorithm in VRML format, as it will be explained in the next section and rendered using standard VRML viewers). If it is required, well established algorithms exist for optimal polygon triangulation [12]. Furthermore, the algorithm can handle cases where more than one polygon is present in the same cube.

The output of the algorithm is in a WWW compliant format, more specifically in VRML 2.0 format [13]. A 3-D structure in VRML format can be rotated and scaled in real time even using hardware as inexpensive as a typical PC. VRML viewers are offered with every computer.

Fig. 4. The triangulation of the 15 predefined cube configurations by the proposed implementation. The display of the triangulated isosurface is in VRML format. The orientation may not always coincide with the one of Fig. 1 to achieve better visualization.
platform, usually with Internet Browsers, allowing interaction with JAVA programs as well as creation of VRML worlds where sensors and multiple users from remote sites can interact, thus enhancing realism.

4. Comparison of the proposed implementation with previous methods

An important advantage of the proposed technique lies on the fact that definition of different cube configurations is not required. Rotations and comparisons between them and the predefined cube configurations that are met in the actual image are no longer necessary. Execution time is therefore reduced. In Fig. 4, the proposed algorithm generates all the
15 predefined cases with the same order as they appear in Fig. 1. The display of the triangulated isosurface is in VRML format. The orientation may not always coincide with the one of Fig. 1 to achieve better visualisation.

As it has already been mentioned, the proposed method does not depend on the complementary symmetry. Thus, by definition, it does not suffer from a specific type of holes that appears due to this symmetry. This surface problem is called the Type A ‘hole problem’ and three extra cases, additional to the 15 standard ones, have been introduced to tackle it [7].

The three extra cube configurations, effectively defining the complementary configurations of Cases 3, 6 and 7, noted as 3(b), 6(b) and 7(b) respectively and can be seen in Fig. 5 (a),(c),(e). The proposed algorithm can generate these additional cases, as it is shown in Fig. 5 (b),(d),(f), respectively.

Furthermore, in Refs. [14,15], six additional predefined cases were introduced to tackle different instances of the Type A ‘hole problem’. These cases are shown in Fig. 6 (Case a to Case f on left side). The proposed algorithm generates these additional cases, as it can be seen in Fig. 6 (Case a–f on right side), in VRML format.

In Figs. 7 and 8, two cases of a Type A ‘hole problem’ are demonstrated. In Fig. 7(a), the standard MC algorithm creates a hole, when the cubes of Case 6 (left side) and Case 3 (right side) are adjacent. The problem is resolved by the introduction of Case 3b, which is the complement of Case 3 shown in Fig. 7(b). (right side) [7]. The proposed algorithm also resolves the Type A ‘hole problem’ as it can be seen in Fig. 7(c). In Fig. 7(c), the polygon ABCD corresponds to hole created in Fig. 7(a). It can be seen that the proposed algorithm resolves the ‘hole problem’. In Fig. 8(a), a similar situation is presented, where the polygon ABCD (shaded area) corresponds to a Type A hole [16]. The problem is resolved using the proposed algorithm, as it can be observed in Fig. 8(b).
5. Experimental results

The proposed algorithm was also able to handle full resolution CT and MRI of human heads, (SGI Indigo 2 with 128 MB RAM). Rendered views produced by the proposed algorithm from 3-D real anatomical data are presented in Fig. 9, after segmentation. Fig. 9(a) shows the 3-D reconstruction of a skull from full resolution CT data of the head, Fig. 9(b) shows the 3-D reconstruction of the bone structure of the same patient, Fig. 9(c) is close up view of the bone structure showing fine details, whereas in Fig. 9(d) the interior of the skull is displayed after a mid-sagittal cut. Finally, Fig. 9(e) shows the largest brain vessels from MRI data whereas Fig. 9(f) displays the lateral ventricles from another MRI brain study.

It is important to note that the proposed algorithm produced VRML files with reduced number of points and polygons. This is considered a great advantage since it allows real time rotation and rendering to be performed in less expensive hardware than the other algorithms.
6. Summary

A novel and efficient implementation has been presented, based on the standard MC algorithm in order to produce triangulated surfaces from volumetric data. The proposed algorithm uses a generic rule capable of generating the correct polygons in any case of cube configurations. It has been shown that the proposed algorithm can reproduce the 15 predefined cases of cube configurations of the standard MC as well as additional cases presented in literature. Furthermore, it does not suffer from the Type A ‘hole problem’, which occurs in the standard MC. The proposed algorithm was implemented in JAVA, generating output files in VRML format from 3-D real medical data; it is therefore suitable for platform independent applications (such as the Telemedicine applications). Future research will be focused on the aspect of reducing the number of triangles produced by the proposed implementation.

References


Konstantinos K. Delibasis was born in Athens in 1967. He received his B. Sc. degree in Physics from the University of Athens, Greece, in 1989, his M.Sc. degree in Medical Physics from the University of Aberdeen and his Ph.D. degree in Medical Imaging from the University of Aberdeen, UK, in 1991 and 1995, respectively. He is currently working as a Research Assistant at the Institute of Communication and Computer Systems, National Technical University of Athens. His research interests include image processing, pattern recognition, optimization techniques and Web based telemmedicine applications.

Nikolaos A. Mouravliansky was born in 1973, in Athens. He received his Diploma in Electrical Engineering in 1996, from the National Technical University of Athens and his Ph.D. in November 2000, from the National Technical University of Athens in medical image processing. His research interests include development of mathematical interpolation techniques, 3-D segmentation, 3-D and 4-D visualization, medical image registration and fusion. He is a member of the Technical Chamber of Greece.

George K. Matsopoulos received his Diploma in Electrical Engineering in 1988 from the National Technical University of Athens. He received his M.Sc. in 1989 and his Ph.D. in Bioengineering Unit in 1993 from the University of Strathclyde, U.K. He is currently working as a Assistant Professor at the Institute of Communication and Computer Systems, National Technical University of Athens. His interests are nonlinear image processing applied to medical applications, 2-D and 3-D registration of medical images and computer vision applications. Dr. Matsopoulos is a member of the IEEE, the Technical Chamber of Greece and the Hellenic Society of Biomedical Engineering.

Konstantina S. Nikita received the Diploma in electrical engineering and the Ph.D. degree from the National Technical University of Athens, Greece, in 1986 and 1990, respectively. She then received the M.D. degree from the University of Athens, Greece, in 1993. Since 1990, she has been working as a Researcher at the Institute of Communication and Computer Systems, National Technical University of Athens. In 1996, she joined the Department of Electrical and Computer Engineering, National Technical University of Athens, where she is currently an Assistant Professor. Her current research interests include applications of electromagnetic waves in medicine, electromagnetic scattering, diffraction tomography, medical imaging and image processing, nonlinear optimization algorithms and applications. Dr. Nikita is a fellow member of IEEE, the Technical Chamber of Greece, Athens Medical Association and Hellenic Society of Biomedical Engineering.