Interactive Retrieval of 3D Shape Models using Physical Objects

Hiroyasu Ichida, Yuichi Itoh, Yoshifumi Kitamura, Fumio Kishino
Graduate School of Information Science and Technology
Osaka University
2-1, Yamada-oka, Suita, Osaka, 565-0871, Japan
{ichida, itoh, kitamura, kishino}@ist.osaka-u.ac.jp

ABSTRACT
We present a novel method for interactive retrieval of 3D shapes using physical objects. Our method is based on simple physical 3D interaction with a set of tangible blocks. As the user connects blocks, the system automatically recognizes the shape of the constructed physical structure and picks similar 3D shape models from a preset model database, in real time. Our system fully supports interactive retrieval of 3D shape models in an extremely simple fashion, which is completely non-verbal and cross-cultural. These advantages make it an ideal interface for inexperienced users, previously barred from many applications that include 3D shape retrieval tasks.

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: Query formulation

General Terms
Design, Algorithms, Human Factors

Keywords
Physical Object, 3D Shape Model, Retrieval System, Voxel Data

1. INTRODUCTION
Recent developments in modeling software and new acquisition techniques, such as 3D scanners, have made the construction of 3D shape models much easier. This has led to an increasing accumulation of 3D shape models both on the web and elsewhere. Following this trend, not only computer graphics experts, but also novice users of any age and culture can easily access many of these models. This has led to a great need for 3D shape retrieval systems [4, 5, 9, 14, 24, 18, 19, 20, 21, 23]. For example, consider a person who wants to build a 3D virtual town that has a lot of houses, stores, cars, and so on. This requires 3D shape models. However, it is hard to design all these models independently, from scratch. Instead, they can be retrieved from a large stock of 3D shape models, using techniques similar to those used for retrieval of text, images, audio, and other media.

However, finding a 3D shape model is much more difficult than retrieving text documents or 2D images. This is because of the complexity and difficulty of expressing and describing 3D shapes and models. Although several web sites or services provide users with 3D shape model retrieval systems which use, for example, text-based keywords, these retrieval systems would not work in many applications because the keywords that humans use to describe shapes depend on language, culture, and many other factors. This makes the text-based query less than ideal in many applications [4, 15].

As an alternative, we propose a novel user interface which allows users to retrieve a 3D shape model using physical blocks. Our system enables a user to tangibly interact with a 3D structure while the shape is being automatically recognized and used as a query shape to a database. By comparing this shape with all of the 3D shape models in a database, the system selects candidates which are most similar to the query shape given by the physical blocks. The method is useful for many applications, for example, story-telling, interactive toy, digital encyclopedia, computer aided design (CAD) and much more. In this paper, we present the system in detail, report its performance, and discuss its applications.

2. RELATED WORK
Some approaches to retrieval of a desired model from a large selection of 3D shape models have already been proposed. Currently, the text-based search method is popular. Indeed, several web site storage systems provide users with a text keyword search. However, these systems have the same problems as any text search: a text description is often too limited, incorrect, ambiguous, and dependent on the language of interaction. In addition, the spatial complexity of 3D shape models is commonly too large to put into a limited number of words.

In order to solve this problem, some researchers have focused on designing systems which query features of 3D shape models other than keywords. In the online demo “Alexandria”, users can control the weights of individual model
attributes, such as “geometry”, “angular geometry”, and “colour” [20, 21]. Suzuki focuses on reducing the size of features used in the matching process by dividing the 3D models into grid cells for the purpose of quick retrieval of 3D shape models [23]. Elad’s system allows users to retrieve 3D shape models from the database by applying relevant feedback. In this system, the use of moments of a 3D shape model is the novel feature. Users mark the models picked by the initial search as either “relevant” or “irrelevant” and then the system changes the weights of the models in the matching function based on those marks [5]. The method by Ohbuchi, first, normalizes the pose of a 3D model by using moments [17]. Then the system computes several inertial properties along the principal axis of the model. Zaharia employed a 3D Hough transformation as the shape feature after pose normalization [27]. Hilaga proposes a technique to calculate the similarity between polyhedral models by comparing skeletal and topological structures [9]. Osada proposes and analyses a method to compute shape signatures with a shape distribution sampled from a shape function measuring global geometric properties of a 3D shape model [18, 19]. Ohbuchi improved Osada’s method by introducing mutual orientation of surfaces, and used a 2D histogram having distance and angle [16]. However, these previous works on retrieving 3D shape models mostly take an algorithmic approach to the problem without considering the user interface and the interaction process.

If it is assumed that users of any ability will be using the retrieval system, it is important that they are able to find the ideal 3D shape model from lots of 3D shape models easily and intuitively. Since novices at computer graphics experience difficulty in designing and manipulating 3D shape models, the main aim should be to produce an effective and intuitive interface for input queries. Some researchers have focused on realizing such interfaces. In Chen’s system [4] and Min’s system [15], three 2D sketches (side view, front view, and top view) are used to represent the 3D structure. However, novices at 3D computer graphics and/or CAD have difficulty in transforming a 3D structure into three 2D images. Moreover, the 3D sketch, like the Teddy system [10], is used to input the query 3D structure [15, 14] and the 3D sketching tool is, arguably, hard to use especially for a person who does not have a talent for painting or drawing.

In this paper, we propose a 3D shape model retrieval system which uses physical objects as the user interface. A user constructs a 3D shape as a query and then the system retrieves similar models to the tangible query shape.

3. 3D SHAPE MODEL RETRIEVAL USING PHYSICAL OBJECTS

3.1 Use of Physical Objects

If a user can construct 3D shapes by simply combining physical blocks, the user interface to generate the query shape becomes intuitive. This leads to other benefits: the user interface does not depend on the language, culture or age group of the users. In addition, users can operate and manipulate the 3D structure by hand without encountering the difficulty of transforming it into 2D images and therefore do not have to be an expert at computer graphics and/or CAD. Moreover, the physical objects used for a query shape could be recognized as the 3D model in the virtual world. Consequently, such an interface might become more usable across the boundary of real space and virtual space. To realize the system described above, we need an interface which can recognize the 3D shape of physical objects in real time.

Research on 3D shape recognition using physical blocks to achieve an intuitive interface was initially carried out in the early days with architecture design [1] using machine-readable models [7, 6]. Later, these ideas were followed by further efforts [2, 22]. Recently, systems have been proposed in which the 3D structure of assembled blocks is recognized by a computer after a user connects them to the computer in real time [8] and in which the geometry of fully assembled Lego-type blocks is recognized by a computer after a user connects them to the computer and powers up [3].

We decided to implement a query interface for retrieval of 3D shape models with physical objects by using the Active-Cube system [12, 26, 25]. The appearance of the ActiveCube system is shown in Figure 1. It consists of a set of rigid cubes with 5-cm sides. Users can construct various 3D structures by combining the cubes as they desire. All the faces of the cubes are the same so that each cube can be connected to any other cube. The 3D structure of connected cubes is recognized by a computer in real time. Therefore, users can construct a 3D virtual model in the computer (i.e., in the virtual environment) that exactly corresponds, spatially and temporally to the physical object built out of the physical cubes. ActiveCube is equipped with sensors that obtain the user’s operational intention and other information from the physical environment. ActiveCube is also equipped with display systems and/or actuators to show the simulated results or internal status of the computer system.

Figure 1: ActiveCube system

3.2 Flexibility in the Shapes Represented by the Construction Blocks

Users construct the intended shape by iteratively connecting and/or disconnecting cubes. Therefore, we have to estimate how many shapes users can construct flexibly by using n cubes. The simulator counting the possible number of shapes, which consist of n cubes, is described here. First, the simulator connects one cube to all shapes which consist of n − 1 cubes acquired in the previous simulation and counts all shapes as candidates. Second, it transforms the candidate shapes into voxel data and gets rid of identical
shapes by using the equation (2) described in section below. As a result, we can acquire all of the unique shapes which consist of \( n \) cubes.

Figure 2 shows the relationship between the number of used cubes and the number of constructible shapes. As you can see, the number of constructible shapes increases phenomenally. Just several cubes can represent an enormous number of shapes as a query. On the other hand, when constructing a detailed structure, users could use hundreds if not thousands of cubes; however, this would become a laborious task. If users could only obtain the intended shape models by constructing such an enormous number of cubes, the implementation would become unrealistic and inefficient. Consequently, we assume that even about 10 cubes are expressive enough to construct a query shape for a database which contains about 100 – 1000 models.

3.3 Proposed System

In creating a query shape, a user constructs the cubes of the ActiveCube system to represent the volume of the intended 3D model. There is also a technique to recognize the shape as the skeleton of the 3D model. However, because each cube has a 3D volume with a 5-cm side, we assumed that the former method is more appropriate than the latter. To represent the 3D volume, we decided to use a voxel data representation for the 3D constructed shape of ActiveCube and for the 3D shape polyhedral models in the database.

As shown in figure 3, our system recognizes a query shape which the user constructs with about 10 cubes of ActiveCube. It next converts the structure into a voxel data representation in real time by using the algorithm described in section 4 below. Then the system calculates the similarities between the voxel data described above and all the models’ voxel data in the database, which has been created in advance. These similarities are calculated with the equation (2) described in section 5; the rate of correspondence between each voxel data is calculated by intersecting them.

However, if two sets of voxel data consist of different numbers of voxels, this causes the following problem. When one set of voxel data consists of more voxels than another set of voxel data, it is plausible that the former may subsume the latter and, if that is the case, the similarity cannot be calculated correctly. Therefore, we define in advance all the voxel data of the 3D shape models as being represented by \( n \) \((n = 1, 2, ..., N) \) voxels. In the process of calculating similarities, the system uses the 3D models’ voxel data represented by the number of voxels as same as the number of the blocks of query shape. Figure 4 shows an example of the definition of \( n \) \((n = 1, 2, ..., N) \) voxel data for a Space Shuttle model. (We describe the details of the definition of these voxel data in section 4.2) Thus, we resolve the scale problem by using a method in which the system changes the target of the voxel data representation for a 3D model to correspond to the number of voxel data of the query shape. Moreover, because the calculation of similarities is simpler, the calculating process is faster. As a result, our system can present some 3D shape models which are well matched to the query shape in real time.

4. VOXEL DATA REPRESENTATION

The 3D shape structure physically constructed by a user is converted into a voxel data representation in real time. A polyhedral model and a voxel data representation for each 3D shape model are stored in the database.

In this section, we describe how the voxel data representation is used for both the query structure and the 3D shape models.
4.1 Voxel Data Representation of Query Shape

Our system recognizes the 3D shape structure constructed with ActiveCube and converts it into a voxel data representation in real time. Figure 5 shows this conversion from query shape to voxel data. In the ActiveCube system, there is a specific cube called the BaseBlock which is equipped with a connector to the Host PC. In the process of converting to voxels, the origin of three coordinate axes is based on the position of the BaseBlock. When a user constructs the query structure as shown in Figure 5(a), the extracted voxel data is defined as shown in Figure 5(b).

4.2 Voxel Data Representation of 3D Shape Models

The polyhedral models are also converted into voxel data representations in the 3D shape model database. Each polyhedral model is represented with \( n \) \((n = 1, 2, ..., N)\) voxels in advance. To do this, first the system obtains all of the voxel data represented by \( n \) voxels. Then, the system compares a polyhedral model with all of the voxel data obtained above, calculates the level of similarity, and picks the highest level of voxel data as the voxel representation of the polyhedral model. The system repeats the above process for all polyhedral models. We describe the details of this procedure in the following set of steps;

(i) Calculate all of the voxel data represented by using \( n \) \((n = 1, 2, ..., N)\) voxels following the procedure described in section 3.2.

(ii) Pick one of polyhedral models and obtain the minimum size of its bounding box. Figure 6(a) shows the definition of the bounding box for one polyhedral model. The premise here is that a polyhedral model is defined in full-faced pose. Each face of the calculated bounding box is vertical to each axis of the coordinates.

(iii) Choose one of the voxel data, calculated at step (i), and obtain the minimum size of its bounding box. Figure 6(b) shows an example of this definition.

(iv) The system resizes both bounding boxes calculated at step (ii) (box 1) and step (iii) (box 2). Concretely, the system adjusts the lengths of the middle-length-sides of each box to maintain the balance between the size of 3D shape model and the size of the query shape. Then, the system overlaps both bounding boxes (Figure 6(c)) and rotates box 2 to maximize the number of the polyhedral model’s points that falls inside the area occupied by voxel data in box 1. Here, we use a polyhedral model as a point set by using Osada’s method [19] for generating unbiased random points with respect to the surface area of a polyhedral model. Figure 7 shows an example of a generated point set for a plane model.

(v) Count the number of points inside the area of the voxel
data in box 1 \((P)\) and compute the similarity \((\text{Model-Voxel Similarity or MV Sim.})\) by calculating the percentage of points contained in the voxel data for all points \((P_{all})\) with equation (1),

\[
\text{MV Sim.} = \left( \frac{P}{P_{all}} \right) \cdot 100 \% \quad (1)
\]

(vi) The system iterates steps (iii) - (v) and chooses the one voxel data in which MV Sim. is maximized. Finally, the system detects the voxel data representation for the chosen 3D shape model by \(n\) voxels.

(vii) The system repeats steps (ii) - (vi) and defines all of the voxel data which consist of \(n\) voxels for all 3D shape models in the database.

These calculating processes are performed off-line. In the matching process, the system can use the \(n\) voxel representation for each model when the user constructs a shape which consists of \(n\) cubes and thus can calculate similarity very quickly.

5. CALCULATION OF SIMILARITY

In this section, we describe the details of the matching algorithm; how our system computes the similarity between the voxel data of the query shape constructed by \(n\) cubes and of a 3D shape model represented by \(n\) voxels.

With the preprocessing described in section 4 above, it is very simple to calculate the 3D similarity between the query shape and the polyhedral model stored in the database. This similarity \((\text{Voxel-Voxel Similarity or VV Sim.})\) is calculated at each connection and disconnection event with equation (2),

\[
\text{VV Sim.} = \left( \frac{i}{n} \right) \cdot 100 \% \quad (2)
\]

where \(i\) is the number of voxels which is an intersection of the voxel data of the query shape and that of the 3D shape model and \(n\) is the number of all voxels in the query shape. VV Sim. is maximized over all possible intersections, \(V_i\), produced by rotating or translating voxels extracted from the query shape.

6. IMPLEMENTATION

We implemented the retrieval system for 96 models in various categories using ActiveCube. Because the number of 3D shape models in the database is about 100, we assume, as mentioned in section 3.2, that the use of about 10 cubes is adequate. Therefore, we substituted 10 for the variable “\(N\)”.

The system was implemented on a notebook PC (Matsushita Let’s Note CF-L1, OS: Microsoft Windows 98 Second Edition, CPU: Intel Pentium 3 600 MHz, RAM: 192 M), and coded in Borland C++ Builder 5 for the retrieval process, Visual Basic 6.0 for GUI, and ActiveCube Library for the control of ActiveCube. After calculating similarities between the query shape and each of the 3D shape models, the 3D models which had more than 80% similarity were presented to a user as candidate models. Figure 8 shows a schoolchild in the lower grades constructing a 3D shape as a query with ActiveCube, presenting the shape to the system, and retrieving some 3D shape models.

Figure 9 shows the voxel data representation for a 3D dog model. Figure 11 shows the transition of similarities of all 3D shape models when a user constructs the query shape by connecting cubes as shown in figure 10. The vertical scale of each graph shows similarity (%), and the horizontal scale shows the number of connected cubes (in this case, the number of cubes is from 1 to 8).

In the initial stage, i.e., when there is only the BaseBlock, all similarities are 100% and also 100% in the case of 2 connected cubes. In the case of 3 connected cubes, the system reduces the number of candidate models to about half because there are two voxel data representations when using 3 voxels. As more blocks are connected, the number of candidate models is reduced. When a user connects 8 cubes in turn according to the procedure shown in figure 10, only 3 models (see top-left of figure 11) have more than 80% simi-
Figure 11: Transition of similarity for all 3D shape models

Figure 12 shows the relationship between the number of blocks and the average retrieval time. As can be seen, even if the query shape is constructed with 8 cubes, it takes only 3 seconds to retrieve candidate models. These results, we believe, clearly demonstrate that our system can fully meet the requirements of practical application.

We also observed a child using our system. The child successfully constructed query shapes by connecting and disconnecting cubes and retrieved 3D shape models in a playful activity. So we also assume that novice users, including children and the older users, will be able to retrieve 3D shape models using our proposed system.

7. DISCUSSION

With our proposed system, a user can present a 3D shape using physical objects as queries, and then choosing from an automatically selected set of candidates. Because this retrieval system uses physical objects, it does not require specific knowledge of computer graphics or 3D geometries and it makes it very easy for a user to intuitively deal with 3D shapes. Therefore, anyone should be able to use 3D shapes and retrieve 3D shape models. Indeed, we qualitatively confirmed that even a novice/child could retrieve several 3D models by only constructing cubes. However, when a user creates a 3D shape to retrieve a desired 3D shape model, the desired model might not always appear. For example, if a user searches for the model “snake”, the result may differ according to whether the user imagined “a coiled-up snake” or “a snake slithering along the ground”. In addition, when defining an n voxel data representation for a polyhedral model, the state of the n – 1 voxel data representation is not considered and, consequently, the candidate models are not always narrowed down as the user intends and models that were not presented in the previ-
ous retrieval may also appear. To solve this problem, it will be necessary to retrieve candidates by taking into account the results of the \( n - 1 \) voxel data representation. In this current implementation, however, the user sometimes gets unexpected models which were not requested and it may therefore be helpful to provide users with some instructions on how to query the 3D models.

As to the retrieval time, this increases exponentially as the number of cubes connected to the query shape increases. This is why our system calculates similarities by rotating and parallel-translating the bounding boxes repeatedly to maximize the correspondence between both types of voxel data. We assume that the retrieval time depends on the size of the database and believe that it is enough practical to apply our method to such applications which use about 100 models. Besides, the database should be structured in order to be able to quickly focus on candidates for a variety of applications which use larger numbers of models. Furthermore, if it is possible and practical to use much more cubes as a query, the benefit of our method must be enhanced more, since the user can search with the volume of the structure.

At present, our system cannot distinguish between 3D shape models that are similar in appearance. Accordingly, we are planning to use all information about the query object, e.g., the input/output functions equipped with cubes and the surface colour of each cube. (ActiveCube is inherently equipped with these information.) Then, users should be able to retrieve the desired 3D shape model more easily and correctly. For example, if a user constructs the shape of an aeroplane and puts the fan-cube at the head of the structure, the system will be able recognize which airplane the user intended to design, that is, a propeller plane or a jet aircraft.

On the other hand, even if users do not know the exact shape of the 3D models, they can still enjoy assembling cubes and retrieving the 3D models. We think that this would be a way in which this system could train users to represent 3D models with physical objects. We are planning to use these features of our proposed system for educational applications and as a training tool.

8. CONCLUSION

In this paper, we proposed the retrieval of 3D shape models using physical objects. We used the ActiveCube system as the physical objects input infrastructure. Our results demonstrated that the proposed system fully supports interactive retrieval of a 3D shape model even by novices at computer graphics or 3D geometries. In order to improve accuracy of the retrieved shape model against the user’s requirement, we are planning to use multiple clues of the constructed object such as input/output functions equipped with blocks, surface color of each blocks, and so on, in addition to the object shape used in the proposed system described in this paper.

A variety of applications can be considered, e.g., storytelling, interactive toy, digital encyclopedia, CAD and much more. For example, in case of application to an interactive toy, users play with a virtual object in the computer which is associated by the constructed physical object in real space [11]. Here, the consistency between the virtual object and the real object is always completely maintained, and the interaction is supported by input and output devices fitted to physical objects.

Future work includes user studies of the proposed system to evaluate how the retrieved model matches the user’s mental image. Although this mental process might be difficult to evaluate accurately, an assessment based on recent neuroimaging studies through the observation of the human brain activities [13] in addition to a conventional experiment on task performance might be interesting.

9. ACKNOWLEDGMENTS

This research was supported in part by “The 21st Century Center of Excellence Program” and Grant-in-Aid for Scientific Research of the Ministry of Education, Culture, Sports, Science and Technology, Japan and by Exploratory Software Project grant of Information-technology Promotion Agency, Japan.

10. REFERENCES


