A Potential-Based Generalized Cylinder Representation

Jen-Hui Chuang (莊仁輝), Chi-Hao Tsai (蔡祈浩), and Cheng-Hui Chen (陳政輝)

Department of Computer and Information Science
National Chiao Tung University, Hsinchu, Taiwan 30010, R.O.C.
Tel: (886) 3-5731979 Fax: (886) 3-5721490
e-mail: jchuang@cis.nctu.edu.tw

Abstract

Generalized cylinder (GC) is a class of parametric shapes that is very flexible and capable of modeling many different types of real-world objects, and have subsequently been the focus of considerable research in the vision community. Most of the related works proposed previously have dealt with the recovery of 3D shape description of objects based on the GC representation from one or more 2D image data. Different from the objective of the previous works, in this paper, we will propose a new approach to obtaining a GC-based shape description of 3D objects. The proposed approach of deriving the GC axis is a further extension of the potential-based skeletonization approach presented in [1]. Simulation results demonstrate that the derived GC representation will yield better approximation of object shape than that based on simpler subclasses of GC since there is, in principle, no restriction on the topology of the GC axis and the shape of the cross-sections. On the other hand, any subclass of the GC which has a simpler GC topology may also be obtained from the potential-based representation through appropriate adjustments of its cross-sections as well as axis direction.

Keywords: generalized cylinder, skeletonization, generalized potential model

1. Introduction

It is well-accepted that a good way to represent a complex 3D object is to decompose it into simpler volumetric parts and then describe the parts and the relationships among them [2], [3], [4], [5]. A popular representation for such volumetric parts is to use generalized cylinder (GC)-based shape representations [2]. A GC is a solid defined by its axis, a cross-section curve, and a scaling function. The solid is obtained by sweeping a planar region, its cross-section, as the region is moved and deformed according to the scaling function along a space curve, its axis. Fig. 1 shows an example of GC, in which a surface meridian is the locus of a specific point on the cross-section curve. In fact, the solid shown in Fig. 1 is one of the major subclasses of GCs called Straight Homogeneous Generalized Cylinder (SHGC) [6], [7], which has a straight axis and cross-sections are of the same shape which may vary in size along the axis. Examples of two other subclasses of GCs include (see Fig. 2):

Fig. 1: A generalized cylinder representation consists of GC axis, cross-sections, scaling function, and meridians.

1. Planar Right Constant Generalized Cylinders (PRCGCs), whose cross-sections are fixed both in shape and size while the axis can be any planar curve.
2. Circular Planar Right Generalized Cylinders (circular PRGCs), which is similar to PRCGC except that its cross-section is fixed in shape (circular) but not in size.

GCs were first proposed by Binford [2] as a class of parametric shapes that is very flexible and capable of modeling many different types of real-world objects, and have subsequently been the focus of considerable research in the vision community. Most of the related works proposed previously have dealt with the recovery of 3D shape description of objects from image data (shape from contour). Recovering 3D shape of objects in a scene from image data remains an important and difficult problem in computer vision. The difficulty arises from the fact that an image is a 2D projection of the scene and the process is not invertible without making some assumptions. Thus, most methods proposed for recovering generalized cylinder descriptions have implicitly or explicitly relied on the assumption that GCs project onto 2D images with some degree of symmetry, and use this assumption as the basis for their recovery.
algorithms. On the other hand, the algorithms are restricted to some subclasses of GCs such as SHGC, PRCGC, and Circular PRGC.

Fig. 2: Three major subclasses of GCs – SHGC, PRCGC, and Circular PRGC.

Agin [8], and Nevatia and Binford [3] used occluding (partial viewed) contours to recover generalized cylinder description from range data. Their methods assume implicitly that the occluding contours of a generalized cylinder are symmetric with respect to its axis. Complete generalized cylinder descriptions are obtained by grouping hypothesized local generalized cylinders, whose axes are found as the midpoints of pairs of occluding contour points. Marr proposed an algorithm in [9] to recover shape from GC contours while assuming that the projection is orthographics and that the occluding contours of an observed SHGC (which he called a generalized cone) are planar.

Horaud and Brady [10] proved that the contours of a solid of revolution (SOR), an important subclass of SHGCs, are symmetric with respect to the image of its axis. An algorithm was given for recovering the position and orientation of a solid of revolution from a single image of its contours using this property. They also argued that the method could be extended to the case of an arbitrary SHGC by approximating it by some best-fitting “perceived surface of revolution”. Nalwa's method [11] was limited to the analysis of solids of revolution and gave some properties of such objects, but does not provide a method for complete reconstruction. Other recovery methods for surfaces of revolution were also considered by Richetin et al. [12] and LaVest [13].

In [7], Shafer proved several general properties of straight homogeneous generalized cylinders, and derived the general equation that a point must satisfy to belong to the occluding contour of a SHGC, without any restriction on the viewing direction. Rao and Nevatia [14] used Shafer's result that the occluding contours of a linear SHGC are planar and symmetric with respect to its axis to segment sparse range data. Ponce, et al. [15] proved several new projective invariant properties of the contours of SHGCs, and used these properties in two implemented algorithms for recovering generalized cylinder descriptions from contour data. These properties were then used to recover SHGCs in several researches. Sato and Binford [16] described a system for recovery of SHGCs in an edge image. Zerroug and Nevatia [17] presented an approach which addressed the problem of shape description and scene segmentation (figure-ground problem) for SHGCs in the presence of broken contours, markings and occlusion. The approach exploited projective invariant properties of SHGCs in order to guide the segmentation and description processes. The method used these rigorous properties in a bottom up, perceptual grouping approach to handle a hierarchy of three levels of features: curves, symmetries and SHGC surface patches. Gross and Boult [18] proposed a methodology which is useful for the recovery of object classes like tubes, where contour and heuristic constraints are shown to be insufficient for shape recovery. They proved that SHGC contours generated under orthography have exactly two degrees of freedom, and showed that the remaining free parameters can be resolved using reflectance-based (intensity) constraints in addition to contours. The reflectance-based recovery algorithm is demonstrated on both synthetic and real SHGC images.

Ulupinar and Nevatia [19] presented techniques for the recovery of two broader classes of curved objects, namely SHGCs and PRCGCs, and proved that they exhibit certain symmetry properties. The method is based on, and is a major generalization of, the technique they had described for inferring 3D shape from contours of zero-Gaussian curvature (or ZGC) surfaces [20]. In [21], Zerroug and Nevatia extended the approach in [17] to handle PRCGCs and circular PRGCs so as to address more complex, compound, objects which consist of parts of some generic shapes which has both straight and curved axes. They derived a number of rigorous geometric invariant and quasi-invariant properties of these parts and developed a system that automatically detects their projections and recovers their 3D shapes from a 2D image. Havaldar and Medioni [22] addressed the problem of recovering high-level, volumetric and segmented (or part-based) descriptions of objects from three intensity images. They derived a way of inferring volumes in terms of GCs, when, in reality, the unseen parts of the object may be quite different from the inferred ones. In such cases, integrating multiple triplets of images from different positions can provide more accurate volumetric descriptions.

More recently, Zerroug and Nevatia described a method in [23] that detects and describes complex objects (i.e., those containing multiple parts). The method recovered descriptions of single-part objects in terms of three subclasses of GCs (SHGCs, PRCGCs, and circular PRGCs). The method addressed the figure-ground problem and part-based shape description problem at the same time and in a generic way (i.e., without prior knowledge of the specific objects being viewed). It has successfully extracted generic 3D volumetric descriptions composed of certain subclasses of generalized cylinders and of their joint relationships, in the presence of noise, boundary breaks, markings, shadows, and partial occlusion.

Different from the objective of the previous works,
in this paper, we will propose a new approach to obtaining a GC-based shape description of 3D objects. The proposed approach of deriving the GC axis is a further extension of the potential-based skeletonization approach presented in [1]. Assume the object surface and its two end cross-sections are given in advance, the GC axis can simply be generated by identifying the potential valley connecting the centroids of them. Cross-sections at different locations along the GC axis are then obtained by a simple algorithm.

The remainder of this article is organized as follows. The generalized potential model in the 3-D space is reviewed in the next section. The potential-based skeletonization algorithm is adopted to derive the GC axis of a 3D object in Section 3, where the algorithm for the derivation of cross-sections will also be presented. Section 4 gives some experimental results and discussion of the proposed approach to deriving the potential-based GC representation. Section 5 concludes this chapter and outlines some possible directions for future research.

2. Review of a Generalized Potential Fields in the 3-D Space

In [24], a potential-based modeling of 3-D workspace for collision avoidance is proposed. It is shown that the Newtonian potential, being harmonic in the 3-D space, can not prevent a point charge from running into an object surface which is uniformly charged. This is because the value of such a potential function is finite at the continuously charged surface. Subsequently, generalized potential models are developed to assure collision avoidance between 3-D objects. The potential function is inversely proportional to the distance between two point charges to the power of an integer \( m \) and the potential and thus its gradient due to a 3-D polygon can be calculated analytically. In particular, it is shown that the repulsive force exerted on a point charge due to polyhedral surfaces can be obtained analytically by evaluating the gradient of the following function

\[
\Phi(x, y, z) = \begin{cases} 
\tan^{-1}\left(\frac{x}{y \sqrt{x^2 + y^2 + z^2}}\right) & \text{for } m=3, \\
\frac{1}{z} & \text{for other values of } m.
\end{cases}
\]

at some \((x, y, z)\)'s for \(m=3\).

In general, the force exerted on a point due to polyhedral object surfaces, which will be used in the proposed skeletonization algorithm discussed next, can be obtained by summing the forces due to individual polygonal object faces. Finally, it is also shown in [24] for \(m=3\) that the generalized potential will diverge for a point charge located on the surface of a polyhedral object, i.e., the basic requirement for the proposed skeletonization algorithm is satisfied.

3. A New Generalized Cylinder Approach

In this section, we will propose a new approach to deriving a GC-based shape description of 3D objects based on the generalized potential model reviewed in the previous section. The GC representation consists of a GC axis and a cross-section function defined along this axis. Assume the two ends (cross-sections) of a 3D object are given in advance, the GC axis can be derived by identifying the potential valley connecting the centroid points of them. Cross-sections at different locations along the GC axis are then obtained by a simple algorithm. Thus, given object surface and its two end cross-sections, the proposed approach to deriving its GC representation can be divided into two phases, as shown in Fig. 3. In the first phase, the GC axis is generated using the potential-based skeletonization algorithm by connecting the two seeds. Then, the cross-sections at different locations along the derived GC axis are obtained in the second phase.

Fig. 3: The two phases for obtaining the GC description of an object.

The algorithms for the derivation of the GC axis and the associated cross-sections of a 3D object will be presented in Sections 3.1 and 3.2, respectively. In Section 3.3, the possibilities of using alternative seed points in the deriving GC representation are discussed.

3.1 GC axis generation – a review

Since the GC axis usually lies in the middle of an object, which is in a way similar to an object skeleton, the proposed approach adopts the potential-based skeletonization algorithm presented in [1], as reviewed next, to obtain the GC axis. The potential-based skeletonization algorithm identifies the potential valleys connecting several input seed points, which are selected as the convex vertices of the object (i.e., end points of the derived object skeleton), to obtain the object skeleton. The algorithm for generating the GC axis is essentially the same as the skeletonization algorithm except that the input seeds are selected as the centroids of two cross-sections of the object instead the convex vertices. With the boundary of a polyhedron charged according to the generalized potential model described in the previous section, the basic requirement for skeletonization, i.e., the potential will diverge at boundaries of polyhedral objects, is guaranteed. Inside the object boundary, the potential valley corresponds to 3D locations where the potential...
has locally minimal value in the 2D subspace perpendicular to the direction of the potential gradient. The skeletonization approach proposed in [1] obtained the object skeleton by, starting from selected seed points, traversing the corresponding potential valleys until a potential minimum is reached. The basic computation of the MAT skeleton, starting with one of the seed point, is described in the following algorithm:

Algorithm Potential_Skeleton

Step 1: Follow the direction of the force to traverse the skeleton (potential valley) until a zero force is obtained, i.e., a potential minimum is reached.

Step 2: Repeat Step 1 for each of the seed points.

Step 3: End the skeleton computation if there is only one potential minimum.

Step 4: Derive additional skeleton branches by identifying potential valleys connecting neighboring potential minima.

In Step 1, the force following is performed inside the polyhedral boundary. For a closed polyhedron, there exists at least one potential minimum since the potential and thus the force field will diverge at its surface [24]. If the number of potential minima obtained from the above force following procedure is equal to one, the complete skeleton is generated and Step 4 is not necessary. Otherwise, the potential minima need to be connected to form the complete MAT skeleton. To estimate the potential skeleton between these potential minima, we first connect each pair of neighboring minima with a virtual line segment. Equally spaced point samples are then selected along the line segment as initial guesses of point samples along the associated skeleton branch. The potential valley between the two minima is then estimated by searching the minimum potential location in the 2D subspace perpendicular to the virtual line segment at each of these samples.

Fig. 4(a) shows the skeleton of a rectangular hexahedron obtained with Potential_Skeleton. The seed points, where the force following procedure of the algorithm start, are selected as the eight convex corners of the object. (In the computer implementation, the selected seed points are located inside the object boundary and are slightly away from those convex corners.) The derived skeleton is closely related to the media surface derived in [25], as discussed in [1]. Obviously, the derived skeleton differs from the GC axis which is defined in Sec. 1 as the principal axis in reflecting the topology of an object. However, if we select the seed points as the centroid points of the two ends (cross-sections) of the object instead of the eight corners, a GC axis will be obtained directly with the same algorithm, as shown in Fig. 4(b).

Fig. 4: (a) A rectangular hexahedron and its MAT skeleton. (b) A GC axis of the rectangular hexahedron.

The main difference between the GC axis and the object skeleton in describing the object shape can be stated most conveniently by considering prismatic objects. While the axis and the cross-section of GC will simply be the principal axis and the cross-section of a prism, respectively, the object skeleton obtained with the proposed approach are more complex near the two ends of the prism where the topology of the skeleton is determined through the selection of seeds (which are convex vertices of the cross-sections in our case). In general, for an object consists of elongated parts, both representations will reflect similar topological (skeletal) properties of each part of the object, except for the above differences, and in turn those of the whole object.

In addition, different from the object skeleton, the GC representation is not unique in general. For example, the GC axis can have either horizontal or vertical direction for a rectangle in the 2D space. As shown in Fig. 5(a), we may select points X and Y as two seeds and generate the GC axis XY. Similarly, points M and N can also be selected as seeds to generate the GC axis in a different direction, as shown in Fig. 5(b). Figs. 5(c) and (d) show the GC cross-sections corresponding to XY.

Fig. 5: The GC axis and cross-sections of a rectangle in the 2D space.

1 In some complex cases where the line segment intersects the object boundary, a more refined estimation based on the visibility graph method is necessary. See [1] for more detailed discussion.

2 Theoretically, the MAT skeleton and GC axis are both continuous curves, however, they are searched step by step in our implementation and then yield a discrete form of MAT skeleton and GC axis.
and $MN$, respectively. In Sec. 3.3, some suggestions for seed point selection will be given from the observation on the different GC axes and cross-sections.

### 3.2 Cross-sections generation

The algorithm for obtaining the cross-sections of a generalized cylinder is based on the definition proposed by Binford: a cross-section of an object is called a cross-section of a generalized cylinder if its surface normal is parallel to the GC axis. For example, polygon $ABCD$ shown in Fig. 6 is one of the GC cross-sections of the rectangular solid since the normal of $ABCD$ is parallel to the GC axis.

Fig. 6: One GC cross-section $ABCD$ of a rectangular solid.

Given object surface and the derived GC axis, the following simple procedure for obtaining the GC cross-sections for the sequence of $n$ sampling points $S_i$, $1 \leq i \leq n$, of the derived GC axis can be summarized as follows.

**Algorithm Cross_Section_Gen**

**Step 1**: Set $i = 1$.

**Step 2**: Find the plane $P_i$ which is perpendicular to $S_iS_{i+1}$ at $S_i$.

**Step 3**: Compute all intersection points and line segments of $P_i$ and the object surfaces.

**Step 4**: Connect these intersections sequentially to form a polygon, i.e., one cross-section of the object.

**Step 5**: $i \leftarrow i + 1$, repeat Step 3 if $i \leq n$.

**Step 6**: End of the algorithm.

According to the above discussion, one can see clearly that the selection of seed points plays a crucial role in the GC algorithm proposed in this paper. In general, seed points should be selected such that a GC axis with smaller curvature can be generated and the corresponding cross-sections will be simpler and neater complete shape description of the object which includes a set of cross-sections is then obtained in Phase 2 using Cross_Section_Gen, as shown in Fig. 7(b). Obviously, the selection of seed points will greatly affect the result of the derived GC axis and cross-sections. A more detailed discussion about this issue will be given in the next section.

### 3.3 Selection of seed points

In this subsection, we will discuss the influence of seed point selection on the derived GC axis and cross-sections. Consider a rectangular hexahedron shown in Fig. 8. The MAT skeleton of the hexahedron obtained by Potential_Skeleton using eight convex corners as seed points is shown in Fig. 8(a). As for the GC axis of the object, Fig. 8(b) shows the axis obtained using two of the eight corner points as seeds whereas centroids of the front and back end surfaces of the object are selected for the axis shown in Fig. 8(c). The corresponding cross-sections obtained by Cross_Section_Gen for Figs. 8(b) and (c) are shown in Figs. 8(d) and (e), respectively.

Fig. 8: The MAT skeleton and different GC representations of a rectangular hexahedron (see text).

Obviously, the cross-sections shown in Fig. 8(d) are too complex due to the large curvature of the GC axis at two locations, as shown in Fig. 8(b). On the other hand, the GC axis shown in Fig. 8(c) goes straight from the front to the back end of the object while maintaining maximum distances from the other four sides of the object. Thus, the cross-sections derived for the GC axis shown in Fig. 8(c) are much simpler and more appropriate as the representation of the rectangular solid.

According to the above discussion, one can see clearly that, the selection of seed points plays a crucial role in the GC algorithm proposed in this paper. In general, seed points should be selected such that a GC axis with smaller curvature can be generated and the corresponding cross-sections will be simpler and neater.
for representing the object. This is especially the case for elongated object with cross-sections of identical size and shape. However, a systematic way of identifying optimal seed points for an object of arbitrary shape is yet to be established.

4. Experimental Results and Discussion

In this section, we will show simulation results of generating the potential-based GC axis and cross-sections of simple as well as more complex objects using the algorithm presented in the previous section. In each of the examples, the seed points used in the force following procedure are assumed to be given in advance that appropriate GC representation of each object can be generated. The complex object models are obtained from a database of 3D objects established by Washington State University, NETLIB scientific data repository. In general, it takes more computation time to derive the GC representation for more complex objects. In some cases, for objects containing multiple parts, a decomposition of an object into some simple parts is needed for the proposed approach to be applied properly. The algorithm is implemented with the C programming language and all examples are tested on a Sun Sparc Ultra-1 workstation with SunOS 4.14 and 128MB Main Memory.

4.1 Simple models

In this subsection, the proposed approach is applied to obtain the GC representations of objects of simple shapes. For each object, it is assumed that the two end cross-sections, as well as the two ends of the GC axis, are given in advance. Accordingly, the latter are used as seed points in the force following procedure of Potential_Skeleton in each example.

Fig. 9: GC representation of echelon hexahedron: (a) GC axis. (b) cross-sections.

Fig. 10: GC representation of dodecahedron: (a) GC axis. (b) cross-sections.

Fig. 11: GC representation of icosahedron: (a) GC axis. (b) cross-sections.

Fig. 12: GC representation of 3D polyhedron with three long rectangular polyhedra crossed over: (a) GC axis. (b) cross-sections.

Fig. 13: GC representation of tetrahedron: (a) GC axis. (b) cross-sections.

Fig. 9 shows the GC representation obtained with the proposed approach for an echelon hexahedron. Since only one potential minimum is obtained with Step 1 of Potential_Skeleton, the connecting process given in Step 4 is not needed. Due to the symmetry of the object shape, the GC axis corresponds to a line segment. The resultant GC axis and cross-sections are generated within 0.098 and 0.008 seconds, respectively. Similar results can be obtained for the potential-based GC representations shown in Figs. 10 and 11. The computation times for the above examples are summarized in Table 1. Note that Figs. 7 and 9 give the SHGC representation (but not Figs. 10 and 11) since there is no change in shape for the cross-sections. In Fig. 11, two object vertices, instead of face centers, are used as seed points, i.e., the area of the two end cross-sections of its GC representation is equal to zero.

The 3D objects considered so far have only one potential minimum in their interior; therefore, additional connecting process given in Step 4 of Potential_Skeleton is not required. Figs. 12 and 13 show two examples in which more than one potential minimum can be found. In

3 For the examples considered in this section, only a subset of the total number of cross-sections calculated are shown in the corresponding figures for better illustrations.
Fig. 12(a) (Fig. 13(a)), distance between the two potential minima obtained with Step 1 of Potential_Skeleton is equal to 3/4 (slightly less than 3/5) the length of the GC axis. Due to additional connecting process (Step 4 of Potential_Skeleton) required in generating the GC axis, more computation time (see Table 1) is needed. More in depth analysis of computational complexity will be discussed later in Sec. 4.3. As for the cross-sections, since there is no restriction on the changes of their size, shape, etc., the changes may not be continuous, e.g., in Fig. 11.

4.2 Complex models

In this subsection, the object models are obtained from a database of 3D objects, which can be found at the website – http://sampl.eng.ohio-state.edu/~sampl/data/3DDB/Models/. Each object in the database is stored in a file format specifying its vertices, polygons, and surfaces. The main difference between simple and complex object models is the latter use a lot of polygons to describe an object surface such that the object looks more real.

Figs. 14 - 17 show experimental results of the proposed approach for some objects represented by complex models wherein one potential minimum is obtained with Potential_Skeleton for each example. The computation times for the above examples are summarized in Table 1. Due to more complex boundary description, these computation time results are in general longer than those for the simple models. Fig. 18 shows another example which requires additional connecting process to connect the two potential minima while deriving the GC axis. It is obviously that the GC representation shown in Fig. 18 can be regarded as two simple SHGCs connected back-to-back. (Similar observations can be made for Fig. 12 but not for Figs. 10 and 11.) In this case, the axes of the two SHGC are connected into a single line segment.

In general, for an object with complex geometry, decomposing the object into simpler parts then obtaining the GC representation for each part may be necessary. One way of obtaining the GC representation for such a complex shaped object is to decompose the object into simpler parts after the GC axis of the complete object is derived. Then, cross-sections of each part of the object are obtained. For example, Fig. 19 shows a Y-shaped object composed of two pipes, a slim and short pipe and a wider and longer one. First, the GC axis can be derived by selecting three seed points\(^4\) using Potential_Skeleton. The connecting process is not required since the force following procedure yields an unique potential minimum, as shown in the top of Fig. 19. The resultant GC axis is generated within 10.256 seconds. Secondly, the object as well as its GC axis (the separated GC axes are not shown in the figure) are divided into two parts. Thirdly, the two parts are processed separately to derive their cross-sections using Cross_Section_Gen. Finally, the entire GC representation of the object is obtained by merging the two GC representation, as shown in the bottom figure of Fig. 19. The computation time spent on obtaining the 22 cross-sections of the left part is 0.256 seconds, while the right part takes 0.114 seconds to generate 9 cross-

\(^{4}\) It is readily observable that the resultant GC representation is not good enough if we select only two seed points. Therefore, three seed points are manually identified corresponding to three end cross-sections (circular faces) in this case.
sections.

Fig. 19: Generating the GC representation of a complex shaped object which need to be divided into two parts and processed separately.

In describing the simple parts of a complex shaped object, some important subclasses of GCs such as SHGC, PRCGC, and Circular PRGC are commonly used. Examples of these three subclasses of GCs are shown in Fig. 2. In the following examples, some of such models are created by 3D Studio Max to test the proposed approach. Figs. 20(a) and (b) show the GC axis and cross-sections of a SHGC, respectively, generated by the proposed approach. Similar results are shown in Figs. 21 and 22 for a PRCGC and a Circular PRGC, respectively.

4.3 Computation time

The proposed approach is not only easy to implement, the associated algorithm is also of low time complexity because the generalized potential field is analytically tractable. Suppose a polyhedron is composed of \(n\) polygons. For each step of the force following procedure, the complexity for calculating the repulsive force exerted on a point sample on the GC axis due to those polygons is \(O(n)\). Thus, for obtaining \(m'\) point samples of the GC axis through force following, the time complexity is \(O(nm')\). Similarly, it is not hard to show that if \(m''\) points are needed to connect the potential minima obtained from the force following procedure, the time complexity for finding these \(m''\) samples of the GC axis is \(O(nm'')\). Thus, the time complexity for finding the complete GC axis is \(O(nm)\), where \(m=m'+m''\) is equal to the total number of point samples of the GC axis. Therefore, the computation time for deriving the GC axis is approximately a linear function of the total number of object polygons multiplying the number of point samples of the GC axis. Table 2, for example, shows the computation time for deriving the GC axis of the experimental examples where the connection process is not required. Fig. 23, which is plotted according to the data listed in Table 2, demonstrates that the computation time is indeed very close to a linear function of Polygons * \(S_{c}\). As for the generation of cross-sections, since the computation complexity is highly depend on the shape of an object as well as the distribution of surface polygons, its analysis is not straightforward in general.

5. Conclusion

In this paper, we have proposed a new approach to deriving a GC-based shape description of 3D objects using the generalized potential model. The proposed approach of deriving the GC axis is an extension of the potential-based skeletonization approach introduced in [1]. The GC representation which consists of this axis and corresponding cross-sections can be generated automatically for the two ends of the axis given as inputs. In general, the derived GC representations will yield better approximation of object shape than that based on simpler
subclasses of GC since there is, in principle, no restriction on the topology of the GC axis and the shape of the cross-sections. On the other hand, any simpler subclass of the GC which has a simpler GC topology may also be obtained from the potential-based GC representation through appropriate simplifications, in cross-section/axis descriptions, of the latter.

References


### Table 1: Computation Times for Different GC Representations

<table>
<thead>
<tr>
<th>Process</th>
<th>Objects</th>
<th>Vertices</th>
<th>Polygons</th>
<th>Number of cross-sections</th>
<th>CPU Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connection Not Required</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rectangular Hexahedron</td>
<td>8</td>
<td>6</td>
<td>47</td>
<td></td>
<td>0.098</td>
</tr>
<tr>
<td>Echelon Hexahedron</td>
<td>8</td>
<td>6</td>
<td>17</td>
<td></td>
<td>0.071</td>
</tr>
<tr>
<td>Dodecahedron</td>
<td>20</td>
<td>12</td>
<td>57</td>
<td></td>
<td>0.51</td>
</tr>
<tr>
<td>Icosahedron</td>
<td>14</td>
<td>20</td>
<td>78</td>
<td></td>
<td>0.786</td>
</tr>
<tr>
<td>Adapter</td>
<td>79</td>
<td>50</td>
<td>16</td>
<td></td>
<td>0.569</td>
</tr>
<tr>
<td>Cap</td>
<td>639</td>
<td>386</td>
<td>15</td>
<td></td>
<td>3.54</td>
</tr>
<tr>
<td>Ball2</td>
<td>994</td>
<td>1024</td>
<td>31</td>
<td></td>
<td>16.207</td>
</tr>
<tr>
<td>Propane</td>
<td>128</td>
<td>66</td>
<td>55</td>
<td></td>
<td>2.333</td>
</tr>
<tr>
<td><strong>Connection Required</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D cross</td>
<td>24</td>
<td>30</td>
<td>89</td>
<td></td>
<td>0.981</td>
</tr>
<tr>
<td>Tetrahedron</td>
<td>6</td>
<td>5</td>
<td>64</td>
<td></td>
<td>12.122</td>
</tr>
<tr>
<td>Column1</td>
<td>392</td>
<td>262</td>
<td>56</td>
<td></td>
<td>45.038</td>
</tr>
<tr>
<td><strong>Special</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bigwye</td>
<td>291</td>
<td>173</td>
<td>22</td>
<td></td>
<td>10.256</td>
</tr>
<tr>
<td><strong>Subclasses of GC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHGC</td>
<td>650</td>
<td>1238</td>
<td>125</td>
<td></td>
<td>64.357</td>
</tr>
<tr>
<td>PRCGC</td>
<td>456</td>
<td>908</td>
<td>97</td>
<td></td>
<td>38.565</td>
</tr>
<tr>
<td>Circular PRGC</td>
<td>456</td>
<td>908</td>
<td>97</td>
<td></td>
<td>36.708</td>
</tr>
</tbody>
</table>

### Table 2: GC Axis Generation Time

<table>
<thead>
<tr>
<th>Process</th>
<th>Objects</th>
<th>Polygons</th>
<th>$S_i$</th>
<th>Polygon * $S_i$</th>
<th>CPU Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connection Not Required</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Echelon Hexahedron</td>
<td>6</td>
<td>34</td>
<td>204</td>
<td></td>
<td>0.071</td>
</tr>
<tr>
<td>Rectangular Hexahedron</td>
<td>6</td>
<td>94</td>
<td>564</td>
<td></td>
<td>0.098</td>
</tr>
<tr>
<td>Dodecahedron</td>
<td>12</td>
<td>114</td>
<td>1368</td>
<td></td>
<td>0.51</td>
</tr>
<tr>
<td>Adapter</td>
<td>50</td>
<td>32</td>
<td>1600</td>
<td></td>
<td>0.569</td>
</tr>
<tr>
<td>Icosahedron</td>
<td>20</td>
<td>156</td>
<td>3120</td>
<td></td>
<td>0.786</td>
</tr>
<tr>
<td>Propane</td>
<td>66</td>
<td>110</td>
<td>7260</td>
<td></td>
<td>2.333</td>
</tr>
<tr>
<td>Cap</td>
<td>386</td>
<td>30</td>
<td>11580</td>
<td></td>
<td>3.54</td>
</tr>
<tr>
<td>Ball2</td>
<td>1024</td>
<td>62</td>
<td>63488</td>
<td></td>
<td>16.207</td>
</tr>
<tr>
<td><strong>Connection Required</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHGC</td>
<td>908</td>
<td>194</td>
<td>176152</td>
<td></td>
<td>36.708</td>
</tr>
<tr>
<td>PRCGC</td>
<td>908</td>
<td>194</td>
<td>176152</td>
<td></td>
<td>38.565</td>
</tr>
<tr>
<td>SHGC</td>
<td>1238</td>
<td>250</td>
<td>309500</td>
<td></td>
<td>64.357</td>
</tr>
</tbody>
</table>