6D-1

Symmetry Identification of Objects Represented by Octrees

Predrag Minovic, Seiji Ishikawa and Kiyoshi Kato
Kyushu Institute of Technology

1. Introduction

In this paper the algorithm for identifying symmetry of a 3D object represented by its octree is described and a symmetry degree, *i.e.* the measure of object symmetry, is introduced. An object can be at an arbitrary position and with arbitrary orientation within the octree space and all types of symmetry, bilateral, axial and point symmetry can be identified.

2. The algorithm outline

Several methods for detecting symmetry of planar images have been reported (see [3] for one approach and related bibliography) but they do not seem easily extendible to the 3D case. If an object is given by its octree, and if it is centered and conveniently oriented in the octree space, the symmetry detection is straightforward (extension of [1]), but this rarely happens in practice. To overcome this problem, object transformation invariant to its position and orientation is necessary. The "principal axes transformation" has this property, it is easy to compute [2], and it facilitates symmetry identification because of the following facts from elementary mechanics [7]:

Theorem 1: Any plane of symmetry of a body is perpendicular to a principal axis.

Theorem 2: Any axis of symmetry of a body is a principal axis.

Since these theorems provide sufficient but not necessary conditions for symmetry, we must perform explicit symmetry check on the transformed octree.

An octree can be defined as a hierarchical data structure which represents the space occupied by an object as a juxtaposition of cubes obtained by recursive, regular decomposition of the 3D space (Fig. 1) using a certain "leaf criterion" as a stopping rule (see [5, 6] for detailed survey). The simplest criterion is to maximize the depth of decomposition introducing the octree resolution. Octree cubes are then classified as BLACK and WHITE (representing presence and absence of the object, respectively), and GRAY or mixed octree nodes.

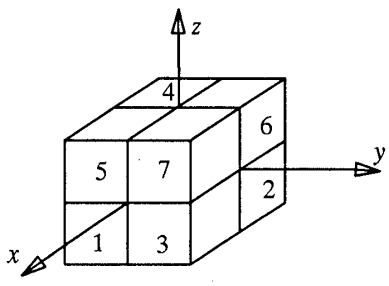

Figure 1: Octree and its subdivision (node 0 is occluded).

It is assumed that for a given object its octree representation is available (for octree construction techniques, refer to [4, 6]). Now we formulate our algorithm as follows:

Step 1: Compute the mass, centroid and inertia matrix for the object given by its octree. Solve the eigenvalue problem for the inertia matrix and compute its eigenvalues and eigenvectors. The latter represent the principal axes of the object.
Step 2: Align the octree from the original coordinate system to the new one, using the centroid as the origin and the principal axes as the new coordinate axes, and build, so called, a "principal octree".
Step 3: Examine appropriate octree nodes and check for bilateral, axial or point symmetry. Compute the corresponding symmetry degree.

In the sequel, these steps will be further elaborated:

2.1. Object's mass, centroid and inertia matrix can be expressed by the means of its moments, and in the case of octree they can be easily computed since its building elements are only cubes (of different size). Refer to [2, 7] for concrete formulae. Here we only note that since an inertia matrix is symmetric, its eigenvalues are real, and corresponding eigenvectors are orthogonal.

2.2. Octree alignment to the new coordinate system is done in two steps: (1) computing the transformation matrix for translation to the centroid and 3D rotation; and (2) octree transformation. For step (2) we implement the method from [8]. It introduces, so called, quantization error which is unavoidable because of the nature of the octree. In order to reduce this error we use higher resolution for the output octree (*e.g.* if the output octree resolution is restricted to r=7, since for higher values the number of nodes will increase too much, the input resolution has to be 5 or 6).

2.3. The measure of symmetry is expressed by the size of the symmetric subset, *i.e.* symmetry degree (sd) is defined as the ratio of the masses of the symmetric subset and the whole object. According to the type of

symmetry we are checking for, four subtrees are extracted and traversed. For each of their BLACK nodes we try to locate its symmetric "brother" using symmetry mapping tables. For example, for locating symmetric nodes with respect to the z-axis (Fig. 1), we extract subtrees 0,1,4,5 and use the table bellow:

d	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
s(d)	*3*	*2*	*1*	*0*	*7*	*6*	*5*	*4*

Actually we simultaneously traverse pairs of symmetric subtrees (*e.g.* 0 and 3, 1 and 2 *etc.*) and whenever we turn in direction *d* on one subtree, we proceed in direction *s(d)* on the other one. As the result the time complexity of this step is equal to the octree traversal time. This method is simpler and faster than [1] where "classes of symmetry" are used.

Furthermore the uniqueness of the eigenvalues can be used as a hint in deciding for which of 7 possible types of symmetry (with respect to 3 planes, 3 axes and the origin) to check first. This relationship can be summarized in the following way:

a) all 3 eigenvalues equal — point symmetry;

b) 2 eigenvalues equal — axial symmetry with respect to the axis corresponding to the unique eigenvalue;

c) all eigenvalues unique — bilateral and point symmetry.

3. Results

In Fig. 2, one test object is shown before and after the transformation together with the program report. Both bilateral and point symmetry are identified. The object is represented by the octree with resolution r=3 on the input, and r=6 on the output. For this synthetic object without noise, the symmetry degrees equal 1 which means that quantization error is "well distributed" in space. However in the presence of noise, a certain threshold value (th) has to be chosen, and symmetry is claimed if sd > th.

Since for steps 1 and 3 octree traversal is done, their time complexity is O(N), where N is the number of nodes in the input octree. Thus, average execution time of our algorithm equals the time necessary for step 2, which is reported [8] to be O(rN), where r is the resolution of the output octree.

4. Conclusion

The algorithm for symmetry identification of a 3D object represented by its octree is given, and some of its aspects are discussed. It can be easily adjusted for 2D images represented by quadtrees.

References

1. N. Alexandridis and A. Klinger, Picture decomposition, tree data structures and identifying directional symmetries as node combination, *CGIP* **8**, 1978, 43–77.

2. C. H. Chien and J. K. Aggarwal, Identification of three dimensional objects from multiple views using quadtrees / octrees, *CVGIP* **36**, 1986, 256–273.

3. G. Marola, On the detection of the axes of symmetry of symmetric and almost symmetric planar images, *IEEE Trans. PAMI* **11**, 1989, 104–108.

4. H. Noborio et al., Construction of the octree approximating three dimensional objects by using multiple views, *IEEE Trans. PAMI* **10**, 1988, 769–782.

5. H. Samet and R. E. Webber, Hierarchical data structures and algorithms for computer graphics, Part I: Fundamentals, *IEEE CG&A* **8**(3), May 1988, 48–68.

6. ——, Hierarchical data structures and algorithms for computer graphics, Part II: Applications, *IEEE CG&A* **8**(4), July 1988, 59–75.

7. K. R. Symon, *Mechanics*, Chapter 10, Addison-Wesley, Reading, Mass., 1965.

8. J. Weng and N. Ahuja, Octrees of objects in arbitrary motion, *CVGIP*, **39**, 1987, 167–185.


```
# 1072 nodes after compaction
bilateral symmetry:
  plane normal (0.707107, 0.707107, 1.000000), sd=0.641811
  plane normal (-0.707107, -0.707107, 1.000000), sd=0.641811
  plane normal (1.000000, -1.000000, 0.000000), sd=1.000000
point symmetry, sd=1.000000
all coordinates with respect to original centroid (8.000000, 8.000000, 12.000000)
```

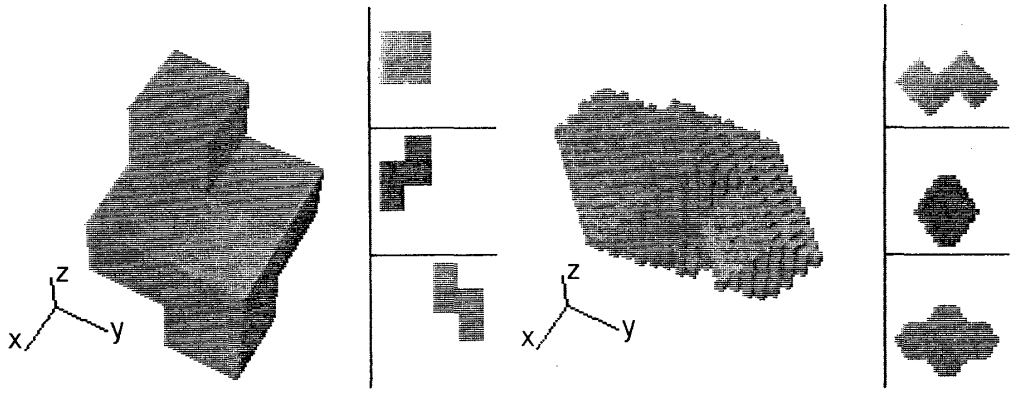

Figure 2:
Program report and a test object before (left) and after (right) the transformation. Projections in both frames: perspective (left) and orthographic (top to bottom) along *z*, *y* and *x* axes.