Proceedings of the 1999 IEEE
International Conference on Robotics & Automation
Detroit, Michigan • May 1999

Efficient Generation of Object Hierarchies from 3D Scenes

Miguel Angel García[†] Angel Domingo Sappa[‡] Luis Basañez[‡]

[†]Department of Computer Science Engineering
Rovira i Virgili University
Ctra. Salou s/n. 43006 Tarragona, Spain
magarcia@etse.urv.es

[‡] Inst. of Organization and Control of Industrial Systems
Polytechnic University of Catalonia
Diagonal 647, planta 11. 08028 Barcelona, Spain
{sappa, basanez}@ioc.upc.es

Abstract

This paper describes an efficient technique for computing a hierarchical representation of the objects contained in a complex 3D scene. First, an adjacency graph keeping the costs of grouping the different pairs of objects in the scene is built. Then the minimum spanning tree (MST) of that graph is determined. A binary clustering tree (BCT) is obtained from the MST. Finally, a merging stage joins the adjacent nodes in the BCT which have similar costs. The final result is an n-ary tree which defines an intuitive clustering of the objects of the scene at different levels of abstraction. Experimental results with synthetic 3D scenes are presented.

1 Introduction

Keeping an efficient representation of the environment where a robot operates is an important issue in order to guarantee the effective application of many algorithms, including path planning and collision avoidance. Hierarchical models are such efficient representations as they allow the description of complex geometric structures at different levels of resolution through the use of basic geometric primitives, such as boxes and spheres.

Hierarchical structures haven been proposed in the world modeling literature for two main purposes. The first purpose has been the representation of individual objects (*intra-object representation*) through hierarchies of boxes (e.g., [5][6][7]) or circles and spheres (e.g., [2][3][4]). The second purpose, which has received less attention, has been the representation of the different objects of a scene (*inter-object representation*) through hierarchies of spheres or convex-hulls. Previous work in this line is reviewed in [8].

Those two types of hierarchical representations are structurally similar but semantically quite different. Intra-object representations pursue the description of single objects with different approximation errors. Each level of the hierarchy contains a set of geometric primitives that approximate the original object at a certain error.

Inter-object representations aim at grouping the objects of a scene into clusters that reflect the spatial distribution of the objects and, ideally, unveil the functional organization of the scene. For example, an ideal objective might be to group the objects lying on a table into one cluster, then those objects, the table and the chairs around it into a new cluster, then the latter and other clusters corresponding to pieces of furniture inside the room into a single cluster, and so forth. Both inter-object and intra-object representations are complementary. Thus, each leaf of an inter-object representation can well be the root of an intra-object representation describing the corresponding object.

Intra-object representations usually apply a refinement (top-down) algorithm that subdivides larger geometric primitives into smaller ones until an approximation error is satisfied. This error-based approach can also be applied to building inter-object representations, by considering that the goal is the approximation of a large object with a certain error. However, a scene is conceptually different to an object in the sense that the latter has physical continuity while the former will usually contain isolated parts (individual objects) separated by empty space of arbitrary size. The extraction of an intra-object representation as it is understood (the geometric primitives at the same level cover the whole object), would lead to the generation of primitives even for covering the regions of empty space in the scene.

A more suitable algorithm for extracting inter-object representations was proposed in [8]. It is based on a bottom-up strategy that starts with the bounding volumes (e.g., spheres, convex hulls) of all the objects in the scene and groups them into bigger bounding volumes by applying heuristic rules that favor the progressive growth of those volumes. The algorithm goes over all the possible pairings of objects in the scene and for those whose distance is below a certain moving threshold (progressively larger) it computes a grouping cost. The pair of objects with lowest cost is grouped and the costs between the

This work has been partially supported by the Government of Spain under the CICYT project TAP96-0868. The second author has been supported by the Spanish Agency for International Cooperation and the National University of La Pampa (Argentina).

0-7803-5180-0-5/99 $10.00 © 1999 IEEE

new and the old clusters are recomputed. The algorithm stops when a single cluster is left. Thus, a binary tree is generated.

At each iteration, the cost function defined in [8] takes into account three heuristics that favor the grouping of the candidate pair of objects (or clusters) whose bounding volume has: *(1)* the smallest volume, *(2)* the smallest amount of empty space inside and *(3)* the most similar ratio between the size of the two objects (or clusters) that constitute that candidate pairing. The cost function is defined by a weighted average of those three criteria.

This paper presents a more efficient approach to the problem of inter-object grouping. Similarly to [8], the algorithm starts with the bounding volumes of the objects contained in the scene and clusters them into a tree. However, the proposed algorithm is more advantageous than the one presented in [8] since: *(1)* efficiency is higher as the costs between all objects of the scene are computed once, *(2)* there are not user-tuned parameters or thresholds and *(3)* the generated tree is n-ary instead of binary and, thus, it is shallower.

The proposed algorithm is described in section 2. Section 3 shows experimental results and a comparison with the technique described in [8]. Section 4 gives conclusions and further improvements.

2 Hierarchical Clustering of Scene Objects

An efficient algorithm is presented for generating a hierarchical clustering of the objects of a complex scene. The algorithm follows a bottom-up strategy that starts with the bounding spheres of the objects in the scene and progressively clusters them to produce an n-ary tree. The bounding spheres are allowed to overlap. Each node of the tree represents a level of abstraction in the scene and has associated the bounding sphere that contains all the bounding spheres that correspond to the clusters (or objects) below that node.

The algorithm is composed of four stages. First, an adjacency graph keeping the costs of grouping the different pairs of objects in the scene is built. Then the *minimum spanning tree* (*MST*) of that graph is obtained. From the MST, a *binary clustering tree* (*BCT*) is defined. Finally, a merging stage joins those adjacent nodes in the BCT that have similar costs. The final result is an n-ary tree which defines an intuitive clustering of the objects of the scene at different levels of abstraction. These stages are described below.

2.1 Adjacency Graph Generation

Let S be the set of N bounding spheres associated with the N objects of a scene. Let S_i and S_j be two bounding spheres contained in S and r_i and r_j their corresponding radius. Let d_{ij} be the distance between the centers of S_i and S_j. Finally, let r_{ij} be the radius of the smallest bounding sphere, S_{ij}, which contains S_i and S_j.

The attraction between S_i and S_j is defined as $\alpha_{ij} = r_i r_j / d_{ij}^2$. The bigger and closer the two spheres are, the larger their attraction will be. Intuitively, spheres with large attractions should tend to be grouped into the same clusters as they would denote close objects.

However, if the clustering algorithm only took the previous attraction into account, the size of the resulting cluster would not be considered, and that could lead to big clusters being created from the beginning of the bottom-up clustering process. Since this process is intended to create a hierarchy in which the leaves, which are the initial spheres in S, are progressively grouped until obtaining the scene's bounding sphere at the root of the tree, a more appropriate strategy will be to favor that small clusters are created first. This policy, which was already applied in [8], favors the generation of balanced trees. In our case, this policy is applied by defining a cost function that increases the cost of grouping two spheres of high attraction when the volume of their smallest bounding sphere is very big. This cost function is defined as $\zeta_{ij} = r_{ij}^3 / \alpha_{ij}$.

The first stage of the hierarchical clustering algorithm defines a fully connected weighted graph with as many nodes as objects in the scene and an edge joining each pair of nodes. Each edge has associated a weight which corresponds to the cost of grouping the spheres represented by the two nodes joined by this edge according to the previous cost function ζ_{ij}. The number of edges, and thus the cost of creating this graph is $O(N^2)$. As in [8], assuming that each object cannot merge with more than a constant number k of bodies the number of edges in the graph becomes $O(N)$. Moreover, if a space partition technique was utilized that graph could be built in $O(N)$ time.

2.2 Minimum Spanning Tree Generation

Given the previous adjacency graph, the next objective is the generation of a hierarchy (tree) of nodes whose edges have a minimum cost. This is done by computing the minimum spanning tree of the previous adjacency graph. The minimum spanning tree (MST) of a graph G is the acyclic subgraph of G that contains all the nodes of G and such that the sum of the costs associated with its edges is minimum. The MST of a graph with M edges and N vertices can be efficiently computed in $O(M \log N)$ by applying *Kruskal's algorithm* [1]. In our case, the cost is $O(N^2 \log N)$ for the worst case (fully connected graph) and $O(N \log N)$ for the best case (k neighbors plus space partitioning). Fig. 1(*left*) shows an example of a connected adjacency graph and its MST.

2.3 Binary Clustering Tree Generation

The minimum spanning tree of the adjacency graph is not directly used to define a clustering of the objects in the scene since, to start with, the root of the tree is not defined. Therefore, a first clustering is done by con-

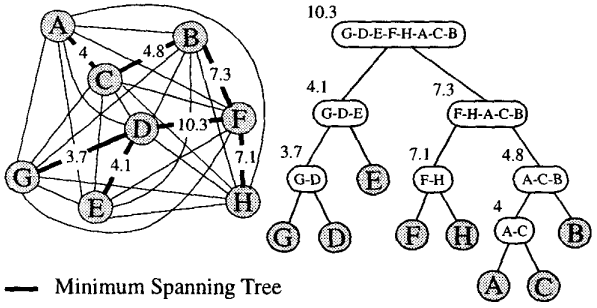

Figure 1. (*left*) Example of adjacency graph and its MST. (*right*) Binary clustering tree obtained from the previous MST.

structing a *binary clustering tree* (*BCT*) from the MST. The BCT is generated by considering the edges from the MST in ascending order of weight. Each considered edge E leads to a new cluster being added to the BCT. The new cluster embodies the clusters that contain the two nodes connected by E, if those clusters exist. The weight (cost function) associated with E is assigned to the new cluster. Each cluster is associated with the smallest bounding sphere that contains the bounding spheres of the nodes or clusters contained in it. The cost of computing the BCT from the MST is $O(N \log N)$

This process is illustrated with the MST shown in Fig. 1(*left*). The obtained BCT is shown in Fig. 1(*right*). The first edge is the one joining nodes G and D, as its weight is the lowest (3.7). Thus, a first cluster, G-D, is formed. The second edge in ascending order is the one joining nodes A and C (weight 4). A new cluster, A-C, is created. The third edge is the one joining nodes D and E (weight 4.1). Since D already belongs to a cluster, G-D, a new cluster, G-D-E, is formed by grouping node E and cluster G-D. The final edge is the one joining nodes D and F (weight 10.3). Since both nodes belong to two clusters at that time (G-D-E and F-H-A-C-B), the root is finally cluster G-D-E-F-H-A-C-B. Each cluster is labeled with the weight of its corresponding edge.

2.4 Cluster Merging

The previously obtained BCT is already a solution to the clustering of the initial objects in the scene. However, binary trees do not always reflect the real structure of the objects of a scene, where a cluster may contain several objects at the same level. Moreover, binary trees tend to become very deep. Hence, n-ary trees in which a cluster can contain two or more clusters or nodes appear to be a more appropriate solution.

The final stage of the proposed algorithm converts the BCT into an n-ary tree in $O(N)$ time by merging compatible clusters. In order to do this, the weights associated with the clusters in the BCT are partitioned into a set of families. Each family of weights denotes a set of objects which have similar grouping cost. After the MST generation stage, a list of the weights associated

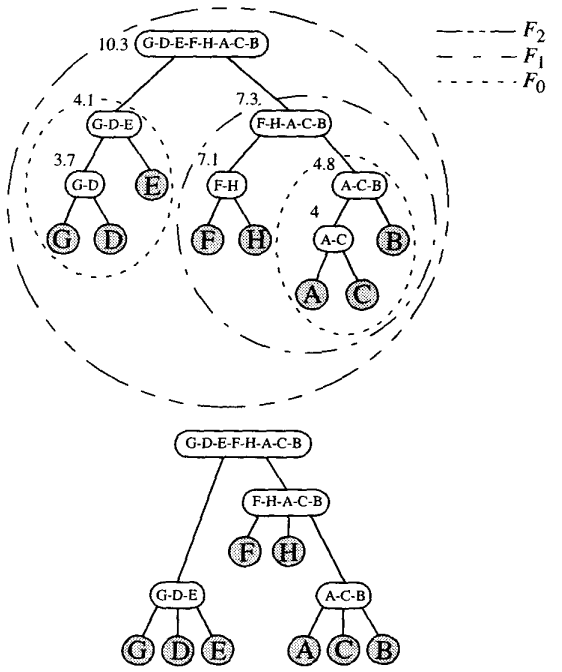

Figure 2. (*top*) Merging process applied to the clusters of the BCT shown in Fig. 1(*right*). (*bottom*) Final structure.

with the edges of the MST sorted in ascending order is obtained. The first two weights in this list define the first family of weights F_0. Let μ_i and σ_i be the mean and standard deviation of the weights that belong to a certain family of weights F_i. A weight w is considered to belong to the family F_i if $w < \mu_i + 2\sigma_i$. If a weight is assigned to a family, the mean and standard deviations of the latter are recomputed.

Starting with the third weight in the sorted list, the first family F_0 is increased with new weights until a weight is found not to belong to F_0 according to the previous criterium. This weight and the next one are the seed for a new family F_1. This grouping process ends up when all the weights in the sorted list have been considered, giving rise to a set of f families. For example, given the weights shown in the example of Fig. 1, three families would be obtained: $F_0 = \{3.7, 4, 4.1, 4.8\}$, $F_1 = \{7.1, 7.3\}$ and $F_2 = \{10.3\}$.

Once the families of weights have been found, the BCT is converted to an n-ary tree by merging adjacent clusters whose weights belong to the same family. This is illustrated in Fig. 2(*top*). The first family of weights, F_0, gives rise to clusters G-D and G-D-E being joined in a single cluster as they have weights that belong to F_0. Similarly, but in a different part of the tree, clusters A-C and A-C-B are also joined. Then, family F_1 leads to the merging of clusters F-H and F-H-A-C-B, as their corresponding weights belong to F_1. Finally, family F_2 does not lead to any clusters being joined as it is only com-

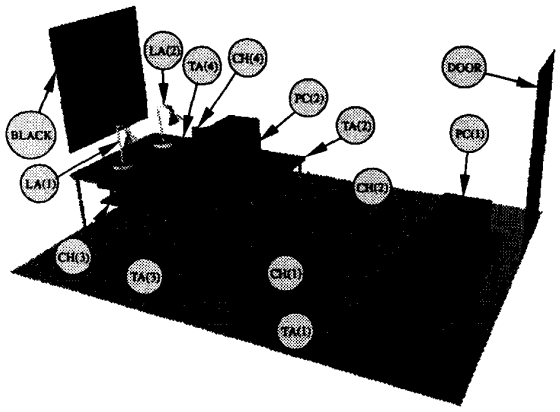

Figure 3. Example of a 3D scene and labels of the different nodes (objects) contained in it.

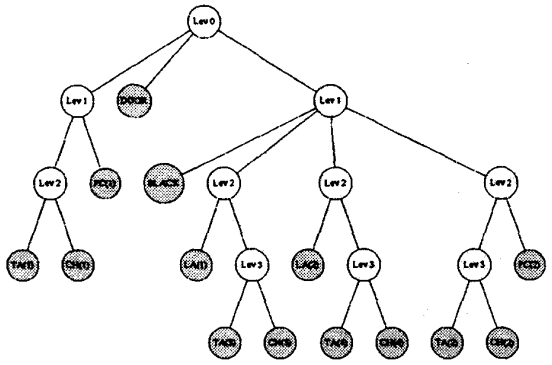

Figure 4. Hierarchical representation automatically generated from the scene shown in Fig. 3.

posed of one weight and, therefore, only one cluster belongs to it. The final n-ary tree representing the objects of the scene is shown in Fig. 2(*bottom*).

In sum, the proposed hierarchical clustering algorithm has a worst-case complexity of $O(N^2 \log N)$ and a best-case one of $O(N \log N)$.

3 Experimental Results

The proposed algorithm has been tested with a set of synthetic scenes. The first scene, shown in Fig. 3, corresponds to an office that contains 14 objects. The hierarchical representation (n-ary tree) generated by the proposed technique is shown in Fig. 4. Notice that some intuitive geometric relationships between the objects of the scene emerge from that grouping. For example, each chair is grouped with its corresponding table. The objects on each table (lamps, computers) are grouped with their corresponding ensembles (table/chair) at a higher level of abstraction. The three sets of tables near the blackboard and the latter are grouped under the same cluster. At the first level of the tree there are three elements: two clusters, which correspond to the two distinctive areas in

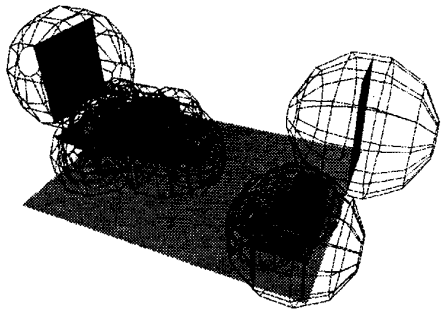

Figure 5. Original objects and their bounding spheres.

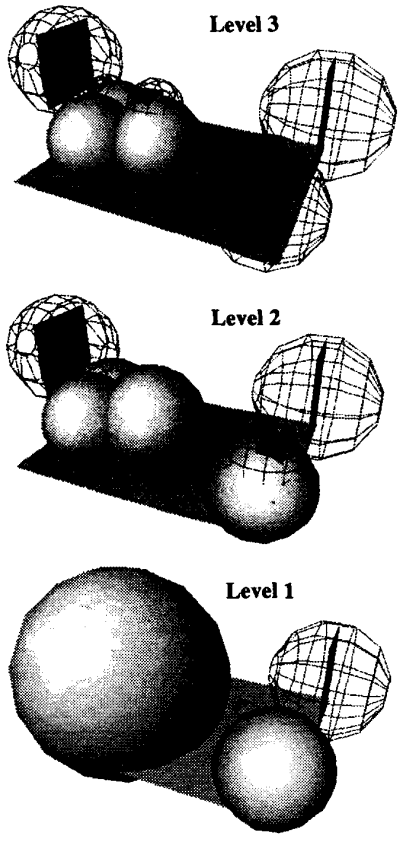

Figure 6. Smallest bounding spheres of clusters at levels 3, 2 and 1 in the generated hierarchical representation for the scene shown in Fig. 3.

the office, and the door. These three elements at the first level of the tree reflect the overall structure of the scene.

The bounding spheres associated with the objects of the scene are shown in Fig. 5. These spheres correspond to the leaves of the generated hierarchical representation. Fig. 6 shows the smallest bounding spheres corresponding to the clusters at the three deepest levels in the generated hierarchy. The root of the tree is a cluster con-

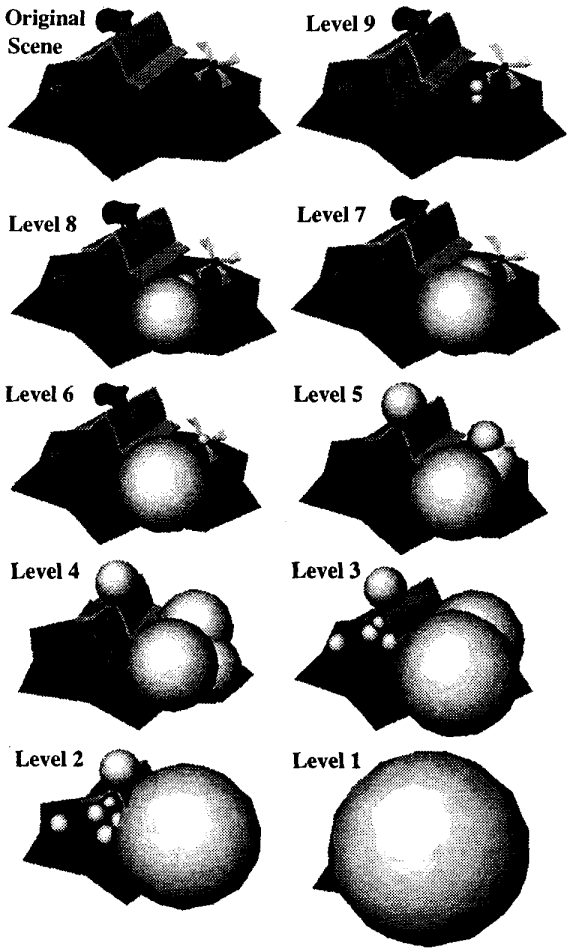

Figure 7. Example of a scene containing 62 objects and the clusters obtained by the proposed algorithm at the different levels of the final tree.

taining the bounding sphere of the whole scene. The CPU time to compute the final n-ary tree for the previous example was 0.01 sec. on a SGI Indigo II.

Fig. 7 shows an example of a scene containing 62 objects and the smallest bounding spheres corresponding to the obtained clusters at the different levels of the final hierarchy shown in Fig. 8. At the deepest levels of the tree, the rocket and the fences around it are grouped (levels 6 to 9). The various parts of the windmill are then grouped at levels 4 to 6. The different parts of the windows of the house are grouped at levels 2 and 3. Finally, the house (a single bounding sphere) with the windows and neon sign, and the cluster of objects beside the house are grouped at the root level. The CPU time to generate this hierarchy was 0.02 sec.

The proposed technique has been compared to a version of the algorithm presented by Xavier [8], considering worst-case conditions for both algorithms,

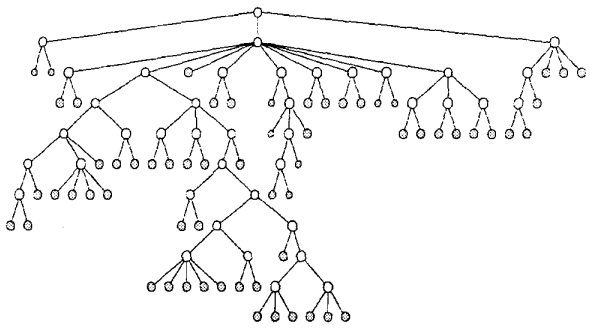

Figure 8. Hierarchical representation automatically generated from the scene shown in Fig. 7 with the proposed algorithm.

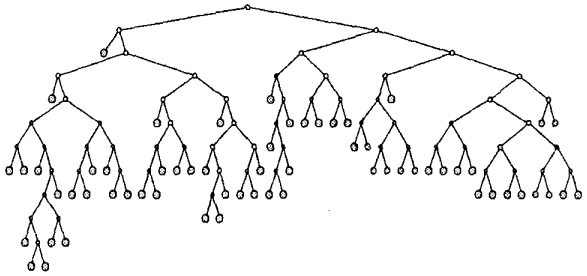

Figure 9. Hierarchical representation automatically generated from the scene shown in Fig. 7 through Xavier's algorithm.

i.e. when no heuristic optimizations are applied. In particular, no careful tuning of the user-defined parameters required by [8] has been done and no space partitioning has been considered. The reason for the former is that no hints are given for deciding the values of those parameters in order to obtain optimal performance. The motivation for not applying space partitioning is that the performance achieved with it is strongly related to the choice of the aforementioned user-defined parameters. Since no optimal parameter tuning can be granted in general, the space partition may not help to improve efficiency and may even lead to performance degradation owing to the overload due to generation and management of the data structure.

Qualitatively, the algorithm presented in [8] produces results comparable to the ones obtained with the proposed algorithm. For instance, Fig. 9 shows the binary tree produced by Xavier's algorithms in 0.05 sec. when it is applied to the scene shown in Fig. 7. Moreover, the asymptotic costs are the same, when the worst and best cases of both algorithms are considered. In order to compare the actual performance of the untuned versions of both the proposed and Xavier's algorithms, much larger scenes were synthetically generated. Those scenes contained randomly placed spheres of similar radius. The CPU times to compute the hierarchical representations with both techniques are shown in Fig. 10.

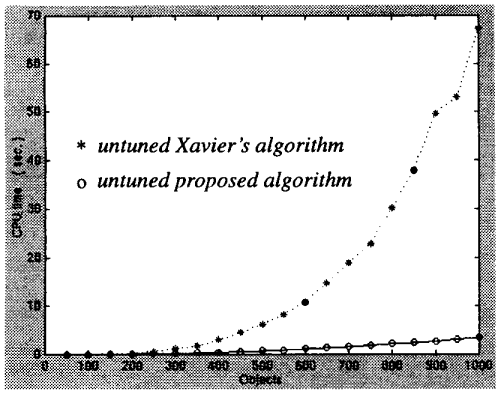

Figure 10. Comparison of performance between untuned versions of both the proposed and Xavier's algorithms. Scene objects are spheres of similar size randomly placed in space.

Every time a new cluster is created, Xavier's algorithm recomputes the costs between the bounding volume of that cluster and the other objects of the scene (both individual objects and already existing clusters). In order to speed-up the process, a dynamic threshold that determines a maximum accepted size is kept. All the objects whose size is larger than that threshold are discarded from the computation of the cost function at that step. This dynamic threshold is progressively increased. In the worst case, where all the objects have the same size, this dynamic threshold does not discard any objects. Moreover, every time a pair of objects is grouped, the cost between the new formed object and the previous ones must be recomputed and the new object must be inserted in the sorted list of candidate object pairs.

Conversely, the most expensive part of the proposed algorithm is the computation of costs between all the objects of the scene, but this stage is applied only once, just before the generation of the MST. The remaining stages are very efficient, leading to a better performance of the whole algorithm. Obviously, a careful tuning of parameters and the application of space partitioning would improve the performance of Xavier's technique, but optimal tuning may be difficult to achieve in practical situations.

4 Conclusions

A new technique for generating a hierarchical representation of the objects of a scene has been presented. The proposed technique is composed of four stages. The first stage computes the cost of grouping all the pairs of objects in the scene and generates an adjacency graph with those costs. Then, the minimum spanning tree (MST) of that graph is determined. A binary clustering tree (BCT) is obtained from the MST. The final stage generates an n-ary tree by merging clusters of the BCT according to a clustering of the costs associated with the edges of the MST.

The proposed technique is advantageous with respect to previous work since: *(1)* efficiency is greatly improved as the costs among the objects of the scene are computed only once, *(2)* there are no explicit user-defined parameters or thresholds to be carefully tuned and *(3)* the trees are n-ary instead of binary (they are shallower).

Further work will consist of the improvement of the first stage in order to avoid the computation of the costs among all the objects of the scene through the application of space partition techniques that prevent distant objects from being considered. Different bounding volumes will also be analyzed. The application of the proposed technique for speeding-up path planning and collision avoidance algorithms will also be studied.

We thank Jaume Gratacos Prats for providing the code that generates the VRML spheres shown in the examples. The images of trees shown in the examples have been generated with daVinci, a public graph visualization software developed at the University of Bremen (Germany).

5 References

[1] K. Rosen, *Discrete Mathematics and its Applications*. McGraw-Hill, Inc., New York, second edition, 1990.

[2] J.Pitt-Francis and R. Featherstone, "Automatic Generation of Sphere Hierarchies from CAD Data", *IEEE Int. Conf. on Robotics and Automation,* Leuven, Belgium, May 1998, 324-329.

[3] S. Quinlan, "Efficient Distance Computation Between Non-Convex Objects", *IEEE Int. Conf. on Robotics and Automation,* San Diego, USA, May 1994, 3324-3329.

[4] B. Martínez, A. delPobil and M. Pérez, "Very Fast Collision Detection for Practical Motion Planning Part I: The Spatial Representation", *IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, May 1998, 624-629.

[5] S. Cameron, "Efficient Bounds in Constructive Solid Geometry", *IEEE Computer Graphics & Applications*, May 1991, 68-74.

[6] S. Gottschalk, M. C. Lin and D. Manocha, "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection", *ACM Siggraph'96.*

[7] M. A. García, "A Hierarchical World-Model Representation Supporting Heterogeneous Multisensory Integration", *Int. Conf. on Advanced Robotics*, Sant Feliu de Guixols, Spain, September 1995, 461-471.

[8] P. Xavier, "A Generic Algorithm for Constructing Hierarchical Representations of Geometric Objects", *IEEE Int. Conf. on Robotics and Automation,* Minneapolis, Minnesota, April 1996, 3644-3651.