Fast Connected Component Labeling Algorithm: A non voxel-based approach

D. Ayala J. Rodríguez[*]
Universitat Politècnica de Catalunya, Barcelona (Spain)
{dolorsa,jrodri}@lsi.upc.es
A. Aguilera
Universidad de las Américas-Puebla, Puebla (México)
Email: `aguilera@mail.udlap.mx`

Abstract

This paper presents a new approach to achieve connected component labeling on both binary images and volumes by using the Extreme Vertices Model (EVM), a representation model for orthogonal polyhedra, applied to digital images and volume datasets recently. In contrast with previous techniques, this method does not use a voxel-based approach but deals with the inner sections of the object. This approach allows us to build data size-independent algorithms and work with volumes from range data or solid modeling field indistinctly. Our proposal, also can be applied to manifold as well as non-manifold data. The algorithm actually detects non-manifold zones and permits to break or not the objects at these zones by an user-specified parameter.

Keywords

Connected component labeling, volume visualization, image understanding.

1. Introduction

Connected components labeling (CCL) of an image is a fundamental step in the segmentation process and consists in identifying and labeling the separate different regions of interest of the image [RP66]. This operation has applications in the fields of image understanding, volume visualization, character recognition, geometric modeling and computer vision. In particular, we have developed it as a part of a major set of operations to edit and manipulate volume datasets from its EVM-representation. In this context, CCL is useful to identify and separate equal-density parts from a greylevel range data like acquired from medical imaging devices .

Algorithms for 2D images have been developed because, traditionally, data have been obtained as a sequence of 2D slices. However nowadays direct 3D acquisition is becoming more available and consequently direct 3D techniques are necessary.

A great diversity of strategies have been devised to solve this problem: from hardware based solutions [Nic95] to methods using optimization techniques. There are also algorithms that simultaneously perform surface tracking and connected-component labeling [UA91]. However the strategy followed by most developed approaches is based on using a more suitable representation instead of the raw data, in order to improve the corresponding algorithms. In 2D, representations such as run-length and quadtrees [Sam81]

[*]This author has been supported by a grant from FONACIT (Venezuelan Council of Research in Science and Technology)

among others have been proposed.

In 3D there has also been an effort to develop new representations for the data. The semiboundary [UO91], created to store a segmented volume, has become a very useful model in fast visualization, manipulation and analysis of structures generated from volume data and in Thurfjell et al's work [TBN95] a model based on the semiboundary is used for performing erosion, dilation and connected component labeling for 3D volume datasets.

Dillencourt, Samet and Tamminen [DST92] formalized a general approach for connected component labeling which can deal with d-dimensional images represented in several data structures such as arrays, quadtrees and bintrees. Finally, in Oikarinem et al's article [OJH99] an algorithm for volume rendering using seed filling and supporting cut planes (restricted to orthogonal planes) is presented. The algorithm does a 2D connected component search on the cut plane within the volume dataset in order to find seed points, which is based on Dillencourt et al's previous work [DST92].

Almost all of these approaches are voxel-based and therefore their performance decreases as the number of voxels of the image increases.

In this paper we present a new approach for connected component labeling based also on an specific representation of the data. We use the EVM which has been proved to be a useful model for binary data [JD01]. From the object's EVM we obtain, in a natural way, a particular kind of spatial partitioning of the object as an ordered union of disjoint boxes (OUDB). Then we apply, onto this set of boxes, an algorithm which follows the basic two-pass strategy first introduced by Rosenfeld and Pfaltz [RP66].

The presented approach works in identical way for images and volume datasets and it is size-independent. In contrast with previous techniques, it does not use a voxel-based approach but deals with the inner sections of the object. The used model EVM is suitable for manifold as well as non-manifold data and the presented algorithm can detect non-manifold regions and permits to break or not the objects at these zones.

The paper is arranged as follows. Next section includes a review of the EVM in which we give the definitions and properties necessary to understand this work and comments the main results that we have obtained using this model in the field of volume visualization. Section three presents our connected component labeling algorithm, explaining how to obtain the OUDB partition and the connected boxes algorithm. Next, section four shows its performance by discussing experimental results. Finally, the last section concludes the paper and outlines future work.

2. Review of the EVM

Orthogonal polyhedra (OP) are polyhedra with all their faces oriented in three orthogonal directions. Orthogonal Pseudo-polyhedra (OPP) is a more general term which refers to regular and orthogonal polyhedra with a possible non-manifold boundary. Figure 1 shows an OPP with a non-manifold vertex and three non-manifold edges. Binary images and volume data sets are regular grids of samples defining orthogonal shapes. Latecki [Lat97] defines the term well-composed for 3D images and relates it to the manifold concept. So we can say that the continuous analog of a well-composed (non well-composed) image is an OP (OPP).

The *Extreme Vertices Model* (*EVM*) is a very concise model in which any OPP can be described using only a subset of its vertices. The EVM is actually a complete (non-ambiguous) solid model. In this paper we use the EVM to represent images and volume data sets.

Definitions

Let P be an OPP. A *brink* is the maximal uninterrupted segment built out of a sequence of collinear and contiguous two-manifold edges of P. The ending vertices of a brink are called *extreme vertices* (EV). See Figure 1. The *EVM* is a model in which any OPP is described by its (and only its) set of EV.

A *plane of vertices* (*plv*) is the set of vertices lying on a plane perpendicular to a main axis of P. A *slice* is the region between two consecutive planes of vertices. A *section* (S) is the resulting polygon from the intersection between P and an orthogonal plane. Each slice has its representing section. See Figure 1.

All these definitions can be extended to any dimension [BMP99]. In this paper we are concerned with dimension ≤ 3. Planes of vertices and sections obtained from a 3D object are, then, 2D orthogonal polygons. From them, we can obtain their 1D lines of vertices and their 2D slices with their corresponding 1D sections. Finally, lines of vertices and 1D sections are 1D objects which are composed of one or several brinks.

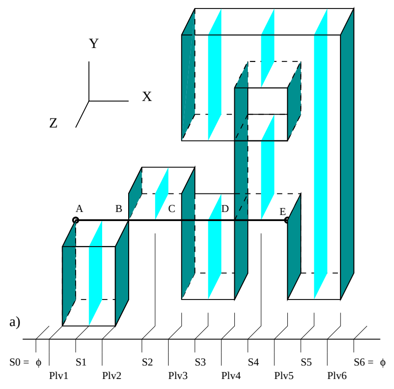


Figure 1: An OPP with a marked brink from vertex A to E (vertices B,C and D are non-extreme). Its planes of vertices and sections perpendicular to the X axis are shown in dark and light grey respectively.

In the EVM model the set of EV can be ordered in six possible ways depending on the coordinate values: XYZ, XZY, YXZ, YZX, ZXY, and ZYX. In an XYZ ordered EVM, planes of vertices perpendicular to the X axis appear ordered from low to high x values and, in each of them, lines of vertices parallel to the Y axis appear also ordered from low to high y values.

Properties of the EVM

All the properties mentioned are proved in [AA98].

Sections can be computed from planes of vertices and vice-versa:

$$S_0(P) = S_n(P) = \emptyset;$$

$$\overline{S_i(P)} = \overline{S_{i-1}(P)} \otimes^* \overline{plv_i(P)}, \quad i = 1 \ldots n \tag{1}$$

$$\overline{plv_i(P)} = \overline{S_{i-1}(P)} \otimes^* \overline{S_i(P)}, \quad i = 1 \ldots n \tag{2}$$

where $\overline{plv_i(P)}$ and $\overline{S_i(P)}$ denote the projections of $plv_i(P)$ and $S_i(P)$ onto a main plane parallel to P, n is the number of planes of vertices and $\otimes^*$ denotes the regularized XOR operation. Note that in order to operate with the projections we need not take into account the coordinate of the extreme vertices that corresponds to the projecting plane.

The following property concerns the XOR operation:

Let P and Q be two d-D (d $\leq$ 3) OPP, having $EVM(P)$ and $EVM(Q)$ as their respective models, then,

$$EVM(P \otimes^* Q) = EVM(P) \otimes EVM(Q) \tag{3}$$

This property means that the XOR operation works in 0D, because it is applied to the EV of the model. Therefore, sections are obtained from planes of vertices and vice-versa by applying the XOR operation to the extreme vertices.

The following property is a corollary of the previous one:

if P and Q are quasi-disjoint, then

$$EVM(P \cup^* Q) = EVM(P) \otimes EVM(Q) \tag{4}$$

Contributions of the EVM

The EVM was first introduced as a solid model [AA98], but now we are studying its application in the field of volume rendering and image processing and we have obtained some results which we summarize in this section.

A boundary extraction algorithm has been devised consisting of two steps: a conversion from a voxelization to the EVM [JAD00] and a conversion from the EVM to a hierarchical boundary representation (B-Rep) which avoids the extreme redundancy of primitives. The obtained B-Rep model is composed by a relatively few large orthogonal faces instead of a huge number of little triangular or rectangular faces, as in marching cubes (MC) like approaches [LC87], or as in other block-form approaches as the semiboundary representation (SB) [UO91]. For several examples, the number of faces obtained with the EVM were about 1/6 of the number of faces obtained with MC and 1/3 of the number of faces obtained with SB [JAD00].

Also in the same work [JAD00] several compression techniques applied to EVM have been presented and evaluated, for storage and transmission purposes.

Finally, we published recently a new method for erosion and dilation using the EVM [JD01]. This approach works in identical way in images and volumes and is independent of the size both of the image and the structuring element. Related operations as opening and closing have also been implemented and proved.

3. EVM-based connected component labeling algorithm

In this section we present our connected component labeling algorithm(EVM-CCL). Our proposal is based in [TBN95] but using the EVM model and its particular partitioning. It works in identical way for images and volumes, is size-independent, reduces the number of labels and equivalences generated in the first pass and deals indistinctly with manifold and non-manifold data.

3.1. Decomposing EVM into an OUDB

As said in the introduction, we first obtain a particular partitioning straightforward from the EVM which consists in an ordered union of disjoint boxes (OUDB).

This partitioning can be thought as a special kind of cell decomposition like octrees, bintrees [DST92] and BSP [SN97] which have been used for representing volume datasets. The OUDB is axis aligned as octrees and bintrees but the partition is done along the object geometry like BSP.

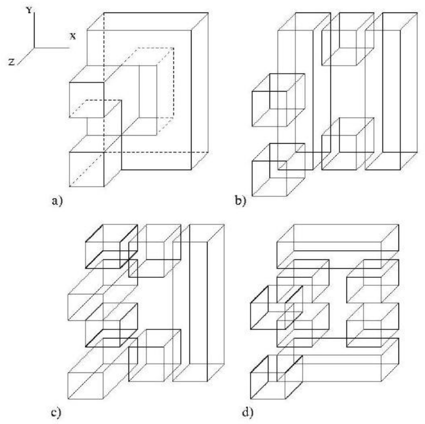


Figure 2: a) An OPP. b) XZ decomposition (6 boxes). c) XY decomposition (7 boxes). d) YZ decomposition (8 boxes).

We obtain the OUDB from EVM by two steps. First, we split the data at every plane of vertices perpendicular to a main axis, say X for instance, obtaining a set of slices. Secondly, we split each slice at every plane of vertices perpendicular to another main axis, say Y, obtaining a set of boxes for each slice. All the obtained boxes constitute the OUDB partitioning. An object can be decomposed into six different sets of boxes depending on the axes we choose to split the data: XY, XZ, YX, YZ, ZX, ZY. Figure 2 shows an object and the XZ, XY and YZ decompositions. Procedure EVMtoOUDB is as follows:

```
procedure EVMtoOUDB (Input p:EVM Input dim:integer Input PCoord,LCoord:integer I/O q: OUDB)
    var br: Brink; S1, S2, plv: EVM endvar
    if dim = 1 then ReadBrink(p, br)
        while ¬ EndEVM(p) do
            AppendBox(q, br, PCoord, LCoord);
            ReadBrink(p, br);
        endwhile
    else dim ≠ 1
        dim:= dim - 1
        S1:= ∅;
        plv:=ReadPlv(p, dim)
        while¬EndEvm(p) do
            S2:= S1 ⊗ plv;
            GetCoord(plv, dim, PCoord, LCoord)
            EVMtoOUDB(S2, dim, PCoord, LCoord, q)
            S1:= S2; plv:=ReadPlv(p, dim)
        endwhile
    endif
endprocedure
```

This procedure uses the basic operations of EVM: $ReadPlv$ and $EndEVM$, which allow us to traverse the EVM. The first one returns the current plane (when $dim = 2$) or line (when $dim = 1$) of vertices of p whereas the another one detects the end of p. $GetCoord$ gets the plane coordinate, $PCoord$ ($dim = 2$), or the line coordinate, $LCoord$ ($dim = 1$). The XOR operation between a section and a plane or line of vertices produces the next section (see equation 1). $ReadBrink$ obtains a brink from a 1D EVM (which consists of a set of collinear brinks) and $AppendBox$ builds a box using the brink br and the values $PCoord$ and $LCoord$ and appends it to the OUDB q.

As most of the algorithms dealing with EVM, this one is also recursive in the dimension. When $dim = 3$, all the 2D sections are computed from the planes of vertices of the model. For each section the procedure is called recursively (then, $dim = 2$) and all the 1D sections are computed from their lines of vertices. Finally, the recursion reaches the base case when $dim = 1$ and each 1D section is processed.

Obtaining the EVM from the OUDB is a very simple method that consists of doing the Xor operation among all the boxes, applying property 4.

```
procedure OUDBtoEVM(Input q:OUDB I/O p:EVM)
    var evq: EVM endvar
    p:=IniEVM()
    for i in [1..NumberOfBoxes(q)] do
        evq:= $BoxToEVM(q_i)$; p:= p $\otimes$ evq
    endfor
endprocedure
```

BoxToEVM converts a box to its EVM representation by putting its 8 vertices in an ordered way.

3.2. Connected boxes labeling algorithm

Once the OUDB partition has been achieved the connected boxes labeling process (CBL) takes place. Previous works use a sequential two-pass strategy which works at voxel-level , [KR89] and [TBN95]. In the first pass all voxels are labeled but there remain some unsolved label equivalences which are sorted out in a second pass. Thurfjell et al.'s algorithm [TBN95] improved this technique by working with the semi-boundary representation of the data, so they only label voxels belonging to the boundary of the object. The method we propose follows the classical two-pass approach (labeling and renumbering pass), but it traverses the set of disjoint boxes(OUDB) instead of a set of boundary voxels, further diminishing the number of elements to label.

3.2.1. The labeling pass

Suppose an OUDB partition obtained by splitting first at the X coordinate (X-slices) and then each X slice at the Y coordinate (XY slices). First, a label is set to the first box. Then, the remaining boxes are traversed and each of them is labeled according to a subset of the previous boxes: its neighbor boxes. This neighborhood includes those boxes in the immediate previous XY-slice and those boxes in the immediate previous X-slice. See algorithm.

```
procedure CCL (I/O obj: OUDB)
    var i, j: integer; mark: boolean endvar
    FirstLabel(obj, 1)
    for i in [2..BoxesNumber(obj)] do
        j:= i-1; mark:=false
        {Skip boxes in the same slice XY}
        while j $\geq$ 1 $\wedge$ SameSliceXY(obj, i, j) do
            j:=j-1
        endwhile
        {Compare with boxes in the previous slice XY}
        while j $\geq$ 1 $\wedge$ SliceXYadjacent(obj, i, j) do
            if NeighboursXY(obj, i, j) then
                AssignLabel(obj,i,Label(obj,j),mark)
                SetEquivalence(EqTable, Label(obj,i), Label(obj,j))
            endif
            j:= j-1
        endwhile
        {Skip boxes in the same slice X and in slices Y
        not adjacents}
        while j $\geq$ 1 $\wedge$ SameSliceX(obj, i, j) do
            j:= j-1
        endwhile
        {Compare with boxes in the previous slice X}
        while j $\geq$ 1 $\wedge$ SlicesXadjacent(obj, i, j) do
            if NeighboursX(obj, i, j) then
                AssignLabel(obj, i, Label(obj, j), mark)
                SetEquivalence(EqTable, Label(obj,i), Label (obj, j))
            endif
            j:= j-1
        endwhile
        if $\neg$mark then NewLabel(obj, i, EqTable) endif
        {End of subset of boxes to compare with box i}
    endfor
endprocedure
```

AssignLabel assigns to box i the label of box j and *NewLabel* assigns a new label to box i.

Figure 3 illustrates the labeling process in 2D, straightforward generalization to 3D is possible. On the figure, the image's EVM has been obtained and its OUDB partition has been computed (a top to down and left to right ordered list). At the begining all the boxes in the list are unlabeled. Then the first box is labeled with 1, afterwards the labeled neighborhood (from previous boxes in the list) of the second box is checked. Any neighbor box is found, so this box is set with a new label: 2. Next the third box is checked. Testing previous boxes in the list (first and second one) only the first one is neighbor to it, then the third box is set to the same label than the first one. This process continues until all the boxes are labeled.

3.2.2. Setting equivalences

In the labeling pass, label equivalences can be detected. When a box i has two or more neighbors labeled with different labels, there is an equivalence between two or more regions. These label equivalences are saved in an *equivalence table*. Following the same labeling strategy proposed by Thurfjell et al. [TBN92], this table has an entry for each label. The index corresponds to the region number and the content to its label. When a new label is set, an element is added to the table with the same value for the region number and label. Procedure *SetEquivalence* updates this table in the way shown below:

```
procedure SetEquivalence(I/O EqTable:Table, Input label1,label2: inetger)
    if EqTable[label2]=label2∨EqTable[label2]=label1 then
        EqTable[label2]=label1
    else
        if EqTable[label2] < label1 then
        SetEquivalence(EqTable,EqTable[label2],label1)
        else {EqTable[label2]>label1}
            SetEquivalence(EqTable,label1,EqTable[label2])
        endif
    endif
endprocedure
```

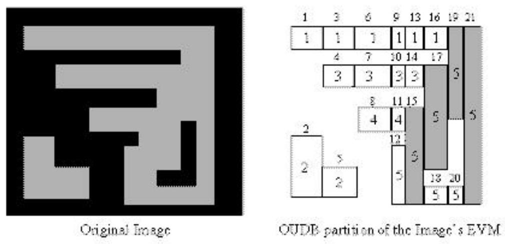
Figure 3 illustrates the progressive updating of the Equivalences table. Each time an equivalence is detected the equivalence table is updated to guarantee an ordered storing of nested equivalences.

3.2.3. The renumbering pass

The renumbering sorts out all the equivalences saved in the labeling pass. Setting equivalences in an order way, as explained above, has the advantage that it makes the renumbering easier. As the equivalences have been saved in increasing order, the nested equivalences are linked in increasing order too. So, the renumbering pass is just a traversal of the table and when $EqTable[i] \neq i$ then $EqTable[i] \leftarrow EqTable[EqTable[i]]$ which has been processed previously. At the end, the nested equivalences will be propagated correctly and all the regions with equivalent labels will have the same label. The main drawback with this strategy deals with the size that the equivalences table can reach because we need one entry per each new detected label. However, our proposal reduces considerably the number of needed labels as we will prove later.

Figure 3 shows the final state of the equivalences table after the renumbering pass.

Finally figure 4 shows the three steps carried out to achieve the EVM-based connected component labeling algorithm in 2D. First, the EVM is extracted from the input data and its OUDB partition is obtained (each grey level is a box), then the labeling of boxes is made and finally the equivalences are solved and the connected components stored as separated objects (each one with its own EVM).



Labels Table

Box	1	2	3	4	5	6	7	8	9	10	11	12	13
Label	1	2	1	3	2	1	3	4	1	3	4	5	1

14	15	16	17	18	19	20	21
3	5	1	5	5	5	5	4

Equivalence Table: Setting equivalences

Region	1	2	3	4	5
Label	1	2	3	4	4

Region	1	2	3	4	5
Label	1	2	3	3	4

Region	1	2	3	4	5
Label	1	2	1	3	4

Renumbering

Region	1	2	3	4	5
Label	1	2	1	1	1

Figure 3: Up:Original image and its labeled OUDB. Equivalences detection in dark grey boxes. Bottom: Successive Equivalence tables. Each dark grey cell in the Labels Table produces an equivalence to be saved in the Equivalence Table.

3.3. Two-manifold property and other applications

When comparing two boxes, we can maintain or not the non-manifold zones. If two boxes share a face or part of a face, then they will have the same label. If they are disjoint, then the new box will have a new label. But if they share an edge, a part of an edge or a vertex, then they can have the same label or not. In the former case, the algorithm will maintain the non-manifold zones. In the latter case, the algorithm will break the object at these zones and the obtained connected components will be 2-manifold. These cases are related to 6 and 26-adjacency respectively using the terminology of voxel based approaches.

A box is represented by the two ending vertices of its main diagonal and the $NeighboursXY$ and $NeighboursX$ functions perform comparisons between the corresponding coordinates. When these

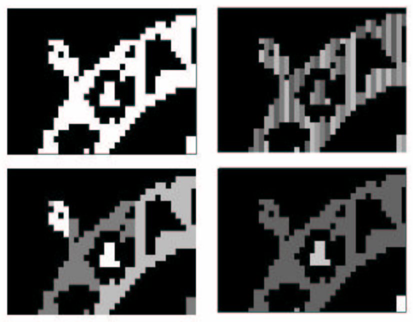


Figure 4: up-left: Original image, up-right: OUDB decomposition, bottom-left: OUDB labeled before renumbering pass and bottom-right: four connected components as separated objects.

comparisons include the equality, non-manifold zones are maintained, otherwise the object is broken at these zones guarantees the two-manifold property for each component. In our implementation this choice is defined by an user-specified parameter.

Figure 5 shows a simple 3D dataset with its components as 2-manifold or non-manifold objects. Each component with a different greylevel.

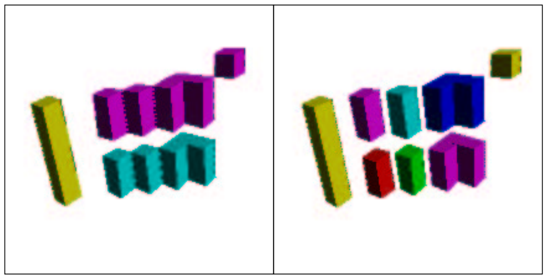


Figure 5: EVM-CCL algorithm applied to a 3D data set with non-manifold detection(right) and without non-manifold detection(left)

Once we have each component we can compute its volume and reject it if this volume is smaller than a specified threshold. Other integral properties like the center of gravity or moments of inertia can be computed as sums of each box contribution for each component [Hof89].

4. Analysis of the algorithm and experimental results

The obtention of the OUDB partition from the EVM of the input data is a needed preprocess for the method. As most of the algorithms concerned with EVM it needs to compute sections. This computation has a quadratic worst case complexity, but the average experimental complexity is slightly greater than linear $nv^{1.2}$, nv being the number of extreme vertices.

The complexity of the EVM-CCL algorithm is $O(nb)$, nb being the number of boxes of the OUDB representation. This number depends on the decomposition (see Figure 2) but we cannot know it a priori from the EVM model. The renumbering pass (as the Thurfjell's algorithm) is $O(nl)$, nl being the number of labels generated in the labeling pass. So our method will have better performance in images(volumes) where the boxes decomposition produces fewer boxes than boundary pixels(voxels). This is true in the most of cases except in extremely sparse datsets where almost each pixel corresponds to a box.

Now we present some results. Our test set consist in two images and a volume dataset (figures 6, 7 and 8 respectively). The algorithms have been implemented in C on a Sun Microsystem ultrasparc60 machine.

Table 1 shows the dimensions of the datasets and the time taken for the OUDB partition (in seconds). The other items on the table are the number of components observed with and without non-manifold detection (on the table 4{6}-ady and 8{26}-ady respectively), and the difference between the number of boundary pixels(or voxels) and the number of boxes resulting by the OUDB decomposition (#bound and #boxes respectively).

Datasets	logoUPC (324x332)	Magallanes (400x159)	pieces (96^2x69)
4(6)-ady	16	29	78
8(26)-ady	16	18	56
#bound	4324	4662	1160
#boxes	177	1113	524
OUDB time	0.01	0.05	0.08

Table 1: Description of datasets

The next table presents per each dataset described in the previous table a comparison between the average execution time using a voxel-based algorithm and with our EVM-CCL.

Datasets	logoUPC	Magallanes	pieces
voxel-based:			
Labeling	0.15	0.06	0.29
Renumbering	0.06	0.03	0.3
Total	0.24	0.1	0.8
EVM-CCL:			
Labeling	0.0001	0.02	0.03
Renumbering	0	0.0001	0.0001
Total	0.01	0.07	0.11

Table 2: Processing time statistics (secs.)

As we said above, our algorithm is image-size independent because its performance is related with the number of boxes and it depends on the orthogonality of the object. To prove this feature the size of the former datasets was augmented by two maintaining the same shape. Table 3 shows the EVM-CCL average execution time (CBL on the table) obtained for these doubled size datasets. Note that the number of boxes remains unchanged, then the time is the same too, instead the number of boundary pixels(or voxels) increases significatively.

1:2-scaled Datasets	logoUPC (648x664)	Magallanes (800x318)	pieces (192^2x138)
#bound	8708	9748	119072
#boxes	177	1113	8414
CBL time	0.01	0.07	0.11

Table 3: Processing time of double size datasets

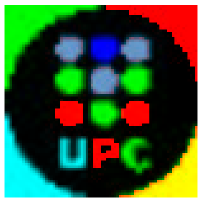


Figure 6: Connected components of 'logoUPC'.

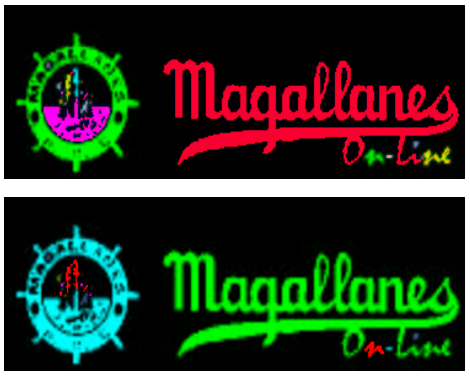


Figure 7: 'Magallanes' with and without non-manifold detection

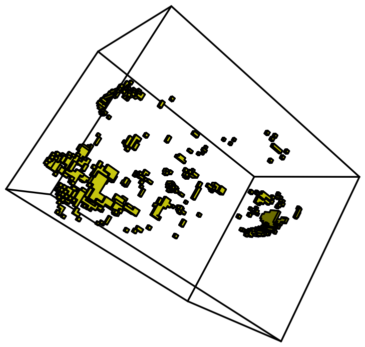


Figure 8: CT skull splitted in several small pieces

Finally, the EVM-CCL algorithm allows significative reductions on the labels and equivalences generated through the labeling pass. It means reduction on setting equivalences time and saving of memory requirements. Figure 9 illustrates the conventional labeling pass (left) and the boxes labeling pass (right). Both images have been processed in the same order, top to down and left to right. Each level of grey represents a different label and you can note how the left image has more labels than the right one. It is possible because the OUDB partition allows us to check adjacency along of many pixels(voxels) at the same time. So, corner pixels, which generates new labels in voxel-based approach, are just a piece of a large box in EVM-CCL. We can check this property in the OUDB partition shown in figure 4. Thus, table 4 presents the number of labels and equivalences generated by two different datasets and figure 9 illustrates one of these cases (Detail).

Datasets	voxel-based		EVM-CCL	
	Labels	Equival.	Labels	Equival.
Detail	29	25	7	3
logoUPC	50	33	16	0
Magallanes	134	115	57	28
pieces	283	69	76	20

Table 4: Labels and equivalences generated in labeling pass

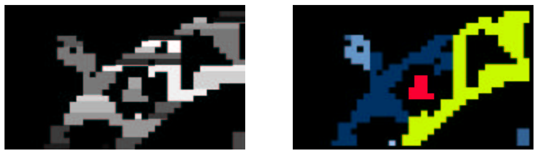


Figure 9: Generation of Equivalences. voxel-based method vs. EVM-based method. Each change of color on the same component indicates an equivalence

5. Conclusions and future work

We have applied the classical two-pass approach to obtain the connected components of an image or volume by using the EVM. The EVM is decomposed as a particular set of disjoint boxes(OUDB) profiting the EVM's knowing about the object geometry. Each box contains many voxels so we can label large regions in only one step. Our approach is not voxel-based but deals with boxes. It produces less labels and equivalences and is faster than the voxel-based ones. It also works in identical way for images and volume datasets with no overhead of memory and it is size-independent. Finally, it also can detect non-manifold zones and separate the object at them. As a future work, we are studying thinning methods, the detection of holes and computation of the genus of 2D and 3D images and other operations using the EVM. Furthermore we are trying to extend it in order to represent and operate with multimodal images.

6. Acknowledgments

This work has been partially supported by a CICYT grant TIC99-1230-C02-02.

References

[AA98] A. Aguilera and D. Ayala. Domain extension for the extreme vertices model (EVM) and set-membership classification. In *CSG'98. Ammerdown (UK)*, pages 33 – 47. Information Geometers Ltd., 1998.

[BMP99] O. Bournez, O. Maler, and A. Pnueli. Orthogonal Polyhedra: Representation and Computation. In *Hybrid Systems: Computation and Control*, LNCS 1569, pages 46 – 60. Springer, 1999.

[DST92] M.B. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labeling for arbitrary image representations. *Journal of the ACM*, 39(2):253 – 280, 1992.

[Hof89] C. M. Hoffmann. *Geometric and Solid Modeling*. Morgan Kauffmann Publishers, Inc., 1989.

[JAD00] J.Rodrguez, A.Aguilera, and D.Ayala. Obtencin de la frontera de una imagen digital 3D usando el modelo EVM. In *CEIG 2000*, pages 203–214, 2000.

[JD01] J.Rodrguez and D.Ayala. Erosion and dilation on 2d and 3d digital images: a new size-independent approach. In *Vision modeling and visualization 2001 (VMV'01)*, pages 143–150, 2001.

[KR89] T. Kong and A. Rosenfeld. Digital topology: Introduction and survey. *Computer Vision, graphics and Image Processing*, 48:357 – 393, 1989.

[Lat97] L. Latecki. 3D Well-Composed Pictures. *Graphical Models and Image Processing*, 59(3):164 – 172, 1997.

[LC87] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surfaces construction algorithm. *Computer Graphics*, 21(4):163 – 169, 1987.

[Nic95] C. J. Nicol. A systolic approach for real time connected component labeling. *Computer Vision and Image Understanding*, 61(1), 1995.

[OJH99] J. Oikarinen, L. Jyrkinen, and R. Hietala. Volume rendering using seed filling acceleration: Supporting cut planes by fast reseeding. *Computer-aided Surgery*, 4(4), 1999.

[RP66] A. Rosenfeld and J.L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM*, 13(4):471 – 494, 1966.

[Sam81] H. Samet. Connected component labeling using quadtrees. *Journal of the ACM*, 28(3):487 – 501, 1981.

[SN97] K. R. Subramanian and B. F. Naylor. Converting Discrete Images to Partitioning Trees. *IEEE Transactions on Visualization and Computer Graphics*, 3(3), 1997.

[TBN92] L. Thurfjell, E. Bengtsson, and B. Nordin. A new three-dimensional connected component labeling algorithm with simultaneous object feature extraction capability. *CVGIP:Graphical Models and Image Processing*, 54(4):357 – 364, 1992.

[TBN95] L. Thurfjell, E. Bengtsson, and B. Nordin. A boundary approach to fast neighborhood operations on three-dimensional binary data. *CVGIP: Graphical Models and Image Processing*, 57(1):13 – 19, 1995.

[UA91] J. K. Udupa and V. G. Ajjanagadde. Boundary and object labeling in three-dimensional images. *Computer Vision, Graphics and Image Processing*, 51(3):355 – 369, 1991.

[UO91] J. Udupa and O. Odhner. Fast visualization, manipulation and analysis of binary volumetric objects. *IEEE Computer Graphics and Applications*, 11(6):53 – 62, 1991.