(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
02 November 2023 (02.11.2023)

WIPO | PCT

(10) International Publication Number
WO 2023/211622 A1

(51) International Patent Classification:
G06T 17/00 (2006.01) *G06T 19/00* (2011.01)

(21) International Application Number:
PCT/US2023/016956

(22) International Filing Date:
30 March 2023 (30.03.2023)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
17/661,193 28 April 2022 (28.04.2022) US

(71) Applicant: **APPLE INC.** [US/US]; One Apple Park Way, Cupertino, California 95014 (US).

(72) Inventors: **MAMMOU, Khaled**; Apple Inc., One Apple Park Way, Cupertino, California 95014 (US). **BIAGIOLI, Adrian A**; Apple Inc., One Apple Park Way, Cupertino, California 95014 (US). **TOLANI, Deepak S**; Apple Inc., One Apple Park Way, Cupertino, California 95014 (US).

(74) Agent: **FLETCHER, Michael G.** et al.; P.O. Box 692289, Houston, Texas 77269 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE,

(54) Title: APPROXIMATE HIERARCHICAL CONVEX DECOMPOSITION

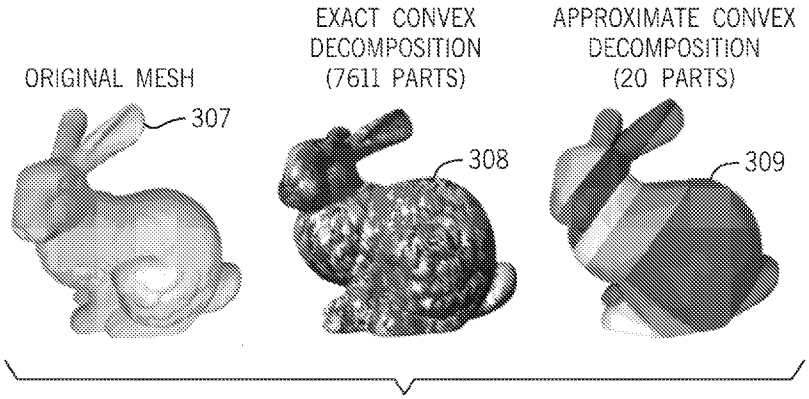

FIG. 3

(57) Abstract: A method of decomposing a three-dimensional representation of an object into a plurality of convex hulls can include instantiating a cluster priority queue in a computing system memory that initially contains a cluster corresponding to the three-dimensional representation of the object, computing with a processor of the computing system a concavity measure for each cluster in the cluster priority queue, and, for the cluster with the highest concavity measure: (1) computing with the processor a cut plane that divides the cluster corresponding to the three-dimensional representation of the object into two new clusters, each of the two new clusters having a corresponding convex hull, wherein computing a cut plane includes performing a hierarchical search of potential cut planes, (2) removing the cluster corresponding to the three-dimensional representation of the object from the cluster priority queue, and (3) adding the two new clusters to the cluster priority queue.

WO 2023/211622 A1

[Continued on next page]

SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

APPROXIMATE HIERARCHICAL CONVEX DECOMPOSITION

BACKGROUND

[0001] Three dimensional objects may be represented in a computing system as a mesh of polygons (*e.g.*, triangles), a point cloud, or other suitable representation. In some applications, for example video games or computer animation, detecting collisions between such objects may be desirable or even essential. To speed computational detection of such collisions, simplified representation of the 3D objects may be used. However, in some applications, many heretofore used simplification techniques have resulted in sub-optimal performance (false collision detection, for example).

SUMMARY

[0002] Thus, what is needed in the art is an improved technique for accurately representing 3D objects in a simplified manner that allows for improved accuracy of collision detection and similar function while also reducing computational complexity to a level that can be accommodate by the computing device of interest.

[0003] A method of decomposing a three-dimensional representation of an object into a plurality of convex hulls can include instantiating a cluster priority queue in a computing system memory that initially contains a cluster corresponding to the three-dimensional representation of the object, computing with a processor of the computing system a concavity measure for each cluster in the cluster priority queue, and, for the cluster with the highest concavity measure: (1) computing with the processor a cut plane that divides the cluster corresponding to the three-dimensional representation of the object into two new clusters, each of the two new clusters having a corresponding convex hull, wherein computing a cut plane includes performing a hierarchical search of potential cut planes, (2) removing the cluster corresponding to the three-dimensional representation of the object from the cluster priority queue, and (3) adding the two new clusters to the cluster priority queue.

[0004] Performing a hierarchical search of potential cut planes can further include a recursive cut plane selection procedure that minimizes concavity of all produced clusters after applying a predetermined number of successive cut planes. Computing with the processor of the computing

system a cut plane can further include accepting user input to select or modify the computed cut plane. Performing a hierarchical search of potential cut planes can further include: (1) determining a first plurality of cut plane orientations and for each of the first plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation, and selecting one of the first plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions; and (2) determining a second plurality of cut plane orientation based on the selected one of the first plurality of cut plane orientations, for each of the second plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation, and selecting one of the second plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions. The first and second pluralities of cut plane orientation can be selected based on successive subdivisions of a unitary octahedron. The one or more cost functions can include a balance cost function that prefers cut planes that produce approximately equally sized clusters. The one or more cost functions can include a cross-section cost function that prefers cut planes that minimize a cross-sectional area of the intersection of the cut plane and the input cluster. The one or more cost functions can include a compactness cost function that prefers cut planes that minimize a sum of surface areas of resulting clusters.

[0005] The above-described method can further include, for each pair of clusters in the priority queue, determining whether the pair of clusters can be merged without increasing concavity above a predetermined threshold. If a pair of clusters in the priority queue can be merged without increasing concavity above a predetermined threshold, the method can still further include merging the pair of clusters, removing the pair of clusters from the priority queue, and placing the merged cluster in the priority queue. The method can further include accepting user input with respect to one or more cluster pairs to merge and—for each cluster pair indicated by the user to be merged—merging the pair of clusters removing the pair of clusters from the priority queue and placing the merged cluster in the priority queue.

[0006] A method of decomposing a three-dimensional representation of an object into a plurality of convex hulls can include computing with a processor of a computing system a cut plane that divides a cluster stored in a memory of the computing system and corresponding to the three-dimensional representation of the object into two new clusters. Each of the two new clusters can have a corresponding convex hull. Computing a cut plane can include performing a hierarchical search of potential cut planes. Performing a hierarchical search of potential cut planes can include

a recursive cut plane selection procedure that minimizes concavity of all produced clusters after applying a predetermined number of successive cut planes. Computing with the processor of the computing system a cut plane can further include accepting user input to select or modify the computed cut plane.

[0007] Performing a hierarchical search of potential cut planes can further include: determining a first plurality of cut plane orientations; computing one or more cost functions associated with the cut plane orientation for each of the first plurality of cut plane orientations; selecting one of the first plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions; determining a second plurality of cut plane orientation based on the selected one of the first plurality of cut plane orientations; for each of the second plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation; and selecting one of the second plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions. The first and second pluralities of cut plane orientation can be selected based on successive subdivisions of a unitary octahedron. The one or more cost functions can include one or more cost functions selected from the group consisting of: (1) a balance cost function that prefers cut planes that produce approximately equally sized clusters, (2) a cross-section cost function that prefers cut planes that minimize a cross-sectional area of the intersection of the cut plane and the input cluster, and (3) a compactness cost function that prefers cut planes that minimize a sum of surface areas of resulting clusters.

[0008] Performing a hierarchical search of potential cut planes can further include a recursive cut plane selection procedure that minimizes concavity of all produced clusters after applying a predetermined number of successive cut planes. The above-described method can further include—for each pair of clusters in the priority queue—determining whether the pair of clusters can be merged without increasing concavity above a predetermined threshold and, if a pair of clusters in the priority queue can be merged without increasing concavity above a predetermined threshold: merging the pair of clusters, removing the pair of clusters from the priority queue, and placing the merged cluster in the priority queue. The method can still further include accepting user input with respect to one or more cluster pairs to merge, and, for each cluster pair indicated by the user to be merged: merging the pair of clusters, removing the pair of clusters from the priority queue and placing the merged cluster in the priority queue.

[0009] A method of performing a hierarchical search of potential cut planes for decomposing a three-dimensional representation of an object into a plurality of convex hulls can include determining a first plurality of cut plane orientations; for each of the first plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation; selecting one of the first plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions; determining a second plurality of cut plane orientation based on the selected one of the first plurality of cut plane orientations; for each of the second plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation; and selecting one of the second plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions.

[0010] Performing a hierarchical search of potential cut planes can include a recursive cut plane selection procedure that minimizes concavity of all produced clusters after applying a predetermined number of successive cut planes. Computing with a processor of the computing system a cut plane can further include accepting user input to select or modify the computed cut plane. The first and second pluralities of cut plane orientations can be selected based on successive subdivisions of a unitary octahedron. The one or more cost functions can include one or more cost functions selected from the group consisting of: a balance cost function that prefers cut planes that produce approximately equally sized clusters; a cross-section cost function that prefers cut planes that minimize a cross-sectional area of the intersection of the cut plane and the input cluster; and a compactness cost function that prefers cut planes that minimize a sum of surface areas of resulting clusters.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] Figure 1 illustrates a 3D object/mesh with a corresponding convex hull and a corresponding approximate convex decomposition.

[0012] Figure 2 illustrates various concepts relating to convexity and concavity for 2D objects, which can be extended to 3D objects.

[0013] Figure 3 illustrates a 3D object mesh with a corresponding exact convex decomposition and an approximate convex decomposition.

[0014] Figures 4 illustrates an algorithm for decomposing a 3D object into a plurality of approximate convex clusters.

[0015] Figure 5 illustrates a 3D object that is voxelized and represented by a plurality of convex pieces.

[0016] Figure 6 illustrates a technique for subdividing a half unit octahedron to generate a hierarchy of prospective cut planes.

[0017] Figures 7 illustrates an algorithm for determining a best cut plane for subdividing a 3D object into a plurality of approximate convex clusters.

[0018] Figure 8 illustrates a recursive cut plane selection technique.

[0019] Figures 9 illustrates a cluster merging algorithm.

[0020] Figure 10 illustrates a simplified block diagram of an exemplary computing system in which the algorithms described herein may be employed.

DETAILED DESCRIPTION

[0021] In the following description, for purposes of explanation, numerous specific details are set forth to provide a thorough understanding of the disclosed concepts. As part of this description, some of this disclosure's drawings represent structures and devices in block diagram form for sake of simplicity. In the interest of clarity, not all features of an actual implementation are described in this disclosure. Moreover, the language used in this disclosure has been selected for readability and instructional purposes, has not been selected to delineate or circumscribe the disclosed subject matter. Rather the appended claims are intended for such purpose.

[0022] Various embodiments of the disclosed concepts are illustrated by way of example and not by way of limitation in the accompanying drawings in which like references indicate similar elements. For simplicity and clarity of illustration, where appropriate, reference numerals have been repeated among the different figures to indicate corresponding or analogous elements. In addition, numerous specific details are set forth to provide a thorough understanding of the implementations described herein. In other instances, methods, procedures, and components have not been described in detail so as not to obscure the related relevant function being described. References to "an," "one," or "another" embodiment in this disclosure are not necessarily to the

same or different embodiment, and they mean at least one. A given figure may be used to illustrate the features of more than one embodiment, or more than one species of the disclosure, and not all elements in the figure may be required for a given embodiment or species. A reference number, when provided in a given drawing, refers to the same element throughout the several drawings, though it may not be repeated in every drawing. The drawings are not to scale unless otherwise indicated, and the proportions of certain parts may be exaggerated to better illustrate details and features of the present disclosure.

[0023] As discussed above, collision detection may be important for many applications, including for example, realistic physical interactions in video games and computer animation. To ensure real-time interactivity with a player/user, video game and 3D modeling software developers usually approximate the 3D models representing objects composing the scene (such as animated characters, static objects, etc.) by a set of simple convex shapes such as ellipsoids, capsules, or convex hulls. Figure 1 illustrates an example of a 3D object 101, which can be represented as a convex hull 102, which can be the smallest convex volume that contains the original object. Figure 1 also illustrates an approximate convex decomposition 103 representing 3D object 101, which uses multiple convex shapes rather than a single hull to represent and contain the original object 101. In some cases, either form of simplified representation can provide poor approximations of the actual surface. Concave surfaces, in particular, may be particularly difficult to represent using convex shapes, and the resulting poor degree of approximation can cause false collision detection and potentially other undesirable effects.

[0024] Figure 2 illustrates in two dimensions a few pertinent concepts for the present disclosure. More specifically, an object or shape 204 is said to be convex if it contains all lines directly connecting any two points within the object or shape. Conversely, an object or shape 205 is concave if it does not contain all lines directly connecting any two points X, Y within the object or shape. The concavity of an object is a measure the difference between the area (or volume) of the object and the smallest convex shape containing the area or volume. Thus star 207 is a concave object, pentagon 206 is the smallest convex shape containing the area or volume, and the sum of unshaded areas 208 provide a measure of concavity. (All of the foregoing concepts may be extended to three dimensional, and in fact higher dimensional, objects.)

[0025] Figure 3 illustrates an alternative to the convex hull approach described above with respect to Fig. 1, in which an exact convex decomposition 308 of an object represented by mesh 307 can be computed. This exact convex decomposition may exactly match the shape of the original mesh, but results in a relatively high number of parts. More specifically, computing an exact convex decomposition can include partitioning the input mesh 307 into a minimal set of convex sub-surfaces. Exact convex decomposition algorithms are NP-hard and therefore may not be practical for many applications in which computational resources and/or time to perform the required calculation are limited. To overcome these limitations, the exact convexity constraint may be relaxed and an approximate convex decomposition 309 of mesh 307 can be computed instead. In such cases, the goal can be to determine a partition of the mesh triangles with a minimal number of clusters, while ensuring that each cluster has a concavity lower than a user defined threshold.

[0026] Pre-existing approximate convex decomposition techniques have included a variety of deficiencies, including for example:

- Requiring high resolution voxelization or tetrahedralization, which can lead to high computational complexity and high memory requirements.
- Considering only axis-aligned cut planes to reduce complexity, which can lead to sub-optimal decompositions.
- Failing to robustly handle meshes with cracks (*i.e.*, not watertight meshes), potentially treating them as open surfaces.
- Potentially over-decomposing convex sub-parts because of approximation errors introduced by voxelization/tetrahedralization.
- Making sub-optimal decomposition decisions because cut planes are determined one at a time (*i.e.*, with no look-ahead strategy).
- Handling mainly or only 3D polygonal meshes (*e.g.*, being unable to handle implicit functions or point clouds).
- Not supporting semi-automatic decompositions (*i.e.*, not providing a user with the ability to alter or guide the decomposition process).

- Employing decomposition algorithms that are fixed, preventing improvement by learning from user-inputs or other inputs/measures.

Disclosed herein are various improved decomposition techniques that may address one or more of the foregoing deficiencies.

[0027] Appendix 1 lists an improved decomposition algorithm in pseudo-code form. Figure 4 illustrates the improved decomposition algorithm as a simplified flow chart. The following description focuses on the flow chart, but it will be appreciated that the pseudo code listing also depicts one technique for implementing the flow chart and may include additional details that may be optional for some embodiments. Such implementation may be provided as software running on a programmed processor, including one located in a server, desktop computer, portable computer, mobile electronic device (*e.g.*, smartphone or tablet computer), etc. Additionally, or alternatively, one or more components of the algorithm could be implemented using hardware assistance or acceleration, *e.g.*, in a graphics processing unit, image processing engine, or other suitable electronic hardware. An exemplary system that can include such implementations is described below with respect to Fig. 10.

[0028] Figure 4 illustrates a flowchart 400 of a decomposition algorithm. Beginning at block 411, the set of all clusters can be represented by "S". In block 413, "PQ" can be a priority queue of the clusters in S, ordered by their concavity. In block 415, it can be determined whether PW is empty and/or if S has reached the maximum cluster count. If so, then the algorithm can end. Otherwise, if the priority queue PQ is not empty and the maximum cluster count has not been reached, flow proceeds to block 419. In block 419, a cluster C can be selected from the priority queue, *i.e.*, the cluster with the highest concavity. In block 423, the best cut plane P for cluster C can be determined. (A best cut plane determination algorithm is described in greater detail below with respect to Appendix 2 and Fig. 7.). Then in block 425, the selected cluster C can be split with the selected best cut plane P to generate two new clusters CL, CR. (Although "left" and "right" nomenclature is used, it should be appreciated that any orientation of cut plane could be used.)

[0029] In block 427, it can be determined whether the cut performed in block 425 has produced a net concavity reduction that is above a selected threshold. This can ensure that the cut performed is "worthwhile." Turning back to Appendix 1, one exemplary concavity measure is given by:

$$\left|\frac{Concavity(C) - Concavity\ (C_L) - Concavity(C_R)}{Concavity(C)}\right| \geq MinConcavityReduction$$

where *Concavity* is a concavity measurement function that, as described above with respect to Fig. 2, measures the volume difference between the convex decomposition clusters and the original cluster and MinConcavityReduction is the threshold. Then, if an acceptable concavity reduction is provided by the split, the cluster C can be removed from priority queue PQ and split clusters CL and CR can be returned to the priority queue. Following this, flow can return to block 419, in which the next cluster C having the highest concavity can be selected. It should be appreciated that this could be either a pre-existing cluster member of S or one of the newly added clusters produced by the split.

[0030] Back at block 427, if the concavity reduction resulting from the cut is not above the selected threshold, then in block 431, it can be determined whether recursive cuts, *i.e.*, performing multiple cuts in sequence are enabled. If not, then the process may end, as the algorithm cannot reduce the concavity of the cluster having the highest concavity, meaning there is nothing further to do. Otherwise, if recursive cuts are enabled, in block 431, P1, P2, ...Pn can be best recursive cut plane candidates. For each candidate, the selected cluster C can be split to generate recursively split planes CLLn, CLRn, CRLn, and CRRn. (Again, left/right nomenclature is used, but it should be appreciated that the cut plane orientation can be arbitrary.) For each set of recursive splits, a concavity reduction rn can be determined, and for the recursive splits having the best concavity reduction, the resulting split clusters CLL, CLR, CRL, and CRR can be returned to the priority queue PQ and set of all clusters S in place of the source cluster C. Further details of recursive cut techniques are described below with respect to Fig. 8.

[0031] Additional aspects may be applied in conjunction with the above-described algorithm. For example, to handle meshes with cracks (*i.e.*, not watertight meshes) and to be able to extend the algorithm to surface representations other than polygonal meshes (*e.g.*, point clouds), a "winding number" may be employed to derive an implicit representation of the surface. Winding numbers are a mathematical construct known to those skilled in the art, and examples of the application of winding numbers to 3D graphics processing are described in "Fast Winding Numbers for Soups and Clouds" by Gavin Barill, *et al.*, which may be found at: https://www.dgp.toronto.edu/projects/fast-winding-numbers/fast-winding-numbers-for-soups-and-clouds-siggraph-2018-barill-et-

al.pdf. In short, a winding number is a number that defines how many times a curve (or, more generally, a surface) wraps around or encompasses a point. Thus, any point may be assigned a winding number derived value that indicates whether the point is inside or outside the surface.

[0032] For example, each 3D point Q can be assigned the scalar value:

$$f(Q) = W(Q) - \frac{1}{2},$$

where $W(Q)$ is the winding number of Q. If a cluster C was generated by applying n cut planes $(P_i)_{i=1\ldots n}$ to the initial surface, the value of the implicit function f_C associated with C at the 3D point Q can be given by:

$$f_C(Q) = \begin{cases} -1 & if\ d(Q,P_i) < 0\ i = 1 \ldots n \\ W(Q) - \frac{1}{2} & otherwise \end{cases}$$

Where $d(Q, P_i)$ is the signed distance of Q with respect to the plane P_i.

[0033] Figure 5 depicts the advantages of the above-described approach. Using a convex hull based approximation of the input surface 541 can allow avoidance high resolution voxelization or tetrahedralization, while offering a more accurate approximation 543 of the input surface S. The idea is to approximate surface S (541) with a set of convex pieces. Computing this initial decomposition makes it possible to support any surface representation (*e.g.*, polygonal meshes, implicit surfaces, point clouds, etc.). To compute the initial decompositions various approaches are possible, for example:

- Voxel-grid guided decomposition: The volume may be segmented into a relatively low-resolution set of volumes 542. Then, for each voxel intersecting the surface, a convex hull of the intersection between the volume enclosed by the surface and the voxel. The union of the volumes defined by these convex hulls provide an approximation of the initial volume bounded by S. It should be noted that that the grid could be uniform or not.
- The voxel-grid guided decomposition could be extended to leverage an octree structure (sparse voxelization) with more refinement around the blocks that intersect the surface. This can allow for reduction of memory requirements and the computational complexity of the ACD process. Exemplary sparse voxelization arrangements are described in https://www.seas.upenn.edu/~pcozzi/OpenGLInsights/OpenGLInsights-SparseVoxelization.pdf

[0034] As noted above, performance of the subdivision algorithm may be improved by using arbitrary cut plane orientations rather than the axially aligned cut planes of prior art methods. Figure 6 illustrates an exemplary hierarchical structure of orientation that may be used to accelerate the search for a best cut plane orientation. The hierarchical structure can be based on normal vectors to the faces of a half unitary octahedron 644. (A half octahedron may be used because the normal vectors of the other half are in the same orientation, just with a different sign.) The initial level 645 of the hierarchy H_0 corresponds to the vertices of a half unitary octahedron. Subsequent hierarchy levels $(H_i)_i$ (646–648) can be obtained by successively applying a 4-1 subdivision. Thus, H_1 (646) can include four additional orientations around a selected normal from H_0, with H_2 657 including four additional orientations around a selected normal from H_1, and so on. The hierarchical search procedure to determine a cut plane orientation is further described below with reference to Appendix 2 and Fig. 7.

[0035] Appendix 2 lists in pseudo-code form a best cut plane selection algorithm that may be used in connection with the decomposition algorithm described above. Figure 7 illustrates the best cut plane selection algorithm as a simplified flow chart. The following description focuses on the flow chart, but it will be appreciated that the pseudo code listing also depicts one technique for implementing the flow chart and may include additional details that may be optional for some embodiments. As described above, such implementation may be provided as software running on a programmed processor. Additionally, or alternatively, one or more components of the algorithm could be implemented using hardware assistance or acceleration, *e.g.*, in a graphics processing unit, image processing engine, or other suitable electronic hardware. An exemplary system that can include such implementations is described below with respect to Fig. 10.

[0036] Figure 7 illustrates a flow chart 700 of a best cut plane selection algorithm that may be employed in conjunction with the segmentation algorithm described above. Beginning at block 751, a best cut plane offset value dn can be computed for the offset plane corresponding to each normal associated with the current iteration through the hierarchy of levels. In the above-described example using subdivisions based on half unitary octahedrons, this can correspond to four cut plane directions. Once an associated offset value dn (*i.e.*, the position of the plane along the direction defined by the normal vector) is determined for each potential orientation, in block 752, the selected cluster C may be split into sub-clusters CLn and CRn using the selected cut planes (where n corresponds to one of the cut planes). In block 753, a cost function value can be computed

for the cuts associated with the respective cut planes. (Exemplary cost functions are described in greater detail below.) The cut plane with the lowest cost function value can be selected as the best cut plane from among the present candidates in block 754. Then, in block 755, it may be determined whether the algorithm has progressed completely through the levels of the hierarchical selection arrangement (*e.g.*, the successive subdivisions shown in Fig. 6). If so, then the process can end at block 756, with the best selected cut plane being used in the remainder of the above-described algorithm. Otherwise, the algorithm can proceed to the next level of the hierarchy by computing the adjacent cut planes for the current "best" cut plane in block 757, with these new adjacent cut plane candidates being passed to block 751, discussed above.

[0037] As mentioned above, a cost function value may be computed for each cut plane to determine which of the cut plane candidates is the "best." There are various cost functions that could be used, either separately or in combination. For example, a BalanceCost function could be defined to prefer cut planes that produce subdivided hulls having the most equal volumes. Such a cost function can be defined as:

$$\text{BalanceCost}(C, C_L, C_R) = \frac{|\text{Vol}(C_L) - \text{Vol}(C_R)|}{\text{Vol}(C_0)}$$

where C_0 is the cluster being subdivided (*i.e.*, the input mesh before any subdivision), C_L and C_R are the respective clusters defined by a split along the selected cut plane, and Vol is the volume of the respective cluster. A cross-section cost function can be defined to prefer cut planes that minimize the cross-sectional area of the intersection of cut plane P with cluster C. For example, such a cost function can be defined as:

$$\text{CrossSectionCost}(C, P) = \frac{\text{CSA}(C, P)}{0.5 \cdot \text{SA}(C_0)}$$

where CSA(C, P) is the cross sectional area defined by the intersection of cluster C with cut plane P and SA(C0) is the surface area of the input cluster. Finally, a compactness cost function can be defined to prefer cut planes that minimize the sum of the surface areas of the resulting split planes C_L and C_R. Such a cost function can be defined as:

$$\text{CompactnessCost}(C, C_L, C_R) = \frac{\text{SA}(C_L) + \text{SA}(C_R) - \text{SA}(C)}{\text{SA}(C_0)}$$

[0038] As noted above, these or other cost functions could be used as appropriate for a given application. Additionally, costs associated with each respective cost function may be combined to define an "aggregate" cost of a particular cut plane split, with the respective individual cost functions weighted accordingly. In one embodiment, an aggregate cut cost function can be defined as:

$$\text{CutCost} = \text{ConcavityCost} \cdot (1 + \text{BalanceCost} + \text{CompactnessCost} + \text{CrossSectionCost})$$

where ConcavityCost for a plane is the sum of the corresponding left and right cluster concavities, and the remaining cost functions are as described above.

[0039] In some cases, selecting one cut plane at a time may lead to sub-optimal decisions. For example. Thus, in at least some applications it may be desirable to provide a recursive cut plane selection procedure, which can optimize the concavity of all produced clusters after applying a selected number of *m* consecutive cut planes. An example is shown with respect to Fig. 8. In Fig. 8, a cluster 860 is depicted as a two-dimensional ring shape having a hollow center portion (although it will be appreciated that the discussed concepts can be extended to three-dimensional or higher order clusters). Cluster 860 can correspond to a convex hull 870, which completely contains cluster 860. Dividing cluster 860 along cut plane 861 can produce two new clusters 862 and 863, corresponding to hulls 871 and 872. In this case, there is no net reduction in concavity, as hulls 871 and 872 still contain the entirety of the space defined by the interior of cluster 860. Thus, a single level cut plane selection might reject this cut, even though a subsequent (recursive) cut along cut plane 864 can produce clusters 866, 867, 868, and 869 that correspond to hulls 873, 874, 875, and 876 in which the net concavity is substantially reduced. Thus, it may be desirable to consider recursive cut planes in which successive cuts are made with cut planes in pre-selected orientations with respect to the initial cut plane in determining the "best" cut plane(s).

[0040] Depending on the vagaries of the particular cluster to be subdivided and the particulars of the algorithm used, it is possible that the cluster may be over-segmented. To remediate such over segmentation, a cluster merging procedure may be employed. The cluster merging procedure can have the objective of merging clusters without introducing a significant increase in concavity. Appendix 3 and Fig. 9 describe an exemplary cluster merging algorithm. More specifically, Appendix 3 lists in pseudo-code form a cluster merging algorithm that may be used in connection

with the decomposition algorithm described above. Figure 9 illustrates the cluster merging algorithm as a simplified flow chart. The following description focuses on the flow chart, but it will be appreciated that the pseudo code listing also depicts one technique for implementing the flow chart and may include additional details that may be optional for some embodiments. As described above, such implementation may be provided as software running on a programmed processor. Additionally, or alternatively, one or more components of the algorithm could be implemented using hardware assistance or acceleration, *e.g.*, in a graphics processing unit, image processing engine, or other suitable electronic hardware. An exemplary system that can include such implementations is described below with respect to Fig. 10

[0041] Figure 9 illustrates a flowchart 900 of a cluster merging algorithm. Beginning with block 981, for each respective pair of clusters CL, CR, a merged cluster Cm can be computed, along with a corresponding cost of merging the clusters. The merger cost may be defined as a function of the increase in total concavity resulting from the merger. From among the mergers of the respective pairs of clusters, the merger having the lowest cost may be selected (block 982), and it can be determined (block 983) whether the selected lowest cost merger remains below a selected threshold. This threshold may be thought of as a maximum increase in concavity or a corresponding aggregate value increase. If the lowest cost merger is not less than the maximum cost, then the merger algorithm may end (block 985) as any further merger of clusters will result in an undesirable increase in concavity. Otherwise, if the lowest cost is below the selected threshold, the merged clusters CL, CR can be removed from the set of clusters S and the merged cluster Cm can be put into the set of clusters S.

[0042] In some cases, automatic decompositions may be improved by user inputs. Thus, the above-described techniques may be implemented as semi-automatic procedures in which a user could alter/refine the choice of a cut plane at any point of the decomposition process. For example, in the algorithm of Fig. 4, a user could supply the initial cut plane 423 (or the cut plane at any subsequent iteration). Additionally, or alternatively, the user could supply a sequence of respective cut planes in block 433. Similarly, in the algorithm of Fig. 7, a user could supply an initial cut plane orientation at varying levels of the hierarchy or could alter the number of hierarchical subdivisions or the number of subdivisions associated with each layer of the hierarchy. In some embodiments, the user could also provide varying degrees of input to the cluster merging algorithm as described in Fig. 9. For example, the user could provide input to merge two or multiple clusters.

A final decomposition may be obtained by switching multiple times between automatically computed cut/merge decisions and user-guided or provided cut/merge decisions.

[0043] Figure 10 illustrates a high-level block diagram of a computing system 1000 that may be employed to implement the graphics processing algorithms described herein. Processing system 1000 can include a processor 1091. This processor can take on a variety of forms, including one or more processors of one or more processor types, including without limitation, microprocessors, graphics processing units (GPUs), image processing units (IPUs), digital signal processors (DSPs), etc. The processor(s) may communicate with memory 1092, which may be in the form of random access memory. In some cases, different processor components may have access to different memory, such as dedicated memory for the CPU and dedicated memory for the GPU or may share access to a unified memory. Memory 1092 can be used to store both the data being operated on as well as program instructions implementing the various algorithms described herein. The processor may also communicate with storage 1093. Storage 1093 may be used for longer term storage of data as well as program instructions, and may include solid state storage, such as NAND flash storage, or other forms of storage such as magnetic storage.

[0044] Processor 1091 may also communicate with a display/user input output system 1094. Depending on the particular application and purpose of computing system 1000, varying display types and technologies may be provided, and different level of user input output (I/O) devices may be provided. For example, for a mobile phone or tablet computer, the display/user IO system can include an LCD or OLED touchscreen. In the case of a notebook computer, the display/user IO system can include an LCD or OLED display together with keyboard and pointer devices (such as a trackpad) for accepting user input, including without limitation the user inputs described above for guiding the selection of cut planes or merger of clusters, etc. In the case of a desktop computer, the display/user IO system can be expanded further to include multiple displays, a keyboard, various pointer devices such as trackpads, mice, digitizer/graphics tablets, etc. (It will also be appreciated that such IO devices may be used with other computer types as well.). Finally, processor 1091 can communicate with Comms system 1095, which can include network or other interfaces for communicating with other devices, including for receiving or transmitting cluster/mesh data to other systems. Such comms systems can include wired interfaces, such as Ethernet, serial ports, etc. or wireless interfaces such as WiFI, Bluetooth, NFC, etc.

[0045] It is anticipated that various permutations of the algorithms described herein can be implemented on a computing system 1000 to improve the functioning of such system by addressing one or more of the above-described deficiencies of pre-existing convex decomposition techniques. That said, it is also intended that such algorithms be used to improve the performance of any computing system to which they are applicable for improving the processing of three-dimensional image data by improving a convex decomposition process.

[0046] Entities implementing the present technology should take care to ensure that, to the extent any sensitive information is used in particular implementations, that well-established privacy policies and/or privacy practices are complied with. In particular, such entities would be expected to implement and consistently apply privacy practices that are generally recognized as meeting or exceeding industry or governmental requirements for maintaining the privacy of users. Implementers should inform users where personally identifiable information is expected to be transmitted in a wireless power transfer system and allow users to "opt in" or "opt out" of participation.

[0047] It is the intent of the present disclosure that personal information data, if any, should be managed and handled in a way to minimize risks of unintentional or unauthorized access or use. Risk can be minimized by limiting the collection of data and deleting data once it is no longer needed. In addition, and when applicable, data de-identification can be used to protect a user's privacy. For example, a device identifier may be partially masked to convey the power characteristics of the device without uniquely identifying the device. De-identification may be facilitated, when appropriate, by removing identifiers, controlling the amount or specificity of data stored (e.g., collecting location data at city level rather than at an address level), controlling how data is stored (e.g., aggregating data across users), and/or other methods such as differential privacy. Robust encryption may also be utilized to reduce the likelihood that communication between inductively coupled devices are spoofed.

[0048] The foregoing describes exemplary embodiments of approximate convex decomposition techniques for three-dimensional object representation in a computing system. Such techniques may be used in a variety of applications but may be particularly advantageous when used in conjunction with improving the performance of collision detection and similar algorithms for 3D gaming, animation, and similar applications. However, any system for which improved

segmentation of 3D graphics objects is desired may advantageously employ the techniques described herein. Although numerous specific features and various embodiments have been described, it is to be understood that, unless otherwise noted as being mutually exclusive, the various features and embodiments may be combined in various permutations in a particular implementation. Thus, the various embodiments described above are provided by way of illustration only and should not be constructed to limit the scope of the disclosure. Various modifications and changes can be made to the principles and embodiments herein without departing from the scope of the disclosure and without departing from the scope of the claims.

APPENDIX 1

```
function AUTOMATICSPLITCLUSTERS ( )
    let S = the list of all clusters

    let PQ = a priority queue of clusters, where clusters with higher concavity have higher priority
    Add all elements of S to PQ
    while PQ is not empty and |S| < maxClusterCount do
        let C = PQ.pop()
        let P = BestCutPlaneForCluster(C)
        let (C_L, C_R) = C.SplitWithPlane(P)
        if (Concavity(C) − Concavity(C_L) − Concavity(C_R)) /Concavity(C) ≥ minConcavityReductionPerSplit
 then
            S.remove(C)
            S.push(C_L, C_R)
            PQ.push(C_L, C_R)
        else if recursiveCutsEnabled and |S| + 3 ≤ maxClusterCount then
            let P_1, P_2, ..., P_k = BestRecursiveCutPlaneCandidates(C)
            (C_LL, C_LR, C_RL, C_RR) := ∅
            r_max := −∞
            for i ∈ 1, 2, ..., k do
                let (C_L^i, C_R^i) = C.SplitWithPlane(P_i)

                let P_L^i = BestCutPlaneForCluster(C_L^i)
                let P_R^i = BestCutPlaneForCluster(C_R^i)

                let (C_LL^i, C_LR^i) = C_L^i.SplitWithPlane(P_L^i)
                let (C_RL^i, C_RR^i) = C_R^i.SplitWithPlane(P_R^i)

                let r = (Concavity(C) − Concavity(C_LL^i) − Concavity(C_LR^i) − Concavity(C_RL^i) − Concavity(C_RR^i)) / Concavity(C)
                if r > r_max then
                    (C_LL, C_LR, C_RL, C_RR) := (C_LL^i, C_LR^i, C_RL^i, C_RR^i)
                    r_max := r
                end if
            end for
            if r_max ≥ minConcavityReductionPerSplit then
                S.remove(C)
                S.push(C_LL, C_LR, C_RL, C_RR)
                PQ.push(C_LL, C_LR, C_RL, C_RR)
            end if
        end if
    end while
end function
```

APPENDIX 2

```
function BestCutPlaneForCluster ( C )
    let j = 2
    let k = 4
    P_best = [a_best, b_best, c_best, d_best] := 0
    cost_best = ∞

    for each normal [a, b, c] in H_{j-1} do
        let d = BestCutPlaneOffsetForNormal(C, [a, b, c])
        let (C_L, C_R) = C.SplitWithPlane([a, b, c, d])
        if CutCost(C, C_L, C_R) < cost_best then
            P_best := [a, b, c, d]
            cost_best := CutCost(C, C_L, C_R)

        end if
    end for

    for i = j; i ≤ k; i := i + 1 do
        let N = the set of cut directions adjacent to [a_best, b_best, c_best] in H_i
        for each [a, b, c] ∈ N do
            let d = BestCutPlaneOffsetForNormal(C, [a, b, c])
            let (C_L, C_R) = C.SplitWithPlane([a, b, c, d])
            if CutCost(C, C_L, C_R) < cost_best then
                P_best := [a, b, c, d]
                cost_best := CutCost(C, C_L, C_R)
            end if
        end for
    end for

    return P_best
end function
```

APPENDIX 3

```
function AutomaticMergeClusters()
    let S = the list of all clusters
    while |S| > 1 do
        (C_L^best, C_R^best) := (∅, ∅)
        cost_best := ∞
        for each pair of clusters C_L, C_R ⊆ S do
            let C_M = MergeClusters(C_L, C_R)
            let cost = (Vol(Hull(C_M)) − Vol(Hull(C_L)) − Vol(Hull(C_R))) / Vol(Hull(C_M))
            if cost < cost_best then
                cost_best := cost
                (C_L^best, C_R^best) := (C_L, C_R)
            end if
        end for
        if cost_best ≤ maxAllowedMergeVolumeIncrease then
            let C_M = MergeClusters(C_L^best, C_R^best)
            Remove C_L^best, C_R^best from S
            Add C_M to S
        else
            break while
        end if
    end while
end function
```

CLAIMS

1. A method, performed by a computing system, of decomposing a three-dimensional representation of an object into a plurality of convex hulls, the method comprising:

instantiating a cluster priority queue in a memory of the computing system, wherein the cluster priority queue initially contains a cluster corresponding to the three-dimensional representation of the object;

computing with a processor of the computing system a concavity measure for each cluster in the cluster priority queue; and

for the cluster with the highest concavity measure:

computing with the processor a cut plane that divides the cluster corresponding to the three-dimensional representation of the object into two new clusters, each of the two new clusters having a corresponding convex hull, wherein computing a cut plane includes performing a hierarchical search of potential cut planes;

removing the cluster corresponding to the three-dimensional representation of the object from the cluster priority queue; and

adding the two new clusters to the cluster priority queue.

2. The method of claim 1 wherein performing a hierarchical search of potential cut planes includes a recursive cut plane selection procedure that minimizes concavity of all produced clusters after applying a predetermined number of successive cut planes.

3. The method of claim 1 wherein computing with the processor of the computing system a cut plane further comprises accepting user input to select or modify the computed cut plane.

4. The method of claim 1 wherein performing a hierarchical search of potential cut planes further comprises:

determining a first plurality of cut plane orientations;

for each of the first plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation;

selecting one of the first plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions;

determining a second plurality of cut plane orientation based on the selected one of the first plurality of cut plane orientations;

for each of the second plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation; and

selecting one of the second plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions.

5. The method of claim 4 wherein the first and second pluralities of cut plane orientation are selected based on successive subdivisions of a unitary octahedron.

6. The method of claim 4 wherein the one or more cost functions include a balance cost function that prefers cut planes that produce approximately equally sized clusters.

7. The method of claim 4 wherein the one or more cost functions include a cross-section cost function that prefers cut planes that minimize a cross-sectional area of the intersection of the cut plane and the input cluster.

8. The method of claim 4 wherein the one or more cost functions include a compactness cost function that prefers cut planes that minimize a sum of surface areas of resulting clusters.

9. The method of claim 1 further comprising:

for each pair of clusters in the priority queue, determining whether the pair of clusters can be merged without increasing concavity above a predetermined threshold; and

if a pair of clusters in the priority queue can be merged without increasing concavity above a predetermined threshold:

merging the pair of clusters;

removing the pair of clusters from the priority queue; and

placing the merged cluster in the priority queue.

10. The method of claim 9 further comprising:

accepting user input with respect to one or more cluster pairs to merge; and

for each cluster pair indicated by the user to be merged:

merging the pair of clusters;

removing the pair of clusters from the priority queue; and

placing the merged cluster in the priority queue.

11. A method, performed by a computing system, of decomposing a three-dimensional representation of an object into a plurality of convex hulls, the method comprising:

computing with a processor of the computing system a cut plane that divides a cluster stored in a memory of the computing system and corresponding to the three-dimensional representation of the object into two new clusters, each of the two new clusters having a corresponding convex hull, wherein computing a cut plane includes performing a hierarchical search of potential cut planes.

12. The method of claim 11 wherein performing a hierarchical search of potential cut planes includes a recursive cut plane selection procedure that minimizes concavity of all produced clusters after applying a predetermined number of successive cut planes.

13. The method of claim 11 wherein computing with the processor of the computing system a cut plane further comprises accepting user input to select or modify the computed cut plane.

14. The method of claim 11 wherein performing a hierarchical search of potential cut planes further comprises:

determining a first plurality of cut plane orientations;

for each of the first plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation;

selecting one of the first plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions;

determining a second plurality of cut plane orientation based on the selected one of the first plurality of cut plane orientations;

for each of the second plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation;

selecting one of the second plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions.

15. The method of claim 14 wherein the first and second pluralities of cut plane orientation are selected based on successive subdivisions of a unitary octahedron.

16. The method of claim 14 wherein the one or more cost functions include one or more cost functions selected from the group consisting of:

a balance cost function that prefers cut planes that produce approximately equally sized clusters;

a cross-section cost function that prefers cut planes that minimize a cross-sectional area of the intersection of the cut plane and the input cluster; and

a compactness cost function that prefers cut planes that minimize a sum of surface areas of resulting clusters.

17. The method of claim 14 wherein performing a hierarchical search of potential cut planes includes a recursive cut plane selection procedure that minimizes concavity of all produced clusters after applying a predetermined number of successive cut planes.

18. The method of claim 14 further comprising:

for each pair of clusters in the priority queue, determining whether the pair of clusters can be merged without increasing concavity above a predetermined threshold; and

if a pair of clusters in the priority queue can be merged without increasing concavity above a predetermined threshold:

merging the pair of clusters;

removing the pair of clusters from the priority queue; and

placing the merged cluster in the priority queue.

19. The method of claim 18 further comprising:

accepting user input with respect to one or more cluster pairs to merge; and

for each cluster pair indicated by the user to be merged:

merging the pair of clusters;

removing the pair of clusters from the priority queue; and

placing the merged cluster in the priority queue.

20. A method, performed by a computing system, of performing a hierarchical search of potential cut planes for decomposing a three-dimensional representation of an object into a plurality of convex hulls, the method comprising:

determining a first plurality of cut plane orientations;

for each of the first plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation;

selecting one of the first plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions;

determining a second plurality of cut plane orientation based on the selected one of the first plurality of cut plane orientations;

for each of the second plurality of cut plane orientations, computing one or more cost functions associated with the cut plane orientation;

selecting one of the second plurality of cut plane orientations having a lowest cost as determined by the one or more cost functions.

21. The method of claim 20 wherein performing a hierarchical search of potential cut planes includes a recursive cut plane selection procedure that minimizes concavity of all produced clusters after applying a predetermined number of successive cut planes.

22. The method of claim 20 wherein computing with a processor of the computing system a cut plane further comprises accepting user input to select or modify the computed cut plane.

23. The method of claim 20 wherein the first and second pluralities of cut plane orientation are selected based on successive subdivisions of a unitary octahedron.

24. The method of claim 20 wherein the one or more cost functions include one or more cost functions selected from the group consisting of:

a balance cost function that prefers cut planes that produce approximately equally sized clusters;

a cross-section cost function that prefers cut planes that minimize a cross-sectional area of the intersection of the cut plane and the input cluster; and

a compactness cost function that prefers cut planes that minimize a sum of surface areas of resulting clusters.

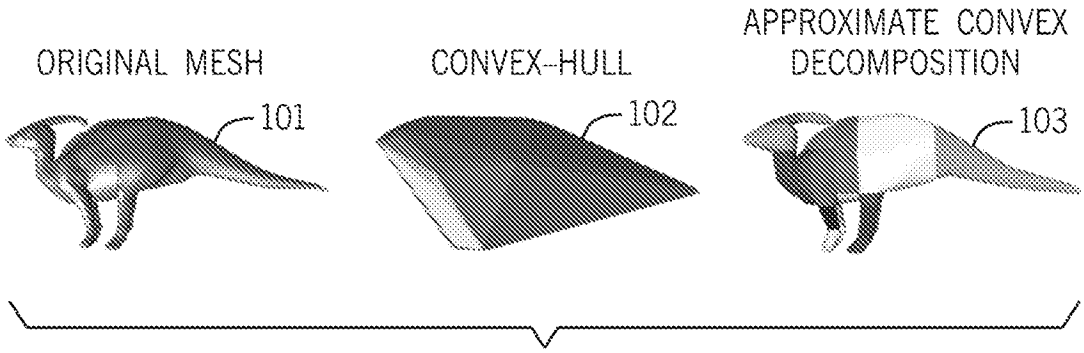

FIG. 1

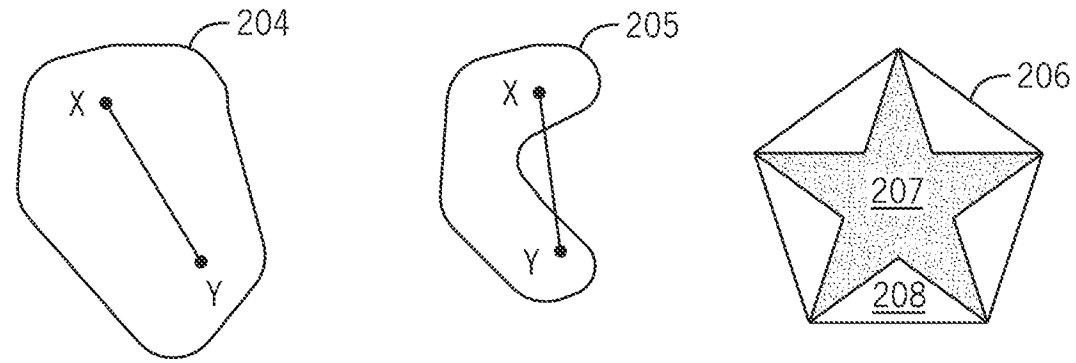

FIG. 2

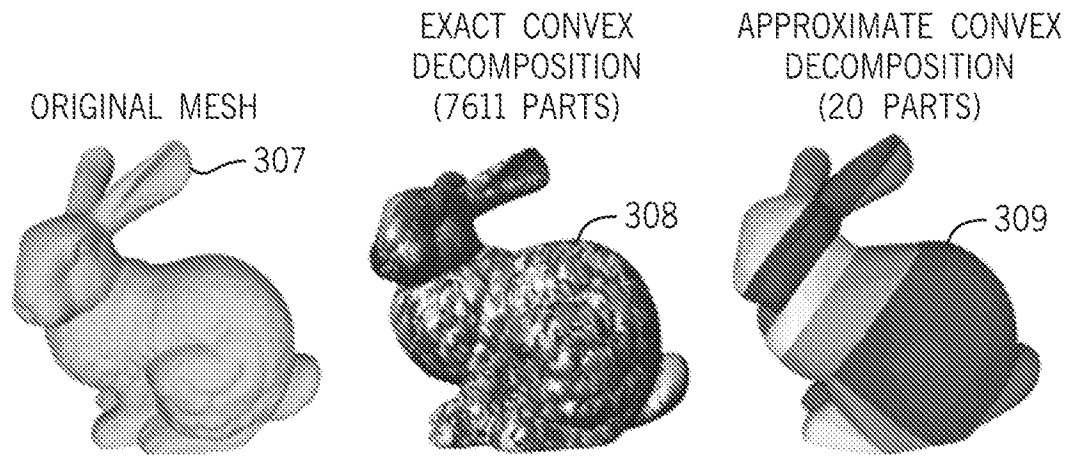

FIG. 3

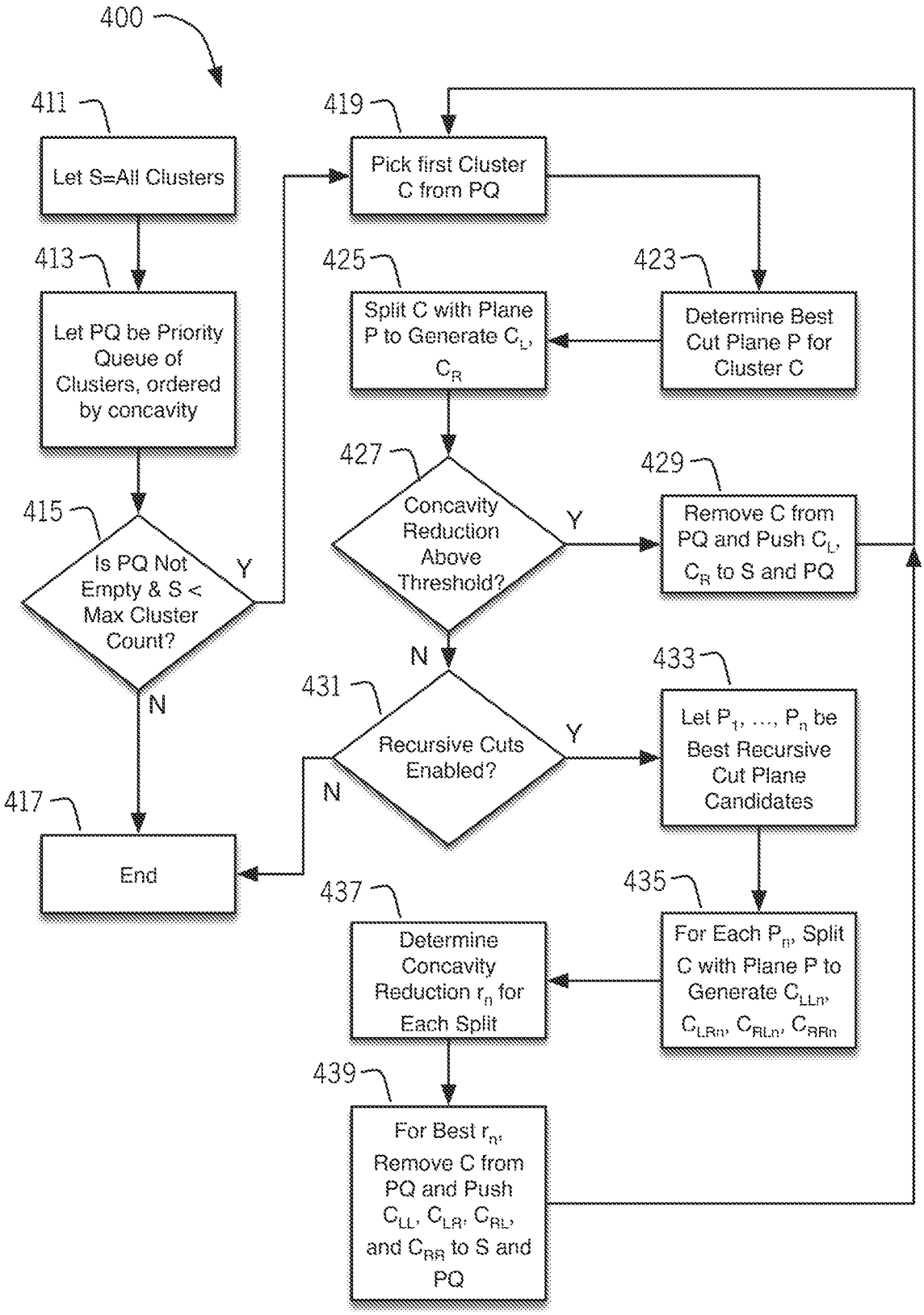
400
411
Let S=All Clusters
413
Let PQ be Priority Queue of Clusters, ordered by concavity
415
Is PQ Not Empty & S < Max Cluster Count?
Y
N
417
End
419
Pick first Cluster C from PQ
423
Determine Best Cut Plane P for Cluster C
425
Split C with Plane P to Generate C_L, C_R
427
Concavity Reduction Above Threshold?
Y
N
429
Remove C from PQ and Push C_L, C_R to S and PQ
431
Recursive Cuts Enabled?
Y
N
433
Let P_1, ..., P_n be Best Recursive Cut Plane Candidates
435
For Each P_n, Split C with Plane P to Generate C_{LLn}, C_{LRn}, C_{RLn}, C_{RRn}
437
Determine Concavity Reduction r_n for Each Split
439
For Best r_n, Remove C from PQ and Push C_{LL}, C_{LR}, C_{RL}, and C_{RR} to S and PQ

FIG. 4

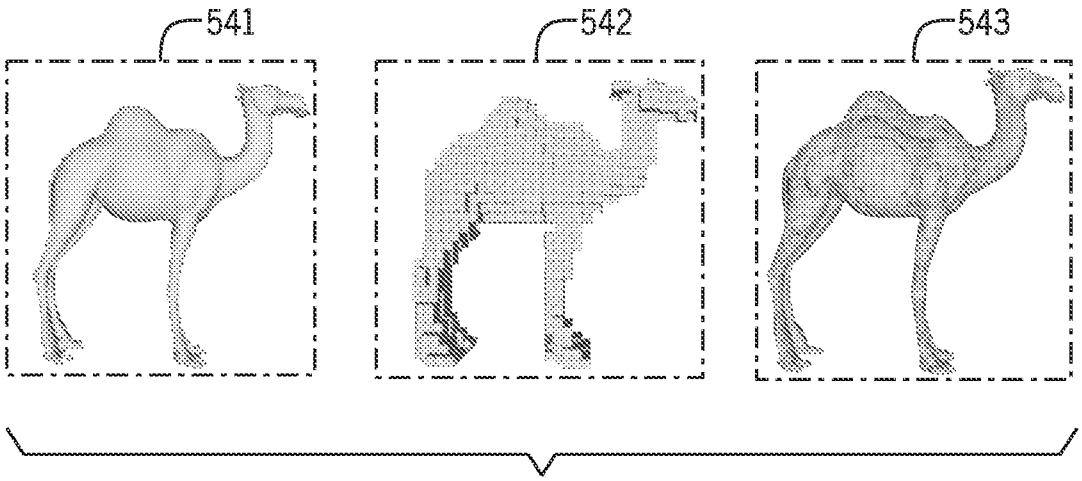

FIG. 5

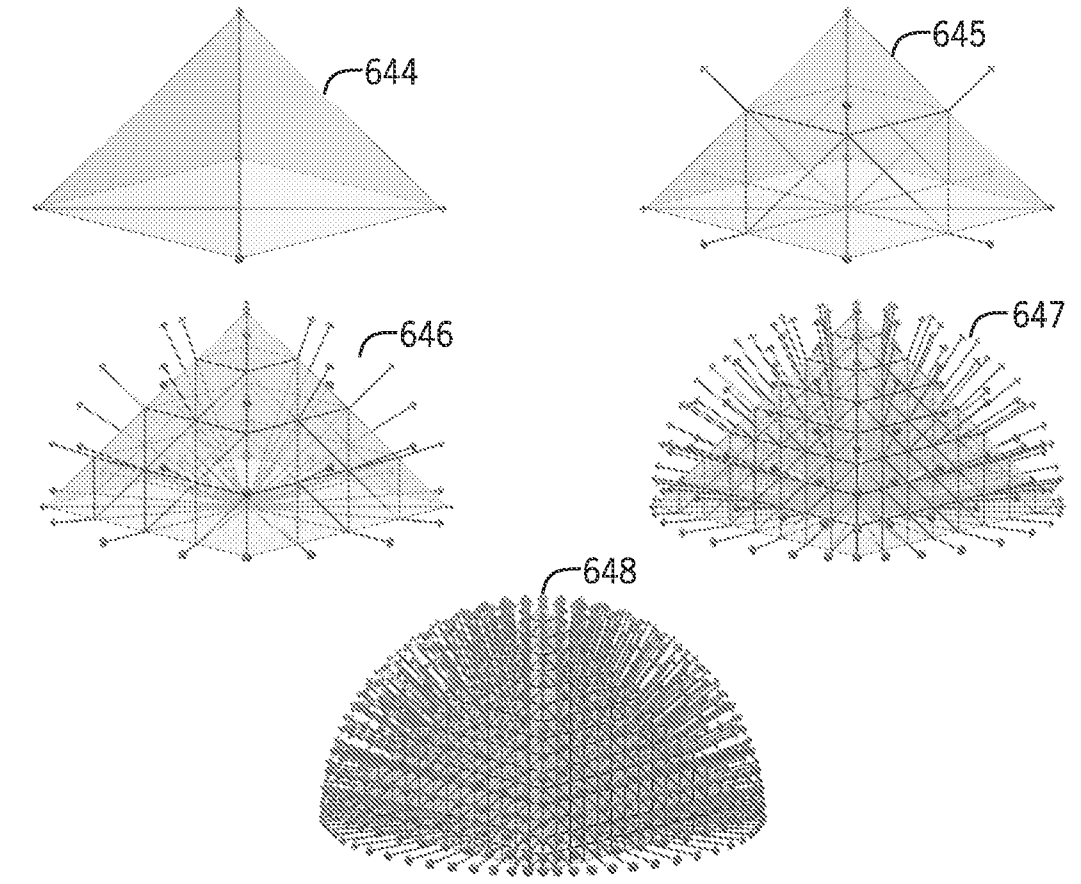

FIG. 6

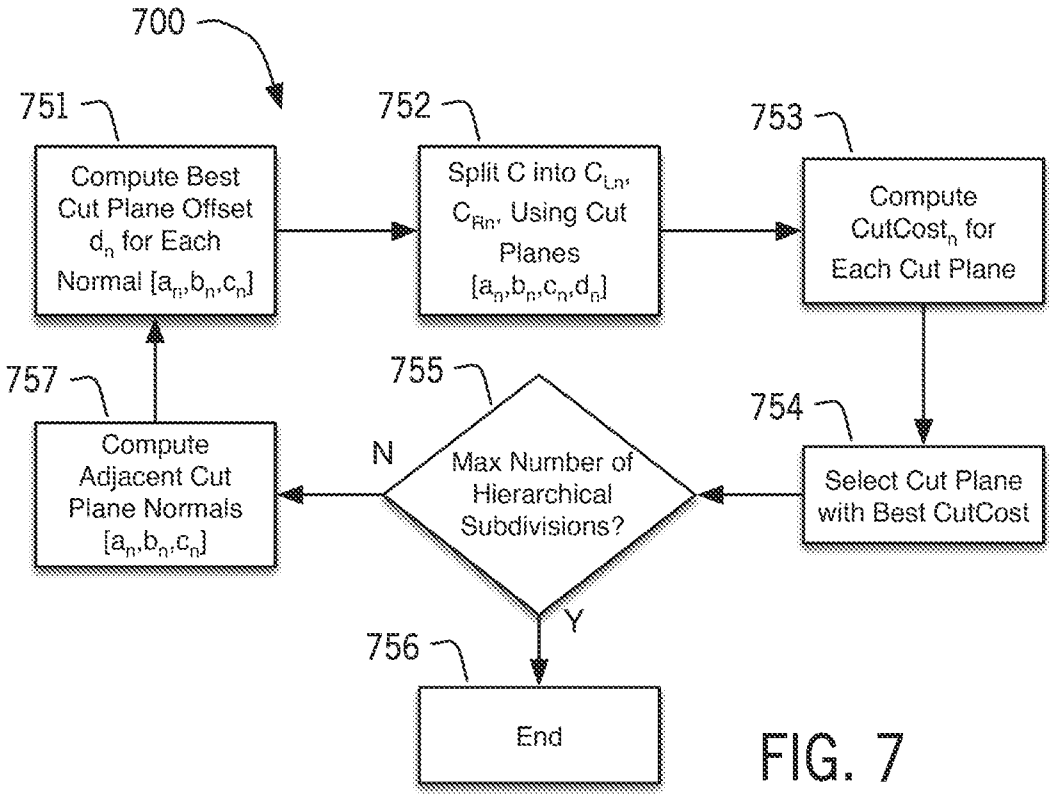
700
751
Compute Best Cut Plane Offset d_n for Each Normal $[a_n,b_n,c_n]$
752
Split C into C_{Ln}, C_{Rn}, Using Cut Planes $[a_n,b_n,c_n,d_n]$
753
Compute $CutCost_n$ for Each Cut Plane
754
Select Cut Plane with Best CutCost
755
Max Number of Hierarchical Subdivisions?
N
Y
757
Compute Adjacent Cut Plane Normals $[a_n,b_n,c_n]$
756
End

FIG. 7

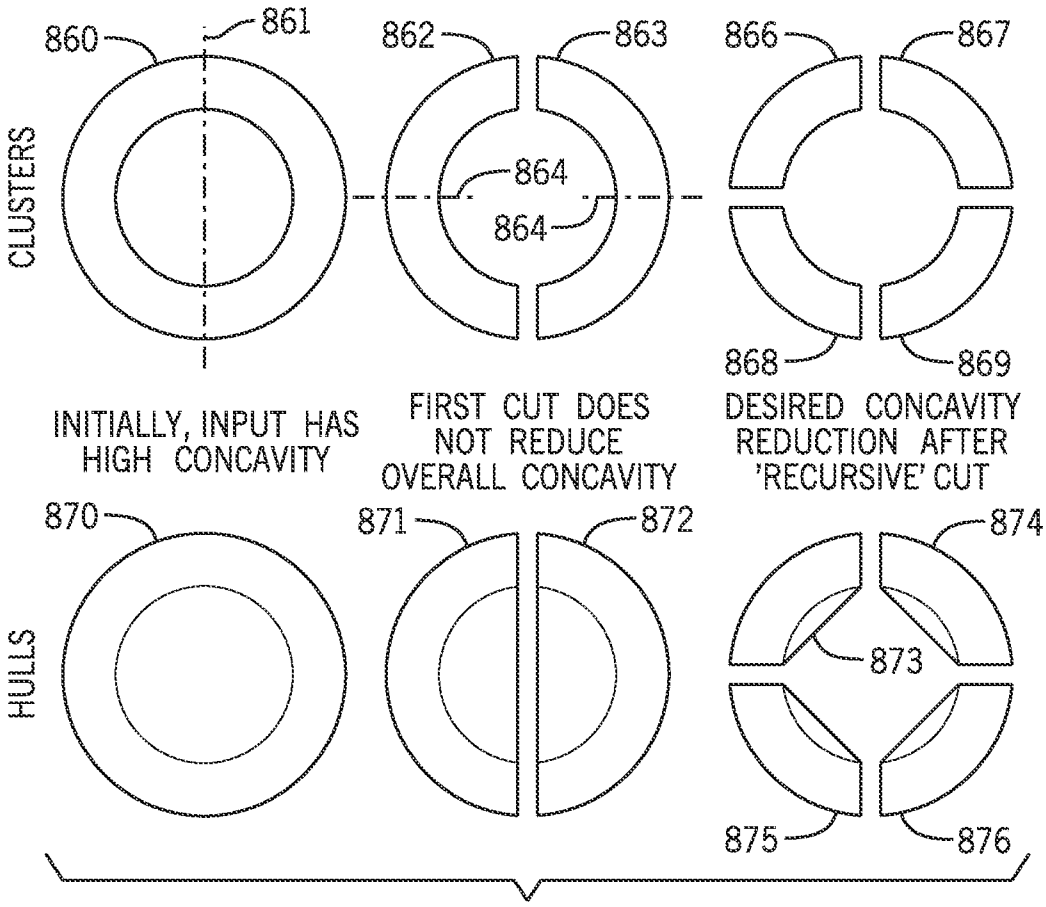
CLUSTERS
860
861
862
863
864
864
866
867
868
869
INITIALLY, INPUT HAS HIGH CONCAVITY
FIRST CUT DOES NOT REDUCE OVERALL CONCAVITY
DESIRED CONCAVITY REDUCTION AFTER 'RECURSIVE' CUT
HULLS
870
871
872
873
874
875
876

FIG. 8

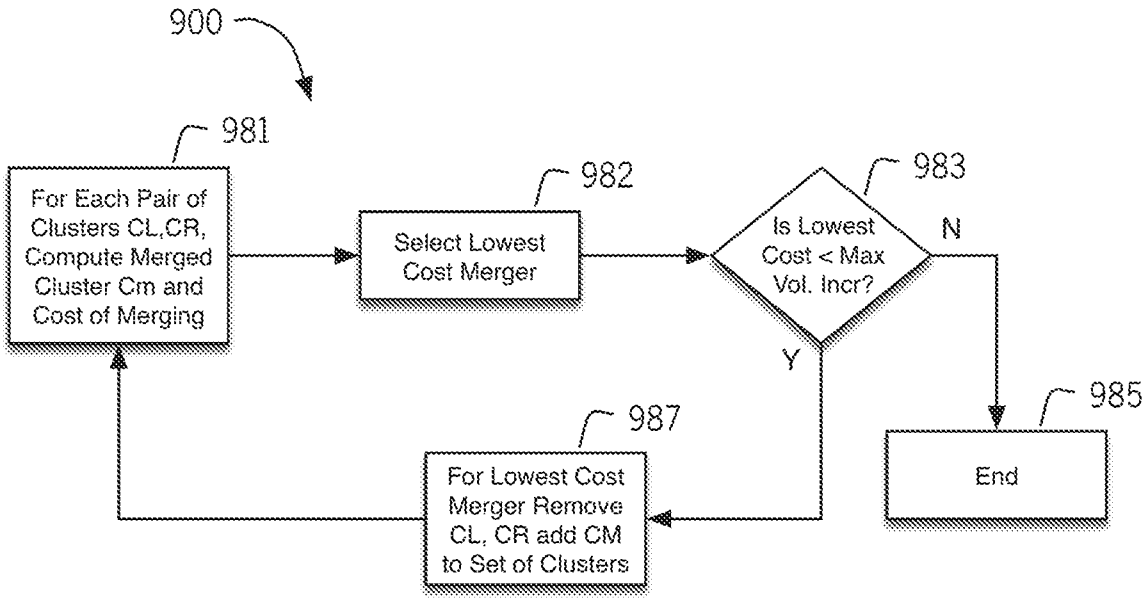
900
981
For Each Pair of Clusters CL,CR, Compute Merged Cluster Cm and Cost of Merging
982
Select Lowest Cost Merger
983
Is Lowest Cost < Max Vol. Incr?
N
Y
985
End
987
For Lowest Cost Merger Remove CL, CR add CM to Set of Clusters

FIG. 9

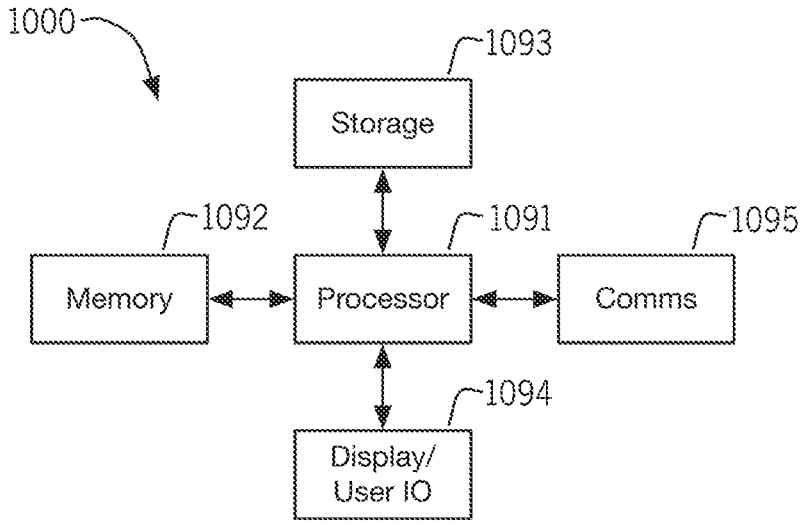
1000
1093
Storage
1092
Memory
1091
Processor
1095
Comms
1094
Display/ User IO

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2023/016956

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06T17/00 G06T19/00
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06T

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2020/327719 A1 (MASON ASHTON [GB]) 15 October 2020 (2020-10-15) abstract; figures 11,12,17 paragraphs [0002] - [0019], [0179], [0197] -----	1-24
X	JYH-MING LIEN ET AL: "Approximate convex decomposition of polyhedra", SOLID AND PHYSICAL MODELING, ACM, 2 PENN PLAZA, SUITE 701 NEW YORK NY 10121-0701 USA, 4 June 2007 (2007-06-04), pages 121-131, XP058273626, DOI: 10.1145/1236246.1236265 ISBN: 978-1-59593-666-0 Abstract, Sections 1, 3 and subsections, in particular Algorithm 1 on page 123 and surrounding text -----	1-24

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance;; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance;; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 July 2023	28/07/2023

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Almeida Garcia, B

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2023/016956

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2020327719 A1	15-10-2020	US 2020327719 A1	15-10-2020
		US 2021074056 A1	11-03-2021